*N/A*, 181 pages, 2024. 
doi:N/A


# *Boolean-Algebraic Subtyping: Intersections, Unions, Negations, and Principal Type Inference*

CHUN YIN CHAU
*The Hong Kong University of Science and Technology (HKUST)*
*Hong Kong, China*
*(e-mail: cychauab@connect.ust.hk)*

LIONEL PARREAUX
*The Hong Kong University of Science and Technology (HKUST)*
*Hong Kong, China*
*(e-mail: parreaux@ust.hk)*

**Abstract**

Intersection and union types are becoming more popular by the day, entering the mainstream in programming languages like TypeScript and Scala 3. But these types are difficult to combine with practical polymorphic type inference and their metatheory has proven difficult to establish, especially in the presence of equirecursive types and distributivity between unions and intersections. We propose *Boolean-algebraic subtyping*, a new subtyping framework for reasoning about type systems with conjunction (a.k.a. intersection), disjunction (a.k.a. union), and negation (a.k.a. complement) connectives. Our framework is *algebraic* in that it does not appeal to some underlying model of types and remains generic/extensible with respect to the specific base type constructors of the underlying language. We also present MLstruct, a programming language based on Boolean-algebraic subtyping and the first language to support principal polymorphic type inference in the presence of union and intersection types. MLstruct is structurally typed but also contains a healthy sprinkle of nominality, enabling the expression of a powerful form of extensible variants that does not require row variables and makes pivotal use of negation types. The algebraic nature of our framework is crucial in defining MLstruct: it allows the addition of nonstandard subtyping rules that would not hold in a classical set-theoretic interpretation of subtyping. With this work, we hope to foster the development of better type inference for present and future programming languages with expressive subtyping systems.

## 1 Introduction

Programming languages with ML-style type inference have traditionally avoided subtyping because of the complexities it brings over a simple unification-based treatment of type constraints. But Dolan and Mycroft (2017) recently showed with MLsub that an *algebraic* account of subtyping resolved many of these difficulties and enabled the inference of precise types that more accurately reflect the flow of expressions in programs. Unfortunately, among other limitations, MLsub does not support unrestricted union and intersection types, which

are emerging as important building blocks in the design of structurally-typed programming languages like TypeScript, Flow, Scala 3, and others.

In this paper, we propose a new algebraic subtyping framework that subsumes MLsub and adds support for first-class union, intersection, and negation types. By *first-class*, we mean that these types can be used without any restrictions, and in particular they can be written down by users in arbitrary type annotations, which is impossible in MLsub.

We present the MLstruct programming language and thereby show that ML-style type inference with subtyping can be generalized to include well-behaved forms of union and intersection types as well as pattern matching on single-inheritance class hierarchies. As a first example, consider the following definitions:

```
class Some[A]: { value: A }
class None: {}
```

```
def flatMap f opt = case opt of
  Some → f opt.value,
  None → None{}
```

The type inferred by our system for `flatMap` is:

$$\mathrm{flatMap}: \forall \alpha, \beta.\ (\alpha \to \beta) \to (\mathrm{Some}[\alpha] \vee \mathrm{None}) \to (\beta \vee \mathrm{None})$$

Interestingly, this is more general than the traditional type given to `flatMap` for `Option` types. Indeed, our `flatMap` does not require the function passed in argument to return either a `None` or a `Some` value, but allows it to return anything it wants (any $\beta$), which gets merged with the `None` value returned by the other branch (yielding type $\beta \vee \mathrm{None}$). For example,

```
let res = flatMap (fun x → x) (Some{value = 42})
```

is given type $42 \vee \mathrm{None}$[1] because the function may return either `42` or `None`. A value of this type can later be inspected with an instance match expression of the form:

```
case res of Int → res, None → 0
```

which is inferred to be of type $42 \vee 0$, a subtype of $\mathrm{Nat}$. This is not the most general version of `flatMap` either. We can also make the function open-ended, accepting either a `Some` value or *anything else*, instead of just `Some` or `None`, by using a default case (denoted by the underscore '_'):

```
def flatMap2 f opt = case opt of Some → f opt.value, _ → opt
```

This `flatMap2` version has the following type inferred, where $\vee$ and $\wedge$ have the usual precedence:

$$\mathrm{flatMap2}: \forall \alpha, \beta.\ (\alpha \to \beta) \to (\mathrm{Some}[\alpha] \vee \beta \wedge \neg\, \#\mathrm{Some}) \to \beta$$

This type demonstrates a central aspect of our approach: the use of *negation types* (also called *complement* types), written $\neg\tau$, which allows us to find principal type solutions in tricky typing situations. Here, type $\#\mathrm{Some}$ is the *nominal tag* of class $\mathrm{Some}$. A nominal tag represents the *identity* of a class, disregarding the values of its fields and type parameters: if a value $v$ has type $\#\mathrm{Some}$, this means $v$ is an instance of $\mathrm{Some}$, while if $v$ has type $\neg\#\mathrm{Some}$, this means it is not. To showcase different usages of this definition, consider the following calls along with their inferred types:[2]

```
ex1 =  flatMap2 (fun x → x + 1) 42                          :  Int
ex2 =  flatMap2 (fun x → Some{value = x}) (Some{value = 12}) :  Some[12]
ex3 =  flatMap2 (fun x → Some{value = x}) 42                 :  Some[⊥] ∨ 42
```

[1] MLstruct supports singleton types for constant literals. For example, 42 is both a value and a type, with $42 : 42 \leqslant \mathrm{Nat} \leqslant \mathrm{Int}$.

[2] Notice that only `ex3` features a union of two distinct type constructors '$\mathrm{Some}[\bot] \vee 42$' because in `ex1` and `ex2` only one concrete type constructor statically flows into the result of the expression (42 and $\mathrm{Some}$, respectively).

It is easy to see that instantiating $\beta$ to $\mathrm{Int}$ and $\mathrm{Some}[12]$ respectively allows `ex1` and `ex2` to type check. In `ex3`, both types $\mathrm{Some}[\gamma]$ and 42 flow into the result, for some type inference variable $\gamma$, but $\gamma$ is never constrained and only occurs positively so it can be simplified, yielding $\mathrm{Some}[\bot] \vee 42$. We can convert `ex3` to `42` through a **`case`** expression using the `impossible` helper function:[3]

```
def impossible x = case x of {}                          :  ⊥ → ⊥
case ex3 of Int → ex3, Some → impossible ex3.value       :  42
```

One may naively think that the following type could fit `flatMap2` as well:

$$\mathrm{flatMap2\_wrong} : \forall \alpha, \beta, \gamma.\ (\alpha \to \beta) \to (\mathrm{Some}[\alpha] \vee \gamma) \to (\beta \vee \gamma)$$

but this type does not work. To see why, consider what happens if we instantiate the type variables to $\alpha = \mathrm{Int}$, $\beta = \mathrm{Int}$, and $\gamma = \mathrm{Some}[\mathrm{Bool}]$. This yields the type:

$$\mathrm{flatMap2\_wrong}' : (\mathrm{Int} \to \mathrm{Int}) \to (\mathrm{Some}[\mathrm{Int}] \vee \mathrm{Some}[\mathrm{Bool}]) \to (\mathrm{Int} \vee \mathrm{Some}[\mathrm{Bool}])$$

which would allow the call `flatMap2 (fun x → x + 1) (Some{value = false})` because $\mathrm{Some}[\mathrm{Bool}] \leqslant \mathrm{Some}[\mathrm{Int}] \vee \mathrm{Some}[\mathrm{Bool}]$. This expression, however, would crash with a runtime type mismatch! Indeed, the shape of the `Some` argument matches the first branch of `flatMap2`'s **`case`** expression, and therefore **`false`** is passed to our argument function, which tries to add `1` to it as though it was an integer... So we *do* need the negation that appears in the correct type of `flatMap2`, as it prevents passing in arguments that are also of the `Some` shape, but with the wrong type arguments.

Finally, let us push the generality of our function further yet, to demonstrate the flexibility of the system. Consider this last twist on `flatMap` for optional values, which we will call `mapSome`:

```
def mapSome f opt = case opt of Some → f opt, _ → opt
```

The difference with the previous function is that this one does not unwrap the `Some` value received in argument, but simply passes it unchanged to its function argument. Its inferred type is:

$$\mathrm{mapSome} : \forall \alpha, \beta.\ (\alpha \to \beta) \to (\alpha \wedge \#\mathrm{Some} \vee \beta \wedge \neg\, \#\mathrm{Some}) \to \beta$$

This type shows that it does not matter what specific subtype of $\mathrm{Some}$ we have in the first branch: as long as the argument has type $\alpha$ ***when it is*** a $\mathrm{Some}$ instance, then $\alpha$ is the type the argument function should take, without loss of generality. This demonstrates that our type system can tease apart different flows of values based on the nominal identities of individual matched classes.

As an example of the additional flexibility afforded by this new function, consider the following:

```
class SomeAnd[A, P]: Some[A] ∧ { payload: P }
let arg = if ⟨arbitrary condition⟩ then SomeAnd{value = 42, payload = 23}
                                  else None{}
in mapSome (fun x → x.value + x.payload) arg
```

of inferred type $\mathrm{Int} \vee \mathrm{None}$. Here, we define a new subclass of `Some` containing an additional `payload` field, and we use this class instead of `Some`, allowing the `payload` field to be used

[3] One may expect $\mathrm{Some}[\bot] \equiv \bot$, but this does not hold in MLstruct, as it would prevent effective principal type inference by requiring some amount of backtracking in the constraint solver and by extension in the type checker as well (Castagna et al., 2016).

from within the function argument we pass to `mapSome`. This is not expressible in OCaml polymorphic variants (Garrigue, 2001) and related systems (Ohori, 1995). More powerful systems with row variables (Rémy, 1994; Pottier, 2003) would still fail here because of their use of *unification*: `mapSome` merges its `opt` parameter with the result, so these systems would yield a unification error at the `mapSome` call site, because the argument function returns an integer instead of a value of the same type as the input:[4] subtyping makes MLstruct more flexible than existing systems based on row variable.

MLscript is a new programming language developed at the Hong Kong University of Science and Technology[5] featuring first-class unions, intersections, negations, and ML-style type inference, among other features. For simplicity, this paper focuses on a *core subset* of MLscript referred to as ***MLstruct***, containing only the features relevant to principal type inference in a Boolean algebra of structural types, used in all examples above. An MLstruct implementation is provided as an artifact (Parreaux et al., 2022) and available at github.com/hkust-taco/mlstruct, with a web demonstration at hkust-taco.github.io/mlstruct. The specific contributions we make are the following:

- We present MLstruct (Section 2), which subsumes both the original ML type system and the newer MLsub (Dolan, 2017), extending the latter with simple class hierarchies and class-instance matching based on union, intersection, and negation type connectives.
- We formalize the Boolean-Algebraic Subtyping framework as a generic theory of subtyping $\mathfrak{S}(\mathcal{T}, \mathcal{R})$ that makes few assumptions on the concrete base type constructors $\mathcal{T}$ of the underlying language and their base subtyping rules $\mathcal{R}$ (Section 3).
- We prove the soundness of Boolean-Algebraic subtyping by showing that $\mathfrak{S}(\mathcal{T}, \mathcal{R})$ does not relate unrelated type constructors (Section 4), a crucial property that underlies the soundness of languages built on top of it, like MLstruct.
- We describe our approach to type inference based on the Boolean-algebraic properties of MLstruct's subtyping system (Section 5). To the best of our knowledge, MLstruct is the first language to support complete polymorphic type inference with union and intersection types. Moreover, it *does not rely on backtracking* and yields principal types that are amenable to simplification.
- We formalize the declarative semantics of MLstruct in the $\lambda^{\neg}$ calculus (Section 6). We state the standard soundness properties of *progress* and *preservation*, whose complete proofs are given in Appendix C.
- We formally describe our type inference algorithm (Section 7). We state its *soundness* and *completeness* theorems. Again, the proofs can be found in Appendix C.

[4] Wrapping the result in `Some` would not work either (as `Some Int` doesn't *unify* with `Some {value: Int, payload: Int}`).
[5] The GitHub repository of the full MLscript language is available at `https://github.com/hkust-taco/mlscript`.

## 2 Presentation of MLstruct

MLstruct is a research language designed to explore type inference with subtyping in the presence of first-class union and intersection types. This minimal language is carved out from the MLscript programming language which is currently being developed as a real-world general-purpose programming language. MLstruct *subsumes* Dolan's MLsub, the previous state of the art in type inference with subtyping, which itself subsumes traditional ML typing: all ML terms are typeable in MLsub and all MLsub terms are typeable in MLstruct. On top of this fertile ML substrate pollinated with MLsub's rich subtyping theory of records and equirecursive types, MLstruct grows structurally-typed abstractions in the form of unions, intersections, negations, structural class types, and class-instance matching. We now present these features along with some examples.

### *2.1 Overview of MLstruct Features*

An MLstruct program is made of top-level statements followed by an expression, the program's body. A statements can be either a type declaration (class or type alias) or a top-level function definition, written `def f = t` or `rec def f = t` when `f` is recursive. MLstruct infers polymorphic types for `def` bindings, allowing them to be used at different type instantiations in the program.

#### *2.1.1 Polymorphism*

Polymorphic types include a set of type variables with bounds, such as $\forall(\alpha \leqslant \mathrm{Int}).\ \mathrm{List}[\alpha] \to \mathrm{List}[\alpha]$. The bounds of polymorphic types are allowed to be cyclic, which can be interpreted as indirectly describing recursive types. For example, $\forall(\alpha \leqslant \top \to \alpha).\ \alpha$ is the principal type scheme of `rec def f = fun a → f` which accepts any argument and returns itself. To simplify the presentation of inferred polymorphic types with recursive bounds, such as $\forall(\alpha \leqslant \alpha \to \beta), \beta.\ \alpha \to \beta$, we may use an equivalent '`as`' shorthand, as follows: $\forall\beta.\ ((\alpha \to \beta)\ \mathbf{as}\ \alpha) \to \beta$.

MLstruct applies aggressive simplification on inferred types, removing redundant type variables and inlining simple type variable bounds (see Section 5.5), so that they are usually fairly concise.

#### *2.1.2 Classes and Inheritance*

Because object orientation is not the topic of this paper, which focuses on functional-style use cases, the basic OO constructs of MLstruct presented here are intentionally bare-bone. Classes are declared with the following syntax:

```
class C[A, B, ...]: D[S, T, ...] ∧ { x: X, y: Y, ... }
```

where `A`, `B`, etc. are type parameters, `S`, `T`, `X`, `Y`, etc. are arbitrary types and `D` is the parent class of `C`, which can be left out if the class has no parents. Along with a *type* constructor $\mathrm{C}[A,\ B,\ \ldots]$, such a declaration also introduces a *data* constructor C of type:

$$\mathrm{C} : \forall\, \beta_1, \beta_2, \ldots, (\alpha_1 \leqslant \tau_1), (\alpha_2 \leqslant \tau_2), \ldots .\ \{\, x_1 : \alpha_1,\ x_2 : \alpha_2,\ \ldots \,\} \to \mathrm{C}[\beta_1, \beta_2 \ldots] \wedge \{\, x_1 : \alpha_1,\ x_2 : \alpha_2,\ \ldots \,\}$$

where $x_i$ are all the fields declared by $\mathrm{C}[\beta_1, \beta_2, \ldots]$ or by any of its ancestors in the inheritance hierarchy, and $\tau_i$ are the corresponding types – if a field is declared in several

classes of the hierarchy, we take the intersection of all the declared types for that field. To retain as precise typing as possible, we let the types of the fields taken in parameters to be arbitrary *subtypes* $\alpha_i$ of the declared $\tau_i$, so we can refine the result type $\mathrm{C}[\beta_1, \beta_2 \ldots] \wedge \{ x_1 : \alpha_1,\ x_2 : \alpha_2,\ \ldots \}$ to retain these precise types. For instance, assuming `class C: { x: Int }`, term `C { x = 1 }` is given the precise type $\mathrm{C} \wedge \{ x : 1 \}$.

Classes are restricted to single-inheritance hierarchies. Like in the work of Muehlboeck and Tate (2018), this has the nice property that it allows union types to be refined by reducing types like $(C_0 \vee \tau) \wedge C_1$ to $C_0 \wedge C_1 \vee \tau \wedge C_1$ by distributivity and to just $\tau \wedge C_1$ when $C_0$ and $C_1$ are unrelated ($C_0 \wedge C_1 \equiv \bot$). But MLstruct can easily be extended to support traits, which are not subject to this restriction, by slightly adapting the definition of type normal forms (our artifact (Parreaux et al., 2022) implements this). Thanks to their use of negation types (described in Section 6.3), the typing rules for pattern matching do not even have to change, and traits can also be pattern-matched. In fact, the full MLscript language supports mixin trait composition (Schärli et al., 2003) similar to Scala (Odersky et al., 2004), whereby traits can be inherited alongside classes, and method overriding is resolved in so-called "linearization order."

### *2.1.3 Shadowing*

Non-recursive `def`s use shadowing semantics,[6] so they can simulate the more traditional field initialization and overriding semantics of traditional class constructors. For instance:

```
class Person: {name: Str, age: Nat, isMajor: Bool}
def Person n a = Person{name = capitalize n, age = a, isMajor = a >= 18}
```

in which the `def`, of inferred type $\mathrm{Person}_1 : \forall(\alpha \leqslant \mathrm{Nat}).\ \mathrm{Str} \to \alpha \to \mathrm{Person} \wedge \{ \mathrm{age} : \alpha \}$, *shadows* the bare constructor of the `Person` class (of type $\mathrm{Person}_0 : \forall(\alpha \leqslant \mathrm{Str}), (\beta \leqslant \mathrm{Nat}), (\gamma \leqslant \mathrm{Bool}).\ \ \{ \mathrm{name} : \alpha,\ \mathrm{age} : \beta,\ \mathrm{isMajor} : \gamma \} \to \mathrm{Person} \wedge \{ \mathrm{name} : \alpha,\ \mathrm{age} : \beta,\ \mathrm{isMajor} : \gamma \}$), forcing users of the class to go through it as the official `Person` constructor. Function `capitalize` returns a Str, so no 'name' refinement is needed ($\mathrm{Person} \wedge \{ \mathrm{age} : \alpha,\ \mathrm{name} : \mathrm{Str} \} \equiv \mathrm{Person} \wedge \{ \mathrm{age} : \alpha \}$).

### *2.1.4 Nominality*

Classes are not equivalent to their bodies. Indeed, they include a notion of "nominal identity", which means that while a class type is a *subtype* of its body, it is not a *supertype* of it. So unlike TypeScript, it is not possible to use a record `{x = 1}` as an instance of a class declared as `class C: {x: Int}`. To obtain a `C`, one must use its constructor, as in `C{x = 1}`. This nominality property is a central part of our type system and is much demanded by users in practice.[7] It comes at no loss of generality, as type synonyms can be used if nominality is not wanted.

[6] Type names, on the other hand, live in a different namespace and are not subject to shadowing.

[7] The lack of nominal typing for classes has been a major pain point in TypeScript. The issue requesting it, created in 2014 and still not resolved, has accumulated more than 500 "thumbs up". See: https://github.com/Microsoft/Typescript/issues/202.

### 2.1.5 Type Aliases

Arbitrary types can be given names using the syntax `type X[A, B, ...] = T`. Type aliases and classes can refer to each other freely and can be mutually recursive.

### 2.1.6 Guardedness Check

Classes and type aliases are checked to ensure they do not inherit or refer to themselves immediately without going through a "concrete" type constructor first (i.e., a function or record type). For instance, the recursive occurrence of `A` in `type A[X] = Id[A[X]] ∨ Int` where `type Id[Y] = Y` is unguarded and thus illegal, but `type A[X] = { x: A[X] } ∨ Int` is fine.

### 2.1.7 Class-Instance Matching

As presented in the introduction, one can match values against class patterns in a form of primitive pattern matching. Consider the following definitions:

```
class Cons[A]: Some[A] ∧ { tail: List[A] }   type List[A] = Cons[A] ∨
None
rec def mapList f ls = case ls of
  Cons → Cons{value = f ls.value, tail = mapList f ls.tail},
  None → None{}
```

of inferred type:[8]

$$\mathrm{mapList} : \forall \alpha, \beta.\ (\alpha \to \beta) \to (\mathrm{Cons}[\alpha] \wedge \{\,\mathrm{tail} : \gamma\,\} \vee \mathrm{None})\ \textbf{as}\ \gamma \to (\mathrm{Cons}[\beta] \wedge \{\,\mathrm{tail} : \delta\,\} \vee \mathrm{None})\ \textbf{as}\ \delta$$

We define a `List` type using `None` as the "nil" list and whose `Cons` constructor *extends* `Some` (from the introduction). A list in this encoding can be passed to any function that expects an option in input — if the list is a `Cons` instance, it is also a `Some` instance, and the `value` field representing the head of the list will be used as the `value` wrapped by the option. This example demonstrates that structural typing lets us mix and match as well as refine different constructors in a flexible way.

As a slightly bigger motivating example, the `List` type thus defined can then be used as follows, defining the classical `unzip` combinator:

```
  def Cons head tail = Cons { value = head, tail = tail }
def None = None{}
  rec def unzip xs = case xs of
    None → { fst = None, snd = None },
    Some → let tmp = unzip xs.tail in { fst = Cons xs.value.fst tmp.fst ,
                                        snd = Cons xs.value.snd tmp.snd }
```

Below are two possible types that may be annotated explicitly by the user for these definitions, and which will be automatically checked by MLstruct for conformance (a.k.a., *subsumption*, see Section 5.4) against their inferred types.[9]

```
def Cons:  α → (β ∧ List[α]) → (Cons[α] ∧ { value: α, tail: β })
def unzip: List[{ fst: α, snd: β }] → { fst: List[α], snd: List[β] }
```

[8] The `where` keyword is used to visually separate the specification of type variable bounds, making them more readable.

[9] Annotating the types of public functions, while not required by MLstruct, is seen as good practice in some communities. Moreover, the subsumption mechanism can be used to provide and check module signatures in an ML-style module system.

### 2.1.8 Records

Record values are built using the syntax `{x1 = t1, x2 = t2, ...}` and are assigned the corresponding types $\{ x_1 : \tau_1,\ x_2 : \tau_2,\ \ldots \}$. Record types are related via the usual *width* and *depth* subtyping relationships. Width subtyping means that, for instance, $\{ x : \tau_1,\ y : \tau_2 \} \leqslant \{ x : \tau_1 \}$, and depth subtyping means that, for instance, $\{ x : \tau_1,\ y : \tau_2 \} \leqslant \{ x : \tau_1,\ y : \tau_3 \}$ if $\tau_2 \leqslant \tau_3$.

## 2.2 Constructing the Lattice of Types

The algebraic subtyping philosophy of type system design is to begin with the subtyping of data types (records, functions, etc.) and to define the order connectives to fit this subtyping order, rather than to follow set-theoretic intuitions. We follow this philosophy and aim to design our subtyping order to tackle the following design constraints:

(A) The order connectives $\wedge$, $\vee$, and $\neg$ should induce a Boolean algebra, so that we can manipulate types using well-known and intuitive Boolean-algebraic reasoning techniques.
(B) Nominal tags and their negations specifically should admit an intuitive set-theoretic understanding, in the sense that for any class $C$, type $\#C$ should denote all instances of $C$ while type $\neg\#C$ should correspondingly denote all instances that are *not* derived from class $C$.[10]
(C) The resulting system should admit principal types as well as an *effective* polymorphic type inference strategy, where "effective" means that it should not rely on backtracking.

### 2.2.1 Lattice Types

*Top*, written $\top$, is the type of all values, a supertype of every other type. Its dual *bottom*, written $\bot$, is the type of no values, a subtype of every other type. For every $\tau$, we have $\bot \leqslant \tau \leqslant \top$. Intersection $\wedge$ and union $\vee$ types are the respective *meet* and *join* operators in the subtyping lattice. It is worth discussing possible treatments one can give these connectives:

1. We can axiomatize them as denoting the intersection $\cap$ and union $\cup$ of the sets of values that their operands denote, which is the approach taken by semantic subtyping.
2. We can axiomatize them as *greatest lower bound* (GLB) and *least upper bound* (LUB) operators, usually written $\sqcap$ and $\sqcup$, whose meaning is given by following the structure of a preexisting lattice of simple types (types without order connectives). In this interpretation, we can *calculate* the results of these operators when their operands are concretely known.
3. Finally, we can view $\wedge$ and $\vee$ as type constructors in their own right, with dedicated subtyping derivation rules. Then unions and intersections are not "computed away"

[10] By contrast, we have no specific requirements on the meaning of negated function and record types, which are uninhabited.

but instead represent proper constructed types, which may or may not be equivalent to existing simple types.

### 2.2.2 Subtyping

We base our approach primarily on (3) but we do include a number of subtyping rules whose goal is to make the order connectives behave like (2) in some specific cases:

- We posit $\#C_1 \wedge \#C_2 \leqslant \bot$ whenever classes $C_1$ and $C_2$ are unrelated.[11] This makes sense because there are no values that can be instances of both classes at the same time, due to single inheritance. We obviously also have $\#C_1 \wedge \#C_2 \geqslant \bot$, meaning the two sides are equivalent (they subtype each other), which we write $\#C_1 \wedge \#C_2 \equiv \bot$. On the other hand, $\#C \leqslant \#D$ for all $C$, $D$ where $C$ inherits from $D$; so when $\#C_1$ and $\#C_2$ *are* related then either $\#C_1 \wedge \#C_2 \equiv \#C_1$ or $\#C_1 \wedge \#C_2 \equiv \#C_2$. Overall, we can always "reduce" intersections of nominal class tags to a single non-intersection type, making $\wedge$ behave like a GLB operator in the class inheritance sublattice, made of nominal tags, $\top$, $\bot$, and $\vee$, evocative of (2).
- We also posit the nonstandard rule $(\tau_1 \to \tau_2) \wedge (\tau_3 \to \tau_4) \leqslant (\tau_1 \vee \tau_3) \to (\tau_2 \wedge \tau_4)$. The other direction holds by function parameter contravariance and result covariance, so again the two sides are made equivalent. $\wedge$ behaves like a GLB operator on function types in a lattice which does not contain subtyping-based overloaded functions types, such as those of Pottier (1998*b*); Dolan (2017). This rule is illogical from the set-theoretic point of view: a function that can be viewed as returning a $\tau_2$ when given a $\tau_1$ and returning a $\tau_4$ when given a $\tau_3$ cannot be viewed as *always* returning a $\tau_2 \wedge \tau_4$. For instance, consider $\lambda x.\, x$, typeable both as $\mathrm{Int} \to \mathrm{Int}$ and as $\mathrm{Bool} \to \mathrm{Bool}$. According to both classical intersection type systems and the semantic subtyping interpretation, this term could be assigned type $(\mathrm{Int} \to \mathrm{Int}) \wedge (\mathrm{Bool} \to \mathrm{Bool})$. But *we posited* that this type is equivalent to $(\mathrm{Int} \vee \mathrm{Bool}) \to (\mathrm{Int} \wedge \mathrm{Bool})$. Thankfully, in $\lambda^{\neg}$ $\lambda x.\, x$ cannot be assigned such an intersection type; instead, its most general type is $\forall \alpha.\ \alpha \to \alpha$, which does subsume both $\mathrm{Int} \to \mathrm{Int}$ and $\mathrm{Bool} \to \mathrm{Bool}$, but not $(\mathrm{Int} \to \mathrm{Int}) \wedge (\mathrm{Bool} \to \mathrm{Bool})$. This explains why intersection types cannot be used to encode overloading in $\lambda^{\neg}$.[12]
- For record intersections, we have the standard rule that $\{\, x : \tau \,\} \wedge \{\, x : \pi \,\} \leqslant \{\, x : \tau \wedge \pi \,\}$, making the two sides equivalent since the other direction holds by depth subtyping. Intersections of distinct record fields, on the other hand, do not reduce and stay as they are — in fact, multi-field record types are encoded, in MLstruct, as intersections of individual single-field record types, following Reynolds (1997). For instance, assuming $x \neq y$, then $\{\, x : \tau_1,\ y : \tau_2 \,\}$ is *not* a core form but merely *syntax sugar* for $\{\, x : \tau_1 \,\} \wedge \{\, y : \tau_2 \,\}$.
- We apply similar treatments to various forms of unions: First, $(\tau_1 \to \tau_2) \vee (\tau_3 \to \tau_4) \equiv (\tau_1 \wedge \tau_3) \to (\tau_2 \vee \tau_4)$, the dual of the function intersection treatment mentioned above. Second, we recognize that $\{\, x : \tau \,\} \vee \{\, y : \pi \,\}$ and $\{\, x : \tau \,\} \vee (\pi_1 \to \pi_2)$, where $x \neq y$, cannot be meaningfully used in a program, as the language has no feature

[11] This class intersection annihilation rule is not novel; for example, Ceylon has a similar one (Muehlboeck and Tate, 2018).

[12] Other forms of overloading, such as type classes and *constructor overloading* (see Section 8), are still possible.

allowing to tease these two components apart, so we identify these types with $\top$, the top type. This is done by adding $\top \leqslant \{ x : \tau \} \vee \{ y : \pi \}$ and $\top \leqslant \{ x : \tau \} \vee (\pi_1 \to \pi_2)$ as subtyping derivation rules.

The full specification of our subtyping theory is presented later, in Section 6 (Figure 16).

### 2.2.3 Soundness

The soundness of subtyping disciplines was traditionally studied by finding semantic models corresponding to types and subtyping, where types are typically understood as predicates on the denotations of $\lambda$ terms (obtained from some $\lambda$ model) and where subtyping is understood as inclusion between the corresponding sets of denotations. In this paper, we take a much more straightforward approach: all we require from the subtyping relation is that it be *consistent*, in the sense that it correctly relate types constructed from the same constructors and that it not relate unrelated type constructors. For instance, $\tau_1 \to \tau_2 \leqslant \pi_1 \to \pi_2$ should hold *if and only if* $\pi_1 \leqslant \tau_1$ and $\tau_2 \leqslant \pi_2$, and $\{ x : \mathrm{Int} \} \leqslant \#C$ should *not* be derivable. This turns out to be a sufficient condition for the usual soundness properties of *progress* and *preservation* to hold in our language. Consistency is more subtle than it may first appear. We cannot identify, e.g., $\#C \vee \{ x : \tau \}$ with $\top$ even though the components of this type cannot be teased apart through instance matching, as doing so is incompatible with distributivity. Notice the conjunctive normal form of $\pi = \#C \wedge \{ x : \tau \} \vee \#D \wedge \{ y : \tau' \}$ is $\pi \equiv (\#C \vee \#D) \wedge (\#C \vee \{ y : \tau' \}) \wedge (\{ x : \tau \} \vee \#D) \wedge (\{ x : \tau \} \vee \{ y : \tau' \})$. We can make $\{ x : \tau \} \vee \{ y : \tau' \}$ equivalent to $\top$ when $x \neq y$ because that still leaves $\pi \equiv (\#C \vee \#D) \wedge (\#C \vee \{ y : \tau' \}) \wedge (\{ x : \tau \} \vee \#D)$, which is equivalent to the original $\pi$ by distributivity and simplification. But making $\#C \vee \{ y : \tau' \}$ and $\{ x : \tau \} \vee \#D$ equivalent to $\top$ would make $\pi \equiv \#C \vee \#D$, losing all information related to the fields, and breaking pattern matching!

### 2.2.4 Negation Types

Finally, we can add Boolean-algebraic negation to our subtyping lattice. In some languages, the values of a negation type $\neg\tau$ are intuitively understood as *all* values that are *not* of the negated type $\tau$. However, nothing dictates that this intuition should always hold in a Boolean-algebraic subtyping system, where negation is interpreted *algebraically* and is not given any *a-priori* meaning in terms of the concrete values that can be typed with it, if any.

MLstruct has negation types out of the box as part of its Boolean-Algebraic subtyping lattice. However, the interpretation of these types is at the same time considerably constrained by the conjunction of the rules already presented in Section 2.2.2 and the existing Boolean-algebraic relationships. In practice, this means that the intuition that the values of $\neg\tau$ are those that are not of type $\tau$ only holds when $\tau$ is a nominal tag in MLstruct. For other constructs, such as functions and records, negations assume a purely algebraic role. For instance, we have relationships like $\neg\{ x : \tau \} \leqslant \pi_1 \to \pi_2$ due to $\{ x : \tau \} \vee \pi_1 \to \pi_2$ being identified with $\top$ (see also Section 3.3.5). Because no values inhabit types like $\neg\{ x : \tau \}$ and $\neg(\pi_1 \to \pi_2)$, these types should be essentially thought of as special *bottom* types that, for algebraic reasons, technically have to contain more static information than $\bot$ and have to possess fewer subtyping relationships.

Negations can express interesting patterns, such as safe division, as seen below, where '`e : T`' is used to ascribe a type `T` to an expression `e`:

```
def div n m = n / (m : Int ∧ ¬0)
div: Int → (Int ∧ ¬0) → Int
```

```
def f x = div x 2
f: Int → Int
```

```
def g (x: Int) = div 100 x    ⇐ error: found Int, expected Int ∧ ¬0
```

```
def div_opt n m = case m of 0 → None{}, _ → Some{value = div n m}
div_opt: Int → Int → (None ∨ Some[Int])
```

Here, '`case m of ...`' is actually a shorthand for the core form '`case m = m of ...`' which shadows the outer `m` with a local variable `m` that is assigned a more refined type in each `case` branch.

As we saw in the introduction, $\neg$ also allows for the sound typing of class-instance matching with default cases. Moreover, together with $\top$, $\bot$, $\wedge$, and $\vee$, our type structure forms a Boolean lattice, whose algebraic properties are essential to enabling principal type inference (see Section 5.3.1).

#### *2.2.5 Structural Decomposition*

We reduce complex object types to simpler elementary parts, which can be handled in a uniform way. Similarly to type aliases, which can always be replaced by their bodies, we can replace class types by their fields intersected with the corresponding nominal tags. For example, $\mathrm{Cons}[\tau]$ as defined in Section 2.1.7 reduces to $\#\mathrm{Cons} \wedge \{\,\mathrm{value} : \tau\,,\ \mathrm{tail} : \mathrm{List}[\tau]\,\}$. Recall that class tags like $\#\mathrm{Cons}$ represent the *nominal identities* of classes. They are related with other class tags by a subtyping relationship that follows the inheritance hierarchy. For instance, given **class** $C[\alpha] : D[\alpha \vee 2] \wedge \{\, x : 0 \vee \alpha \,\}$ and **class** $D[\beta] : \{\, x : \beta,\ y : \mathrm{Int} \,\}$, then we have $\#C \leqslant \#D$. Moreover, the refined class type $C[1] \wedge \{\, y : \mathrm{Nat} \,\}$ *reduces to* the equivalent $\#C \wedge \{\, x : 0 \vee 1 \,\} \wedge \{\, x : 1 \vee 2,\ y : \mathrm{Int} \,\} \wedge \{\, y : \mathrm{Nat} \,\}$, which reduces further to $\#C \wedge \{\, x : 1,\ y : \mathrm{Nat} \,\}$.

Decomposing class types into more elementary types makes MLstruct's approach fundamentally structural, while retaining the right amount of nominality to precisely reflect the semantics of runtime class-instance matching (i.e., pattern matching based on the runtime class of objet values). It also means that there is no primitive notion of nominal type constructor *variance* in MLstruct: the covariance and contravariance of type parameters simply arise from the way class and alias types desugar into basic structural components.

### *2.3 Limitations*

While MLstruct features very flexible and powerful type inference, this naturally comes with some limitations, necessary to ensure the decidability and tractability of the type system. We already mentioned in Section 2.2.2 that intersections cannot be used to type overloading. Here we explain several other significant limitations.

#### *2.3.1 Regular Structural Types*

We restrict the shapes of MLstruct data types to be *regular trees* to make the problem of deciding whether one subsumes another decidable: concretely, occurrences of a class or alias type transitively reachable through the body of that type must have the same shape as the type's head declaration. For instance, the following are disallowed:

```
class C[A]: {x: C[Int]}    class C[A]: C[{x: List[A]}]
class C[A]: {x: C[C[A]]}
```

We conjecture that allowing such definitions would give our types the expressive power of context-free grammars, for which language inclusion is undecidable, making subtyping undecidable.[13] To replace illegal non-regular class fields, one can use either top-level functions or *methods*. The latter solve this problem by having their types known in advance and not participating in structural subtype checking. Methods are implemented in MLstruct but not presented in this paper.

### 2.3.2 Simplified Treatment of Unions

MLstruct keeps the expressiveness of unions in check by identifying $\{ x : \tau_1 \} \vee \{ y : \tau_2 \}$ $(x \neq y)$ and $\{ x : \tau_1 \} \vee (\tau_2 \to \tau_3)$ with $\top$, as described in Section 2.2.2. To make unions of different fields useful, one needs to "tag" the different cases with class types, as in $C_1 \wedge \{ x : \tau_1 \} \vee C_2 \wedge \{ y : \tau_2 \}$, allowing us to separately handle these cases through instance matching '**case** $v$ **of** $C_1 \to \ldots\ v.x\ \ldots$, $C_2 \to \ldots\ v.y\ \ldots$', whereas this is not necessary in, e.g., TypeScript.

A direct consequence of this restriction is that in MLstruct, there is no difference between $\{ x : \mathrm{Int},\ y : \mathrm{Int} \} \vee \{ x : \mathrm{Str},\ y : \mathrm{Str} \}$ and $\{ x : \mathrm{Int} \vee \mathrm{Str},\ y : \mathrm{Int} \vee \mathrm{Str} \}$ (still assuming $x \neq y$). Indeed, remember that $\{ x : \tau_1,\ y : \tau_2 \}$ is syntax sugar for $\{ x : \tau_1 \} \wedge \{ y : \tau_2 \}$ and by distributivity of unions over intersections, we can take $\{ x : \mathrm{Int},\ y : \mathrm{Int} \} \vee \{ x : \mathrm{Str},\ y : \mathrm{Str} \}$ to

$$(\{ x : \mathrm{Int} \} \vee \{ x : \mathrm{Str} \}) \wedge (\{ x : \mathrm{Int} \} \vee \{ y : \mathrm{Str} \}) \wedge (\{ y : \mathrm{Int} \} \vee \{ x : \mathrm{Str} \}) \wedge (\{ y : \mathrm{Int} \} \vee \{ y : \mathrm{Str} \})$$

and since $\{ x : \tau_1 \} \vee \{ y : \tau_2 \}$ is identified with $\top$, as explained in Section 2.2.2, this reduces to

$$(\{ x : \mathrm{Int} \} \vee \{ x : \mathrm{Str} \}) \wedge (\{ y : \mathrm{Int} \} \vee \{ y : \mathrm{Str} \})$$

which reduces by field merging to $\{ x : \mathrm{Int} \vee \mathrm{Str} \} \wedge \{ y : \mathrm{Int} \vee \mathrm{Str} \}$, i.e., $\{ x : \mathrm{Int} \vee \mathrm{Str},\ y : \mathrm{Int} \vee \mathrm{Str} \}$.

Another consequence is that, e.g., $\mathrm{List}[\mathrm{Int}] \vee \mathrm{List}[\mathrm{Str}]$ is identified with $\mathrm{List}[\mathrm{Int} \vee \mathrm{Str}]$. Again, to distinguish between these two, one should prefer the use of class-tagged unions or, equivalently, proper sum types such as $\mathrm{Either}[\mathrm{List}[\mathrm{Int}], \mathrm{List}[\mathrm{Str}]]$, defined in terms of Left and Right classes.

### 2.3.3 Fewer Relationships

Unlike in semantic subtyping approaches, but like in most practical programming languages, we do not have $\{x : \bot\} \leqslant \bot$. This would in fact lead to unsoundness in MLstruct: consider $\pi = (\{ x : \mathrm{Some}[\mathrm{Int}],\ y : \tau_1 \} \vee \{ x : \mathrm{None},\ y : \tau_2 \}) \wedge \{ x : \mathrm{None} \}$; we would have $\pi \equiv \{ x : \bot,\ y : \tau_1 \} \vee \{ x : \mathrm{None},\ y : \tau_2 \} \equiv \{ x : \mathrm{None},\ y : \tau_2 \}$ by distributivity and *also* $\pi \equiv \{ x : \bot \vee \mathrm{None},\ y : \tau_1 \vee \tau_2 \}$ by using (2.3.2) before distributing, but $\tau_1 \not\equiv \tau_1 \vee \tau_2$ in general.

### 2.3.4 No intersection overloading

Unlike languages like TypeScript, we do not permit the use of intersection types to encode inclusive function overloading (Pierce, 1991). Thankfully, simpler forms of overloading compatible with MLstruct exist; we briefly discuss one in Section 8.

[13] TypeScript does allow such definitions, meaning its type checker would necessarily be either unsound or incomplete.

Core syntax

$$\begin{array}{rrcl}
\textit{Type} & \tau, \pi & ::= & T\ \overline{\tau^+}\ \overline{\tau^-}\ \overline{\tau^0} \mid \alpha \mid \top^{\diamond} \mid \tau \vee^{\diamond} \tau \mid \neg\tau \\
\textit{Mode} & \diamond, \circ & ::= & \cdot \mid \curvearrowright \\
\textit{Polymorphic type} & \sigma & ::= & \forall\, \Xi.\, \tau
\end{array}$$

Contexts

$$\begin{array}{rrcl}
\textit{Subtyping context} & \Sigma, \Delta & ::= & \epsilon \mid \Sigma \cdot (\tau \leqslant \tau) \mid \Sigma \cdot \triangleright(\tau \leqslant \tau) \\
\textit{Bounds context} & \Xi & ::= & \epsilon \mid \Xi \cdot (\alpha \leqslant \tau) \mid \Xi \cdot (\tau \leqslant \alpha)
\end{array}$$

Fig. 1. Syntax of types, terms, and contexts of $\mathfrak{S}(\mathcal{T}, \mathcal{R})$.

$$\begin{array}{rrcl}
\textit{Type constructors} & T\ \overline{\tau^+}\ \overline{\tau^-}\ \overline{\tau^0} & ::= & (\rightarrow)\ \tau^+\ \tau^- \mid \{x\}\ \tau^+ \mid \#C \\
\textit{(also written as)} & & ::= & \tau \rightarrow \tau \mid \{x : \tau\} \mid \#C
\end{array}$$

Fig. 2. Instantiated syntax specific to $\lambda^{\neg}$.

## 3 Formalization of Boolean-Algebraic Subtyping

In this section, we present $\mathfrak{S}(\mathcal{T}, \mathcal{R})$[14], a generic Boolean-algebraic subtyping system, and prove some of its important formal properties.

Along with presenting the generic theory of Boolean-Algebraic subtyping as realized in $\mathfrak{S}(\mathcal{T}, \mathcal{R})$, we also *instantiate* that theory to the constructors of MLstruct as a running example to aid intuition. We do this by taking the type constructors $\mathcal{T} = \{ (\rightarrow)\ \tau^+\ \tau^- \} \cup \bigcup_i \{ \{x_i\}\ \tau^+ \} \cup \bigcup_j \{ \#C_j \}$ of *functions* (of syntax $(\rightarrow)\ \tau_0^+\ \tau_1^-$, also written $\tau_0 \rightarrow \tau_1$), *records* (of syntax $\{x\}\ \tau^+$, also written $\{x : \tau\}$), and *class tags* (of syntax $\#C$, where $C$ is a class name), and $\mathcal{R} = \{$ S-ClsSub, S-ClsBot, S-RcdTop $\}$, the subtyping rules associated to single-class inheritance and record widening (presented later in Figure 4). This yields system $\mathfrak{S}_{\rightarrow\{x\}\#C}$, the subtyping system of $\lambda^{\neg}$, which is the core language of MLstruct[15], presented in Section 6.

### 3.1 Syntax

The syntax of $\mathfrak{S}(\mathcal{T}, \mathcal{R})$ is presented in Figure 1. We use the notation $\overline{E_i}^i$ to denote a repetition of $i = 0$ to $n$ occurrences of a syntax form $E$, and we use the shorthand $\overline{E}$ when $i$ is not needed for disambiguation.

The *mode* $\diamond$ or $\circ$ of a syntactic form is used to deduplicate sentences that refer to unions and intersections as well as top and bottom, which are respective duals and can therefore often be treated symmetrically. For instance, $\top^{\diamond}$ is to be understood as either $\top^{\cdot}$ when $\diamond = \cdot$, i.e., $\top$, or as $\top^{\curvearrowright}$ when $\diamond = \curvearrowright$, i.e., $\bot$. A similar idea was developed independently

[14] As difficult to read as it is, '$\mathfrak{S}$' is supposed to be a stylized '$S$', which stands for "subtyping".

[15] Although $\lambda^{\neg}$ was already presented in our previous work (Parreaux and Chau, 2022), its subtyping system $\mathfrak{S}_{\rightarrow\{x\}\#C}$ was not given a name at the time.

by d. S. Oliveira et al. (2020) to cut down on boilerplate and repetition in formalizing subtyping systems.

$\mathfrak{S}(\mathcal{T}, \mathcal{R})$ is parametrized by a set of type constructors $\mathcal{T}$ and a set of subtyping rules $\mathcal{R}$ in addition to the Boolean algebraic rules, as well as depth subtyping and merge rules for the type constructors. The parameter lists $\overline{\tau^+}$, $\overline{\tau^-}$, and $\overline{\tau^0}$ of $T$ are the covariant, contravariant, and invariant parameters of $T$ respectively. Naturally, we will impose some restrictions on the rules in $\mathcal{R}$ (in Section 3.7), so that they are well-behaved with respect to the subtyping system as a whole.

Figure 2 shows the instantiation of type constructors $\mathcal{T} = \{ (\to)\ \tau^+\ \tau^- \} \cup \bigcup_i \{ \{x_i\}\ \tau^+ \} \cup \bigcup_j \{ \#C_j \}$ needed to obtain the syntax of $\lambda^\neg$.

### 3.2 Subtyping and Bounds Contexts

*Subtyping* contexts $\Sigma$ record assumptions about subtyping relationships, with some of these assumptions potentially hidden behind a $\triangleright$ (explained in Section 3.3.1). On the other hand, *bounds* contexts $\Xi$ contain bounds on type variables that can be generalized as part of a polymorphic type.

### 3.3 Subtyping Rules

The subtyping rules of $\mathfrak{S}(\mathcal{T}, \mathcal{R})$ are presented in Figure 3. and the rules of $\lambda^\neg$ that we use in its instantiation via $\mathcal{R} = \{$ S-ClsSub, S-ClsBot, S-RcdTop $\}$ are presented in Figure 4. Note that the fully specialized subtyping rules of $\lambda^\neg$ are later presented on their own, for clarity, in Figure 16.

Remember that the mode syntax $\diamond$ is used to factor in dual formulations. For instance, $\tau \leqslant^{\diamond} \top^{\diamond}$ is to be understood as either $\tau \leqslant \top^{\cdot}$ when $\diamond = \cdot$, i.e., $\tau \leqslant \top$, or as $\tau \leqslant^{\supset} \top^{\supset}$ when $\diamond = \supset$, i.e., $\tau \geqslant \bot$, also written $\bot \leqslant \tau$. The purpose of rule S-Weaken is solely to make rules which need no context slightly more concise to state. In this paper, we usually treat applications of S-Weaken implicitly.

#### 3.3.1 Subtyping Recursive Types

A consequence of our syntactic account of subtyping is that we do *not* define types as some fixed point over a generative relation, as done in, e.g., (Pierce, 2002; Dolan, 2017). Instead, we have to account for the fact that we manipulate *finite* syntactic type trees, in which recursive types have to be manually unfolded to derive things about them. This is the purpose of the S-Exp rules, which substitute a possibly-recursive type with its body to expose one layer of its underlying definition. As remarked by Amadio and Cardelli (1993, §3.2), to subtype recursive types, it is not enough to simply allow unfolding them a certain number of times. Moreover, in our system, recursive types may *arise* from cyclic type variable constraints (which is important for type inference), and thus not be attached to any explicit recursive binders. Thus, we cannot simply follow Castagna (2012, §1.3.4) in admitting a $\mu$ rule, which would still be insufficient.

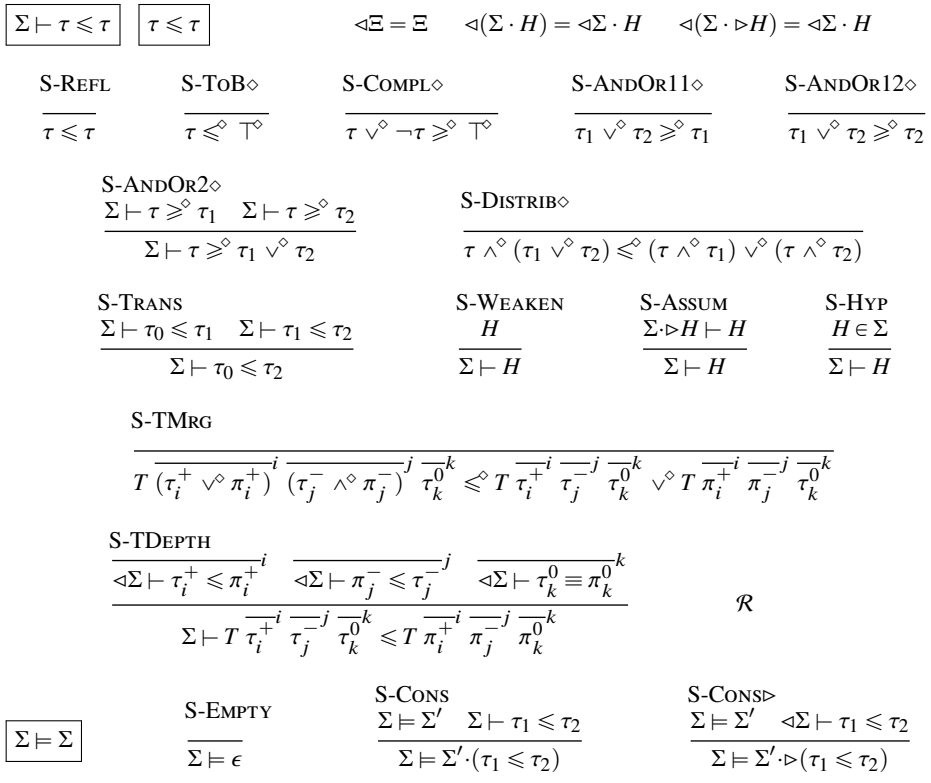


Fig. 3. Subtyping rules of $\mathfrak{S}(\mathcal{T}, \mathcal{R})$.

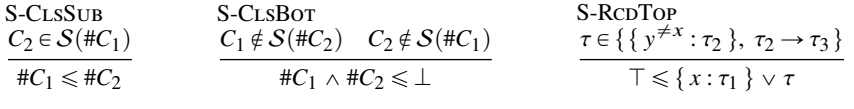


Fig. 4. Subtyping rules specific to $\lambda^{\neg}$.

### *3.3.2 Subtyping Hypotheses*

We make use of the $\Sigma$ environment to store subtyping hypotheses via S-Assum, to be leveraged later using the S-Hyp rule. We should be careful not to allow the use of a hypothesis right after assuming it, which would obviously make the system unsound (as it could derive any subtyping). In the specification of their constraint solving algorithm, Hosoya et al. (2005) use two distinct judgments $\vdash$ and $\vdash'$ to distinguish from places where the hypotheses can or cannot be used. We take a different, but related approach. Our S-Assum subtyping rule resembles the Löb rule described by Appel et al. (2007), which uses the "later" modality $\rhd$ in order to delay the applicability of hypotheses — by placing this symbol in front of the hypothesis being assumed, we prevent its immediate usage by

S-Hyp. We eliminate $\triangleright$ when passing through a function or record constructor: the dual $\triangleleft$ symbol is used to remove all $\triangleright$ from the set of hypotheses, making them available for use by S-Hyp. These precautions reflect the "guardedness" restrictions used by Dolan (2017) on recursive types, which prevents usages of $\alpha$ that are not guarded by $\to$ or $\{ \ldots \}$ in a recursive type $\mu\alpha.\,\tau$. Such productivity restriction is also implemented by our guardedness check, preventing the definition of types such as **type** $A = A$ and **type** $A = \neg A$ (Section 2.1.6).[16]

### 3.3.3 Example

As an example, let us try to derive $A_1 \leqslant A_2$ where $A_1 = \tau \to \tau \to A_1$ and $A_2 = \tau \to A_2$, which states that the type of a function taking two curried $\tau$ arguments an arbitrary number of times is a *special case* of the type of a function taking a single $\tau$ argument an arbitrary number of times. To facilitate the development, we use the shorthand $H = A_1 \leqslant A_2$. We start by deriving that the respective unfoldings of the recursive types are subtypes; that is, that $(1)\ \tau \to \tau \to A_1 \leqslant \tau \to A_2$. Note that for conciseness, we omit the applications of S-Weaken in the derivations below:

$$\textsc{Fun}\ \frac{\dfrac{\textsc{Refl}\ \dfrac{}{H \vdash \tau \leqslant \tau} \qquad \textsc{Fun}\ \dfrac{\textsc{Refl}\ \dfrac{}{H \vdash \tau \leqslant \tau} \qquad \textsc{Hyp}\ \dfrac{(A_1 \leqslant A_2) \in H}{H \vdash A_1 \leqslant A_2}}{H \vdash \tau \to A_1 \leqslant \tau \to A_2} \qquad \textsc{Exp}\ \dfrac{}{H \vdash \tau \to A_2 \leqslant A_2}}{H \vdash \tau \to A_1 \leqslant A_2}\ \textsc{Trans}}{\triangleright H \vdash \tau \to \tau \to A_1 \leqslant \tau \to A_2 \quad (1)}$$

Then, we simply have to fold back the unfolded recursive types, using Exp and Trans:

$$\textsc{Assum}\ \frac{\dfrac{\textsc{Trans}\ \dfrac{\textsc{Exp}\ \dfrac{}{\triangleright H \vdash A_1 \leqslant \tau \to \tau \to A_1} \quad (1)}{\triangleright H \vdash A_1 \leqslant \tau \to A_2} \qquad \textsc{Exp}\ \dfrac{}{\triangleright H \vdash \tau \to A_2 \leqslant A_2}}{\triangleright H \vdash A_1 \leqslant A_2}\ \textsc{Trans}}{A_1 \leqslant A_2}$$

### 3.3.4 A Boolean Algebra

The subtyping preorder in $\mathfrak{S}(\mathcal{T}, \mathcal{R})$ gives rise to a Boolean *lattice* or *algebra* when taking the equivalence relation '$\tau_1 \equiv \tau_2$' to be the relation induced by '$\tau_1 \leqslant \tau_2$ *and* $\tau_2 \leqslant \tau_1$'. To see why, let us inspect the standard way of defining Boolean algebras, which is as the set of *complemented distributive lattices*. We can define a lattice equivalently as either:

- An algebra $\langle L,\ \wedge,\ \vee \rangle$ such that $\wedge$ and $\vee$ are idempotent, commutative, associative, and satisfy the absorption law, i.e., $\tau \wedge (\tau \vee \pi) \equiv \tau \vee (\tau \wedge \pi) \equiv \tau$. Then $\tau_1 \leqslant \tau_2$ is taken to mean $\tau_1 \equiv \tau_1 \wedge \tau_2$ or (equivalently) $\tau_1 \vee \tau_2 \equiv \tau_2$.
- A partially-ordered set $\langle L, \leqslant \rangle$ (i.e., $\leqslant$ is reflexive, transitive, and antisymmetric) where every two elements $\tau_1$ and $\tau_2$ have a least upper bound $\tau_1 \vee \tau_2$ (supremum) and a greatest lower bound $\tau_1 \wedge \tau_2$ (infimum). That is, $\forall\, \pi \leqslant \tau_1, \tau_2.\ \pi \leqslant \tau_1 \wedge \tau_2$ and $\forall\, \pi \geqslant \tau_1, \tau_2.\ \pi \geqslant \tau_1 \vee \tau_2$.

[16] Perhaps counter-intuitively, it is *not* a problem to infer types like '$\forall(\alpha \leqslant \alpha).\ \tau$' and '$\forall(\alpha \leqslant \neg\alpha).\ \tau$' because such "funny" cyclic bounds, unlike unproductive recursive types, do not actually allow concluding incorrect subtyping relationships.

The latter is most straightforward to show: we have reflexivity by S-Refl, transitivity by S-Trans, antisymmetry by definition of $\equiv$, and the supremum and infimum properties are given directly by S-AndOr2$\cdot$ and S-AndOr2$\supset$ respectively.

Moreover, to be a Boolean algebra, our lattice needs to be:

- a *complemented* lattice, which is
  - bounded: $\top$ and $\bot$ are respective *least* and *greatest* elements (S-ToB$\diamond$);
  - such that every $\tau$ has a complement $\neg\tau$ where $\tau \vee \neg\tau \equiv \top$ and $\tau \wedge \neg\tau \equiv \bot$ (S-Compl$\diamond$);[17]
- a *distributive* lattice, meaning that $\tau \wedge^\diamond (\tau_1 \vee^\diamond \tau_2) \equiv (\tau \wedge^\diamond \tau_1) \vee^\diamond (\tau \wedge^\diamond \tau_2)$ for $\diamond \in \{\supset, \cdot\}$.
  The first direction $\leqslant^\diamond$ of distributivity is given directly by S-Distrib. The other direction $\geqslant^\diamond$ is admissible: since $\tau_1 \vee^\diamond \tau_2 \geqslant^\diamond \tau_1$ (S-AndOr11$\diamond$) and $\tau_1 \vee^\diamond \tau_2 \geqslant^\diamond \tau_2$ (S-AndOr12$\diamond$), we can easily derive $\tau \wedge^\diamond (\tau_1 \vee^\diamond \tau_2) \geqslant^\diamond \tau \wedge^\diamond \tau_1$ and $\tau \wedge^\diamond (\tau_1 \vee^\diamond \tau_2) \geqslant^\diamond \tau \wedge^\diamond \tau_2$, and by (S-AndOr2$\diamond$) we conclude that $\tau \wedge^\diamond (\tau_1 \vee^\diamond \tau_2) \geqslant^\diamond (\tau \wedge^\diamond \tau_1) \vee^\diamond (\tau \wedge^\diamond \tau_2)$.

A useful property of Boolean algebras is that the usual De Morgan's laws hold, which will allow us to massage constrains into normal forms during type inference.

### 3.3.5 Purely Algebraic Rules

We call S-FunMrg and S-RcdTop *purely algebraic* subtyping rules because they do not follow from a set-theoretic interpretation of order connectives ($\wedge$, $\vee$, $\neg$). S-FunMrg and S-RcdMrg respectively make function and record types lattice homomorphisms,[18] which is required to make type inference complete — this allows the existence of well-behaved normal forms. Though one can still think of types as sets of values, as in the *semantic subtyping* approach, in $\lambda^\neg$ the sets of values of $\tau_1 \wedge \tau_2$ is *not* the intersection of the sets of values of $\tau_1$ and $\tau_2$ (unless $\tau_1$ and $\tau_2$ are nominal tags or records), and similarly for unions and complements. These purely algebraic rules are sound in $\lambda^\neg$ because of the careful use we make of unions and intersections, e.g., not using intersections to encode overloading. Notably, S-RcdTop implies surprising relationships like $\neg(\tau_1 \to \tau_2) \leqslant \{\, x : \pi \,\}$ and $\neg\{\, x : \pi \,\} \leqslant \{\, y : \pi \,\}$ $(x \neq y)$, exemplifying that negation in $\lambda^\neg$ is essentially *algebraic*.

## 3.4 Some Useful Subtyping Relationships

Next, we demonstrate a few useful subtyping rules that can be derived in our system as well as in any Boolean algebra of types (i.e., a Boolean algebra where ordering is interpreted as subtyping). These are all proven in Appendix A.3.

Figure 5 lists some of these rules that can be expressed as simple inference rules.

**Lemma 3.1.** *For all* $\Sigma$*, we have* $\Sigma \vdash \tau_1 \vee^\diamond \tau_2 \leqslant^\diamond \tau_3 \iff \Sigma \vdash \tau_1 \leqslant^\diamond \tau_3 \ \wedge\ \Sigma \vdash \tau_2 \leqslant^\diamond \tau_3$.

[17] We can also show that our lattice is *uniquely* complemented, i.e., $\neg\tau_1 \equiv \neg\tau_2$ implies $\tau_1 \equiv \tau_2$ (Theorem 3.2).

[18] A lattice homomorphism $f$ is such that $f(\tau \vee \pi) \equiv f(\tau) \vee f(\pi)$ and $f(\tau \wedge \pi) \equiv f(\tau) \wedge f(\pi)$. Function types are lattice homomorphisms in their parameters in the sense that $f(\tau) = (\neg\tau) \to \pi$ is a lattice homomorphism.

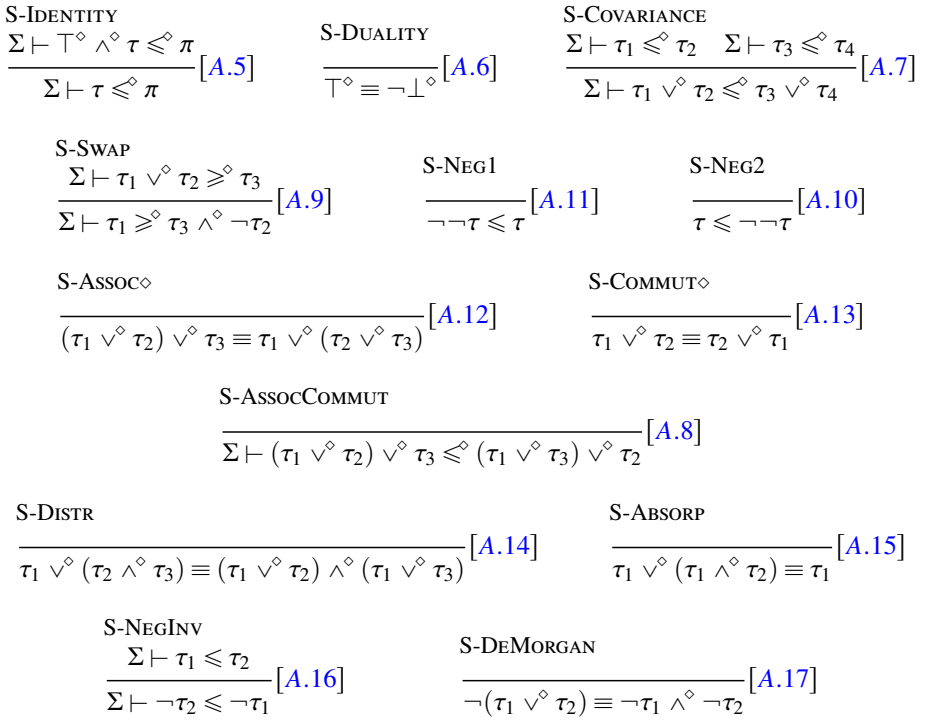


Fig. 5. Some useful subtyping relationships that hold in $\mathfrak{S}(\mathcal{T}, \mathcal{R})$ as well as in any other Boolean algebra of types.

**Theorem 3.2** (Unique Complementation)**.** *For all* $\tau_1$ *and* $\tau_2$, $\neg\tau_1 \equiv \neg\tau_2$ *implies* $\tau_1 \equiv \tau_2$*, i.e., "*$\neg\tau_1 \leqslant \neg\tau_2$ *and* $\neg\tau_2 \leqslant \neg\tau_1$*" imply "*$\tau_1 \leqslant \tau_2$ *and* $\tau_2 \leqslant \tau_1$*".*

### *3.5 Type Variables & Polymorphism*

In line with ML-style type inference, which is based on *prenex polymorphism*, we seek to assign *type schemes* to the term definitions of a program, where a type scheme is a normal type that references type variables that are quantified at its outermost level.

We could write such type schemes $\forall\overline{\alpha}\{\Xi\}.\,\tau$, as we do in our work on first-class polymorphism (Parreaux et al., 2024), where $\overline{\alpha}$ are the type variables being quantified and $\Xi$ is their bounds. However, since in this work we focus on polymorphism only for top-level definitions (we do not support nested let polymorphism, although the system could be extended to support it), we instead use the more compact notation $\forall\Xi.\,\tau$, whereby all variables mentioned in $\Xi$ are implicitly quantified.

Polymorphism type schemes are implicitly and eagerly instantiated whenever the corresponding definition is used, so that type inference and constraint solving only ever have to deal with *monomorphic* types: the polymorphism is only at the top level and not part of the core subtyping system.

### *3.6 Consistency of Bounds Contexts*

A crucial aspect of polymorphic type inference with bounds is that we must ensure that these bounds are *consistent*, in the sense that they are "meaningful" and do not lead to contradictions in the type system.

For example, we must prevent typing definitions with such bounds as $\forall(\mathrm{Bool} \leqslant \alpha \leqslant \mathrm{Int}).\, \tau$ — which is a shorthand for $\forall(\mathrm{Bool} \leqslant \alpha) \cdot (\alpha \leqslant \mathrm{Int}).\, \tau$. Indeed, in the body of the corresponding definition, this would allow one to implicitly *upcast* any value of type $\mathrm{Int}$ into a value of type $\mathrm{Bool}$, due to the assumptions on the bounds of $\alpha$ and the transitivity of subtyping implying that $\mathrm{Bool} \leqslant \mathrm{Int}$.

So we need to make sure that $\Xi$ contexts are consistent, which we write $\Xi$ ***cons.***. But there are several possible ways we could define such consistency criterion so that it preserves the soundness of the type system.

#### *3.6.1 Classical Consistency*

Using the most obvious approach, consistency could be defined in the classical way:

$$\frac{\models \rho(\Xi)}{\Xi \textbf{ \textit{cons.}}}$$

That is, a bounds context is considered consistent if there exists a substitution $\rho$ that makes all the constraints hold in the empty context, written $\epsilon \models \rho(\Xi)$ or just $\models \rho(\Xi)$.

While this definition is quite simple and intuitive, it describes a rather *strong* consistency criterion. To see that, consider the bounds context $\alpha \leqslant \top \to \alpha$, which is cyclic and essentially describes a *recursive type*. Such type schemes are important to support since they are required for complete & principal type inference, so we cannot simply reject them. But to show that this bounds context is classically consistent requires the existence of some form of first-class recursive or infinite types as primitives of the underlying type system. Here, the substitution $\rho$ would have to map $\alpha$ to a type that is a function type from $\top$ to itself. We usually write these types using a $\mu$ binder, as in $\mu X.\, \top \to X$.

While the requirement that $\mu$ types (or equivalent) should be available as one of the core type constructors of the system is not a fundamental problem, it has two major disadvantages:

- It can *complicate* the formal developments, requiring the handling of all possible uses of $\mu$ types in the metatheory.
- It is *unsatisfying* from a practical and theoretical point of view, in that the real type system of the programming language under study may already have its own notion of recursive types (in MLstruct, these are user-defined type aliases and class types) and $\mu$ types would play double duty with them, flaunting the principle of *economy of concepts*. We would really rather like for a generic theory of Boolean-algebraic subtyping like $\mathfrak{S}(\mathcal{T}, \mathcal{R})$ to not make strong assumptions on the constructors of the underlying language beyond the existence of the base Boolean-algebraic connectives.

While this strong consistency definition is sufficient to achieve soundness (making impossible to, say, upcast integers to Booleans), it is in fact not *necessary*.

### 3.6.2 Weak Consistency

As it turns out, it is possible to design an alternative, *weaker* definition of consistency that does *not* assume the existence of recursive structures in the base type forms and instead relies purely on type variables and the $\triangleright$ modality:

$$\frac{\triangleright\Xi \vDash \rho(\Xi)}{\Xi \ \textbf{\textit{cons.}}}$$

This version of consistency is correct and sufficient for all intents and purposes, but we will instead use the *slightly* stronger one below just because we can:

$$\frac{\overline{\triangleright(\alpha \equiv \tau)}^{(\alpha \mapsto \tau) \in \rho} \vDash \rho\Xi}{\Xi \ \textbf{\textit{cons.}}}$$

This definition says that $\Xi$ is consistent if there is a substitution $\rho$ that makes all the constraints hold in a *guarded* context where each substituted type variable is equated with its substitution.

Note that for simplicity of the definition, this assumes substitutions may substitute a type variable $\alpha$ with a type that still contains occurrences of $\alpha$. So these are not "proper" substitutions in the usual sense, where a proper substitution is supposed to be *idempotent*. We could call our pseudo-substitutions *partial substitutions*, but by abuse of terminology we will usually just call them *substitutions*.

Using this definition, we can show that our running example $\Xi = (\alpha \leqslant \top \to \alpha)$ is consistent by the partial substitution $[\alpha \mapsto (\top \to \alpha)]$, as demonstrated in the derivation below:

$$\dfrac{\dfrac{\textsc{Refl}\ \dfrac{}{(\alpha \equiv \top \to \alpha) \vdash \top \leqslant \top} \qquad \dfrac{}{(\alpha \equiv \top \to \alpha) \vdash \alpha \leqslant \top \to \alpha}\ \textsc{Hyp}}{\textsc{Fun}\ \dfrac{}{\triangleright(\alpha \equiv \top \to \alpha) \vdash \top \to \alpha \leqslant \top \to \top \to \alpha}}}{(\alpha \leqslant \top \to \alpha)\ \textbf{\textit{cons.}}}$$

This is sufficient to derive the soundness of the declarative type system of MLstruct, but not quite enough to show the correctness of its type inference algorithm. For that, we need to enrich the consistency definition in two steps, first to *parameterized weak consistency* and then to *algorithmic consistency*.

### 3.6.3 Parameterized Weak Consistency

To allow reasoning about consistency in the context of principal type inference, we need to first generalize the definition to allow for subtyping context assumptions:

$$\frac{\overline{\triangleright(\alpha \equiv \tau)}^{(\alpha \mapsto \tau) \in \rho} \cdot \rho\Sigma \vDash \rho\Xi}{\Sigma \vdash \Xi \ \textbf{\textit{cons.}}}$$

This is the definition of weak consistency we will retain, as it is more general.

A central property of weak consistency is that it implies the ability to *inline type variable bounds*, when they are consistent, into an existing subtyping derivation.

**Lemma 3.3** (Inlining of consistent bounds)**.** *If* $\Sigma \vdash \Xi$ ***cons.*** *and* $\Sigma{\cdot}\Xi \vdash \tau \leqslant \tau'$*, then* $\rho\Sigma{\cdot}\overline{\triangleright(\alpha \equiv \pi)}^{(\alpha \mapsto \pi) \in \rho} \vdash \rho\tau \leqslant \rho\tau'$ *for some* $\rho$*.*

**Lemma 3.4** (Inlining of consistent bounds on guarded derivations)**.** *If* $\Sigma \vdash \Xi$ ***cons.*** *and* $\Sigma{\cdot}\Xi \vdash \tau \leqslant \tau'$ *and* $TTV(\tau) \cup TTV(\tau') = \varnothing$*, then* $\rho\Sigma{\cdot}\overline{\triangleright(\alpha \equiv \pi)}^{(\alpha \mapsto \pi) \in \rho} \vdash \tau \leqslant \tau'$ *for some* $\rho$*.*

### *3.6.4 Algorithmic Consistency*

When proving facts about the type inference algorithm, we will need to rely on a *very specific* way of achieving consistency of the bounds contexts involved. Indeed, we will need for this consistency criterion to *precisely mirror* the way the constraint solving part of the algorithm ensures consistency.

This idea gives rise to the following definition, which now precisely specifies the way type variables should be substituted one by one to ensure consistency. We chose the substitution $[\alpha \mapsto \alpha \wedge ub_\Xi(\alpha) \vee lb_\Xi(\alpha)]$ because the substituted bounds context would be derivable if and only if the lower bounds are subtypes of the upper bounds. Upon closer inspection, the definition of algorithmic consistency shares a lot of similarities with that of parametrized weak consistency: the substituted bounds context should be entailed by the substituted subtyping context assumptions together with delayed assumption that the type variables are equivalent to their respective images. The only difference is that in the definition of algorithmic consistency, the even stronger delayed assumption of the bounds themselves (which implies that the type variables are equivalent to their images under the specified substitution) is used, and the type variables are substituted one by one.

The attractiveness of this definition is that we will be able to perform *inversion* on it, allowing us to modify the substitution of a specific type variable without worrying about invalidating the evidence for the other type variables, in the inductive proofs of constraint solving soundness and completeness.

$\boxed{\Sigma \vdash \triangleright\Xi{\cdot}\Xi\,;\ \rho\ \textbf{\textit{cons.}}}$ *Assuming* $\Sigma$ *holds, then bounds* $\triangleright\Xi{\cdot}\Xi$ *are consistent, as witnessed by* $\rho$*.*

$$\Xi\ \textbf{\textit{cons.}} \equiv \exists\rho.\ \epsilon \vdash \Xi\,;\ \rho\ \textbf{\textit{cons.}}$$

$$\frac{}{\Sigma \vdash \triangleright\Xi\,;\ [\,]\ \textbf{\textit{cons.}}} \qquad \frac{\begin{array}{c} split_\alpha(\Xi,\ dom(\rho')) = (\Xi_\alpha,\ \Xi_{\not\alpha}) \qquad \rho = [\alpha \mapsto \alpha \wedge ub_\Xi(\alpha) \vee lb_\Xi(\alpha)] \\ \triangleright\Xi_\triangleright{\cdot}\triangleright\Xi_\alpha{\cdot}\rho\Xi_{\not\alpha}{\cdot}\rho\Sigma \vDash \rho\Xi_\alpha \qquad \rho\Sigma \vdash \triangleright\Xi_\triangleright{\cdot}\triangleright\Xi_\alpha{\cdot}\rho\Xi_{\not\alpha}\,;\ \rho'\ \textbf{\textit{cons.}} \end{array}}{\Sigma \vdash \triangleright\Xi_\triangleright{\cdot}\Xi\,;\ \rho' \circ \rho\ \textbf{\textit{cons.}}}$$

$$split_\alpha(\Xi,\ \{\,\overline{\gamma}\,\}) =$$

$$(\overline{(\tau \leqslant \pi)}^{(\tau \leqslant \pi) \in \Xi \,|\, \alpha \in \{\,\tau,\,\pi\,\}},\ \overline{(\tau \leqslant \pi)}^{(\tau \leqslant \pi) \in \Xi \,|\, \alpha \notin \{\,\tau,\,\pi\,\}} \cdot \overline{(\alpha \leqslant^\diamond \beta)}^{(\alpha \leqslant^\diamond \beta) \in \Xi \,|\, \beta \in \{\,\overline{\gamma}\,\}})$$

Where *lb* and *ub* are defined in Definition 3.5 below.

**Definition 3.5** (Upper and lower bounds)**.** *We use the following definitions of lower and upper bounds $lb_\Xi(\alpha)$ and $ub_\Xi(\alpha)$ of a type variable $\alpha$ inside a constraining context $\Xi$:*

$$\boxed{lb_\Xi(\alpha)} : \tau$$

$$lb_{\Xi\cdot(\tau\leqslant\alpha)}(\alpha) = \tau \vee lb_\Xi(\alpha)$$
$$lb_{\Xi\cdot(\tau\leqslant\beta)}(\alpha) = lb_\Xi(\alpha) \qquad (\alpha \neq \beta)$$
$$lb_{\Xi\cdot(\beta\leqslant\tau)}(\alpha) = lb_\Xi(\alpha)$$
$$lb_\epsilon(\alpha) = \bot$$

$$\boxed{ub_\Xi(\alpha)} : \tau$$

$$ub_{\Xi\cdot(\tau\leqslant\beta)}(\alpha) = ub_\Xi(\alpha)$$
$$ub_{\Xi\cdot(\alpha\leqslant\tau)}(\alpha) = \tau \wedge ub_\Xi(\alpha)$$
$$ub_{\Xi\cdot(\beta\leqslant\tau)}(\alpha) = ub_\Xi(\alpha) \qquad (\alpha \neq \beta)$$
$$ub_\epsilon(\alpha) = \top$$

Algorithmic consistency shows a way of achieving consistency by picking each type variable one by one and substituting it with a type that is equivalent to the original variable but with its bounds inlined. One can understand this definition as getting rid of all the unguarded type variable bounds, ending with an equivalent subtyping context where these bounds are integrated into the type variable occurrences themselves. In a sense, this is reminiscent of the *bisubstitution* process of Dolan (2017), except that we do not care about polarity and always integrate *both* upper and lower bounds with each occurrence.

Naturally, we can show that algorithmic consistency implies weak consistency:

**Lemma 3.6** (Algorithmic consistency implies weak consistency)**.** *If $\Sigma \vdash \Xi$ ; $\rho$ **cons.**, then $\Sigma \vdash \Xi$ **cons.**.*

### *3.7 Requirements on Base Subtyping Rules*

As explained before, we place some requirements on the base subtyping rules $\mathcal{R}$ of the underlying type system so that these rules do not threaten the proof structure of the general subtyping system $\mathfrak{S}(\mathcal{T}, \mathcal{R})$.

For each rule in $\mathcal{R}$ with conclusion $\Sigma \vdash \tau \leqslant \pi$, we require each of its premises $\Sigma' \vdash \tau' \leqslant \pi'$ to adhere to the following restrictions:

- $\lhd\Sigma$ ***cons.*** implies $\lhd\Sigma'$ ***cons.***
- $\max(depth(\tau'), depth(\pi')) \leqslant \max(depth(\tau), depth(\pi))$
- If $\max(depth(\tau'), depth(\pi')) = \max(depth(\tau), depth(\pi))$, then $\Sigma$ ***cons.*** implies $\Sigma'$ ***cons.***

The first restriction ensures that the base subtyping rules do not introduce inconsistencies in their premises. The two other restrictions ensure that our proofs by inductions can go through without running into well-foundedness issues.

We are confident that the reader can convince themselves that these rules are eminently reasonable and should be easily satisfied by any practical underlying type system.

### *3.8 Subtyping Derivation Shapes*

We now give a few definitions characterizing the shapes of subtyping derivations, and prove properties about them.

**Definition 3.7** (Right-leaning derivations)**.** *A subtyping derivation is said to be* right-leaning *if all its applications of rule S-Trans have a first premise which is not itself an application of rule S-Trans.*

It is easy to see that any subtyping derivation can be rewritten into an equivalent right-leaning derivation of the same size by reorganizing its uses of S-Trans.

**Definition 3.8** (Bottom-level rules)**.** *A rule is used* at the bottom level *in a derivation if it is one of the following:*

1. *the last rule used in the derivation;*
2. *either premise of a bottom-level application of rule S-Trans;*
3. *the premise of a bottom-level application of rule S-Exp∘;*
4. *the first premise of a bottom-level application of rule T-Subs.*

**Definition 3.9** (Unassuming derivation)**.** *An unassuming derivation is a subtyping derivation that does not make use of S-Assum at the bottom level.*

**Lemma 3.10** (Unassuming derivation)**.** *Any subtyping derivation can be rewritten to an equivalent unassuming derivation.*

**Definition 3.11** (Subsumption-normalized derivation)**.** *A subsumption-normalized derivation is a typing derivation that makes at most one use of T-Subs at the bottom level.*

**Lemma 3.12** (Subsumption-normalized derivation)**.** *Any typing derivation can be rewritten to an equivalent subsumption-normalized derivation.*

## 4 Soundness of Boolean-Algebraic Subtyping

The reason we can soundly incorporate rules such as S-FunMrg, S-RcdMrg, and S-RcdTop is that they do not threaten any of the properties we actually need for the type soundness proofs. As a first step towards showing that, and in order to support the next important lemmas, we want to prove that *subtyping is sound* in $\mathfrak{S}_{\to\{x\}\#C}$.

In this section, we demonstrate the soundness of Boolean-Algebraic subtyping $\mathfrak{S}(\mathcal{T}, \mathcal{R})$ by proving that, assuming a few rather conservative assumptions on the language's parameterized type constructors and their subtyping relationships (which are, naturally, upheld by the $\mathfrak{S}_{\to\{x\}\#C}$ instance), then Boolean-Algebraic subtyping will not relate unrelated constructors. By extension, this demonstrates that subtyping in $\lambda^{\neg}$ is sound, which is a key ingredient in showing that MLstruct can be used as a type-safe programming language whose well-typed programs "*do not go wrong*".

### 4.1 High-Level Goal

Essentially, we want to prove the following property, which we have here instantiated to $\mathfrak{S}_{\to\{x\}\#C}$ type constructors for the sake of intuition:

**Theorem 4.1** (Soundness of $\lambda^{\sqcap}$ subtyping $\mathfrak{S}_{\to\{x\}\#C}$)**.** *If* $\Xi$ ***cons.*** *and* $\Xi \vdash \tau \leqslant \pi$*, where:*

$$\tau \in \{\ \bot,\ \top,\ \#C,\ \tau_1 \to \tau_2,\ \{\ \overline{x_i : \tau_i}^{\,i}\ \}\ \}$$
$$\pi \in \{\ \bot,\ \top,\ \#C',\ \pi_1 \to \pi_2,\ \{\ x' : \pi_1\ \}\ \}$$

*then exactly one of the following is true:*

*(a)* $\tau = \bot$ *or* $\pi = \top$*;*
*(b)* $\tau = \#C$ *and* $\pi = \#C'$ *and* $C' \in \mathcal{S}(\#C)$*;*
*(c)* $\tau = \tau_1 \to \tau_2$ *and* $\pi = \pi_1 \to \pi_2$ *and* $\Xi \vdash \pi_1 \leqslant \tau_1$ *and* $\Xi \vdash \tau_2 \leqslant \pi_2$*;*
*(d)* $\tau = \{\ \overline{x_i : \tau_i}^{\,i}\ \}$ *and* $\pi = \{\ x_k : \pi_1\ \}$ *and* $\Xi \vdash \tau_k \leqslant \pi_1$ *for some* $k$*.*

This property can be read as follows: if the right-hand side of a subtyping relation is a function type and the left-hand side is a specific constructed type, then that constructed type must be either bottom or a function type with compatible argument and return types, and similarly for the other base type constructors. This describes how the basic type constructors of the language should or should not relate by subtyping, and in particular prevents wrong relations, such as function types subtyping record types.

The structure of the soundness proof is quite complex and requires additional syntax forms and inductive relations. This is because the Boolean-algebraic rules are so general and flexible that we must find a way of somehow giving them more "structure" by restricting the way they may be used to a form amenable for inductive reasoning.

Indeed, proving the theorem stated above cannot proceed by the standard technique of induction on subtyping derivations. Due to the restricted shape of the type forms involved on both sides of $\leqslant$, the inductive hypothesis cannot be applied to the premises of S-Trans, as the middle type introduced may not adhere to that restricted shape.

### 4.2 Splitting up Boolean-Algebraic Subtyping

A quick inspection reveals that the problem lies within S-AndOr2. While some usages of S-AndOr2 can be removed by rewritting the derivation, not all usages can be removed.

The solution we adopt is to split the full $\leqslant$ subtyping relation into two, with $\subseteq$ covering the pure Boolean-algebraic relation and $\preceq$ covering the remaining relation between the atoms and coatoms of the system, referred to as *elementary type forms*. In a sense, $\preceq$ defines what base type constructors are considered related or unrelated at the level of the underlying language (for instance, functions and records are unrelated, but derived classes are related to their base classes), whereas $\subseteq$ is only concerned with deciding what is related in terms of pure Boolean-algebraic structure. Crucially, $\preceq$ can be defined in a way that does not require a rule for *transitivity*, greatly simplifying the corresponding proofs.

This allows us to refine the statement of the inductive lemma by stating required properties on these two aspects of subtyping separately (see Lemma 4.22).

### 4.2.1 Pure Boolean-Algebraic Subtyping

First, we define $\subseteq$ as the standard Boolean lattice order.

**Definition 4.2** (Pure Boolean-Algebraic Subtyping)**.** *We define $\tau_1 \subseteq \tau_2$ to mean that $\tau_1 \leqslant \tau_2$ can be derived by using only "Boolean Lattice" subtyping rules, which are those that that are not specific to $\mathcal{T}$ types and simply encode their Boolean-Algebraic structure. More specifically, these rules are: S-Refl, S-ToB, S-Compl, S-AndOr11, S-AndOr12, S-AndOr2, S-Distrib, and S-Trans.*

**Theorem 4.3** (Standard Boolean Lattice Order)**.** *$\subseteq$ holds in every Boolean lattice, i.e., it does not introduce any extra relations between its atoms, which are the base types $\mathcal{T}$ of $\mathfrak{S}(\mathcal{T}, \mathcal{R})$.*

Since $\subseteq$ is itself a Boolean Algebra (see Section 3.3.4), this means our rules for $\subseteq$ are a proper axiomatization of Boolean Algebras.

**Proof** We show that the $\subseteq$ rules follow from the pure Boolean algebra axioms. In this proof, $\equiv$ denotes the pure Boolean algebra equivalence, defined by the following axioms (Huntington, 1904):

$$\begin{aligned}
\text{B-Iden}\diamond: &\quad \tau \wedge^\diamond \top^\diamond \equiv \tau \\
\text{B-Commut}\diamond: &\quad \tau_1 \vee^\diamond \tau_2 \equiv \tau_2 \vee^\diamond \tau_1 \\
\text{B-Distrib}\diamond: &\quad \tau \wedge^\diamond (\tau_1 \vee^\diamond \tau_2) \equiv (\tau \wedge^\diamond \tau_1) \vee^\diamond (\tau \wedge^\diamond \tau_2) \\
\text{B-Compl}\diamond: &\quad \tau \vee^\diamond \neg\tau \equiv \top^\diamond
\end{aligned}$$

The following laws follow from the axioms (Huntington, 1904):

$$\begin{aligned}
\text{B-Idem}\diamond: &\quad \tau \vee^\diamond \tau \equiv \tau \\
\text{B-Bound}\diamond: &\quad \tau \vee^\diamond \top^\diamond \equiv \top^\diamond \\
\text{B-Absorp}\diamond: &\quad \tau_1 \wedge^\diamond (\tau_1 \vee^\diamond \tau_2) \equiv \tau_1 \\
\text{B-DeMorgan}\diamond: &\quad \neg(\tau_1 \vee^\diamond \tau_2) \equiv (\neg\tau_1 \wedge^\diamond \neg\tau_2) \\
\text{B-Assoc}\diamond: &\quad (\tau_1 \vee^\diamond \tau_2) \vee^\diamond \tau_3 \equiv \tau_1 \vee^\diamond (\tau_2 \vee^\diamond \tau_3)
\end{aligned}$$

In the context of Boolean algebras thus axiomatized, $\tau_1 \subseteq \tau_2$ is understood to mean $\tau_1 \equiv \tau_1 \wedge \tau_2$ (Section 3.3.4). So all we have to show is that all the conclusions of the form $\tau_1 \subseteq \tau_2$ given by our subtyping rules are so that $\tau_1 \equiv \tau_1 \wedge \tau_2$ holds by the axioms above.

**S-Refl.**

$$\tau \equiv \tau \wedge \tau \qquad \text{by B-Idem}\diamond$$

**S-ToB$\cdot$.**

$$\tau \equiv \tau \wedge \top \qquad \text{by B-Iden}\cdot$$

**S-ToB⤾.**

$$\begin{aligned} \bot &\equiv \tau \wedge \bot && \text{by B-Bound⤾} \\ &\equiv \bot \wedge \tau && \text{by B-Commut⤾} \end{aligned}$$

**S-Compl·.**

$$\begin{aligned} \top &\equiv \tau \vee \neg\tau && \text{by B-Compl·} \\ &\equiv (\tau \vee \neg\tau) \wedge \top && \text{by B-Iden·} \\ &\equiv \top \wedge (\tau \vee \neg\tau) && \text{by B-Commut⤾} \end{aligned}$$

**S-Compl⤾.**

$$\begin{aligned} \tau \wedge \neg\tau &\equiv \bot && \text{by B-Compl⤾} \\ &\equiv (\tau \wedge \neg\tau) \wedge \bot && \text{by B-Bound⤾} \end{aligned}$$

**S-AndOr11·.**

$$\begin{aligned} \tau_1 &\equiv \tau_1 \wedge (\tau_1 \vee \tau_2) && \text{by B-Absorp·} \end{aligned}$$

**S-AndOr11⤾.**

$$\begin{aligned} \tau_1 \wedge \tau_2 &\equiv (\tau_1 \wedge \tau_1) \wedge \tau_2 && \text{by B-Idem⤾} \\ &\equiv \tau_1 \wedge (\tau_1 \wedge \tau_2) && \text{by B-Assoc⤾} \\ &\equiv (\tau_1 \wedge \tau_2) \wedge \tau_1 && \text{by B-Commut⤾} \end{aligned}$$

**S-AndOr12·.**

$$\begin{aligned} \tau_2 &\equiv \tau_2 \wedge (\tau_2 \vee \tau_1) && \text{by B-Absorp·} \\ &\equiv \tau_2 \wedge (\tau_1 \vee \tau_2) && \text{by B-Commut·} \end{aligned}$$

**S-AndOr12⤾.**

$$\begin{aligned} \tau_1 \wedge \tau_2 &\equiv \tau_1 \wedge (\tau_2 \wedge \tau_2) && \text{by B-Idem⤾} \\ &\equiv (\tau_1 \wedge \tau_2) \wedge \tau_2 && \text{by B-Assoc⤾} \end{aligned}$$

**S-AndOr2·.**

$$\begin{aligned} \tau_1 \vee \tau_2 &\equiv (\tau_1 \wedge \tau) \vee \tau_2 && \text{by assumption } \tau_1 \subseteq \tau \Leftrightarrow \tau_1 \equiv \tau_1 \wedge \tau \\ &\equiv (\tau_1 \wedge \tau) \vee (\tau_2 \wedge \tau) && \text{by assumption } \tau_2 \subseteq \tau \Leftrightarrow \tau_2 \equiv \tau_2 \wedge \tau \\ &\equiv (\tau \wedge \tau_1) \vee (\tau_2 \wedge \tau) && \text{by B-Commut⤾} \\ &\equiv (\tau \wedge \tau_1) \vee (\tau \wedge \tau_2) && \text{by B-Commut⤾} \\ &\equiv \tau \wedge (\tau_1 \vee \tau_2) && \text{by B-Distrib·} \\ &\equiv (\tau_1 \vee \tau_2) \wedge \tau && \text{by B-Commut⤾} \end{aligned}$$

**S-AndOr2⤾.**

$$\begin{aligned} \tau &\equiv \tau \wedge \tau_2 && \text{by assumption } \tau \subseteq \tau_2 \Leftrightarrow \tau \equiv \tau \wedge \tau_2 \\ &\equiv (\tau \wedge \tau_1) \wedge \tau_2 && \text{by assumption } \tau \subseteq \tau_1 \Leftrightarrow \tau \equiv \tau \wedge \tau_1 \\ &\equiv \tau \wedge (\tau_1 \wedge \tau_2) && \text{by B-Assoc⤾} \end{aligned}$$

**S-DISTRIB$\cdot$.**

$$\begin{aligned}\tau \wedge (\tau_1 \vee \tau_2) &\equiv (\tau \wedge (\tau_1 \vee \tau_2)) \wedge (\tau \wedge (\tau_1 \vee \tau_2)) && \text{by B-IDEM}\curvearrowright\\ &\equiv (\tau \wedge (\tau_1 \vee \tau_2)) \wedge ((\tau \wedge \tau_1) \vee (\tau \wedge \tau_2)) && \text{by B-DISTRIB}\cdot\end{aligned}$$

**S-DISTRIB$\curvearrowright$.**

$$\begin{aligned}(\tau \vee \tau_1) \wedge (\tau \vee \tau_2) &\equiv ((\tau \vee \tau_1) \wedge (\tau \vee \tau_2)) \wedge ((\tau \vee \tau_1) \wedge (\tau \vee \tau_2)) && \text{by B-IDEM}\curvearrowright\\ &\equiv ((\tau \vee \tau_1) \wedge (\tau \vee \tau_2)) \wedge (\tau \vee (\tau_1 \wedge \tau_2)) && \text{by B-DISTRIB}\curvearrowright\end{aligned}$$

**S-TRANS.**

$$\begin{aligned}\tau_0 &\equiv \tau_0 \wedge \tau_1 && \text{by assumption } \tau_0 \subseteq \tau_1 \Leftrightarrow \tau_0 \equiv \tau_0 \wedge \tau_1\\ &\equiv \tau_0 \wedge (\tau_1 \wedge \tau_2) && \text{by assumption } \tau_1 \subseteq \tau_2 \Leftrightarrow \tau_1 \equiv \tau_1 \wedge \tau_2\\ &\equiv (\tau_0 \wedge \tau_1) \wedge \tau_2 && \text{by B-ASSOC}\curvearrowright\\ &\equiv \tau_0 \wedge \tau_2 && \text{by assumption } \tau_0 \subseteq \tau_1 \Leftrightarrow \tau_0 \equiv \tau_0 \wedge \tau_1\end{aligned}$$

■

Contrary to full $\leqslant$-subtyping, $\subseteq$ only relates concrete type constructors (function, record, and nominal class tag types) in an obvious and syntactic way, making it easy to reason about. For example, notice that $\{\, x : \tau_1 \,\} \subseteq \{\, y : \tau_2 \,\}$ holds iff $x = y$ *and* $\tau_1 = \tau_2$ (i.e., iff they are *syntactically* the same).

**Definition 4.4** (Boolean algebra equivalence)**.** *We define* $(\cong)$ *as Boolean Algebra equivalence:*

$$\tau_1 \cong \tau_2 \;\Leftrightarrow\; \tau_1 \subseteq \tau_2 \text{ and } \tau_2 \subseteq \tau_1$$

Remark in passing: It is easy to show that $\tau_1 \cong \tau_1' \vee \tau_2$ implies $\tau_2 \subseteq \tau_1$. Indeed, it implies $\tau_1' \vee \tau_2 \subseteq \tau_1$, which implies $\tau_2 \subseteq \tau_1$. Similarly, $\tau_1' \wedge \tau_2 \cong \tau_1$ implies $\tau_1 \subseteq \tau_2$.

### *4.2.2 Elementary type forms*

The second step in our quest to *split* the Boolean-Algebraic subtyping relation in two is to define the *elementary type forms* making up the "meat" of a type system. This is the part where one gives formal meaning to whether any two type constructors are considered "related" or "unrelated".

We first define the elementary type forms for the $\mathfrak{S}_{\to\{x\}\#C}$ instantiation of $\mathfrak{S}(\mathcal{T}, \mathcal{R})$, and then explain how to derive similar rules for any other particular instantiation of the type system. Up until now, we have assumed that the subtyping rules in $\mathcal{R}$ can take arbitrary shapes. In the following discussion about elementary type forms, we limit rules in $\mathcal{R}$ to have at most two type constructors on the top level, as in S-FUNMRG, S-RCDMRG, and S-RCDTOP. We foresee that generalizing to more complex subtyping rules is possible, albeit non-trivial.

**Definition 4.5** (Constructors and negated constructors)**.** *The syntax of constructors and negated constructors is presented in Figure 6.*

The constructors and negated constructors for the elementary type forms, denoted by $C$, are straightforward. Each type constructor in $\mathcal{T}$, together with top and bottom, has one corresponding base (non-negated) constructor $B$. A constructor can be negated as in $\not{C}$. Negating a negated constructor cancels out the negations.

$$B ::= \to \mid x \mid \#C \mid \bot \mid \top$$
$$C, D ::= B \mid \not{B}$$

Notation: $\not{C} = \begin{cases} \not{B} & \text{if } C = B \\ B & \text{if } C = \not{B} \end{cases}$

Fig. 6. Syntax of constructor and negated constructor.

The elementary type forms are also straightforward, given our simplifying assumption. There are two kinds of elementary type forms: elementary union types $U^C$ and elementary intersection types $X^C$. Each type constructor in $\mathcal{T}$ corresponds to a kind of elementary union type and a kind of elementary intersection type, which are unions and intersections of an arbitrary number of instances of the type constructor. The top constructor is exclusive to the elementary union types, and likewise the bottom constructor is exclusive to the elementary intersection types. This notably means that empty unions and empty intersections cannot be represented as elementary union types and elementary intersection types respectively. Each subtyping rule in $\mathcal{R}$ adds an extra form to either $U^\top$ or $X^\bot$. The choice is arbitrary: using Theorem A.9, we can either move both type constructors in a rule to the LHS and add a form to $X^\bot$, or move them to the RHS and add a form to $U^\top$. We chose below such that the type constructors do not appear under a negation. For elementary type forms of negated constructors, we can conveniently represent elementary union types of a negated constructor as negations of an elementary intersection types of the non-negated constructor and vice versa thanks to the de Morgan rule.

In order to generalize elementary type forms to subtyping rules of arbitrary shapes, they would need to be parametrized by *sets* of constructors instead of single constructors.

**Definition 4.6** (Elementary type forms)**.** *The "elementary" type forms are defined in Figure 7. These are conceptually the type forms we need to care about for the system to be sound.*

**Lemma 4.7** (Inversion of negated elementary types)**.**

*(A) For all $C$ and $U^C$, we have $\neg U^C \cong X^{\not{C}}$ for some $X^{\not{C}}$.*
*(B) For all $C$ and $X^C$, we have $\neg X^C \cong U^{\not{C}}$ for some $U^{\not{C}}$.*

**Proof** By case analysis on $C$.

$$
\begin{array}{rl}
 & \underline{\text{Elementary union types}} \quad \boxed{U^C,\ V^C} \\
U^{\rightarrow} & ::= (\tau_1 \to \pi_1) \vee \cdots \vee (\tau_n \to \pi_n) \\
U^{x} & ::= \{\, x : \tau_1 \,\} \vee \cdots \vee \{\, x : \tau_n \,\} \\
U^{\#C} & ::= \#C \\
U^{\top} & ::= \top \mid \{\, x_1 : \tau_1 \,\} \vee \{\, x_2 : \tau_2 \,\} \quad (\text{where } x_1 \neq x_2) \\
 & \quad \mid \{\, x_1 : \tau_1 \,\} \vee (\tau \to \pi) \\
U^{\not B} & ::= \neg X^B
\end{array}
$$

$$
\begin{array}{rl}
 & \underline{\text{Elementary intersection types}} \quad \boxed{X^C,\ Y^C} \\
X^{\rightarrow} & ::= (\tau_1 \to \pi_1) \wedge \cdots \wedge (\tau_n \to \pi_n) \\
X^{x} & ::= \{\, x : \tau_1 \,\} \wedge \cdots \wedge \{\, x : \tau_n \,\} \\
X^{\#C} & ::= \#C \\
X^{\bot} & ::= \bot \mid \#C_1 \wedge \#C_2 \quad (\text{where } C_1 \text{ and } C_2 \text{ are unrelated}) \\
X^{\not B} & ::= \neg U^B
\end{array}
$$

Fig. 7. Elementary type form definition.

(A) If $C = B$ for some B, then pick $X^{\not C} = X^{\not B} = \neg U^B = \neg U^C$. If $C = \not B$ for some $B$, then $U^C = U^{\not B} = \neg X^B$ by the definition of $U^{\not B}$, so $\neg U^C = \neg\neg X^B \cong X^B = X^{\not C}$.
(B) If $C = B$ for some B, then pick $U^{\not C} = U^{\not B} = \neg X^B = \neg X^C$. If $C = \not B$ for some $B$, then $X^C = X^{\not B} = \neg U^B$ by the definition of $X^{\not B}$, so $\neg X^C = \neg\neg U^B \cong U^B = U^{\not C}$.

■

**Definition 4.8** (Helper pseudo-subtyping relation)**.** *The rules of the pseudo-subtyping are defined in Figure 8. It is easy to show that $\preceq$ implies $\leqslant$.*

The helper pseudo-subtyping relation relates elementary type forms with each other. The relation for elementary union types has an intersection of the same kind on the LHS, and a single one, possibly with a different constructor, on the RHS. Similarly, the relation for elementary intersection types has a union of the same kind on the RHS, and a single one, possibly with a different constructor, on the LHS.

The relation is made up of a few components:

- Two rules relating the top and bottom elementary types forms to any other constructors, serving the purpose of S-ToB.
- Two rules applying negations and inverting the two sides, serving the purpose of S-NegInv.
- Five rules for each type constructor (except the nullary ones): one for depth subtyping, two for applying the merge rule to the LHS of each kind of elementary type, and two for applying to the RHS.

- One rule for each nullary constructor, serving as reflexivity.
- One rule for each subtyping rule in $\mathcal{R}$, which can be obtained by moving one type constructor to each side, then applying S-AndOr2 to generalize it to fit the syntax of the relation (i.e., an intersection of unions on the LHS and a union on the RHS, or a union of intersections on the RHS and an intersection on the LHS).

In order to generalize the pseudo-subtyping relation to subtyping rules of arbitrary shapes, we would need to lift the restriction of the elementary type form components of unions or intersections having the same constructor, together with parametrizing elementary types forms by sets of constructors as above.

$$\boxed{\begin{array}{l}\Sigma \vdash \bigwedge_i U_i^C \preceq V^D \\ \Sigma \vdash X^C \preceq \bigvee_i Y_i^D\end{array}} \qquad \frac{}{\Sigma \vdash \bigwedge_i U_i^C \preceq V^\top} \qquad \frac{}{\Sigma \vdash X^\bot \preceq \bigvee_i Y_i^C}$$

$$\frac{\Sigma \vdash Y^D \preceq \bigvee_i X_i^C}{\Sigma \vdash \bigwedge_i U_i^{\not C} \preceq V^{\not D}} \qquad \frac{\Sigma \vdash \bigwedge_i V_i^D \preceq U^C}{\Sigma \vdash X^{\not C} \preceq \bigvee_i Y_i^{\not D}} \qquad \frac{\triangleleft\Sigma \vdash \tau' \leqslant \tau \quad \triangleleft\Sigma \vdash \pi \leqslant \pi'}{\Sigma \vdash \tau \to \pi \preceq \tau' \to \pi'}$$

$$\frac{\Sigma \vdash U^C \preceq (\bigwedge_i \tau_i) \to (\bigvee_i \pi_i)}{\Sigma \vdash U^C \preceq \bigvee_i \tau_i \to \pi_i} \qquad \frac{\Sigma \vdash (\bigvee_i \tau_i) \to (\bigwedge_i \pi_i) \preceq Y^C}{\Sigma \vdash \bigwedge_i \tau_i \to \pi_i \preceq Y^C}$$

$$\frac{\Sigma \vdash (\bigvee_i \bigwedge_j \tau_{ij}) \to (\bigwedge_i \bigvee_j \pi_{ij}) \preceq U^C}{\Sigma \vdash \bigwedge_i \bigvee_j \tau_{ij} \to \pi_{ij} \preceq U^C} \qquad \frac{\Sigma \vdash X^C \preceq (\bigwedge_i \bigvee_j \tau_{ij}) \to (\bigvee_i \bigwedge_j \pi_{ij})}{\Sigma \vdash X^C \preceq \bigvee_i \bigwedge_j \tau_{ij} \to \pi_{ij}}$$

$$\frac{\triangleleft\Sigma \vdash \tau \leqslant \tau'}{\Sigma \vdash \{\, x : \tau \,\} \preceq \{\, x : \tau' \,\}} \qquad \frac{\Sigma \vdash U^C \preceq \{\, x : \bigvee_i \tau_i \,\}}{\Sigma \vdash U^C \preceq \bigvee_i \{\, x : \tau_i \,\}} \qquad \frac{\Sigma \vdash \{\, x : \bigwedge_i \tau_i \,\} \preceq Y^C}{\Sigma \vdash \bigwedge_i \{\, x : \tau_i \,\} \preceq Y^C}$$

$$\frac{\Sigma \vdash \{\, x : \bigwedge_i \bigvee_j \tau_{ij} \,\} \preceq U^C}{\Sigma \vdash \bigwedge_i \bigvee_j \{\, x : \tau_{ij} \,\} \preceq U^C} \qquad \frac{\Sigma \vdash X^C \preceq \{\, x : \bigvee_i \bigwedge_j \tau_{ij} \,\}}{\Sigma \vdash X^C \preceq \bigvee_i \bigwedge_j \{\, x : \tau_{ij} \,\}} \qquad \frac{C_2 \in \mathcal{S}(\#C_1)}{\Sigma \vdash \bigwedge_i \#C_1 \preceq \#C_2}$$

$$\frac{C_2 \in \mathcal{S}(\#C_1)}{\Sigma \vdash \#C_1 \preceq \bigvee_i \#C_2} \qquad \frac{x \neq y}{\Sigma \vdash \bigwedge_i U_i^{\not x} \preceq V^y} \qquad \frac{}{\Sigma \vdash \bigwedge_i U_i^{\not x} \preceq V^{\to}}$$

$$\frac{}{\Sigma \vdash \bigwedge_i U_i^{\not\to} \preceq V^x} \qquad \frac{C_1 \notin \mathcal{S}(\#C_2) \quad C_2 \notin \mathcal{S}(\#C_1)}{\Sigma \vdash X^{\#C_1} \preceq \bigvee_i Y^{\#\not C_2}}$$

Fig. 8. Helper pseudo-subtyping relation rules.

We can now express the soundness of subtyping at the level of elementary types, which is much easier to verify as their elementary subtyping relation is quite straightforward and notably does not include transitivity:

**Lemma 4.9** (Soundness of elementary subtyping)**.**

*(A) If* $\bigwedge_i U_i^C \preceq V^D$*, then either one of the following is true:*

- $D \in \{ C, \top, \not\perp \}$
- $C = \#C_1$ *and* $D = \#C_2$ *and* $C_2 \in \mathcal{S}(\#C_1)$
- $C = \#\not{C}_1$ *and* $D = \#\not{C}_2$ *and* $C_1 \in \mathcal{S}(\#C_2)$
- $C = \not{x}$ *and* $D = y \neq x$
- $C = \not{x}$ *and* $D = \rightarrow$
- $C = \not\rightarrow$ *and* $D = x$
- $C = \#C_1$ *and* $D = \#\not{C}_2$ *and* $C_1 \notin \mathcal{S}(\#C_2)$ *and* $C_2 \notin \mathcal{S}(\#C_1)$

*(B) If* $X^C \preceq \bigvee_i Y_i^D$*, then either one of the following is true:*

- $C \in \{ D, \bot, \not\top \}$
- $D = \#C_1$ *and* $C = \#C_2$ *and* $C_1 \in \mathcal{S}(\#C_2)$
- $D = \#\not{C}_1$ *and* $C = \#\not{C}_2$ *and* $C_2 \in \mathcal{S}(\#C_1)$
- $D = x$ *and* $C = \not{y} \neq \not{x}$
- $D = \rightarrow$ *and* $C = \not{x}$
- $D = x$ *and* $C = \not\rightarrow$
- $D = \#\not{C}_1$ *and* $C = \#C_2$ *and* $C_1 \notin \mathcal{S}(\#C_2)$ *and* $C_2 \notin \mathcal{S}(\#C_1)$

The soundness of elementary subtyping for $\mathfrak{S}(\mathcal{T}, \mathcal{R})$ can be read off the conclusions of its helper pseudo-subtyping relations rules.

Notice that Lemma 4.9 defines a binary relation between the constructor on the two sides of $\preceq$. Given a set of constructors $S$, we write $f^{\diamond}_{\preceq}$ for the set of possible constructors that can appear on the other side:

$$f^{\cdot}_{\preceq}(S) = \bigcup_{C \in S} \{ D \mid \exists \overline{U_i^C}^i, V^D.\ \textstyle\bigwedge_i U_i^C \preceq V^D \}$$
$$f^{\diamond}_{\preceq}(S) = \bigcup_{D \in S} \{ C \mid \exists X^C, \overline{Y_i^D}^i.\ X^C \preceq \textstyle\bigvee_i Y_i^D \}$$

**Lemma 4.10.** *For* $\tau \in \{ \tau_1 \rightarrow \tau_2, \{ x : \tau_1 \}, \#C \}$*,*

*(A) If* $U^C \subseteq \tau$*, then* $U^C = \bigvee_i \tau$*.*
*(B) If* $\tau \subseteq X^C$*, then* $X^C = \bigwedge_i \tau$*.*

**Corollary 4.11.** *For* $\tau \in \{ \tau_1 \rightarrow \tau_2, \{ x : \tau_1 \}, \#C \}$*,*

*(A) If* $U^C \subseteq \neg\tau$*, then* $U^C = \bigvee_i \neg\tau$*.*
*(B) If* $\neg\tau \subseteq X^C$*, then* $X^C = \bigwedge_i \neg\tau$*.*

**Proof**

(A) We have $U^C = \bigvee_i U_i^{C_i}$ for some $\overline{U_i^{C_i}}^i$, where $\overline{U_i^{C_i}}^i$ are not unions. Then by S-NegInv, Theorem A.10, Theorem A.11, and Theorem A.17, we have $\tau \subseteq \bigwedge_i U_i^{\not{C}_i}$, which implies $\bigwedge_i U_i^{\not{C}_i} = \bigwedge_i \tau$ by Lemma 4.10, i.e., $\overline{U_i^{\not{C}_i} = \tau}^i$. Then we have $U^C = \bigvee_i \neg\tau$.

(B) We have $X^C = \bigwedge_i X_i^{C_i}$ for some $\overline{X_i^{C_i}}^i$, where $\overline{X_i^{C_i}}^i$ are not intersections. Then by S-NegInv, Theorem A.10, Theorem A.11, and Theorem A.17, we have $\bigvee_i X_i^{\not C_i} \subseteq \tau$, which implies $\bigvee_i X_i^{\not C_i} = \bigvee_i \tau$ by Lemma 4.10, i.e., $\overline{X_i^{\not C_i} = \tau}^i$. Then we have $X^C = \bigwedge_i \neg\tau$.

■

### *4.3 A First Attempt at an Inductive Lemma*

Now that the subtyping relation is properly split between its Boolean-algebraic fragment and its elementary types fragment, we can start sketching what the inductive lemma of subtyping soundness should look like. However, we will see later that the full statement of that lemma is quite complex and requires more scaffolding still. So it helps to first look at the statement of our first attempt, which *does not hold* in general:

1. If $\triangleright\Sigma \vdash \tau \leqslant \pi$ and $\tau \cong \bigwedge_i \left(\tau_i' \vee U_i^{C_i}\right)$, then there exists some $\overline{\pi_j'}^j$ and $\overline{D_j}^j$ and $\overline{V_j^{D_j}}^j$ such that $\pi \cong \bigwedge_j \left(\pi_j' \vee V_j^{D_j}\right)$ and $\overline{\triangleright\Sigma \vdash \bigwedge_{i \in S_j} U_i^{C_i} \preceq V_j^{D_j}}^j$ for some $\overline{S_j}^j$.
2. If $\triangleright\Sigma \vdash \tau \leqslant \pi$ and $\pi \cong \bigvee_j \left(\pi_j' \wedge Y_j^{D_j}\right)$, then there exists some $\overline{\tau_i'}^i$ and $\overline{C_i}^i$ and $\overline{X_i^{C_i}}^i$ such that $\tau \cong \bigvee_i \left(\tau_i' \wedge X_i^{C_i}\right)$ and $\overline{\triangleright\Sigma \vdash X_i^{C_i} \preceq \bigvee_{j \in S_i} Y_j^{D_j}}^i$ for some $\overline{S_i}^i$.

The first direction of this already quite complex statement should be understood intuitively as follows: *Assuming $\tau \leqslant \pi$, if it is possible to rewrite $\tau$ through the rules of pure Boolean algebra into a big intersection $\bigwedge_i \left(\tau_i' \vee U_i^{C_i}\right)$ of arbitrary types $\tau_i'$ unioned with some elementary types $U_i^{C_i}$ of constructors $C_i$, then $\pi$ should not be completely "arbitrary" and should somehow "include" the elementary types $U_i^{C_i}$ after some potential consolidation (taking the intersection of all $U_i^{C_i}$ together) and some widening into elementary types $V_j^{D_j}$, i.e., $\bigwedge_{i \in S_j} U_i^{C_i} \preceq V_j^{D_j}$.* The other direction reads similarly. Intuitively, no matter how we manage to reorganize the intersected components of the left-hand side $\tau$ through Boolean-algebraic massaging (notably distributivity), we should *only* be able to reach, through subtyping, a right-hand side type that is itself no more specific. Finally, notice that we here assume a *guarded* subtyping context $\triangleright\Sigma$ without loss of generality thanks to the properties of unassuming derivations (Section 3.8).

Unfortunately, this lemma does not work as is. Its proof still cannot proceed by standard induction due to the interaction between S-AndOr2 and S-Distrib. As an example, consider the following derivation for some $\tau \in \{\ \bot,\ \top,\ \#C',\ \tau_1 \to \tau_2,\ \{\ \overline{x_i : \tau_i}^i\ \}\ \}$ and unrelated

classes $C$ and $D$:

$$\dfrac{\dfrac{}{\#C \wedge (\#D \vee \neg\#C) \leqslant \#C \wedge \#D \vee \#C \wedge \neg\#C}\ \text{S-Distrib}\cdot \qquad \dfrac{\vdots}{\#C \wedge \#D \vee \#C \wedge \neg\#C \leqslant \bot}}{\textbf{(1)}\ \#C \wedge (\#D \vee \neg\#C) \leqslant \bot}\ \text{S-Trans}$$

$$\dfrac{\dfrac{\dfrac{\vdots}{\tau \leqslant \#C} \qquad \dfrac{\vdots}{\tau \leqslant \#D \vee \neg\#C}}{\tau \leqslant \#C \wedge (\#D \vee \neg\#C)}\ \text{S-AndOr2}\supset \qquad \textbf{(1)}}{\tau \leqslant \bot}\ \text{S-Trans}$$

According to the goal of our Theorem 4.1, $\tau$ can only be $\bot$. However, from the subderivations for $\tau \leqslant \#C$ and $\tau \leqslant \#D \vee \neg\#C$, nothing locally restricts $\tau$ to be $\bot$. This is because S-Distrib can split a complement into two separate subderivations to be later merged back together by S-AndOr2. To overcome this difficulty, we normalize the shape of subtyping derivations by introducing the CDN- and DCN-normalized type forms and derivations, which respectively stand for *conjunctions-disjunctions-negations* and *disjunctions-conjunctions-negations*.

CDN- and DCN-normalized derivations require S-Distrib$\diamond$ to be followed immediately by S-AndOr2$\diamond$. We show that all types and subtyping derivations can be translated into an equivalent CDN-normalized one and an equivalent DCN-normalized one. This will allow us to carry out the proof of the full inductive lemma (4.22) by induction on CDN- and DCN-normalized subtyping derivations.

As we mentioned before, the above simplified version of inductive soundness does not hold in general. The problematic cases arise when $\tau \equiv \bot$ for direction 1 and $\pi \equiv \top$ for direction 2. Since the relation holds by S-Trans with S-ToB for any type on the other side, we should not be able to conclude anything about it. Fortunately, we do not need to care about such cases for proving Theorem 4.1. Therefore, we can exclude them by adding side conditions on the elementary type forms, and making sure that they are preserved in the conclusion of the lemma, allowing us to apply it successively within a transitivity chain. For direction 1, in order to reject cases where $\tau \cong \bot$, we require $\bigwedge_i U_i^{C_i}$ to be complement-free, then we have $\tau \cong \bigwedge_i \left(\tau_i' \vee U_i^{C_i}\right) \supseteq \bigwedge_i U_i^{C_i} \not\in \bot$, which implies $\tau \not\in \bot$ by the antisymmetry and boundedness of Boolean algebras. For direction 2, we symmetrically require $\bigvee_j Y_j^{D_j}$ to be complement-free. To reject cases where $\tau \equiv \bot$ but $\tau \not\cong \bot$ for direction 1, we add restrictions on the set of elementary type constructors $\{\, \overline{C_i}^{\,i} \,\}$. For example, since we can derive $\tau_1 \to \tau_2 \leqslant \tau_3 \to \tau_4$ for some $\overline{\tau_i}^{\,i \in 1..4}$, which implies $\tau_1 \to \tau_2 \wedge \neg(\tau_3 \to \tau_4) \leqslant \bot$ by Theorem A.9, we reject cases where both $\to \in \{\, \overline{C_i}^{\,i} \,\}$ and $\not\to \in \{\, \overline{C_i}^{\,i} \,\}$. We can derive similar restrictions from other subtyping rules, and symmetric restrictions for direction 2.

### *4.4 CDN- and DCN-normalized type forms and derivations*

Since the intersection, union, and negation connectives can freely nest within and intertwine with each other, they introduce significant difficulty for the proof of subtyping consistency. We introduce the CDN- and DCN-normalized forms to order them one after the other, using only the Boolean-algebraic relation, i.e., not normalizing deeply under constructors. This is

by contrast to the RDNF form we will introduce later as part of type inference (Section 7.2), where deep normalization is important to ensure termination.

We also present alternative sets of subtyping rules where only the respective normalized forms appear in the top level, and show that any subtyping derivations can be translated into a normalized one. Thus we can prove any property by induction on normalized derivations.

### *4.4.1 CDN-normalized type forms*

**Definition 4.12** (CDN-normalized form)**.** *The syntax of CDN-normalized (conjunction-disjunction-negation) form is presented in Figure 9.*

$$
\begin{aligned}
\tau^{0} &::= T\ \overline{\tau^{+}}\ \overline{\tau^{-}}\ \overline{\tau^{0}} \mid \alpha \mid \top \\
\tau^{\mathrm{n}} &::= \tau^{0} \mid \neg\tau^{0} \\
\tau^{\mathrm{dn}} &::= \tau^{\mathrm{n}} \mid \tau^{\mathrm{n}} \vee \tau^{\mathrm{dn}} \\
\tau^{\mathrm{cdn}} &::= \tau^{\mathrm{dn}} \mid \tau^{\mathrm{dn}} \wedge \tau^{\mathrm{cdn}}
\end{aligned}
$$

Fig. 9. Syntax of CDN-normalized form.

In the proofs below, we sometimes abuse the notations $\tau_1^{\mathrm{dn}} \vee \tau_2^{\mathrm{dn}}$ and $\tau_1^{\mathrm{cdn}} \wedge \tau_2^{\mathrm{cdn}}$ to mean their properly associated versions, i.e., $\mathrm{dis}(\tau_1^{\mathrm{dn}}, \tau_2^{\mathrm{dn}})$ and $\mathrm{con}(\tau_1^{\mathrm{cdn}}, \tau_2^{\mathrm{cdn}})$ in Figure 10 respectively.

**Definition 4.13** (CDN-normalized form translation)**.** *The translation from arbitrary types into CDN-normalized types* $\mathrm{cdn}(\cdot)$ *is defined in Figure 10.*

**Lemma 4.14.** *For any* $\tau$, $\mathrm{cdn}(\tau) \cong \tau$.

**Proof** By straightforward induction. ∎

**Definition 4.15** (Complement-free CDN-normalized form)**.** *We say that a CDN-normalized form* $\tau^{\mathrm{cdn}}$ *is* complement-free *if* $\tau^{\mathrm{cdn}} = \bigwedge_i \bigvee_{j \in 1..n_i} \tau_{ij}^{\mathrm{n}}$, *where* $\overline{\forall j_i \in 1..n_i}^{\,i}.\ \forall i',\ \mathrm{neg}(\tau_{i'j}^{\mathrm{n}}) \notin \{\ \overline{\tau_{ij}^{\mathrm{n}}}^{\,i \neq i'}\ \}$.

It is easy to see that if $\tau^{\mathrm{cdn}} = \bigwedge_i \bigvee_{j \in 1..n_i} \tau_{ij}^{\mathrm{n}}$ is complement-free, then $\overline{\forall j_i \in 1..n_i}^{\,i}.\ \bigwedge_i \tau_{ij_i}^{\mathrm{n}} \not\leqslant \bot$.

**Definition 4.16** (CDN-normalized subtyping context)**.** $\Sigma$ *is CDN-normalized if for all* $H \in \Sigma$, *either one of the following is true:*

1. $H = (\top \leqslant \bigvee_i \tau_i^{\mathrm{n}})$, *where* $\forall\alpha.\ \{\ \alpha,\ \neg\alpha\ \} \cap \{\ \overline{\tau_i^{\mathrm{n}}}^{\,i}\ \} = \varnothing$;
2. $H = (\alpha \leqslant \bigvee_i \tau_i^{\mathrm{n}})$, *where the following are true:*

$$\boxed{\mathrm{cdn}(\tau)} : \tau^{\mathrm{cdn}}$$
$$\mathrm{cdn}(\tau^0) = \tau^0$$
$$\mathrm{cdn}(\bot) = \neg\top$$
$$\mathrm{cdn}(\neg\tau) = \mathrm{neg}(\mathrm{cdn}(\tau))$$
$$\mathrm{cdn}(\tau_1 \vee \tau_2) = \mathrm{dis}(\mathrm{cdn}(\tau_1),\ \mathrm{cdn}(\tau_2))$$
$$\mathrm{cdn}(\tau_1 \wedge \tau_2) = \mathrm{con}(\mathrm{cdn}(\tau_1),\ \mathrm{cdn}(\tau_2))$$

$$\boxed{\mathrm{neg}(\tau^{\mathrm{cdn}})} : \tau^{\mathrm{cdn}}$$
$$\mathrm{neg}(\tau^0) = \neg\tau^0$$
$$\mathrm{neg}(\neg\tau^0) = \tau^0$$
$$\mathrm{neg}(\tau_1^{\mathrm{n}} \vee \tau_2^{\mathrm{dn}}) = \mathrm{con}(\mathrm{neg}(\tau_1^{\mathrm{n}}),\ \mathrm{neg}(\tau_2^{\mathrm{dn}}))$$
$$\mathrm{neg}(\tau_1^{\mathrm{dn}} \wedge \tau_2^{\mathrm{cdn}}) = \mathrm{dis}(\mathrm{neg}(\tau_1^{\mathrm{dn}}),\ \mathrm{neg}(\tau_2^{\mathrm{cdn}}))$$

$$\boxed{\mathrm{dis}(\tau^{\mathrm{cdn}},\ \tau^{\mathrm{cdn}})} : \tau^{\mathrm{cdn}}$$
$$\mathrm{dis}(\tau_{11}^{\mathrm{dn}} \wedge \tau_{12}^{\mathrm{cdn}},\ \tau_2^{\mathrm{cdn}}) = \mathrm{con}(\mathrm{dis}(\tau_{11}^{\mathrm{dn}},\ \tau_2^{\mathrm{cdn}}),\ \mathrm{dis}(\tau_{12}^{\mathrm{cdn}},\ \tau_2^{\mathrm{cdn}}))$$
$$\mathrm{dis}(\tau_{11}^{\mathrm{n}} \vee \tau_{12}^{\mathrm{dn}},\ \tau_2^{\mathrm{cdn}}) = \mathrm{dis}(\tau_{11}^{\mathrm{n}},\ \mathrm{dis}(\tau_{12}^{\mathrm{dn}},\ \tau_2^{\mathrm{cdn}}))$$
$$\mathrm{dis}(\tau_1^{\mathrm{n}},\ \tau_{21}^{\mathrm{dn}} \wedge \tau_{22}^{\mathrm{cdn}}) = \mathrm{con}(\mathrm{dis}(\tau_1^{\mathrm{n}},\ \tau_{21}^{\mathrm{dn}}),\ \mathrm{dis}(\tau_1^{\mathrm{n}},\ \tau_{22}^{\mathrm{cdn}}))$$
$$\mathrm{dis}(\tau_1^{\mathrm{n}},\ \tau_2^{\mathrm{dn}}) = \tau_1^{\mathrm{n}} \vee \tau_2^{\mathrm{dn}}$$
$$\mathrm{Dis}_{i \in m..n}\ \tau_i^{\mathrm{cdn}} = \mathrm{dis}(\tau_m^{\mathrm{cdn}},\ \mathrm{Dis}_{i \in m+1..n}\ \tau_i^{\mathrm{cdn}})$$
$$\mathrm{Dis}_{i \in n..n}\ \tau_i^{\mathrm{cdn}} = \tau_n^{\mathrm{cdn}}$$

$$\boxed{\mathrm{con}(\tau^{\mathrm{cdn}},\ \tau^{\mathrm{cdn}})} : \tau^{\mathrm{cdn}}$$
$$\mathrm{con}(\tau_{11}^{\mathrm{dn}} \wedge \tau_{12}^{\mathrm{cdn}},\ \tau_2^{\mathrm{cdn}}) = \mathrm{con}(\tau_{11}^{\mathrm{dn}},\ \mathrm{con}(\tau_{12}^{\mathrm{cdn}},\ \tau_2^{\mathrm{cdn}}))$$
$$\mathrm{con}(\tau_1^{\mathrm{dn}},\ \tau_2^{\mathrm{cdn}}) = \tau_1^{\mathrm{dn}} \wedge \tau_2^{\mathrm{cdn}}$$
$$\mathrm{Con}_{i \in m..n}\ \tau_i^{\mathrm{cdn}} = \mathrm{con}(\tau_m^{\mathrm{cdn}},\ \mathrm{Con}_{i \in m+1..n}\ \tau_i^{\mathrm{cdn}})$$
$$\mathrm{Con}_{i \in n..n}\ \tau_i^{\mathrm{cdn}} = \tau_n^{\mathrm{cdn}}$$

Fig. 10. CDN-normalized form translation

- $\{\ \alpha,\ \neg\alpha\ \} \cap \{\ \overline{\tau_i^{\mathrm{n}}}^{\,i}\ \} = \varnothing$;
- $\forall\beta \in \{\ \overline{\tau_i^{\mathrm{n}}}^{\,i}\ \}.\ \neg\beta \notin \{\ \overline{\tau_i^{\mathrm{n}}}^{\,i}\ \}$;
- $\forall\beta \in \{\ \overline{\tau_i^{\mathrm{n}}}^{\,i}\ \}.\ \exists(\bigwedge_j \pi_j^{\mathrm{n}} \leqslant \beta) \in \Sigma.\ \{\ \overline{\pi_j^{\mathrm{n}}}^{\,j}\ \} = \{\ \overline{\mathrm{neg}(\tau_i^{\mathrm{n}})}^{\,i \mid \tau_i^{\mathrm{n}} \neq \beta},\ \alpha\ \}$;
- $\forall\neg\beta \in \{\ \overline{\tau_i^{\mathrm{n}}}^{\,i}\ \}.\ \exists(\beta \leqslant \bigvee_j \pi_j^{\mathrm{n}}) \in \Sigma.\ \{\ \overline{\pi_j^{\mathrm{n}}}^{\,j}\ \} = \{\ \overline{\tau_i^{\mathrm{n}}}^{\,i \mid \tau_i^{\mathrm{n}} \neq \neg\beta},\ \neg\alpha\ \}$;

3. $H = (\bigwedge_i \tau_i^{\mathrm{n}} \leqslant \alpha)$, *where the following are true:*

- $\{\ \alpha,\ \neg\alpha\ \} \cap \{\ \overline{\tau_i^{\mathrm{n}}}^{\,i}\ \} = \varnothing$;
- $\forall\beta \in \{\ \overline{\tau_i^{\mathrm{n}}}^{\,i}\ \}.\ \neg\beta \notin \{\ \overline{\tau_i^{\mathrm{n}}}^{\,i}\ \}$;

- $\forall \beta \in \{\, \overline{\tau_i^{\mathrm{n}}}^{\,i} \,\}.\ \exists (\beta \leqslant \bigvee_j \pi_j^{\mathrm{n}}) \in \Sigma.\ \{\, \overline{\pi_j^{\mathrm{n}}}^{\,j} \,\} = \{\, \overline{\mathrm{neg}(\tau_i^{\mathrm{n}})}^{\,i \mid \tau_i^{\mathrm{n}} \neq \beta},\ \alpha \,\}$;
- $\forall \neg \beta \in \{\, \overline{\tau_i^{\mathrm{n}}}^{\,i} \,\}.\ \exists (\bigwedge_j \pi_j^{\mathrm{n}} \leqslant \beta) \in \Sigma.\ \{\, \overline{\pi_j^{\mathrm{n}}}^{\,j} \,\} = \{\, \overline{\tau_i^{\mathrm{n}}}^{\,i \mid \tau_i^{\mathrm{n}} \neq \neg \beta},\ \neg \alpha \,\}$;

**Definition 4.17** (CDN-normalized subtyping context translation)**.** *The translation from arbitrary subtyping contexts into CDN-normalized subtyping contexts* $\mathrm{cdn}(\cdot)$ *is defined in Figure 11.*

$$\boxed{\mathrm{cdn}(\Sigma)} : \Sigma$$

$$\mathrm{cdn}(\Sigma) = \overline{\mathrm{cdn}(\top \leqslant \mathrm{cdn}(\neg \tau \vee \pi))}^{(\tau \leqslant \pi) \in \Sigma} \cdot \overline{\triangleright H}^{\triangleright H \in \Sigma}$$

$$\boxed{\mathrm{cdn}(\top \leqslant \tau^{\mathrm{cdn}})} : \Sigma$$

$$\mathrm{cdn}(\top \leqslant \textstyle\bigwedge_i \bigvee_{j_i} \tau_{ij_i}^{\mathrm{n}}) = \overline{\mathrm{cdn}(\top \leqslant \bigvee_{j_i} \tau_{ij_i}^{\mathrm{n}})}^{\,i}$$

$$\mathrm{cdn}(\top \leqslant \textstyle\bigvee_i \tau_i^{\mathrm{n}}) = \begin{cases} \epsilon & \text{if } \exists \alpha.\ \{\, \alpha,\ \neg \alpha \,\} \subseteq \{\, \overline{\tau_i^{\mathrm{n}}}^{\,i} \,\} \\ \overline{(\bigwedge_{i \mid \tau_i^{\mathrm{n}} \neq \alpha} \mathrm{neg}(\tau_i^{\mathrm{n}}) \leqslant \alpha)}^{\alpha \in \{\, \overline{\tau_i^{\mathrm{n}}}^{\,i} \,\}} \cdot \overline{(\alpha \leqslant \bigvee_{i \mid \tau_i^{\mathrm{n}} \neq \neg \alpha} \tau_i^{\mathrm{n}})}^{\alpha \mid \neg \alpha \in \{\, \overline{\tau_i^{\mathrm{n}}}^{\,i} \,\}} & \text{if } (\exists \alpha.\ \{\, \alpha,\ \neg \alpha \,\} \cap \{\, \overline{\tau_i^{\mathrm{n}}}^{\,i} \,\} \neq \varnothing) \text{ and } (\forall \alpha \in \{\, \overline{\tau_i^{\mathrm{n}}}^{\,i} \,\}.\ \neg \alpha \notin \{\, \overline{\tau_i^{\mathrm{n}}}^{\,i} \,\}) \\ (\top \leqslant \bigvee_i \tau_i^{\mathrm{n}}) & \text{if } \forall \alpha.\ \{\, \alpha,\ \neg \alpha \,\} \cap \{\, \overline{\tau_i^{\mathrm{n}}}^{\,i} \,\} = \varnothing \end{cases}$$

Fig. 11. CDN-normalized subtyping context translation

**Lemma 4.18.** *For any* $\Sigma$*, we have* $\Sigma \vDash \mathrm{cdn}(\Sigma)$ *and* $\mathrm{cdn}(\Sigma) \vDash \Sigma$*.*

**Proof** Straightforward, notably making use of Theorem A.9 and Lemma 4.14. ∎

### *4.4.2 CDN-normalized derivations*

For each rule in $\mathcal{R}$ with conclusion $\Sigma \vdash \tau \leqslant \pi$, we assume without loss of generality that $\mathrm{cdn}(\pi \vee \neg \tau) = \pi^{\mathrm{dn}}$ for some $\pi^{\mathrm{dn}}$, since we can otherwise split the rule into multiple simpler rules while keeping the original rule admissible.

**Definition 4.19** (CDN-normalized derivations)**.** *The CDN-normalized subtyping relation* $\leqslant^{\mathrm{cdn}}$ *is defined in Figure 12. The following are the difference compared to the full subtyping relation* $\leqslant$ *in Figure 16:*

- *On the top level, the relation is restricted to* $\Sigma \vdash \tau^{\mathrm{cdn}} \leqslant \tau^{\mathrm{cdn}}$*.*
- *On the top level, all occurrences of* $\bot$ *are replaced with* $\neg \top$*.*
- *The rule S-*Distrib$\diamond$ *is replaced by S-*DistribCdn$\diamond$*, which requires an application of S-*Distrib$\diamond$ *to be followed immediately by an application of S-*AndOr2$\cdot$ *in a transitivity chain by merging the two rules into one.*

- *For each rule in $\mathcal{R}$ with conclusion $\Sigma \vdash \tau \leqslant \pi$ and premises $\Sigma' \vdash \tau' \leqslant \pi'$, we transform them into the equivalent CDN-normalized derivation rule in $\mathcal{R}^{\mathrm{cdn}}$ by performing the following:*
  - *Transform the conclusion into $\Sigma \vdash \mathrm{cdn}(\tau) \leqslant^{\mathrm{cdn}} \mathrm{cdn}(\pi)$*
  - *If $\max(depth(\tau'), depth(\pi')) < \max(depth(\tau), depth(\pi))$, keep the premises as is*
  - *If $\max(depth(\tau'), depth(\pi')) = \max(depth(\tau), depth(\pi))$, then transform the premises into $\Sigma' \vdash \mathrm{cdn}(\tau') \leqslant^{\mathrm{cdn}} \mathrm{cdn}(\pi')$*

*Notice that S-TDepth is treated the same way as rules in $\mathcal{R}$, so its premises still refer to the full $\leqslant$ relation, even though its conclusion is about the $\leqslant^{\mathrm{cdn}}$ relation.*

*The CDN-normalized boolean subtyping relation $\subseteq^{\mathrm{cdn}}$ is defined similarly.*

Notice that Lemma A.7 and Lemma 3.1 extend to CDN-normalized derivations. In the proofs below, we also make use of extended versions of commutativity ($\tau_1 \vee^\diamond \tau_2 (\vee^\diamond \tau_3) \leqslant^{\mathrm{cdn}} \tau_2 \vee^\diamond \tau_1 (\vee^\diamond \tau_3)$) and idempotence ($\tau_1 \vee^\diamond \tau_1 (\vee^\diamond \tau_2) \leqslant^{\mathrm{cdn}} \tau_1 (\vee^\diamond \tau_2)$).

**Lemma 4.20.** *$\Sigma \vdash \tau_1^{\mathrm{cdn}} \leqslant \tau_2^{\mathrm{cdn}}$ if $\Sigma \vdash \tau_1^{\mathrm{cdn}} \leqslant^{\mathrm{cdn}} \tau_2^{\mathrm{cdn}}$. Similarly, $\tau_1^{\mathrm{cdn}} \subseteq \tau_2^{\mathrm{cdn}}$ if $\tau_1^{\mathrm{cdn}} \subseteq^{\mathrm{cdn}} \tau_2^{\mathrm{cdn}}$.*

**Proof** It is easy to see that every rule of $\leqslant^{\mathrm{cdn}}$ is admissible in $\leqslant$. ∎

**Lemma 4.21.** *If $\Sigma \vdash \tau \leqslant \pi$, then $\mathrm{cdn}(\Sigma) \vdash \mathrm{cdn}(\tau) \leqslant^{\mathrm{cdn}} \mathrm{cdn}(\pi)$. Similarly, if $\tau \subseteq \pi$, then $\mathrm{cdn}(\tau) \subseteq^{\mathrm{cdn}} \mathrm{cdn}(\pi)$.*

#### *4.4.3 DCN-normalized type forms and derivations*

The DCN-normalized (disjunction-conjunction-negation) type forms and derivations are symmetric to its CDN counterpart, except that the order of unions and intersections are swapped. Its detailed description can be found in the appendix.

### *4.5 Soundness of Subtyping*

We can now finally state the general inductive lemma supporting the subtyping soundness theorem (4.1). The main intuition over out first attempt in Section 4.3 is to add preconditions to exclude sets of constructors that can reduce to bottom or top in directions for the elementary union and intersection types respectively, and ensure that the possible set of constructors in the conclusion preserves this property.

We can obtain these restrictions on the sets of constructors by examining the subtyping rules. S-Compl forbids any constructor to appear together its negated counterpart. By moving all the type constructors in the rules in $\mathcal{R}$ to one side, we can read out the restrictions they impose. We then ensure that for all sets of constructors $\{ \overline{C} \}$ that satisfies this restriction, $f_\leq^\diamond(\{ \overline{C} \})$ also satisfies it. This does not hold in general. Adding more rules to $\mathcal{R}$ results in $f_\leq^\diamond$ mapping to a larger set of constructors, while simultaneously adding to the restrictions here. So the rules in $\mathcal{R}$ must be designed carefully in order to ensure the soundness of subtyping.

$$\boxed{\Sigma \vdash \tau^{\mathrm{cdn}} \leqslant^{\mathrm{cdn}} \tau^{\mathrm{cdn}}} \qquad \boxed{\tau^{\mathrm{cdn}} \leqslant^{\mathrm{cdn}} \tau^{\mathrm{cdn}}}$$

$$\triangleleft \Xi = \Xi \qquad \triangleleft(\Sigma \cdot H) = \triangleleft\Sigma \cdot H \qquad \triangleleft(\Sigma \cdot \triangleright H) = \triangleleft\Sigma \cdot H$$

S-Refl
$$\frac{}{\tau^{\mathrm{cdn}} \leqslant^{\mathrm{cdn}} \tau^{\mathrm{cdn}}}$$

S-ToB$\cdot$
$$\frac{}{\tau^{\mathrm{cdn}} \leqslant^{\mathrm{cdn}} \top}$$

S-ToB$\rotatebox{180}{$\cdot$}$
$$\frac{}{\neg\top \leqslant^{\mathrm{cdn}} \tau^{\mathrm{cdn}}}$$

S-Compl$\cdot$
$$\frac{}{\top \leqslant^{\mathrm{cdn}} \tau^0 \vee \neg\tau^0}$$

S-Compl$\rotatebox{180}{$\cdot$}$
$$\frac{}{\tau^0 \wedge \neg\tau^0 \leqslant^{\mathrm{cdn}} \neg\top}$$

S-AndOr1$\cdot$
$$\frac{S \subseteq \{\, \overline{i} \,\}}{\bigvee_{i' \in S} \tau^{\mathrm{n}}_{i'} \leqslant^{\mathrm{cdn}} \bigvee_i \tau^{\mathrm{n}}_i}$$

S-AndOr1$\rotatebox{180}{$\cdot$}$
$$\frac{S \subseteq \{\, \overline{i} \,\}}{\bigwedge_i \tau^{\mathrm{dn}}_i \leqslant^{\mathrm{cdn}} \bigwedge_{i' \in S} \tau^{\mathrm{dn}}_{i'}}$$

S-AndOr2$\cdot$
$$\frac{\overline{\Sigma \vdash \tau^{\mathrm{n}}_i \leqslant^{\mathrm{cdn}} \tau^{\mathrm{cdn}}}^{\,i}}{\Sigma \vdash \bigvee_i \tau^{\mathrm{n}}_i \leqslant^{\mathrm{cdn}} \tau^{\mathrm{cdn}}}$$

S-AndOr2$\rotatebox{180}{$\cdot$}$
$$\frac{\overline{\Sigma \vdash \tau^{\mathrm{cdn}} \leqslant^{\mathrm{cdn}} \tau^{\mathrm{dn}}_i}^{\,i}}{\Sigma \vdash \tau^{\mathrm{cdn}} \leqslant^{\mathrm{cdn}} \bigwedge_i \tau^{\mathrm{dn}}_i}$$

S-DistribCdn$\cdot$
$$\frac{\overline{\Sigma \vdash \tau^{\mathrm{n}}_i \wedge \tau^{\mathrm{cdn}} \leqslant^{\mathrm{cdn}} \pi^{\mathrm{cdn}}}^{\,i}}{\Sigma \vdash (\bigvee_i \tau^{\mathrm{n}}_i) \wedge \tau^{\mathrm{cdn}} \leqslant^{\mathrm{cdn}} \pi^{\mathrm{cdn}}}$$

S-DistribCdn$\rotatebox{180}{$\cdot$}$
$$\frac{\Sigma \vdash \tau^{\mathrm{n}} \leqslant^{\mathrm{cdn}} \pi^{\mathrm{cdn}} \quad \Sigma \vdash \bigwedge_i \tau^{\mathrm{dn}}_i \leqslant^{\mathrm{cdn}} \pi^{\mathrm{cdn}}}{\Sigma \vdash \bigwedge_i (\tau^{\mathrm{n}} \vee \tau^{\mathrm{dn}}_i) \leqslant^{\mathrm{cdn}} \pi^{\mathrm{cdn}}}$$

S-Trans
$$\frac{\Sigma \vdash \tau^{\mathrm{cdn}}_0 \leqslant^{\mathrm{cdn}} \tau^{\mathrm{cdn}}_1 \quad \Sigma \vdash \tau^{\mathrm{cdn}}_1 \leqslant^{\mathrm{cdn}} \tau^{\mathrm{cdn}}_2}{\Sigma \vdash \tau^{\mathrm{cdn}}_0 \leqslant^{\mathrm{cdn}} \tau^{\mathrm{cdn}}_2}$$

S-Weaken
$$\frac{H}{\Sigma \vdash H}$$

S-Assum
$$\frac{\Sigma \cdot \triangleright H \vdash H}{\Sigma \vdash H}$$

S-Hyp
$$\frac{H \in \Sigma}{\Sigma \vdash H}$$

S-TMrg$\diamond$
$$\frac{}{T\ \overline{(\tau^+_i \vee^{\diamond} \pi^+_i)}^{\,i}\ \overline{(\tau^-_j \wedge^{\diamond} \pi^-_j)}^{\,j}\ \overline{\tau^0_k}^{\,k} \leqslant^{\diamond\mathrm{cdn}} T\ \overline{\tau^+_i}^{\,i}\ \overline{\tau^-_j}^{\,j}\ \overline{\tau^0_k}^{\,k} \vee^{\diamond} T\ \overline{\pi^+_i}^{\,i}\ \overline{\pi^-_j}^{\,j}\ \overline{\tau^0_k}^{\,k}}$$

S-TDepth
$$\frac{\overline{\triangleleft\Sigma \vdash \tau^+_i \leqslant \pi^+_i}^{\,i} \quad \overline{\triangleleft\Sigma \vdash \pi^-_j \leqslant \tau^-_j}^{\,j} \quad \overline{\triangleleft\Sigma \vdash \tau^0_k \equiv \pi^0_k}^{\,k}}{\Sigma \vdash T\ \overline{\tau^+_i}^{\,i}\ \overline{\tau^-_j}^{\,j}\ \overline{\tau^0_k}^{\,k} \leqslant^{\mathrm{cdn}} T\ \overline{\pi^+_i}^{\,i}\ \overline{\pi^-_j}^{\,j}\ \overline{\pi^0_k}^{\,k}} \qquad \mathcal{R}^{\mathrm{cdn}}$$

Fig. 12. CDN-normalized subtyping rules for $\mathfrak{S}(\mathcal{T}, \mathcal{R})$.

**Lemma 4.22** (Soundness of subtyping (inductive))**.**

*(A) If* $\triangleright\Sigma \vdash \tau \leqslant \pi$ *and* $\tau \cong \bigwedge_i \left(\tau'_i \vee U_i^{C_i}\right)$*, where the following are true:*

- $\bigwedge_i U_i^{C_i}$ *is a complement-free CDN-normalized form*
- $\rightarrow \notin \{\, \overline{C_i}^{\,i} \,\}$ *or* $\nrightarrow \notin \{\, \overline{C_i}^{\,i} \,\}$
- $\forall \mathscr{x} \in \{\, \overline{C_i}^{\,i} \,\}.\ \mathscr{x} \notin \{\, \overline{C_i}^{\,i} \,\}$
- $\forall \#C \in \{\, \overline{C_i}^{\,i} \,\}.\ \#\mathscr{C} \notin \{\, \overline{C_i}^{\,i} \,\}$
- $\forall \#C_1 \in \{\, \overline{C_i}^{\,i} \,\}, \#C_2 \in \{\, \overline{C_i}^{\,i} \,\}.\ C_1 \in \mathcal{S}(\#C_2)$ *or* $C_2 \in \mathcal{S}(\#C_1)$
- $|\{\, \mathscr{x} \mid \mathscr{x} \in \{\, \overline{C_i}^{\,i} \,\} \,\}| \leqslant 1$
- $|\{\, \mathscr{x} \mid \mathscr{x} \in \{\, \overline{C_i}^{\,i} \,\} \,\}| = 0$ *or* $\nrightarrow \notin \{\, \overline{C_i}^{\,i} \,\}$

*then there exists some* $\overline{\pi'_j}^{\,j}$ *and* $D_j \in \{\, \overline{C_i}^{\,i} \,\} \cup \{\, \top, \not\perp \,\} \cup \{\, \overline{x}^{\,x \notin \{\, \overline{C_i}^{\,i} \,\}} \,\} \cup \{\, \overline{\#\not C}^{\,\#C \notin \{\, \overline{C_i}^{\,i} \,\}} \,\}^{\,j}$ *and* $\overline{V_j^{D_j}}^{\,j}$ *such that* $\pi \cong \bigwedge_j \left(\pi'_j \vee V_j^{D_j}\right)$ *and* $\bigwedge_j V_j^{D_j}$ *is a complement-free CDN-normalized form and* $\overline{\triangleright\Sigma \vdash \bigwedge_{i \in S_j} U_i^{C_i} \preceq V_j^{D_j}}^{\,j}$ *for some* $\overline{S_j}^{\,j}$.

*(B) If* $\triangleright\Sigma \vdash \tau \leqslant \pi$ *and* $\pi \cong \bigvee_j \left(\pi'_j \wedge Y_j^{D_j}\right)$, *where the following are true:*

- $\bigvee_j Y_j^{D_j}$ *is a complement-free DCN-normalized form*
- $\rightarrow \notin \{\, \overline{D_j}^{\,j} \,\}$ *or* $\not\rightarrow \notin \{\, \overline{D_j}^{\,j} \,\}$
- $\forall x \in \{\, \overline{D_j}^{\,j} \,\}.\, x \notin \{\, \overline{D_j}^{\,j} \,\}$
- $\forall \#\mathcal{C} \in \{\, \overline{D_j}^{\,j} \,\}.\, \#C \notin \{\, \overline{D_j}^{\,j} \,\}$
- $\forall \#\not{\mathcal{C}}_1 \in \{\, \overline{C_i}^{\,i} \,\}, \#\not{\mathcal{C}}_2 \in \{\, \overline{C_i}^{\,i} \,\}.\, C_1 \in \mathcal{S}(\#C_2)$ *or* $C_2 \in \mathcal{S}(\#C_1)$
- $|\{\, x \mid x \in \{\, \overline{D_j}^{\,j} \,\} \,\}| \leqslant 1$
- $|\{\, x \mid x \in \{\, \overline{D_j}^{\,j} \,\} \,\}| = 0$ *or* $\rightarrow \notin \{\, \overline{D_j}^{\,j} \,\}$

*then there exists some* $\overline{\tau'_i}^{\,i}$ *and* $C_i \in \{\, \overline{D_j}^{\,j} \,\} \cup \{\, \bot, \not\top \,\} \cup \{\, \overline{x}^{\,x \notin \{\, \overline{D_j}^{\,j} \,\}} \,\} \cup \{\, \overline{\#C}^{\,\#\mathcal{C} \notin \{\, \overline{D_j}^{\,j} \,\}} \,\}^{\,i}$ *and* $\overline{X_i^{C_i}}^{\,i}$ *such that* $\tau \cong \bigvee_i \left(\tau'_i \wedge X_i^{C_i}\right)$ *and* $\bigvee_i X_i^{C_i}$ *is a complement-free DCN-normalized form and* $\overline{\triangleright\Sigma \vdash X_i^{C_i} \preceq \bigvee_{j \in S_i} Y_j^{D_j}}^{\,i}$ *for some* $\overline{S_i}^{\,i}$.

As usual, the proof is given in appendix. It relies on all the definitions and lemmas we have carefully developed throughout this section as well as some additional less interesting technical lemmas stated only in the appendix.

Notice that the property to prove has a conclusion that can itself be used as a hypothesis for another application of the property. When proving (A), we consider the leftmost rule application in a transitivity chain, show the property for it, and this allows us to apply the induction hypothesis on the rest of the chain; this works even if the chain is of length 1 (with no uses of S-Trans). When proving (B), we proceed in the same way but from the right. So we do not have to consider uses of S-Trans explicitly, and only consider uses of the other rules here.

### *4.6 Soundness of* $\mathfrak{S}(\mathcal{T}, \mathcal{R})$ *Subtyping*

We have only stated the soundness of $\lambda^{\neg}$ subtyping (i.e., $\mathfrak{S}_{\rightarrow\{x\}\#C}$) and reserved the statement of the soundness of $\mathfrak{S}(\mathcal{T}, \mathcal{R})$ for the end of this section as it depends on its elementary type forms and helper pseudo-subtyping relation. Specifically, by considering a directed graph with the constructors as nodes and binary relation in Lemma 4.9 as edges, we can read out what we can state in the soundness statement by considering an induced subgraph. For types restricted to the constructors corresponding to the nodes of the induced subgraph, only the subtyping relations corresponding to the edges hold. We can also read obtain additional information about these types from the premises of the helper pseudo-subtyping relation rules. For instance, if we consider the non-negated constructors for $\lambda^{\neg}$, the induced subgraph would only contain self-loops, edges from bottom to every other

node, and edges from every other node to top. Then we obtain the following soundness statement:

**Theorem 4.23** (Soundness of $\lambda^{\neg}$ subtyping (weak))**.** *If* $\Xi$ ***cons.*** *and* $\Xi \vdash \tau \leqslant \pi$*, where:*

$$\tau \in \{\ \bot,\ \top,\ \#C,\ \tau_1 \to \tau_2,\ \{\ x : \tau\ \}\ \}$$
$$\pi \in \{\ \bot,\ \top,\ \#C',\ \pi_1 \to \pi_2,\ \{\ x' : \pi\ \}\ \}$$

*then exactly one of the following is true:*

*(a)* $\tau = \bot$ *or* $\pi = \top$*;*
*(b)* $\tau = \#C$ *and* $\pi = \#C'$ *and* $C' \in \mathcal{S}(\#C)$*;*
*(c)* $\tau = \tau_1 \to \tau_2$ *and* $\pi = \pi_1 \to \pi_2$ *and* $\Xi \vdash \pi_1 \leqslant \tau_1$ *and* $\Xi \vdash \tau_2 \leqslant \pi_2$*;*
*(d)* $\tau = \{\ x : \tau\ \}$ *and* $\pi = \{\ x' : \pi\ \}$ *and* $x = x'$ *and* $\Xi \vdash \tau_k \leqslant \pi_1$ *for some* $k$*.*

We can then modify the record type on the LHS while ensuring that it still satisfies the restriction in Lemma 4.22 to obtain the full Theorem 4.1.

### *4.7 Contexts and Type Variables*

So far, we have ignored the subtyping context by requiring it to be guarded. Our handling of type variables and the subtyping context relies on two key insights: for Theorem 4.1, we do not care about type variables on the top level; and we do not care about all possible subtyping contexts, only the ones produced by type inference. We have previously defined the consistency of constraining contexts, and by ensuring that we only manipulate consistent contexts (as type inference will do), this allows us to guard the context in any subtyping derivations under consistent contexts and with no type variables on the top level by Lemma 3.4, which are all we care about for the remaining soundness and completeness proofs.

## 5 Inferring Principal Types for MLstruct

We now informally describe our general approach to principal type inference in MLstruct.

### *5.1 Algebraic Subtyping*

MLstruct follows Dolan's *algebraic subtyping* (2017) discipline, which distinguishes itself from so-called *semantic subtyping* approaches in that it focuses on the *algebraic* properties of types, instead of focusing on set-theoretic semantics. In algebraic subtyping, some subtyping relationships are *not necessary* and cannot be justified if one were to look at types purely as denotations for sets of values. These algebraic relationships are nevertheless sound to have in the type system, and in turn enable principal type inference and type simplification.

As an example, consider $(\tau_1 \to \tau_2) \wedge (\tau_3 \to \tau_4) \leqslant (\tau_1 \vee \tau_3) \to (\tau_2 \wedge \tau_4)$, which holds in Dolan's MLsub. While the other direction holds by simple contravariance of function

parameters and covariance of function results, this direction is a lot more contentious. It does not make sense from the set-theoretic point of view: a function that can be viewed as returning $\tau_2$ when given a $\tau_1$ and returning $\tau_4$ when given a $\tau_3$ cannot be viewed as *always* returning a $\tau_2 \wedge \tau_4$. For instance, consider $\lambda x.\ x$, typable both as $\mathrm{Int} \to \mathrm{Int}$ and as $\mathrm{Bool} \to \mathrm{Bool}$, and which *could* therefore be assigned type $(\mathrm{Int} \to \mathrm{Int}) \wedge (\mathrm{Bool} \to \mathrm{Bool})$. Surely, this function never returns an $\mathrm{Int} \wedge \mathrm{Bool}$ value (an uninhabited type) when called with an $\mathrm{Int} \vee \mathrm{Bool}$ argument. But in MLsub, $\lambda x.\ x$ by design *cannot* be assigned such an intersection type; instead, its most general type is $\forall \alpha.\ \alpha \to \alpha$, which does subsume both $\mathrm{Int} \to \mathrm{Int}$ and $\mathrm{Bool} \to \mathrm{Bool}$ though not $(\mathrm{Int} \to \mathrm{Int}) \wedge (\mathrm{Bool} \to \mathrm{Bool})$. This explains the restriction that intersections cannot be used to encode overloading in MLsub and MLstruct.

In MLstruct, we define further additional algebraic subtyping relationships, such as $\top \leqslant \{\, x : \tau_1 \,\} \vee (\tau_2 \to \tau_3)$, as hinted in Section 2.3.2. We similarly ensure that this relationship does not threaten soundness by making sure the language cannot meaningfully distinguish between values of these two types (i.e., one cannot pattern match on record or function types).

### *5.2 Basic Type Inference Idea*

We base the core of our type inference algorithm on a simple formulation of MLsub type inference we formulated in previous work (Parreaux, 2020). The constraint solver attaches a set of *lower* and *upper* bounds to each type variable, and maintain the transitive closure of these constraints, i.e., it makes sure that at all times the union of all lower bounds of a variable remains a *subtype* of the intersection of all its upper bounds. This means that when registering a new constraint of the form $\alpha \leqslant \tau$, we not only have to add $\tau$ to the upper bounds of $\alpha$, but also to constrain $\mathit{lowerBounds}(\alpha) \leqslant \tau$ in turn. One has to be particularly careful to maintain a "cache" of subtyping relationships currently being constrained, as the graphs formed by type variable bounds may contain cycles. Because types are regular, there is always a point, in a cyclic constraint, where we end up checking a constraint we are already in the process of checking (it is in the cache), in which case we can assume that the constraint holds and terminate. Constraints of the general form $\tau_1 \leqslant \tau_2$ are handled by *losslessly* decomposing them into smaller constraints, until we arrive at constraints on type variables, which is made possible by the algebraic subtyping rules. The losslessness of this approach is needed to ensure that we only infer *principal* types. In other words, when decomposing a constraint, we must produce a set of smaller constraints that is *equivalent* to the original constraint. For example, we can decompose the constraint $\tau_1 \vee (\tau_2 \to \tau_3) \leqslant \tau_4 \to \tau_5$ into the equivalent set of constraints: $\tau_1 \leqslant \tau_4 \to \tau_5$ ; $\tau_4 \leqslant \tau_2$ ; and $\tau_3 \leqslant \tau_5$. If we arrive at a constraint between two incompatible type constructors, such as $\tau_1 \to \tau_2 \leqslant \{\, x : \tau_3 \,\}$, an error is reported.

### 5.3 Solving Constraints with Unions and Intersections

By contrast with MLsub, MLstruct supports union and intersections types in a *first-class* capacity, meaning that one can use these types in both positive *and* negative positions. [19] This is particularly important to type check instance matching, which requires unions in negative positions, and class types, which require intersections in positive positions (both illegal in MLsub).

The main problem that arises in this setting is: ***How to resolve constraints with the shapes*** $\tau_1 \leqslant \tau_2 \vee \tau_3$ ***and*** $\tau_1 \wedge \tau_2 \leqslant \tau_3$ ***?*** Such constraints cannot be easily decomposed into simpler constraints without losing information — which would prevent us from achieving complete type inference — and without having to perform backtracking — which would quickly become intractable, even in non-pathological cases, and would yield a set of possible types instead of a single principal type. When faced with such constraints, we distinguish two cases: (1) there is a type variable among $\tau_1$, $\tau_2$, and $\tau_3$; and (2) conversely, none of these types are type variables.

#### 5.3.1 Negation Types

We use negation types to reformulate constraints involving type variables into forms that allow us to make progress, relying on the Boolean-algebraic properties of negation. A constraint such as $\tau_1 \leqslant \tau_2 \vee \alpha$ can be rewritten to $\tau_1 \wedge \neg\tau_2 \leqslant \alpha$ by turning the "positive" $\tau_2$ on the right into a "negative" on the left, as these are equivalent in a Boolean algebra.[20] Therefore, it is sufficient *and* necessary to constrain $\alpha$ to be a supertype of $\tau_1 \wedge \neg\tau_2$ to solve the constraint at hand. Similarly, we can solve $\alpha \wedge \tau_1 \leqslant \tau_2$ by constraining $\alpha$ to be a subtype of $\tau_2 \vee \neg\tau_1$.[21] When both transformations are possible, one may pick one or the other equivalently. The correctness of these transformations is formally demonstrated in Theorem A.9.. This approach provides a solution to case (1), but in a way it only pushes the problem around, delaying the inevitable apparition of case (2).

#### 5.3.2 Normalization of Constraints

To solve problem (2), we *normalize* constraints until they are in the shape "$\tau_{\text{con}} \leqslant \tau_{\text{dis}}$", where (using a horizontal overline to denote 0 to $n$ repetitions):

- $\tau_{\text{con}}$ represents $\top$, $\bot$, or the intersection of any non-empty subset of $\{\, \#C,\ \tau_1 \to \tau_2,\ \{\, \overline{x : \tau} \,\} \,\}$.
- $\tau_{\text{dis}}$ represents types of the form $\top$, $\bot$, $(\tau_1 \to \tau_2)\ \overline{\vee\, \#C}$, $\{\, x : \tau \,\}\ \overline{\vee\, \#C}$, or $\#C\ \overline{\vee\, \#C'}$.

[19] Positive positions correspond to the types that a term *outputs*, while negative positions correspond to the types that a term *takes in* as input. For instance, in $(\tau_0 \to \tau_1) \to \tau_2$, type $\tau_2$ is in positive position since it is the output of the main function, and the function type $(\tau_0 \to \tau_1)$ is in negative position, as it is taken as an input to the main function. On the other hand, $\tau_1$, which is returned *by the function taken as input* is in negative position (since it is provided by callers via the argument function), and $\tau_0$ is in positive position (since it is provided by the main function when calling the argument function).

[20] Aiken and Wimmers (1993) used a similar trick, albeit in a more specific *set-theoretic* interpretation of unions/intersections.

[21] If it were not for pattern matching, we could avoid negation types by adopting a more complicated representation of type variable bounds that internalizes the same information. That is, instead of $\alpha \leqslant \tau$ and $\alpha \geqslant \tau$ for a given type variable $\alpha$, we would have bounds of the form $\alpha \wedge \pi \leqslant \tau$ and $\alpha \vee \pi \geqslant \tau$, representing $\alpha \leqslant \tau \vee \neg\pi$ and $\alpha \geqslant \tau \wedge \neg\pi$ respectively. But reducing several upper/lower bounds into a single bound, which previously worked by simply intersecting/taking the union of them, would now be impossible without generalizing bounds further. Type simplification would also become difficult.

Let us consider a few examples. First, given a constraint like $(\tau_1 \vee \tau_2) \wedge \tau_3 \leqslant \tau_4$, we can distribute the intersection over the union thanks to the rules of Boolean algebras (see Section 3.3.4), which results in $(\tau_1 \wedge \tau_3) \vee (\tau_2 \wedge \tau_3) \leqslant \tau_4$, allowing us to solve $\tau_1 \wedge \tau_3 \leqslant \tau_4$ and $\tau_2 \wedge \tau_3 \leqslant \tau_4$ independently. Second, given a constraint like $\tau_1 \leqslant \{ x : \tau_2 \} \vee \tau_3 \to \tau_4$, we simply use the fact that $\{ x : \tau_2 \} \vee \tau_3 \to \tau_4 \equiv \top$ (as explained in Section 2.2.2) to reduce the constraint to $\tau_1 \leqslant \top$, a tautology. Third, with constraints containing intersected nominal class tags on the left, we can compute their greatest lower bound based on our knowledge of the single-inheritance class hierarchy. We eventually end up with constraints of the shape "$\tau_{\text{con}} \leqslant \tau_{\text{dis}}$" and there always exists a $\tau_i \in \tau_{\text{con}}$ and $\tau'_j \in \tau_{\text{dis}}$ such that we can reduce the constraint to an *equivalent* constraint $\tau_i \leqslant \tau'_j$. Notice that if two related nominal tags appears on each side, it is always safe to pick that comparison, as doing so does not entail any additional constraints. If there are no such related nominal tags, the only other choice is to find a type in the right-hand side to match a corresponding type in the left-hand side, and the syntax of these normal forms prevents there being more than one possible choice. All in all, our Boolean algebra of types equipped with various algebraic simplification laws ensures that we have a lossless way of resolving the complex constraints that arise from union and intersection types, enabling principal type inference.

The constraint solving algorithm described in Section 7.3 and implemented in the artifact uses the ideas explored above but puts the entire constraint into a *normal form*, instead of normalizing constraints on the fly. This helps to efficiently guarantee termination by maintaining a cache of currently-processed subtyping relationships in normal forms, which is straightforward to query.

### *5.4 Subsumption Checking*

*Subsumption checking*, denoted by $\leqslant^{\forall}$, is important to check that definitions conform to given signatures. Contrary to MLsub, which syntactically separates positive from negative types (the *polarity restriction*), and therefore requires different algorithms for constraint solving and subsumption checking, in MLstruct we can immediately *reuse* the constraint solving algorithm for subsumption checking, without requiring much changes to the type system. To implement $\forall \Xi_1.\ \tau_1 \leqslant^{\forall} \forall \Xi_2.\ \tau_2$, we instantiate all the type variables in $\Xi_1$, with their bounds, to fresh type variables, and we turn all the variables in $\Xi_2$ into *rigid* variables (so-called "skolems"). The latter can be done by turning these type variables into fresh *flexible* nominal tags and by inlining their bounds, expressing them in terms of unions, intersections, and recursive types. Since there is no polarity restrictions in our system, the resulting types can be compared directly using the normal constraint solving algorithm.

Flexible nominal tags $\#F$ are just like nominal class tags $\#C$, except that they can coexist with unrelated tags without reducing to $\bot$. For example, while $\#C_1 \wedge \#C_2$ is equivalent to $\bot$ in MLstruct when $C_1$ and $C_2$ are unrelated, $\#F \wedge \#C_2$ is not.[22] Flexible nominal tags are also the feature used to encode the nominal tags of *traits*, necessary to implement mixin traits as described in Section 2.1.2.

For lack of space, we do not formally describe subsumption checking in this paper.

[22] This requires extending the syntax of normal forms in a straightforward way to $\tau'_{\text{con}} ::= \tau_{\text{con}} \overline{\wedge \#F}$ and $\tau'_{\text{dis}} ::= \tau_{\text{dis}} \overline{\vee \#F}$.

### 5.5 Simplification and Presentation of Inferred Types

Type simplification and pretty-printing are important components of any practical implementation of MLsub and MLstruct. They indeed perform a lot of the heavy-lifting of type inference, massaging inferred types, which are often big and unwieldy, into neat and concise equivalent type expressions. In this section, we briefly explain how simplification is performed in MLstruct.

#### 5.5.1 Basic Simplifications

For basic simplifications, we essentially follow Parreaux (2020) — we remove polar occurrences of type variables, remove type variables "sandwiched" between identical bounds, and we perform some hash consing to simplify inferred recursive types. The simplification of unions, intersections, and negations is not fully addressed by Parreaux, since MLsub does not fully supports these features. In MLstruct, we apply standard Boolean algebra simplification techniques to simplify these types, such as putting them into disjunctive normal forms, simplifying complements, and factorizing common conjuncts. We also reduce types as they arise, based on Section 2.2.2.

#### 5.5.2 Bound Inlining

Many types can be represented equivalently using either bounded quantification or inlined intersection and union types, so we often have to choose between them. For instance, $\forall(\alpha \leqslant \mathrm{Int})\cdot(\beta \geqslant \mathrm{Int}).\ \alpha \to \alpha \to \beta$ is much better expressed as the equivalent $\mathrm{Int} \to \mathrm{Int} \to \mathrm{Int}$. But whether $(\alpha \wedge \mathrm{Int}) \to (\alpha \wedge \mathrm{Int}) \to \alpha$ is better than the equivalent $\forall(\alpha \leqslant \mathrm{Int}).\ \alpha \to \alpha \to \alpha$ may depend on personal preferences. As a general rule of thumb, we only inline bounds when doing so would not duplicate them and when they are not cyclic (i.e., we do not inline recursive bounds).

### 5.6 Implementation

MLstruct is implemented in ~5000 lines of Scala code, including advanced type simplification algorithms and error reporting infrastructure.[23] We have an extensive tests suite consisting of more than 4000 lines of well-typed *and* ill-typed MLstruct expressions, for which we automatically check the output of the type simplifier and error reporting for regressions. Running this test suite in parallel takes ~2s on a 2020 iMac with a 3.8 GHz 8-Core Intel Core i7 and 32 GB 2667 MHz DDR4.

## 6 Formal Semantics of MLstruct

In this section, we introduce $\lambda^{\neg}$, a formal calculus which reflects the core features of MLstruct.

[23] This does not include about 1200 additional lines of code to generate JavaScript (the tests are run through NodeJS).

Core syntax

$$
\begin{array}{rrcl}
\textit{Type} & \tau, \pi & ::= & \tau \to \tau \mid \{ x : \tau \} \mid N[\overline{\tau}] \mid \#C \mid \alpha \mid \top^{\diamond} \mid \tau \vee^{\diamond} \tau \mid \neg \tau \\
\textit{Mode} & \diamond, \circ & ::= & \cdot \mid \curvearrowright \\
\textit{Type name} & N & ::= & A \mid C \\
\textit{Polymorphic type} & \sigma & ::= & \forall \Xi. \tau \\
\textit{Term} & s, t & ::= & x, y, z \mid t : \tau \mid \lambda x. t \mid t\ t \mid t.x \mid C\ \{ \overline{x = t} \} \mid \textbf{case}\ x = t\ \textbf{of}\ M \\
\textit{Case branches} & M & ::= & \epsilon \mid \_ \to t \mid C \to t,\ M \\
\textit{Value} & v, w & ::= & \lambda x. t \mid C\ \{ \overline{x = v} \} \\
\textit{Program} & P & ::= & t \mid \textbf{def}\ x = t;\ P \\
\textit{Top-level declaration} & d & ::= & \textbf{class}\ C[\overline{\alpha}] : \tau \mid \textbf{type}\ A[\overline{\alpha}] = \tau
\end{array}
$$

Contexts

$$
\begin{array}{rrcl}
\textit{Declarations context} & \mathcal{D} & ::= & \epsilon \mid \mathcal{D} \cdot d \\
\textit{Typing context} & \Gamma & ::= & \epsilon \mid \Gamma \cdot (x : \tau) \mid \Gamma \cdot (x : \sigma) \\
\textit{Subtyping context} & \Sigma, \Delta & ::= & \Xi \mid \Sigma \cdot (\tau \leqslant \tau) \mid \Sigma \cdot \triangleright (\tau \leqslant \tau) \\
\textit{Constraining context} & \Xi & ::= & \epsilon \mid \Xi \cdot (\alpha \leqslant \tau) \mid \Xi \cdot (\tau \leqslant \alpha)
\end{array}
$$

Fig. 13. Syntax of types, terms, and contexts.

### *6.1 Syntax*

The syntax of $\lambda^{\neg}$ is presented in Figure 13.

#### *6.1.1 Core Syntax*

The core syntax of $\lambda^{\neg}$ follows the MLstruct source language presented previously quite closely, though it introduces a syntactic novelty: the *mode* $\diamond$ or $\circ$ of a syntactic form is used to deduplicate sentences that refer to unions and intersections as well as top and bottom, which are respective duals and can therefore often be treated symmetrically. For instance, $\top^{\diamond}$ is to be understood as either $\top^{\cdot}$ when $\diamond = \cdot$, i.e., $\top$, or as $\top^{\curvearrowright}$ when $\diamond = \curvearrowright$, i.e., $\bot$. A similar idea was developed independently by d. S. Oliveira et al. (2020) to cut down on boilerplate and repetition in formalizing subtyping systems.

Parametric polymorphism in $\lambda^{\neg}$ is attached solely to top-level '**def**' bindings, whose semantics, as in languages like Scala, is to re-evaluate their right-hand side every time they are referred to in the program. In contrast, local let bindings are desugared to immediately-applied lambdas, and are treated monomorphically. *Let polymorphism* is orthogonal to the features presented in this paper, and can be handled by using a level-based algorithm (Parreaux, 2020) on top of the core algorithm we describe here, as well as a value restriction if the language is meant to incorporate mutation.

In $\lambda^{\neg}$, **def** bindings are never recursive. This simplification is made without loss of generality, as recursion can be recovered using a Z fixed point combinator, typeable in MLsub (Dolan, 2017) and thus also in $\lambda^{\neg}$. This combinator is defined as $t_Z = \lambda f.\ t'_Z\ t'_Z$ where $t'_Z = \lambda x.\ f\ (\lambda v.\ x\ x\ v)$. One can easily verify that $t_Z$ can be typed as $((\alpha \to \beta) \to ((\alpha \to \beta) \wedge \gamma)) \to \gamma$.

To keep the formalism on point, we only present class object types, and ignore uninteresting primitive and built-in types like $\mathrm{Int}$ and $\mathrm{Bool}$, which can be encoded as classes. Note that singleton types like 1, 2, and **true**, as we use them in the introduction, are easily encoded as subclasses $1_C$, $2_C$, and $\mathbf{true}_C$ of the corresponding built-in types.

Finally, the syntax of pattern matching '**case** $x = t$ **of** . . .' includes a variable binding because the rules for typing it will *refine* the type of that variable in the different branches. We do not use '**case** $x$ **of** . . .' as the core form in order to allow for simple substitution of variables with terms.

### 6.1.2 Contexts

We use four kinds of contexts. Declarations contexts $\mathcal{D}$ hold the type declarations of the program. Throughout this paper, **we assume an ambient declarations context** (i.e., our formal developments are implicitly parameterized by $\mathcal{D}$). Typing contexts $\Gamma$ bind both *monomorphic* and *polymorphic* types, the latter corresponding to '**def**' bindings. Subtyping contexts $\Sigma$ record assumptions about subtyping relationships, with some of these assumptions potentially hidden behind a $\triangleright$ (explained in Section 3.3.1). Finally, *bounds* contexts $\Xi$ contain bounds on type variables. The typing rules will ensure that in a polymorphic type $\forall \Xi.\, \tau$, context $\Xi$ is *consistent*, which implies $\boldsymbol{err} \notin \Xi$. Note that $\Sigma$ contexts are *rooted* in $\Xi$ contexts because subtyping judgments require the former but are invoked from typing judgments, which use the latter for polymorphism. While this rooting is not strictly required here (indeed, bounds contexts would be a strict subset of subtyping contexts even if the latter was not rooted in the former), it will become convenient once we extend bounds context to also possibly contain an error marker in Secsec:constr-ctxs.

### 6.1.3 Shorthands

Throughout this paper, we make use of the following notations and shorthands:

$$R ::= \{\, \overline{x = v} \,\} \qquad H ::= \tau \leqslant \tau \qquad N \equiv N[\epsilon] \qquad C \to t \equiv C \to t, \epsilon$$

$$\{\, \overline{x : \tau_x}^{\,x \in S},\ y : \tau_y \,\} \equiv \{\, \overline{x : \tau_x}^{\,x \in S} \,\} \wedge \{\, y : \tau_y \,\} \quad (y \notin S) \qquad \textbf{let } x = t_1 \textbf{ in } t_2 \equiv (\lambda x.\, t_2)\, t_1$$

$$\textbf{case } y \textbf{ of } M \equiv \textbf{case } x = y \textbf{ of } [y \mapsto x]M \quad (x \notin FV(M))$$

## 6.2 Evaluation Rules

The small-step reduction semantics of $\lambda^{\neg}$ is shown in Figure 14. The relation $P \rightsquigarrow P'$ reads "program $P$ evaluates to program $P'$ in one step." Note that $P$ here may refer to a simple term $t$.

We write $\{\, x = v_2 \,\} \in v_1$ to say that $v_1$ is a value of the form '$C\ \{\, \overline{z = w},\ x = v_2 \,\}$' or of the form '$C\ \{\, \overline{z = w},\ y = v_2' \,\}$' where $y \neq x$ and $\{\, x = v_2 \,\} \in C\ \{\, \overline{z = w} \,\}$. Class instances are constructed via the $C\ R$ introduction form, where $R$ is a record of the fields of the instance. Instance matching works by inspecting the runtime instance of a scrutinee value, in order to determine which corresponding branch to evaluate. This is done through the *superclasses* function $\mathcal{S}(\tau)$. Note that a term of the shape '**case** $x = v$ **of** $\epsilon$' is stuck.

$$E[\square] ::= \square\ t \mid v\ \square \mid \square.x \mid C\,\{\overline{x = v},\ y = \square,\ \overline{z = t}\} \mid \textbf{case}\ x = \square\ \textbf{of}\ M$$

| | | |
|---|---|---|
| E-Ctx | $E[t] \rightsquigarrow E[t']$ | if $t \rightsquigarrow t'$ |
| E-Def | $\textbf{def}\ x = t;\ P \rightsquigarrow [x \mapsto t]P$ | |
| E-App | $(\lambda x.\ t)\ v \rightsquigarrow [x \mapsto v]t$ | |
| E-Asc | $t : \tau \rightsquigarrow t$ | |
| E-Proj | $v_1.x \rightsquigarrow v_2$ | if $\{ x = v_2 \} \in v_1$ |
| E-CaseCls1 | $\textbf{case}\ x = C_1\ R\ \textbf{of}\ C_2 \to t,\ M \rightsquigarrow [x \mapsto C_1\ R]t$ | if $C_2 \in \mathcal{S}(\#C_1)$ |
| E-CaseCls2 | $\textbf{case}\ x = C_1\ R\ \textbf{of}\ C_2 \to t,\ M \rightsquigarrow \textbf{case}\ x = v\ \textbf{of}\ M$ | if $C_2 \notin \mathcal{S}(\#C_1)$ |
| E-CaseWld | $\textbf{case}\ x = v\ \textbf{of}\ \_ \to t \rightsquigarrow [x \mapsto v]t$ | |

Fig. 14. Small-step evaluation rules.

**Definition 6.1** (Superclasses)**.** *We define the superclasses $\mathcal{S}(\tau)$ of a type $\tau$ as the set of classes transitively inherited by type $\tau$, assuming $\tau$ is a class type or the expansion of a class type. The full definition is given in appendix (Definition B.1).*

### *6.3 Declarative Typing Rules*

*Program*-typing judgments $\Xi, \Gamma \vdash^\star P : \tau$ are used to type programs while *term*-typing judgments $\Xi, \Gamma \vdash t : \tau$ are used to type **def** right-hand sides and program bodies. The latter judgement is read "under type variable bounds $\Xi$ and in context $\Gamma$, term $t$ has type $\tau$." We present only the rules for the latter judgment in Figure 15, as they are the more interesting ones, and relegate the auxiliary *program-typing* ($\Xi, \Gamma \vdash^\star P : \tau$), *consistency* ($\Sigma$ ***cons.***) and *subtyping entailment* ($\Sigma \vdash \sigma \leqslant^\forall \sigma$ and $\Sigma \models \Sigma$) rules to the appendix (Appendix B.1). The consistency judgment is used to make sure we type `def`s and program bodies under *valid* (i.e., consistent) bounds only.[24]

Rule T-Obj features a few technicalities deserving of careful explanations. First, notice that its result type is an intersection of the nominal class tag $\#C$ with a record type of all the fields passed in the instantiation. Importantly, these fields may have any types, including ones not compatible with the field declarations in $C$ or its parents. This simplifies the meta theory (especially type inference) and is done without loss of generality: indeed, we can *desugar* '`C {x = t, ...}`' instantiations in MLstruct into a type-ascribed instantiation '$C\{x = t, \ldots\} : C[\overline{\alpha}]$' in $\lambda^\neg$,[25] where all $\overline{\alpha}$ are fresh, which will ensure that the provided fields satisfy their declared types in $C$.

T-Obj also requires $C$ to be "final" using the $C$ ***final*** judgment (formally defined in Figure 26). This means that $C$ is not extended by any other classes in $\mathcal{D}$. It ensures that, at runtime, for every class pattern $D$, pattern-matching scrutinees are always instances of a class $D'$ that is either a subclass of $D$ (meaning $\#D' \leqslant \#D$) or an unrelated class (meaning

[24] Indeed, under inconsistent bounds, ill-typed terms become typeable. For example, we have $(\mathrm{Int} \leqslant \mathrm{Int} \to \mathrm{Int}) \vdash 1\ 1 : \mathrm{Int}$.

[25] The alternative desugaring '**let** $tmp = C\{x = t, \ldots\}$ **in let** $\_ = tmp : C[\overline{\alpha}]$ **in** $tmp$' is nicer because it allows the user to retain refined field types (as described in Section 2.1.2) as well as any new fields that were not declared in $C$ or its parents.

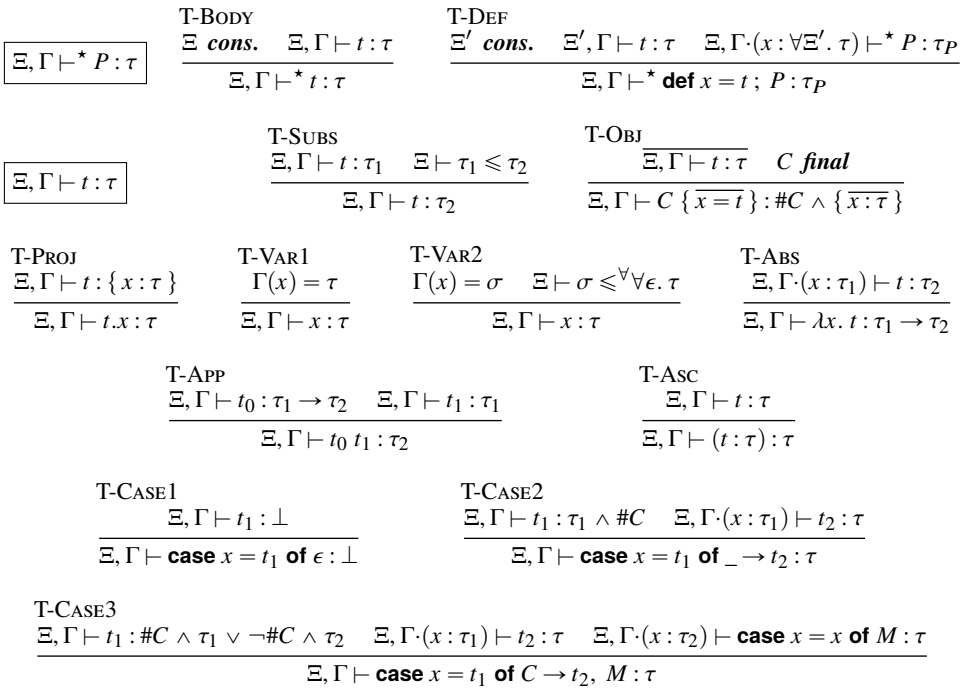


Fig. 15. Program and term typing rules.

$\#D' \leqslant \neg\#D$). Without this property, type preservation would technically not hold. Indeed, consider the program:

```
class C1   class C2: C1   class C3
case x = C1{} of C2 → C3{}, _ →x
```

This program can be given type $\neg C_2$ since $C_1 \leqslant C_2 \vee \neg C_2 \equiv \top$ (in T-Case3, we pick $\tau_2 = \neg C_2$), but it reduces to $C_1\{\}$, which does *not* have type $\neg C_2$ because $C_1$ and $C_2$ are *not* unrelated classes.

This finality requirement is merely a technicality of $\lambda^\neg$ and it does not exist in MLstruct, where non-final classes can be instantiated. This can be understood as each MLstruct class $C$ implicitly defining a final version $C^F$ of itself, which is used upon instantiation. So the MLstruct program above would actually denote the following desugared $\lambda^\neg$ program:

```
class C1   class C1F: C1   class C2: C1   class C3   class C3F: C3
case x = C1F{} : C1 of C2 → C3F{} : C3, _ →x
```

The refined program above now evaluates to $C_1^F\{\}$, of type $C_1^F$, which *is* a subtype of $\neg C_2$.

In T-Subs, we use the current constraining context $\Xi$ as a subtyping context $\Sigma$ when invoking the subtyping judgement $\Xi \vdash \tau_1 \leqslant \tau_2$ (presented in the next subsection), which is possible since the syntax of constraining contexts is a special case of the syntax of subtyping contexts.

Rule T-Var2 uses the *entailment* judgment $\Xi \vdash \sigma \leqslant^\forall \forall \epsilon.\ \tau$ defined in appendix to *instantiate* the polymorphic type found in the context.

The typing of instance matching is split over three rules. Rule T-Case1 specifies that no scrutinee can be matched by a **case** expression with no branches, which is expressed by assigning type $\bot$ (the type inhabited by no value) to the scrutinee.

Rule T-Case2 handles **case** expressions with a single, default case, which is equivalent to a let binding, where the body $t_2$ of the default case is typed within a typing context extended with the case-bound variable $x$ and the type of the scrutinee. This rule requires the scrutinee to have a class type $\#C$; this is to prevent functions from being matched, because that would technically break preservation in a similar way as described above (since we *do not* have $\pi_1 \to \pi_2 \leqslant \neg\#D$[26]).

T-Case3 is the more interesting instance matching rule. We first assume that the scrutinee $t_1$ has some type $\tau_1$ in order to type the first **case** branch, and then assume $t_1$ has type $\tau_2$ to type the rest of the instance matching (by reconstructing a smaller **case** expression binding a new variable $x$ which shadows the old variable occurring in $M$). Then, we make sure that the scrutinee $t_1$ can be typed at $\#C \wedge \tau_1 \vee \neg\#C \wedge \tau_2$, which ensures that if $t_1$ is an instance of $C$, then it is also of type $\tau_1$, and if not, then it is of type $\tau_2$. In this rule, $\tau_1$ can be picked to be anything, so assuming $\Gamma{\cdot}(x : \tau_1)$ to type $t_2$ is sufficient, and there is no need to assume $\Gamma{\cdot}(x : \tau_1 \wedge \#C)$. If the $t_2$ branch needs $\tau_1$ to be a subtype of $\#C$, we can always pick $\tau_1 = \tau_1' \wedge \#C$. Notice that the required type for $t_1$ still has the same shape $\#C \wedge \tau_1 \vee \neg\#C \wedge \tau_2 \equiv \#C \wedge (\#C \wedge \tau_1') \vee \neg\#C \wedge \tau_2 \equiv \#C \wedge \tau_1' \vee \neg\#C \wedge \tau_2$.

### *6.4 Declarative Subtyping Rules*

The declarative subtyping rules were already mostly presented as the runnign example of Section 3. They are solidified and recalled in Figure 16.

Remember that the mode syntax $\diamond$ is used to factor in dual formulations. For instance, $\tau \leqslant^{\diamond} \top^{\diamond}$ is to be understood as either $\tau \leqslant \top$ when $\diamond = \cdot$, i.e., $\tau \leqslant \top$, or as $\tau \leqslant^{\curvearrowright} \top^{\curvearrowright}$ when $\diamond = \curvearrowright$, i.e., $\tau \geqslant \bot$, also written $\bot \leqslant \tau$.

The presented rules extend those of $\mathfrak{S}(\mathcal{T}, \mathcal{R})$ shown in Figure 3, except that the depth subtyping and merge rules are specialized to the type constructors of $\lambda^{\neg}$ and that we have a new rule S-Exp$\diamond$ which is used to expand named types (type synonyms and class types).

#### *6.4.1 Desugaring Named Types*

The reason we did not present the $N[\overline{\tau}]$ type form and the S-Exp$\diamond$ rule as being a core part of $\mathfrak{S}_{\to\{x\}\#C}$ in Section 3 is that these can easily be *desugared* into core $\mathfrak{S}_{\to\{x\}\#C}$.

To do this, observe that the regularity requirement on $\lambda^{\neg}$ (and MLstruct) type definitions means that there is always only a finite number of type argument lists $\overline{\tau}$ passed in named type applications $N[\overline{\tau}]$, so we can represent each such application through a unique type variable $\alpha_{N[\overline{\tau}]}$ associated to it.

We then simply type all definitions and the body of the program under consideration by including a bounds context $\Xi_{\text{init}}$ that equates each such named type application type variable $\alpha_{N[\overline{\tau}]}$ to the corresponding expanded body of the type, here $[\overline{\beta \mapsto \tau}]\pi$ if we assume, for example, that $N$ was defined as **type** $N[\overline{\beta}] = \pi$.

[26] We *cannot* support this without breaking subtyping consistency, because it would mean that $\#C \wedge (\tau_1 \to \tau_2) \leqslant \ldots$

This context $\Xi_{\text{init}}$ essentially makes all named type applications type variables equivalent to the expansions of the corresponding type applications in the type system, which is the intent of the original conceived named types and their S-Exp$\diamond$ rule.

### *6.5 Soundness of the Declarative Type System*

We now state the main soundness theorems for $\lambda^{\neg}$'s type system, proven in Section C.1 and C.2. In the following, $\vdash^{\star}$ is used as the syntax for *program*-typing judgments (see Figure 25 in appendix).

**Theorem 6.2** (Progress). *If $\vdash^{\star} P : \tau$ and $P$ is not a value, then $\vdash P \rightsquigarrow P'$ for some $P'$.*

**Theorem 6.3** (Preservation). *If $\vdash^{\star} P : \tau$ and $\vdash P \rightsquigarrow P'$, then $\vdash^{\star} P' : \tau$.*

$\boxed{\Sigma \vdash \tau \leqslant \tau}$ $\boxed{\tau \leqslant \tau}$ $\quad \triangleleft\Xi = \Xi \quad \triangleleft(\Sigma \cdot H) = \triangleleft\Sigma \cdot H \quad \triangleleft(\Sigma \cdot \triangleright H) = \triangleleft\Sigma \cdot H$

S-Refl $\dfrac{}{\tau \leqslant \tau}$ S-ToB$\diamond$ $\dfrac{}{\tau \leqslant^{\diamond} \top^{\diamond}}$ S-Compl$\diamond$ $\dfrac{}{\tau \vee^{\diamond} \neg\tau \geqslant^{\diamond} \top^{\diamond}}$ S-AndOr11$\diamond$ $\dfrac{}{\tau_1 \vee^{\diamond} \tau_2 \geqslant^{\diamond} \tau_1}$ S-AndOr12$\diamond$ $\dfrac{}{\tau_1 \vee^{\diamond} \tau_2 \geqslant^{\diamond} \tau_2}$

S-AndOr2$\diamond$ $\dfrac{\Sigma \vdash \tau \geqslant^{\diamond} \tau_1 \quad \Sigma \vdash \tau \geqslant^{\diamond} \tau_2}{\Sigma \vdash \tau \geqslant^{\diamond} \tau_1 \vee^{\diamond} \tau_2}$ S-Distrib$\diamond$ $\dfrac{}{\tau \wedge^{\diamond} (\tau_1 \vee^{\diamond} \tau_2) \leqslant^{\diamond} (\tau \wedge^{\diamond} \tau_1) \vee^{\diamond} (\tau \wedge^{\diamond} \tau_2)}$

S-Trans $\dfrac{\Sigma \vdash \tau_0 \leqslant \tau_1 \quad \Sigma \vdash \tau_1 \leqslant \tau_2}{\Sigma \vdash \tau_0 \leqslant \tau_2}$ S-Weaken $\dfrac{H}{\Sigma \vdash H}$ S-Assum $\dfrac{\Sigma \cdot \triangleright H \vdash H}{\Sigma \vdash H}$ S-Hyp $\dfrac{H \in \Sigma}{\Sigma \vdash H}$ S-ClsSub $\dfrac{C_2 \in \mathcal{S}(\#C_1)}{\#C_1 \leqslant \#C_2}$

S-ClsBot $\dfrac{C_1 \notin \mathcal{S}(\#C_2) \quad C_2 \notin \mathcal{S}(\#C_1)}{\#C_1 \wedge \#C_2 \leqslant \bot}$ S-FunDepth $\dfrac{\triangleleft\Sigma \vdash \tau_0 \leqslant \tau_1 \quad \triangleleft\Sigma \vdash \tau_2 \leqslant \tau_3}{\Sigma \vdash \tau_1 \to \tau_2 \leqslant \tau_0 \to \tau_3}$

S-FunMrg$\diamond$ $\dfrac{}{(\tau_1 \vee^{\diamond} \tau_3) \to (\tau_2 \wedge^{\diamond} \tau_4) \geqslant^{\diamond} \tau_1 \to \tau_2 \wedge^{\diamond} \tau_3 \to \tau_4}$ S-Exp$\diamond$ $\dfrac{\tau \ \textbf{\textit{exp.}} \ \tau'}{\tau \geqslant^{\diamond} \tau'}$ S-RcdDepth $\dfrac{\triangleleft\Sigma \vdash \tau_1 \leqslant \tau_2}{\Sigma \vdash \{x : \tau_1\} \leqslant \{x : \tau_2\}}$

S-RcdMrg$\diamond$ $\dfrac{}{\{x : \tau_1 \vee^{\diamond} \tau_2\} \leqslant^{\diamond} \{x : \tau_1\} \vee^{\diamond} \{x : \tau_2\}}$ S-RcdTop $\dfrac{\tau \in \{\{y^{\neq x} : \tau_2\}, \tau_2 \to \tau_3\}}{\top \leqslant \{x : \tau_1\} \vee \tau}$

$\boxed{\tau \ \textbf{\textit{exp.}} \ \tau}$ S-AlsExp $\dfrac{(\textbf{type } A[\overline{\alpha_i}^{\,i \in S}] = \tau) \in \mathcal{D}}{A[\overline{\tau_i}^{\,i \in S}] \ \textbf{\textit{exp.}} \ [\overline{\alpha_i \mapsto \tau_i}^{\,i \in S}]\tau}$ S-ClsExp $\dfrac{(\textbf{class } C[\overline{\alpha_i}^{\,i \in S}] : \tau) \in \mathcal{D}}{C[\overline{\tau_i}^{\,i \in S}] \ \textbf{\textit{exp.}} \ \#C \wedge [\overline{\alpha_i \mapsto \tau_i}^{\,i \in S}]\tau}$

Fig. 16. Declarative subtyping rules. These are essentially the same as the rules of Figure 3 but specialized to the type constructors of $\lambda^{\neg}$.

$\boxed{\Gamma \Vdash^\star P : \tau \Rightarrow \Xi}$

$$\textsc{I-Body}\ \frac{\Gamma \Vdash t : \tau \Rightarrow \Xi}{\Gamma \Vdash^\star t : \tau \Rightarrow \Xi} \qquad \textsc{I-Def}\ \frac{\Gamma \Vdash t : \tau \Rightarrow \Xi \quad \Gamma{\cdot}(x : \forall \Xi.\ \tau) \Vdash^\star P : \pi \Rightarrow \Xi'}{\Gamma \Vdash^\star \textbf{def}\ x = t\ ;\ P : \pi \Rightarrow \Xi'}$$

$\boxed{\Xi, \Gamma \Vdash t : \tau \Rightarrow \Xi}$

$$\textsc{I-Proj}\ \frac{\Xi_0, \Gamma \Vdash t : \tau \Rightarrow \Xi_1 \quad \alpha\ \textbf{\textit{fresh}} \quad \Xi_0{\cdot}\Xi_1 \vdash \tau \ll \{\, x : \alpha \,\} \Rightarrow \Xi_2}{\Xi_0, \Gamma \Vdash t.x : \alpha \Rightarrow \Xi_1{\cdot}\Xi_2}$$

$$\textsc{I-Obj}\ \frac{\begin{array}{c}\Xi_0, \Gamma \Vdash t_1 : \tau_1 \Rightarrow \Xi_1 \\ \Xi_0{\cdot}\Xi_1, \Gamma \Vdash t_2 : \tau_2 \Rightarrow \Xi_2 \quad \ldots \quad \Xi_0{\cdot}\Xi_1{\cdot}\ldots{\cdot}\Xi_{n-1}, \Gamma \Vdash t_n : \tau_n \Rightarrow \Xi_n \quad C\ \textbf{\textit{final}}\end{array}}{\Xi_0, \Gamma \Vdash C\ \{\, x_1 = t_1;\ x_2 = t_2;\ \ldots;\ x_n = t_n \,\} : \#C \wedge \{\, x_1 : \tau_1;\ x_2 : \tau_2;\ \ldots;\ x_n : \tau_n \,\} \Rightarrow \Xi_1{\cdot}\ldots{\cdot}\Xi_n}$$

$$\textsc{I-Var1}\ \frac{\Gamma(x) = \tau}{\Xi, \Gamma \Vdash x : \tau \Rightarrow \epsilon} \qquad \textsc{I-Var2}\ \frac{\Gamma(x) = \forall \Xi_1.\ \tau_1 \quad TV(\forall \Xi_1.\ \tau_1) = S \quad \overline{\gamma_\alpha\ \textbf{\textit{fresh}}}^{\alpha \in S}}{\Xi_0, \Gamma \Vdash x : [\overline{\alpha \mapsto \gamma_\alpha}^{\alpha \in S}]\tau_1 \Rightarrow [\overline{\alpha \mapsto \gamma_\alpha}^{\alpha \in S}]\Xi_1}$$

$$\textsc{I-Abs}\ \frac{\alpha\ \textbf{\textit{fresh}} \quad \Xi_0, \Gamma{\cdot}(x : \alpha) \Vdash t : \tau \Rightarrow \Xi_1}{\Xi_0, \Gamma \Vdash \lambda x.\ t : \alpha \to \tau \Rightarrow \Xi_1} \qquad \textsc{I-App}\ \frac{\Xi_0, \Gamma \Vdash t_1 : \tau_1 \Rightarrow \Xi_1 \quad \Xi_0{\cdot}\Xi_1, \Gamma \Vdash t_2 : \tau_2 \Rightarrow \Xi_2 \quad \alpha\ \textbf{\textit{fresh}} \quad \Xi_0{\cdot}\Xi_1{\cdot}\Xi_2 \vdash \tau_1 \ll \tau_2 \to \alpha \Rightarrow \Xi_3}{\Gamma,\ \Xi_0 \Vdash t_1\ t_2 : \alpha \Rightarrow \Xi_1{\cdot}\Xi_2{\cdot}\Xi_3}$$

$$\textsc{I-Asc}\ \frac{\Xi_0, \Gamma \Vdash t : \tau_1 \Rightarrow \Xi_1 \quad \Xi_0{\cdot}\Xi_1 \vdash \tau_1 \ll \tau_2 \Rightarrow \Xi_2}{\Xi_0, \Gamma \Vdash (t : \tau_2) : \tau_2 \Rightarrow \Xi_1{\cdot}\Xi_2} \qquad \textsc{I-Case1}\ \frac{\Xi_0, \Gamma \Vdash t_1 : \tau_1 \Rightarrow \Xi_1 \quad \Xi_0{\cdot}\Xi_1 \vdash \tau_1 \ll \bot \Rightarrow \Xi_2}{\Xi_0, \Gamma \Vdash \textbf{case}\ x = t_1\ \textbf{of}\ \epsilon : \bot \Rightarrow \Xi_1{\cdot}\Xi_2}$$

$$\textsc{I-Case2}\ \frac{\Xi_0, \Gamma \Vdash t_1 : \tau_1 \Rightarrow \Xi_1 \quad \Xi_0{\cdot}\Xi_1 \vdash \tau_1 \ll \#C \Rightarrow \Xi_2 \quad \Xi_0{\cdot}\Xi_1{\cdot}\Xi_2, \Gamma{\cdot}(x : \tau_1) \Vdash t_2 : \tau \Rightarrow \Xi_3}{\Xi_0, \Gamma \Vdash \textbf{case}\ x = t_1\ \textbf{of}\ \_ \to t_2 : \tau \Rightarrow \Xi_1{\cdot}\Xi_2{\cdot}\Xi_3}$$

$$\textsc{I-Case3}\ \frac{\begin{array}{c}\Xi_0, \Gamma \Vdash t_1 : \tau_1 \Rightarrow \Xi_1 \quad \alpha\ \textbf{\textit{fresh}} \\ \Xi_0{\cdot}\Xi_1, \Gamma{\cdot}(x : \alpha) \Vdash t_2 : \tau_2 \Rightarrow \Xi_2 \quad \beta\ \textbf{\textit{fresh}} \quad \Xi_0{\cdot}\Xi_1{\cdot}\Xi_2, \Gamma{\cdot}(x : \beta) \Vdash \textbf{case}\ x = x\ \textbf{of}\ M : \tau_3 \Rightarrow \Xi_3 \\ \Xi_0{\cdot}\Xi_1{\cdot}\Xi_2{\cdot}\Xi_3 \vdash \tau_1 \ll \#C \wedge \alpha \vee \neg\#C \wedge \beta \Rightarrow \Xi_4\end{array}}{\Xi_0, \Gamma \Vdash \textbf{case}\ x = t_1\ \textbf{of}\ C \to t_2,\ M : \tau_2 \vee \tau_3 \Rightarrow \Xi_1{\cdot}\Xi_2{\cdot}\Xi_3{\cdot}\Xi_4}$$

Fig. 17. Algorithmic type inference rules.

## 7 Principal Type Inference for $\lambda^\neg$

We now formally describe the type inference algorithm which was presented in Section 5.

In this section, we assume that bounds contexts $\Xi$ are refined to also potentially contain *error* markers ***err*** as shown below. We call this refined class of contexts *constraining contexts*.

*Constraining context* $\qquad \Xi ::= \epsilon \mid \Xi \cdot (\alpha \leqslant \tau) \mid \Xi \cdot (\tau \leqslant \alpha) \mid \Xi \cdot \textbf{\textit{err}}$

### 7.1 Type Inference Rules

Our type inference rules are presented in Figure 17. The judgments $\Gamma \Vdash^\star P : \tau \Rightarrow \Xi$ and $\Xi, \Gamma \Vdash t : \tau \Rightarrow \Xi$ are similar to their declarative typing counterparts, except that they are *algorithmic* and produce constraining contexts $\Xi$ containing inferred type variables bounds.

We give the following formal meaning to premises of the form '$\alpha$ ***fresh***', and in the rest of this paper, we implicitly only consider well-formed derivations:

**Definition 7.1** (Well-formed derivations)**.** *A type inference or constraining derivation is said to be well-formed if, for every $\alpha$, the '$\alpha$ **fresh**' premise appears at most once in the entire derivation and, if it does, $\alpha$ does not occur in any user-specified type (i.e., on the right of ascription trees '$t : \tau$').*

The *program*-typing inference rules I-Body and I-Def mirror their declarative counterparts. In I-Def, notice how the output context corresponding to the definition's body is the one used to quantify the corresponding type in the typing context. Notice that in these rules, the consistency condition (which can be seen in the declarative typing rules in Figure 25) has disappeared, because type inference only produces consistent contexts by design.

The main difference between type inference rules and declarative typing rules is that in the former, we immediately produce a type for each subexpression irrelevant of its context, using type variables for local unknowns, and we then use a *constraining* judgement $\Sigma \vdash \tau \ll \pi \Rightarrow \Xi$ (explained in the next subsection) to make sure that the inferred type $\tau$ conforms to the expected type $\pi$ in this context. So whenever we need to guess a type (such as the type of a lambda's parameter in I-Abs), we simply introduce a fresh type variable. As an example, in I-Proj, we infer an unconstrained type $\tau$ for the field projection's prefix $t$, and then make sure that this is a subtype of a record type by constraining $\Xi_0 \vdash \tau \ll \{ x : \alpha \} \Rightarrow \Xi_1$ — where $\Xi_1$ is the output context containing the type variable bounds necessary to make this relationship hold. Rules I-App, I-Asc, I-Case1, I-Case2, and I-Case3 all work according to the same principles, threading the set of constraining contexts currently inferred through the next type inference steps, which is necessary to make sure that all inferred type variable bounds are consistent with each other. Rule I-Var2 *refreshes* all the variables of a type $\forall \Xi.\, \tau$ obtained from the typing context, which includes both variables that occur in the constraining context $\Xi$ as well as those that occur in the underlying type $\tau$, even when some of the latter may not be mentioned in $\Xi$; indeed, in $\lambda^\neg$ all type variables are implicitly quantified.

### 7.2 Reduced Disjunctive Normal Forms

To facilitate constraint solving, it is useful to massage types into a normal form which we call RDNF, for *reduced disjunctive normal form*. This normal form is similar to a classical disjunctive normal form (DNF) except that we reduce all "incompatible" intersections and unions to $\bot$ and $\top$ respectively. Here, *incompatible* means that the type holds no useful information, either because it is inhabited by no value or because it cannot be used meaningfully, as explained in Section 2.2.2.

The syntax of RDNF is given below. It is indexed by a level $n$ and there are two possible levels: level-0 RDNF, written $\mathrm{D}^0$ does not contain any occurrence of class or alias types at the top level (they will have been expanded); whereas level-1 RDNF, written $\mathrm{D}^1$, allows them. **Notation:** we will often write D as a shorthand for $\mathrm{D}^1$ (and similarly for the other indexed syntax forms).

$$\begin{array}{rcl} \mathrm{D}^n & ::= & \bot \mid \mathrm{C}^n \mid \mathrm{D}^n \vee \mathrm{C}^n \\ \mathrm{I}^1 & ::= & \mathrm{I}^0 \mid \mathrm{I}^1 \wedge N[\overline{\mathrm{D}^1}] \\ \mathrm{U}^1 & ::= & \mathrm{U}^0 \mid \mathrm{U}^1 \vee N[\overline{\mathrm{D}^1}] \end{array} \qquad \begin{array}{rcl} \mathrm{C}^n & ::= & \mathrm{I}^n \wedge \neg \mathrm{U}^n \mid \mathrm{C}^n \wedge \alpha \mid \mathrm{C}^n \wedge \neg\alpha \\ \mathrm{I}^0 & ::= & \mathcal{I}^{\mathcal{N}}[\mathcal{N}] \mid \mathcal{I}^{\rightarrow}[\mathcal{F}] \mid \mathcal{I}^{\{\}}[\mathcal{R}] \\ \mathrm{U}^0 & ::= & \bot \mid \mathrm{D}^1 \to \mathrm{D}^1 \mid \{\, x : \mathrm{D}^1 \,\} \mid \mathrm{U}^0 \vee \#C \end{array}$$

where the $\mathcal{I}^{\cdot}$ contexts stand for combinations of nominal tags $\mathcal{N}$, functions $\mathcal{F}$, and records $\mathcal{R}$:

$$\begin{array}{rcl} \mathcal{I}^{\mathcal{N}}[\square] & ::= & \square \wedge \mathcal{F} \wedge \mathcal{R} \\ \mathcal{I}^{\rightarrow}[\square] & ::= & \mathcal{N} \wedge \square \wedge \mathcal{R} \\ \mathcal{I}^{\{\}}[\square] & ::= & \mathcal{N} \wedge \mathcal{F} \wedge \square \end{array} \qquad \begin{array}{rcl} \mathcal{N} & ::= & \top \mid \#C \\ \mathcal{F} & ::= & \top \mid \mathrm{D}^1 \to \mathrm{D}^1 \\ \mathcal{R} & ::= & \top \mid \{\, \overline{x : \mathrm{D}^1} \,\} \end{array} \qquad \begin{array}{rcl} \mathcal{I}[\square] & ::= & \mathcal{I}^{\mathcal{N}}[\square] \mid \mathcal{I}^{\rightarrow}[\square] \mid \mathcal{I}^{\{\}}[\square] \\ \top^3 & ::= & \top \wedge \top \wedge \top \end{array}$$

As an example, '$\mathrm{D}_1 = \#C \wedge \top \wedge \{\, x : \top \,\} \wedge C[\mathrm{Int}, \mathrm{Bool}] \wedge A[\mathrm{Str}] \wedge \neg\bot \wedge \neg\alpha$' is a valid level-1 RDNF, but not a valid level-0 one because $C[\mathrm{Int}, \mathrm{Bool}]$ and $A[\mathrm{Str}]$ occur at the top level and are not expanded, while '$\mathrm{D}_2^n = \top \wedge \top \wedge \{\, x : C[\mathrm{Int}, \mathrm{Bool}] \,\} \wedge \neg\bot$' is well-defined for both $n \in \{0,\ 1\}$.

### *7.2.1 Algorithm*

Figures 18 and 19 give an algorithm to convert types $\tau$ to level-$n$ RDNFs, written $\mathrm{dnf}^n(\tau)$. The task is essentially straightforward, if relatively tedious. Essentially, $\mathrm{dnf}^n$ pushes negations in using DeMorgan laws, distributes intersections over unions, and at the same time ensures that all constructed conjunctions are de-duplicated and as reduced as possible, so that for instance intersections of unrelated classes are reduced to $\bot$ and function and record types are merged with themselves. We write $(\neg)\tau$ as a shorthand for either $\tau$ or $\neg\tau$ (used uniformly in a rule) and make use of auxiliary functions $\mathrm{union}^n(\mathrm{D}^n, \mathrm{D}^n)$ and $\mathrm{inter}^n(\mathrm{D}^n, \mathrm{D}^n)$, which rely on the following context definitions $S^+[\cdot]$ and $S^-[\cdot]$, used to "dig into" the various shapes of $\mathrm{C}^n$ syntaxes:

$$\begin{array}{rcl} S^+[\square] & ::= & \mathcal{I}[\square] \mid S^+[\square] \wedge \alpha \mid S^+[\square] \wedge \neg\alpha \mid S^+[\square] \wedge \neg \mathrm{U} \mid S^+[\square] \wedge N[\overline{\mathrm{D}^1}] \\ S^-[\square] & ::= & S^-[\square] \wedge \alpha \mid S^-[\square] \wedge \neg\alpha \mid \mathrm{I} \wedge \neg S^-[\square] \\ S^-[\square] & ::= & \square \mid S^-[\square] \vee N[\overline{\mathrm{D}^1}] \mid S^-[\square] \vee \#C \mid \mathrm{U} \vee \square \end{array}$$

For example, we can decompose $\mathrm{C}^n = \mathrm{I}^n \wedge \neg((\mathrm{D}_1^n \to \mathrm{D}_2^n) \vee \#C) \wedge \alpha$ as $\mathrm{C}^n = S^-[\mathrm{D}_1^n \to \mathrm{D}_2^n]$ where $S^-[\square] = \mathrm{I}^n \wedge \neg(\square \vee \#C) \wedge \alpha$.

The algorithm is well-defined on well-formed types $\tau$ ***wf***, assuming a well-formed declarations context $\mathcal{D}$ ***wf***. These notions of well-formedness are defined formally in Appendix B.2.

**Lemma 7.2** (Well-Defined dnf)**.** *If $\mathcal{D}$ **wf**, $\tau$ **wf**, and $n \in \{0,\ 1\}$, then $\mathrm{dnf}^n(\tau) = \mathrm{D}^n$ for some $\mathrm{D}^n$.*

**Lemma 7.3** (Correctness of dnf)**.** *For all $\tau$, $n \in \{0,\ 1\}$, and $\mathrm{D}^n = \mathrm{dnf}^n(\tau)$, we have $\tau \equiv \mathrm{D}^n$.*

$$\boxed{\mathrm{dnf}^n(\tau)} : \mathrm{D}^n$$

$$\mathrm{dnf}^n(\top) = \mathrm{dnf}^n(\neg\bot) = \top^3 \wedge \neg\bot \tag{7.1}$$
$$\mathrm{dnf}^n(\bot) = \mathrm{dnf}^n(\neg\top) = \bot \tag{7.2}$$
$$\mathrm{dnf}^n(\alpha) = \top^3 \wedge \neg\bot \wedge \alpha \tag{7.3}$$
$$\mathrm{dnf}^n(\#C) = \#C \wedge \top \wedge \top \wedge \neg\bot \tag{7.4}$$
$$\mathrm{dnf}^n(\tau_1 \to \tau_2) = \top \wedge \mathrm{dnf}^1(\tau_1) \to \mathrm{dnf}^1(\tau_2) \wedge \top \wedge \neg\bot \tag{7.5}$$
$$\mathrm{dnf}^n(\{\, x : \tau \,\}) = \{\, x : \mathrm{dnf}^1(\tau) \,\} \wedge \top \wedge \top \wedge \neg\bot \tag{7.6}$$
$$\mathrm{dnf}^0(N\,[\overline{\tau}]) = \mathrm{dnf}^0(\tau') \qquad \textit{when} \quad N\,[\overline{\tau}] \ \textbf{\textit{exp.}}\ \tau' \tag{7.7}$$
$$\mathrm{dnf}^1(N\,[\overline{\tau}]) = \top^3 \wedge N\,[\,\overline{\mathrm{dnf}^1(\tau)}\,] \wedge \neg\bot \tag{7.8}$$
$$\mathrm{dnf}^n(\tau_1 \wedge \tau_2) = \mathrm{inter}(\mathrm{dnf}^n(\tau_1),\ \mathrm{dnf}^n(\tau_2)) \tag{7.9}$$
$$\mathrm{dnf}^n(\tau_1 \vee \tau_2) = \mathrm{union}(\mathrm{dnf}^n(\tau_1),\ \mathrm{dnf}^n(\tau_2)) \tag{7.10}$$
$$\mathrm{dnf}^n(\neg\alpha) = \top^3 \wedge \neg\bot \wedge \neg\alpha \tag{7.11}$$
$$\mathrm{dnf}^n(\neg\#C) = \top^3 \wedge \neg(\bot \vee \#C) \tag{7.12}$$
$$\mathrm{dnf}^n(\neg\{\, x : \tau \,\}) = \top^3 \wedge \neg\{\, x : \mathrm{dnf}^1(\tau) \,\} \tag{7.13}$$
$$\mathrm{dnf}^n(\neg(\tau_1 \to \tau_2)) = \top^3 \wedge \neg(\mathrm{dnf}^1(\tau_1) \to \mathrm{dnf}^1(\tau_2)) \tag{7.14}$$
$$\mathrm{dnf}^0(\neg N\,[\overline{\tau}]) = \mathrm{dnf}^0(\neg\tau') \qquad \textit{when} \quad N\,[\overline{\tau}] \ \textbf{\textit{exp.}}\ \tau' \tag{7.15}$$
$$\mathrm{dnf}^1(\neg N\,[\overline{\tau}]) = \top^3 \wedge \neg(\bot \vee N\,[\,\overline{\mathrm{dnf}^1(\tau)}\,]) \tag{7.16}$$
$$\mathrm{dnf}^n(\neg(\tau_1 \wedge \tau_2)) = \mathrm{union}(\mathrm{dnf}^n(\neg\tau_1),\ \mathrm{dnf}^n(\neg\tau_2)) \tag{7.17}$$
$$\mathrm{dnf}^n(\neg(\tau_1 \vee \tau_2)) = \mathrm{inter}(\mathrm{dnf}^n(\neg\tau_1),\ \mathrm{dnf}^n(\neg\tau_2)) \tag{7.18}$$

$$\boxed{\mathrm{union}(\mathrm{D}^n,\ \mathrm{D}^n)} : \mathrm{D}^n$$

$$\mathrm{union}(\mathrm{D}^n,\ \bot) = \mathrm{D}^n \tag{7.19}$$
$$\mathrm{union}(\mathrm{D}^n,\ \mathrm{C}^n) = \begin{cases} \mathrm{D}^n & \textit{when}\ \ \mathrm{C}^n \in \mathrm{D}^n \\ \mathrm{D}^n \vee \mathrm{C}^n & \textit{otherwise} \end{cases} \tag{7.20}$$
$$\mathrm{union}(\mathrm{D}_1^n,\ \mathrm{D}_2^n \vee \mathrm{C}^n) = \mathrm{union}(\mathrm{union}(\mathrm{D}_1^n,\ \mathrm{C}^n),\ \mathrm{D}_2^n) \tag{7.21}$$

$$\boxed{\mathrm{inter}(\mathrm{D}^n,\ \mathrm{D}^n)} : \mathrm{D}^n$$

$$\mathrm{inter}(\bot,\ \mathrm{D}^n) = \mathrm{inter}(\mathrm{D}^n,\ \bot) = \bot \tag{7.22}$$
$$\mathrm{inter}(\mathrm{D}_1^n \vee \mathrm{C}^n,\ \mathrm{D}_2^n) = \mathrm{union}(\mathrm{inter}(\mathrm{D}_1^n,\ \mathrm{D}_2^n),\ \mathrm{inter}(\mathrm{C}^n,\ \mathrm{D}_2^n)) \tag{7.23}$$
$$\mathrm{inter}(\mathrm{C}_1^n,\ \mathrm{D}^n \vee \mathrm{C}_2^n) = \mathrm{union}(\mathrm{inter}(\mathrm{C}_1^n,\ \mathrm{D}^n),\ \mathrm{inter}(\mathrm{C}_1^n,\ \mathrm{C}_2^n)) \tag{7.24}$$

Fig. 18. Normal form construction algorithm.

### *7.3 Type Constraining Rules*

The type constraining rules are defined in Figure 20. They are defined for *any* pairs of types and input subtyping contexts, returning an output context containing ***err*** in case the constraining fails. We need ***err*** cases to distinguish an infinite loop in the algorithm from a subtype constraining error, i.e., we want to justify that we have a proper algorithm and not just a semi-algorithm.

In top-level constraining judgments, of the form $\Sigma \vdash \tau \ll \tau \Rightarrow \Xi$, we check whether a subtyping relationship is currently in the assumptions; if not, we extend the set of assumptions with the current constraint (guarded by a $\triangleright$) and call the nested constraining

(7.25)

$$\boxed{\mathrm{inter}(\mathrm{C}^n \mid \bot,\ \mathrm{C}^n \mid \mathrm{I}^n \mid \neg\mathrm{U}^n)} : \mathrm{C}^n \mid \bot$$

$$\mathrm{inter}(\bot,\ \_) = \bot \tag{7.26}$$

$$\mathrm{inter}(\mathrm{C}_1^n,\ \mathrm{C}_2^n \wedge (\neg)\boldsymbol{\alpha}) = \begin{cases} \mathrm{inter}(\mathrm{C}_1^n,\ \mathrm{C}_2^n) & \textit{when } (\neg)\boldsymbol{\alpha} \in \mathrm{C}_1^n \\ \bot & \textit{when } \boldsymbol{\alpha}, \neg\boldsymbol{\alpha} \in \mathrm{C}_1^n \wedge (\neg)\boldsymbol{\alpha} \\ \mathrm{inter}(\mathrm{C}_1^n \wedge (\neg)\boldsymbol{\alpha},\ \mathrm{C}_2^n) & \textit{otherwise} \end{cases} \tag{7.27}$$

$$\mathrm{inter}(\mathrm{C}^n,\ \mathrm{I}^n \wedge \neg\mathrm{U}^n) = \mathrm{inter}(\mathrm{inter}(\mathrm{C}^n,\ \mathrm{I}^n),\ \neg\mathrm{U}^n) \tag{7.28}$$

$$\mathrm{inter}(\mathrm{C}^1,\ \mathrm{I}^1 \wedge N[\,\overline{\mathrm{D}^1}\,]) = \mathrm{inter}(\mathrm{inter}(\mathrm{C}^1,\ \mathrm{I}^1),\ N[\,\overline{\mathrm{D}^1}\,]) \tag{7.29}$$

$$\mathrm{inter}(\mathrm{C}^n,\ \mathcal{N} \wedge \mathcal{F} \wedge \mathcal{R}) = \mathrm{inter}(\mathrm{inter}(\mathrm{inter}(\mathrm{C}^n,\ \mathcal{N}),\ \mathcal{F}),\ \mathcal{R}) \tag{7.30}$$

$$\mathrm{inter}(\mathrm{C}^1,\ \neg(\mathrm{U}^1 \vee N[\,\overline{\mathrm{D}^1}\,])) = \mathrm{inter}(\mathrm{inter}(\mathrm{C}^1,\ \neg\mathrm{U}^1),\ \neg N[\,\overline{\mathrm{D}^1}\,]) \tag{7.31}$$

$$\mathrm{inter}(\mathrm{C}^n,\ \neg\bot) = \mathrm{C}^n \tag{7.32}$$

$$\mathrm{inter}(S^-[\mathrm{U}_1^n],\ \neg\mathrm{U}_2^n) = \top^3 \qquad \textit{when} \quad (\mathrm{U}_1^n,\ \mathrm{U}_2^n) \in \left\{ \begin{array}{l} (\_ \to \_,\ \{\, x : \_ \,\})\,; \\ (\{\, x : \_ \,\},\ \_ \to \_)\,; \\ (\{\, x : \_ \,\},\ \{\, y^{\neq x} : \_ \,\}) \end{array} \right\} \tag{7.33}$$

$$\mathrm{inter}(S^-[\mathrm{D}_1^1 \to \mathrm{D}_2^1],\ \neg(\mathrm{D}_3^1 \to \mathrm{D}_4^1)) = S^-[\mathrm{inter}(\mathrm{D}_1^1,\ \mathrm{D}_3^1) \to \mathrm{union}(\mathrm{D}_2^1,\ \mathrm{D}_4^1)] \tag{7.34}$$

$$\mathrm{inter}(S^-[\{\, x : \mathrm{D}_1^1 \,\}],\ \neg\{\, x : \mathrm{D}_2^1 \,\}) = S^-[\{\, x : \mathrm{union}(\mathrm{D}_1^1,\ \mathrm{D}_2^1) \,\}] \tag{7.35}$$

$$\mathrm{inter}(S^-[\mathrm{U}_1^n],\ \neg(\mathrm{U}_2^n \vee \#\boldsymbol{C})) = \begin{cases} \mathrm{inter}(S^-[\mathrm{U}_1^n],\ \neg\mathrm{U}_2^n) & \textit{when } \#\boldsymbol{C} \in \mathrm{U}_1^n \\ \mathrm{inter}(S^-[\mathrm{U}_1^n \vee \#\boldsymbol{C}],\ \neg\mathrm{U}_2^n) & \textit{otherwise} \end{cases} \tag{7.36}$$

$$\mathrm{inter}(S^-[\bot],\ \neg\mathrm{U}^n) = S^-[\mathrm{U}^n] \tag{7.37}$$

$$\boxed{\mathrm{inter}(\mathrm{D}^1 \mid \mathrm{C}^1,\ (\neg)N[\,\overline{\mathrm{D}^1}\,])} : \mathrm{D}^1$$

$$\mathrm{inter}(\bot,\ (\neg)N[\,\overline{\mathrm{D}^1}\,]) = \bot \tag{7.38}$$

$$\mathrm{inter}(\mathrm{D}_0^1 \vee \mathrm{C}^1,\ (\neg)N[\,\overline{\mathrm{D}^1}\,]) = \mathrm{inter}(\mathrm{D}_0^1,\ (\neg)N[\,\overline{\mathrm{D}^1}\,]) \vee \mathrm{inter}(\mathrm{C}^1,\ (\neg)N[\,\overline{\mathrm{D}^1}\,]) \tag{7.39}$$

$$\mathrm{inter}(\mathrm{C}^1 \wedge \boldsymbol{\alpha},\ (\neg)N[\,\overline{\mathrm{D}^1}\,]) = \mathrm{inter}(\mathrm{C}^1,\ (\neg)N[\,\overline{\mathrm{D}^1}\,]) \wedge \boldsymbol{\alpha} \tag{7.40}$$

$$\mathrm{inter}(\mathrm{C}^1 \wedge \neg\boldsymbol{\alpha},\ (\neg)N[\,\overline{\mathrm{D}^1}\,]) = \mathrm{inter}(\mathrm{C}^1,\ (\neg)N[\,\overline{\mathrm{D}^1}\,]) \wedge \neg\boldsymbol{\alpha} \tag{7.41}$$

$$\mathrm{inter}(\mathrm{I}^1 \wedge \neg\mathrm{U}^1,\ N[\,\overline{\mathrm{D}^1}\,]) = \begin{cases} \mathrm{I}^1 \wedge \neg\mathrm{U}^1 & \textit{when } N[\,\overline{\mathrm{D}^1}\,] \in \mathrm{I}^1 \\ \mathrm{I}^1 \wedge N[\,\overline{\mathrm{D}^1}\,] \wedge \neg\mathrm{U}^1 & \textit{otherwise} \end{cases} \tag{7.42}$$

$$\mathrm{inter}(\mathrm{I}^1 \wedge \neg\mathrm{U}^1,\ \neg N[\,\overline{\mathrm{D}^1}\,]) = \begin{cases} \mathrm{I}^1 \wedge \neg\mathrm{U}^1 & \textit{when } N[\,\overline{\mathrm{D}^1}\,] \in \mathrm{U}^1 \\ \mathrm{I}^1 \wedge \neg(\mathrm{U}^1 \vee N[\,\overline{\mathrm{D}^1}\,]) & \textit{otherwise} \end{cases} \tag{7.43}$$

$$\boxed{\mathrm{inter}(\mathrm{C}^n,\ \mathcal{N} \mid \mathcal{F} \mid \mathcal{R})} : \mathrm{C}^n \mid \bot$$

$$\mathrm{inter}(\mathrm{C}^n,\ \top) = \mathrm{C}^n \tag{7.44}$$

$$\mathrm{inter}(S^+[\mathcal{I}^{\mathcal{N}}[\top]],\ \#\boldsymbol{C}) = S^+[\mathcal{I}^{\mathcal{N}}[\#\boldsymbol{C}]] \tag{7.45}$$

$$\mathrm{inter}(S^+[\mathcal{I}[\#\boldsymbol{C}_1]],\ \#\boldsymbol{C}_2) = \begin{cases} \bot & \textit{when } \boldsymbol{C}_1 \notin \mathcal{S}(\#\boldsymbol{C}_2) \textit{ and } \boldsymbol{C}_2 \notin \mathcal{S}(\#\boldsymbol{C}_1) \\ S^+[\mathcal{I}[\#\boldsymbol{C}_2]] & \textit{when } \boldsymbol{C}_1 \in \mathcal{S}(\#\boldsymbol{C}_2) \\ S^+[\mathcal{I}[\#\boldsymbol{C}_1]] & \textit{when } \boldsymbol{C}_2 \in \mathcal{S}(\#\boldsymbol{C}_1) \end{cases} \tag{7.46}$$

$$\mathrm{inter}(S^+[\mathcal{I}^{\to}[\top]],\ \mathrm{D}_1^1 \to \mathrm{D}_2^1) = S^+[\mathcal{I}^{\to}[\mathrm{D}_1^1 \to \mathrm{D}_2^1]] \tag{7.47}$$

$$\mathrm{inter}(S^+[\mathcal{I}[\mathrm{D}_1^1 \to \mathrm{D}_2^1]],\ \mathrm{D}_3^1 \to \mathrm{D}_4^1) = S^+[\mathcal{I}[\mathrm{union}(\mathrm{D}_1^1,\ \mathrm{D}_3^1) \to \mathrm{inter}(\mathrm{D}_2^1,\ \mathrm{D}_4^1)]] \tag{7.48}$$

$$\mathrm{inter}(\mathrm{C}^n,\ \{\, x : \mathrm{D}_x^1,\ \overline{y : \mathrm{D}_y^1} \,\}) = \mathrm{inter}(\mathrm{inter}(\mathrm{C}^n,\ \{\, x : \mathrm{D}_x^1 \,\}),\ \{\, \overline{y : \mathrm{D}_y^1} \,\}) \tag{7.49}$$

$$\mathrm{inter}(S^+[\mathcal{I}^{\{\}}[\top]],\ \{\, x : \mathrm{D}^1 \,\}) = S^+[\mathcal{I}^{\{\}}[\{\, x : \mathrm{D}^1 \,\}]] \tag{7.50}$$

$$\mathrm{inter}(S^+[\mathcal{I}[\{\, \overline{x : \mathrm{D}_x^1}^{\,x \in S} \,\}]],\ \{\, y : \mathrm{D}^1 \,\}) = \begin{cases} S^+[\mathcal{I}[\{\, \overline{x : \mathrm{D}_x^1}^{\,x \in S \setminus \{y\}},\ y : \mathrm{inter}(\mathrm{D}_y^1,\ \mathrm{D}^1) \,\}]] & \textit{when } y \in S \\ S^+[\mathcal{I}[\{\, \overline{x : \mathrm{D}_x^1}^{\,x \in S},\ y : \mathrm{D}^1 \,\}]] & \textit{otherwise} \end{cases} \tag{7.51}$$

Fig. 19. Normal form construction algorithm (continued).

$\boxed{\Sigma \vdash \tau \ll \tau \Rightarrow \Xi}$

C-Hyp
$$\frac{(\tau_1 \leqslant \tau_2) \in \Sigma}{\Sigma \vdash \tau_1 \ll \tau_2 \Rightarrow \epsilon}$$

C-Assum
$$\frac{(\tau_1 \leqslant \tau_2) \notin \Sigma \qquad \Sigma{\cdot}\triangleright(\tau_1 \leqslant \tau_2) \vdash \mathrm{dnf}^0(\tau_1 \wedge \neg\tau_2) \Rightarrow \Xi}{\Sigma \vdash \tau_1 \ll \tau_2 \Rightarrow \Xi}$$

$\boxed{\Sigma \vdash \mathrm{D}^0 \Rightarrow \Xi}$

C-Or
$$\frac{\Sigma \vdash \mathrm{D}^0 \Rightarrow \Xi \qquad \Xi{\cdot}\Sigma \vdash \mathrm{C}^0 \Rightarrow \Xi'}{\Sigma \vdash \mathrm{D}^0 \vee \mathrm{C}^0 \Rightarrow \Xi{\cdot}\Xi'}$$

C-Bot
$$\frac{}{\Sigma \vdash \bot \Rightarrow \epsilon}$$

C-NotBot
$$\frac{}{\Sigma \vdash \mathrm{I}^0 \wedge \neg\bot \Rightarrow \textbf{\textit{err}}}$$

C-Cls1
$$\frac{C_2 \in \mathcal{S}(\#C_1)}{\Sigma \vdash \mathcal{I}[\#C_1] \wedge \neg(\mathrm{U} \vee \#C_2) \Rightarrow \epsilon}$$

C-Cls2
$$\frac{C_2 \notin \mathcal{S}(\#C_1) \qquad \Sigma \vdash \mathcal{I}[\#C_1] \wedge \neg\mathrm{U} \Rightarrow \Xi}{\Sigma \vdash \mathcal{I}[\#C_1] \wedge \neg(\mathrm{U} \vee \#C_2) \Rightarrow \Xi}$$

C-Cls3
$$\frac{\Sigma \vdash \mathcal{I}^{\mathcal{N}}[\top] \wedge \neg\mathrm{U} \Rightarrow \Xi}{\Sigma \vdash \mathcal{I}^{\mathcal{N}}[\top] \wedge \neg(\mathrm{U} \vee \#C) \Rightarrow \Xi}$$

C-Fun1
$$\frac{\triangleleft\Sigma \vdash \mathrm{D}_3 \ll \mathrm{D}_1 \Rightarrow \Xi \qquad \Xi{\cdot}\triangleleft\Sigma \vdash \mathrm{D}_2 \ll \mathrm{D}_4 \Rightarrow \Xi'}{\Sigma \vdash \mathcal{I}[\mathrm{D}_1 \to \mathrm{D}_2] \wedge \neg(\mathrm{D}_3 \to \mathrm{D}_4) \Rightarrow \Xi{\cdot}\Xi'}$$

C-Fun2
$$\frac{}{\Sigma \vdash \mathcal{I}^{\to}[\top] \wedge \neg(\mathrm{D}_1 \to \mathrm{D}_2) \Rightarrow \textbf{\textit{err}}}$$

C-Rcd1
$$\frac{y \in S \qquad \triangleleft\Sigma \vdash \mathrm{D}_y \ll \mathrm{D} \Rightarrow \Xi}{\Sigma \vdash \mathcal{I}[\{\overline{x : \mathrm{D}_x}^{\,x \in S}\}] \wedge \neg\{\, y : \mathrm{D} \,\} \Rightarrow \Xi}$$

C-Rcd2
$$\frac{y \notin S}{\Sigma \vdash \mathcal{I}[\{\overline{x : \mathrm{D}_x}^{\,x \in S}\}] \wedge \neg\{\, y : \mathrm{D} \,\} \Rightarrow \textbf{\textit{err}}}$$

C-Rcd3
$$\frac{}{\Sigma \vdash \mathcal{I}^{\{\}}[\top] \wedge \neg\{\, x : \mathrm{D} \,\} \Rightarrow \textbf{\textit{err}}}$$

C-Var1
$$\frac{\Sigma{\cdot}(\alpha \leqslant \neg\mathrm{C}) \vdash lb_\Sigma(\alpha) \ll \neg\mathrm{C} \Rightarrow \Xi}{\Sigma \vdash \mathrm{C} \wedge \alpha \Rightarrow \Xi{\cdot}(\alpha \leqslant \neg\mathrm{C})}$$

C-Var2
$$\frac{\Sigma{\cdot}(\mathrm{C} \leqslant \alpha) \vdash \mathrm{C} \ll ub_\Sigma(\alpha) \Rightarrow \Xi}{\Sigma \vdash \mathrm{C} \wedge \neg\alpha \Rightarrow \Xi{\cdot}(\mathrm{C} \leqslant \alpha)}$$

Fig. 20. Normal form constraining rules.

rules with the two sides $\tau_1$ and $\tau_2$ merged into a single $\mathrm{dnf}^0(\tau_1 \wedge \neg\tau_2)$ normal form.[27] Nested constraining judgments have syntax $\Sigma \vdash \mathrm{D}^0 \Rightarrow \Xi$; they implicitly solve the constraint $\mathrm{D}^0 \leqslant \bot$. We can do this because for all $\tau_1$ and $\tau_2$, the subtyping relationship $\Sigma \vdash \tau_1 \leqslant \tau_2$ is formally equivalent to $\Sigma \vdash \tau_1 \wedge \neg\tau_2 \leqslant \bot$. This technique was inspired by Pearce (2013), who also puts constraints into this form to solve subtyping problems involving unions, intersections, and negations. Our constraining rules are deterministic except for C-Var1 and C-Var2. By convention, we always pick C-Var1 in case both can be applied.

The *lb* and *ub* functions are defined in Definition 3.5.

Notice how the C-Var1/2 rules solve tricky constraints involving type variables by moving the rest of a type expression to the other side of the inequality, relying on negation types and on the properties of Boolean algebras (see Theorem A.9). Moreover, C-Var1/2 look up the existing bounds of the type variable being constrained and perform a recursive call to ensure that the new bound is consistent with these existing ones. This is required to ensure we only produce consistent output contexts, and it explains why we have to thread

[27] The real implementation is a little smarter and does not always put the entire constraint into DNF to avoid needless work in common cases. It also uses a mutable cache to reuse previous computations and avoid exponential blowups (Pierce, 2002).

constraining contexts throughout all type inference derivations. As part of this recursive call, we extend the subtyping assumptions context with the bound being recorded. For example, C-Var2 recurses with context $\Sigma{\cdot}(\mathrm{C} \leqslant \alpha)$ instead of just $\Sigma$. This is crucial for two reasons: First, it is possible that new upper bounds $\tau_i$ be recorded for $\alpha$ as part of the recursive call. By adding C to the current lower bounds of $\alpha$ within the recursive call, we make sure that any such new upper bounds $\tau_i$ will be checked against C as part of the resulting $lb_\Sigma(\alpha) \ll \tau_i$ constraining call performed when adding bound $\tau_i$. Second, it is quite common for type inference to result in direct type variable bound cycles, such as $\alpha \leqslant \beta,\ \beta \leqslant \alpha$, which can for instance arise from constraining $\beta \to \beta \leqslant \alpha \to \alpha$. These cycles do not lead to divergence of type inference thanks to the use of $\Sigma{\cdot}(\mathrm{C} \leqslant \alpha)$ instead of $\Sigma$ in the recursive call, ensuring that any constraint resulting from a type variable bound cycle will end up being caught by C-Hyp.

The other constraining rules are fairly straightforward. The "beauty" of the RDNF is that it essentially makes constraint solving with $\lambda^\neg$ types obvious. In each case, there is always an obvious choice to make: either (1) the constraint is unsatisfiable (for example with $\top \leqslant \bot$ in C-NotBot, which yields an ***err***); or (2) the constraint needs to unwrap an irrelevant part of the type to continue (for example with $\mathrm{D}_1 \to \mathrm{D}_2 \leqslant \mathrm{U} \vee \#C$ in C-Cls3, which can be solved iff $\mathrm{D}_1 \to \mathrm{D}_2 \leqslant \mathrm{U}$ itself can be solved, because function types are unrelated to nominal class tags); or (3) we can solve the constraint in an obvious, unambiguous way (for example with $\{\, \overline{x : \mathrm{D}_x}^{\,x \in S} \,\} \leqslant \{\, y : \mathrm{D} \,\}$ where $y \in S$ in C-Rcd1).

Normalizing types deeply (i.e., not solely on the outermost level) makes the termination of constraining (Theorem B.9) straightforward. If we did not normalize nested types and for example merged $\{\, x : \tau_1 \,\} \wedge \{\, x : \tau_2 \,\}$ syntactically as $\{\, x : \tau_1 \wedge \tau_2 \,\}$, constraining recursive types in a way that repetitively merges the same type constructors together could lead to unbounded numbers of equivalent types being constrained, such as $\{\, x : \tau_1 \wedge \tau_1 \wedge \tau_1 \wedge \ldots \,\}$, failing to terminate by C-Hyp.

**Example** Consider the constraint $\tau = \{\, x : \mathrm{Nat},\ y : \mathrm{Nat} \,\} \ll \pi = \{\, x : \mathrm{Int},\ y : \top \,\}$. After adding the pair to the set of hypotheses, C-Assum computes the RDNF $\mathrm{dnf}^0(\tau \wedge \neg\pi) = \{\, x : \mathrm{Nat},\ y : \mathrm{Nat} \,\} \wedge \neg\{\, x : \mathrm{Int} \,\} \vee \{\, x : \mathrm{Nat},\ y : \mathrm{Nat} \,\} \wedge \neg\{\, y : \top \,\}$. Then this constrained type is decomposed into two smaller constrained types $\{\, x : \mathrm{Nat},\ y : \mathrm{Nat} \,\} \wedge \neg\{\, x : \mathrm{Int} \,\}$ and $\{\, x : \mathrm{Nat},\ y : \mathrm{Nat} \,\} \wedge \neg\{\, y : \top \,\}$ by C-Or, and each one is solved individually by C-Rcd1, which requires constraining respectively $\mathrm{Nat} \ll \mathrm{Int}$ and $\mathrm{Nat} \ll \top$. The former yields RDNF $\#\mathrm{Nat} \wedge \neg\#\mathrm{Int}$, which is solved by C-ClsCls1, and the latter yields RDNF $\bot$, which is solved by C-Bot.

### *7.4 Correctness of Type Inference*

We conclude this section by presenting the main correctness lemmas and theorems of type inference.

**Theorem 7.4** (Soundness of type inference)**.** *If the type inference algorithm successfully yields a type for program P, then P has this type. Formally: if* $\Vdash^\star P : \tau \Rightarrow \Xi$ *and* $\mathbf{err} \notin \Xi$*, then* $\Xi \vdash^\star P : \tau$.

**Lemma 7.5** (Sufficiency of Constraining). *Successful type constraining ensures subtyping: if* $\Sigma$ ***cons.*** *and* $\Sigma \vdash \tau \ll \pi \Rightarrow \Xi$ *and* $\mathbf{err} \notin \Xi$*, then* $\Xi{\cdot}\Sigma$ ***cons.*** *and* $\Xi{\cdot}\Sigma \vdash \tau \leqslant \pi$.

**Theorem 7.6** (Constraining Termination). *For all* $\tau, \pi, \mathcal{D}, \Sigma$ ***wf****,* $\Sigma \vdash \tau \ll \pi \Rightarrow \Xi$ *for some* $\Xi$.

**Theorem 7.7** (Completeness of type inference). *If a program P can be typed at type* $\sigma$*, then the type inference algorithm derives a type* $\sigma'$ *such that* $\sigma' \leqslant^{\forall} \sigma$*. Formally: if* $\Xi \vdash^{\star} P : \tau$*, then* $\Vdash^{\star} P : \tau' \Rightarrow \Xi'$ *for some* $\Xi'$ *and* $\tau'$ *where* $\Xi'$ ***cons.*** *and* $\forall \Xi'.\, \tau' \leqslant^{\forall} \forall \Xi.\, \tau$.

In the following lemma, which is crucial for proving the above theorem, $\rho$ refers to type variable substitutions and $\Xi \models \Xi'$ denotes that $\Xi$ entails $\Xi'$ (both defined formally in Appendix C).

**Lemma 7.8** (Completeness of Constraining). *If there is a substitution* $\rho$ *that makes* $\rho(\tau_1)$ *a subtype of* $\rho(\tau_2)$ *in some consistent* $\Xi$*, then constraining* $\tau_1 \ll \tau_2$ *succeeds and only introduces type variable bounds that are entailed by* $\Xi$ *(modulo* $\rho$*).*
*Formally and slightly more generally: if* $\Xi$ ***cons.*** *and* $\Xi \vdash \rho(\tau_1) \leqslant \rho(\tau_2)$ *and* $\Xi \models \rho(\Xi_0)$*, then* $\Xi_0 \vdash \tau_1 \ll \tau_2 \Rightarrow \Xi_1$ *for some* $\Xi_1$ *so that* $\mathbf{err} \notin \Xi_1$ *and* $\Xi \models \rho(\Xi_1)$.

## 8 Related Work

We now relate the different aspects of MLstruct and $\lambda^{\neg}$ with previous work.

**Intersection type systems.** Intersection types for lambda calculus were pioneered by Coppo and Dezani-Ciancaglini (1980); Barendregt et al. (1983), after whom the "BCD" type system is named. BCD has the very powerful "T-$\wedge$-I" rule, stating: if $\Gamma \vdash t : \tau_1$ and $\Gamma \vdash t : \tau_2$, then $\Gamma \vdash t : \tau_1 \wedge \tau_2$. Such systems have the interesting property that typeability coincides with strong normalization (Ghilezan, 1996), making type inference undecidable. Thankfully, we do not need something as powerful as T-$\wedge$-I — instead, we introduce intersections in less general ways (i.e., through T-Obj), and we retain decidability of type inference. Most intersection type systems, including MLstruct and $\lambda^{\neg}$, do admit the following standard BCD subtyping rules given by Barendregt et al.: (1) $\tau_1 \wedge \tau_2 \leqslant \tau_1$; (2) $\tau_1 \wedge \tau_2 \leqslant \tau_2$; and (3) if $\tau_1 \leqslant \tau_2$ and $\tau_1 \leqslant \tau_3$, then $\tau_1 \leqslant \tau_2 \wedge \tau_3$. Some systems use intersection types to encode a form of overloading (Pierce, 1991). However, Smith (1991) showed that ML-style type inference with such a general form of overloading and subtyping is undecidable (more specifically, finding whether inferred sets of constraints are satisfiable is) and proposed *constructor overloading*, a restricted form of overloading with more tractable properties, sufficient to encode many common functions, such as addition on different primitive types as well as vectors of those types. Constructor overloading is eminently compatible with MLstruct and MLscript. Another design decision for intersection systems is whether and how this connective should *distribute* over function types. BCD subtyping states[28] $(\tau \rightarrow$

[28] This rule together with T-$\wedge$-I was shown unsound in the presence of imperative features by Davies and Pfenning (2000).

$\pi_1) \wedge (\tau \to \pi_2) \leqslant \tau \to (\pi_1 \wedge \pi_2)$ and Barbanera et al. (1995) also propose $(\tau_1 \to \pi) \wedge (\tau_2 \to \pi) \leqslant (\tau_1 \vee \tau_2) \to \pi$. Together, these correspond to the minimal relevant logic B+ (Dezani-Ciancaglini et al., 1998). Approaches like that of Pottier (1998*b*) use a *greatest lower bound* connective $\sqcap$ that resembles type intersection $\wedge$ but admits a more liberal rule that generalizes the previous two: $(\tau_1 \to \pi_1) \wedge (\tau_2 \to \pi_2) \leqslant (\tau_1 \vee \tau_2) \to (\pi_1 \wedge \pi_2)$, which we will refer to as *(full) function distributivity*. However, notice that in a system with primitives, full function distributivity is incompatible with T-$\wedge$-I and thus precludes intersection-based overloading.[29]

**Union and intersection types in programming.** Union types are almost as old as intersection types, first introduced by MacQueen et al. (1986),[30] and both have a vast (and largely overlapping) research literature, with popular applications such as refinement types (Freeman and Pfenning, 1991). These types have seen a recent resurgence, gaining a lot of traction both in academia (Muehlboeck and Tate, 2018; Huang and Oliveira, 2021; Castagna et al., 2022; Rehman et al., 2022; Dunfield, 2012; Binder et al., 2022; Alpuim et al., 2017) and in industry,[31] with several industry-grade programming languages like TypeScript, Flow, and Scala 3 supporting them, in addition to a myriad of lesser-known research languages. It is worth noting that many modern type systems with intersection types do not support T-$\wedge$-I in its full generality. For example, in TypeScript, a term can only assume an overloaded intersection type if that term is a function with a list of pre-declared type signatures, and in Scala intersections can only be introduced through inheritance. Unions and intersections have also found uses in program analysis. Palsberg and Pavlopoulou (1998) showed that polyvariant analysis can be related formally to a subtyping system with union, intersection, and recursive types. Unions model sets of abstract values and intersections model each usage of an abstract value. Their system conspicuously does not feature polymorphism, but it is well-known that there is a correspondence between intersection types and polymorphism — a polymorphic type can be viewed as an infinite intersection of all its possible instantiations (Aiken and Wimmers, 1993). Smith and Wang (2000) propose inferring polymorphic types, rather than intersections, for function definitions, which is more flexible and composable as it can process unrelated definitions separately, whereas the approach based solely on intersections is a global process. We believe that having both intersections and polymorphism, as in MLscript, represents the best of both worlds.

**Type inference for unions and intersections.** None of the previous approaches we know have proposed a satisfactory ML-style type inference algorithm for full union and intersection types. By *satisfactory*, we mean that the algorithm should infer principal polymorphic types without backtracking. Earlier approaches used heavily-restricted forms of unions and

[29] For instance, term $id = \lambda x.\ x$ has both types $\mathrm{Int} \to \mathrm{Int}$ and $\mathrm{Bool} \to \mathrm{Bool}$ so by T-$\wedge$-I it would also have type $(\mathrm{Int} \to \mathrm{Int}) \wedge (\mathrm{Bool} \to \mathrm{Bool})$. But by function distributivity and subsumption, this would allow typing $id$ as $(\mathrm{Int} \vee \mathrm{Bool}) \to (\mathrm{Int} \wedge \mathrm{Bool})$ and thus typing $id\ 0$ (which reduces to 0) as $\mathrm{Int} \wedge \mathrm{Bool}$, breaking type preservation.

[30] Funnily, MacQueen et al. reported at the time that "*type-checking difficulties seem to make intersection and union awkward in practice; moreover it is not clear if there are any potential benefits from their use*," clearly not anticipating their enduring popularity.

[31] The first author of this paper has received emails from various people reimplementing Simple-sub (Parreaux, 2020) and wanting to know how to add support for first-class union and intersection types, showing the enduring interest in these.

intersections. For instance, Aiken and Wimmers (1993); Aiken et al. (1994) impose very strict restrictions on negative unions (they must be disjoint) and on positive intersections (they must not have free variables and must be "upward closed"). Trifonov and Smith (1996) go further and restrict intersections to *negative* or *input* positions (those appearing on the right of $\leqslant$ constraints) and unions types to *positive* or *output* positions (those appearing on the left). Pottier (1998*b*); Dolan (2017); Parreaux (2020); Binder et al. (2022) all follow the same idea. In these systems, unions and intersections are not *first-class* citizens: they cannot be used freely in type annotations. Frisch et al. (2008) infer set-theoretic types (see *semantic subtyping* below) for a higher-order language with overloading but do not infer polymorphic types. Castagna et al. (2016) propose a complete polymorphic set-theoretic type inference system, but their types are not principal so their algorithm returns several solutions, which leads to the need for backtracking. It seems this should have severe scalability issues, as the number of possible types for an expression would commonly grow exponentially.[32] Petrucciani (2019) describes ways to reduce backtracking, but recognizes it as fundamentally "unavoidable."

**Negation or complement types.** Negation types have not been nearly as ubiquitous as unions and intersection in mainstream programming language practice and theory, except in the field of *semantic subtyping* (see below). Nevertheless, our use of negation types to make progress while solving constraints is not new — Aiken and Wimmers (1993) were the first to propose using complement types in such a way. However, their complement types are less precise than our negation types,[33] and in their system $\alpha \wedge \tau_1 \leqslant \tau_2$ and $\alpha \leqslant \tau_2 \vee \neg\tau_1$ are *not* always equivalent.

**Recursive types.** Recursive types in the style of MLstruct, where a recursive type is *equivalent* to its unfolding (a.k.a. *equirecursive* types, not to be confused with *iso-recursive* types), have a long history in programming languages research (MacQueen et al., 1986; Amadio and Cardelli, 1993; Abadi and Fiore, 1996; Pierce, 2002; Hosoya et al., 2005; Appel et al., 2007), dating as far back as Morris' thesis, where he conjectured their use under the name of *cyclic types* (Morris, 1969, pp.122–124). Recursive types with subtyping were developed in the foundational work of Amadio and Cardelli (1993) and Brandt and Henglein (1998) gave a coinductive axiomatization of such recursive types. Jim and Palsberg (1999) described a co-inductive formalization of recursive types as arbitrary infinite trees which is more general than approaches like ours, which only allows reasoning about *regular* types. Nevertheless, the algorithms they gave were unsurprisingly restricted to regular types. Gapeyev et al. (2002); Pierce (2002) reconciled the representation as infinite regular trees with the representation as $\mu$ types, and described the standard algorithms to decide the corresponding subtyping relationship. An important aspect of practical recursive type algorithms is that one needs to maintain the cache of discovered subtyping relationships *across recursive calls* to avoid exponential blowup (Gapeyev et al., 2002). Our implementation of MLstruct follows the same principle, as a naive implementation of $\lambda^{\neg}$ would lead to exactly

[32] Hindley-Milner type inference and derived systems like MLsub and MLstruct can also infer types that grow exponentially in some situations, but these mostly occur in pathological cases, and not in common human-written programs.

[33] For example, in their system $\neg(\tau \to \pi)$ is the type of all values that are not functions, *regardless* of $\tau$ and $\pi$.

the same blowup. Also refer to Section 3.3.2 for more parallels between the handling of recursive types in $\lambda^{\neg}$ and previous work.

**Early approaches to subtype inference.** The problem of type inference in the presence of subtyping was kick-started in the 1980s (Mitchell, 1984; Stansifer, 1988; Fuh and Mishra, 1989) and studied extensively in the 1990s (Fuh and Mishra, 1990; Curtis, 1990; Smith, 1991; Aiken and Wimmers, 1993; Kozen et al., 1994; Palsberg et al., 1997; Pottier, 1998*a*,b; Jim and Palsberg, 1999), mostly through the lens of constraint solving on top of Hindley-Milner-style type inference (Hindley, 1969; Milner, 1978; Damas and Milner, 1982). These approaches often involved combinations of record, intersection, union, and recursive types, but as far as we know none proposed an effective (i.e., without backtracking) principal type inference technique for a system with all of these combined. Odersky et al. (1999) gave them a unified account by proposing a general framework called HM(X), where the 'X' stands for a constraint solver to plug into their generic system. While these approaches often claimed a form of principal type inference (also called *minimality*[34]), the *constrained types* they inferred were often large and unwieldy. Beyond inferring constraint sets and ensuring their satisfiability, the related problem of *simplification* to produce more readable and efficiently-processable types was also studied, often by leveraging the connection between regular type trees and finite-state automata (Eifrig et al., 1995; Aiken, 1996; Pottier, 1996, 1998b, 2001; Simonet, 2003). A major stumbling block with all of these approaches was the problem of *non-structural subtyping entailment*[35] (NSSE), which is to decide whether a given type scheme, which consists in a polymorphic type along with its constraints on type variables, subsumes another. Solving this issue is of central importance because it is needed to check implementations against user-provided interfaces and type signatures, and because it provides a foundation from which to derive sound type simplification techniques. However, to this day NSSE remains an open problem, and it is not known whether it is even decidable (Dolan, 2017). Due to these difficulties, interest in this very powerful form of subtyping all but faded in the subsequent decade, in what we interpret as a minor "subtype inference winter." Indeed, many subsequent approaches were developed in reaction to this complexity with the aim of being simpler to reason about (e.g., polymorphic variants — see below).

**Algebraic subtyping.** Approaches like that of Pottier (1998*b*) used a lattice-theoretic construction of types inspired by the connection between types and term automata. Meet $\sqcap$ and join $\sqcup$ operators resembling intersection and union types are used to compactly representing conjunctions of constraints, but these are not *first-class* types, in that they are restricted to appearing respectively in negative and positive positions only. Full function distributivity (defined above, in *intersection type systems*) holds in these approaches due to the lattice structure. Pottier's system still suffered from a lack of complete entailment algorithm due

[34] Some authors like Aiken et al. (1994) make a distinction between a concept of *principality* which is purely syntactic (relating types by a substitution instance relationship) and *minimality* which involve a *semantic* interpretation of types.

[35] "*Non-structural*" here is by opposition to so-called *structural subtyping*, which is a more tractable but heavily restricted form of subtyping that only relates type constructors of identical arities (Palsberg et al., 1997) (precluding, e.g., $\{ x : \tau \} \leqslant \top$).

to NSSE. Dolan (2017); Dolan and Mycroft (2017) later built upon that foundation and proposed an *algebraic* construction of types which allowed breaking free of NSSE and finally enjoying a sound and complete entailment algorithm. Two magical ingredients allowed this to be possible: 1. the definition of "extensible" type semantics based on constructing types as a distributive lattice of coproducts; and 2. a different treatment of type variables than in previous work, representing them as part of the lattice of types and not as unknowns ranging over a set of ground types. In this paper, we in turn build on these foundations, although we only retain the latter innovation, somehow forgoing the "extensible" construction of types.[36] Together with our generalization of the subtyping lattice to a Boolean one by adding negations and with the additional structure we impose on types (such as reducing unions of unrelated records to $\top$), this turns out to be sufficient for allowing principal type inference and decidable entailment (though we only sketched the latter in this paper for lack of space). Ingredient 1 allowed Dolan to show the soundness of his system in a very straightforward way, relying on the property (called Proposition 12 by Dolan (2017)) that any constraint of the form $\bigwedge_i \tau_i \leqslant \bigvee_i \pi_i$ holds iff there is a $k$ such that $\tau_k \leqslant \pi_k$ when all $\tau_i$ have distinct constructors and all $\pi_i$ similarly. By contrast, we allow some intersections of unrelated type constructors to reduce to $\bot$ and some unions of them to $\top$, and we are thus not "extensible" in Dolan's terminology. This is actually desirable in the context of pattern matching, where we want to eliminate impossible cases by making the intersections of unrelated class types empty. It is also needed in order to remove the ambiguity from constraints like $(\tau_1 \to \tau_2) \wedge \{ x : \pi \} \leqslant (\tau_1' \to \tau_2') \vee \{ x : \pi' \}$ which in our system reduces to $(\tau_1 \to \tau_2) \wedge \{ x : \pi \} \leqslant \top$. The present paper also takes heavy inspiration from our earlier operationally-focused take on Dolan's type inference algorithm (Parreaux, 2020). While Dolan shirks from explicitly representing constraints, which he prefers to inline inside types on the fly as $\sqcap$ and $\sqcup$ types, we use an approach closer to the original constrained-types formulation followed by Pottier. Besides being much easier to implement, our approach has other concrete advantages, such as the ability to deal with invariance seamlessly (`class C[A]: {f: A → A}`, which is invariant in `A`, is valid in MLstruct) and a simpler treatment of cyclic type variable constraints.

**Semantic subtyping and set-theoretic types.** The *semantic subtyping* approaches (Frisch et al., 2002, 2008; Castagna et al., 2016; Petrucciani, 2019; Castagna et al., 2022) view types as sets of values which inhabit them and define the subtyping relationship as set inclusion, giving set-based meaning to *union*, *intersection*, and *negation* (or *complement*) connectives. This is by contrast to algebraic subtyping, which may admit subtyping rules that *violate* the set-theoretic interpretation, such as function distributivity, to ensure that the subtyping lattice has desirable algebraic properties. For more detailed discussions contrasting semantic subtyping with other approaches, we refer the reader to Parreaux (2020) and Muehlboeck and Tate (2018).

**Occurrence and flow typing.** Occurrence typing was originally introduced by Tobin-Hochstadt and Felleisen (2008) for Typed Scheme, and was later incorporated into TypeScript and Flow, where it is known as *flow typing*. It allows the types of variables

[36] As discussed in prior work (Parreaux, 2020), we believe the argument for Dolan's notion of extensibility to be rather weak.

to be locally refined based on path conditions encountered in the program. Negation types are pervasive in this context, though they are often only used at the meta-theoretic level. Instance-matching in MLstruct can be understood as a primitive form of occurrence typing in that it refines the types of scrutinee variables in `case` expressions, similarly to the approach of Rehman et al. (2022). Occurrence typing was also recently extended to the semantic subtyping context (Castagna et al., 2021, 2022), where negation types are first-class types. The latter work proposes a powerful type inference approach that can infer overloaded function signatures as intersections types; however, this approach does not support polymorphism and likely does not admit principal types. The idea of simplifying the definition of core object-oriented type languages by using class tags (or *brands*) in addition to structural typing is not new and was notably developed by Jones et al. (2015); Lee et al. (2015).

**Polymorphic records/variants and row polymorphism.** Polymorphic *records* are structurally-typed products whose types admit the usual width and depth subtyping relationships. Their dual, polymorphic *variants*, are another useful language feature (Garrigue, 1998, 2001), used to encode structural sum types. In their simplest expression, polymorphic records (resp. variants) do not support ad-hoc field extension (resp. default match cases). Previous approaches have thus extended polymorphic records and variants with *row polymorphism*, which uses a new kind of variables, named "row" variables, to record the presence and absence of fields (resp. cases) in a given type. Some approaches, like OCaml's polymorphic variants and object types, use row polymorphism exclusively to *simulate* subtype polymorphism, in order to avoid subtyping in the wider languages. However, row polymorphism and subtyping actually complement each other well, and neither is as flexible without the other (Pottier, 1998*b*, Chapter 14.7). There are also techniques for supporting variant and record extensibility through union, intersection, and negation types, as shown by Castagna et al. (2016), who also explain that their system resolves long-standing limitations with OCaml-style row polymorphism. In our system, we solve many (though not all) of these limitations, but we also support principal type inference. It is worth pointing out that OCaml's polymorphic variants (Garrigue, 2001) and related systems based on kinds (Ohori, 1995) lack support for *polymorphic extension* (White, 2015; Gaster and Jones, 1996), whereas MLstruct does (see `mapSome` in the introduction). As a simpler example, `def foo x dflt els = case x of { Apple → dflt | _ → els x }` would be assigned a too restrictive type in OCaml and as a consequence `foo (Banana{}) 0 (fun z → case z of { Banana → 1 })` would not type check (OCaml would complains that the function argument does not handle `Apple`). A more expressive row-polymorphic system exposing row variables to users would support this use case (Rémy, 1994; Gaster and Jones, 1996), but as explained in the introduction, even these have limitations compared to our subtyped unions.

## 9 Conclusion and Future Work

In this paper, we developed a general theory of Boolean-algebraic subtyping called $\mathfrak{S}(\mathcal{T}, \mathcal{R})$ and showed how to prove its soundness, a particularly challenging endeavour given the great

flexibility of the corresponding subtyping rules. We instantiated this theory to $\mathfrak{S}_{\rightarrow\{x\}\#C}$, the subtyping theory of $\lambda^{\neg}$, the core language a new research language called MLstruct. We saw that with MLstruct, polymorphic type inference for first-class union, intersection, and negation types is possible, enabling features such as class-instance matching patterns yielding very precise types, comparable in expressiveness to row-polymorphic variants. We also saw that this type inference approach relies on two crucial aspects of MLstruct's type system, only made possible by our novel Boolean-algebraic approach to subtyping: 1. using the full power of Boolean algebras to normalize types and massage constraints into shapes amenable to constraint solving without backtracking; and 2. approximating some unions and intersections, most notably unions of records and intersections of functions, in a way that does not naturally follow from a typical set-theoretic interpretation of subtyping, in order to remove potential ambiguities during constraint solving without threatening the soundness of the system.

**Future Work.** In the future, we intend to explore more advanced forms of polymorphism present in MLscript, such as first-class and ad-hoc polymorphism, as well as how to remove some of the limitations of regular types, which currently prevent fully supporting object-oriented programming idioms.

**Acknowledgements.** We would like to sincerely thank the anonymous reviewers as well as François Pottier, Didier Rémy, Alan Mycroft, Bruno C. d. S. Oliveira, Andong Fan, and Anto Chen for their constructive and helpful comments on earlier versions of this paper. We are particularly grateful to Stephen Dolan, who gave us some invaluable feedback and mathematical intuitions on the development of this new algebraic subtyping system.

### *Conflicts of Interest*

None.

## References

Martin Abadi and Marcelo P. Fiore. 1996. Syntactic considerations on recursive types. In *Proceedings 11th Annual IEEE Symposium on Logic in Computer Science*. IEEE, 242–252.

Alexander Aiken. 1996. Making set-constraint program analyses scale. In *In CP96 Workshop on Set Constraints*.

Alexander Aiken and Edward L. Wimmers. 1993. Type Inclusion Constraints and Type Inference. In *Proceedings of the Conference on Functional Programming Languages and Computer Architecture* (Copenhagen, Denmark) *(FPCA '93)*. Association for Computing Machinery, New York, NY, USA, 31–41. https://doi.org/10.1145/165180.165188

Alexander Aiken, Edward L. Wimmers, and T. K. Lakshman. 1994. Soft Typing with Conditional Types. In *Proceedings of the 21st ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages* (Portland, Oregon, USA) *(POPL '94)*. Association for Computing Machinery, New York, NY, USA, 163–173. https://doi.org/10.1145/174675.177847

João Alpuim, Bruno C. d. S. Oliveira, and Zhiyuan Shi. 2017. Disjoint Polymorphism. In *Programming Languages and Systems*, Hongseok Yang (Ed.). Springer Berlin Heidelberg, Berlin, Heidelberg, 1–28.

Roberto M. Amadio and Luca Cardelli. 1993. Subtyping Recursive Types. *ACM Trans. Program. Lang. Syst.* 15, 4 (Sept. 1993), 575–631. https://doi.org/10.1145/155183.155231

Andrew W. Appel, Paul-André Melliès, Christopher D. Richards, and Jérôme Vouillon. 2007. A Very Modal Model of a Modern, Major, General Type System. In *Proceedings of the 34th Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages* (Nice, France) *(POPL '07)*. Association for Computing Machinery, New York, NY, USA, 109–122. https://doi.org/10.1145/1190216.1190235

F. Barbanera, M. Dezaniciancaglini, and U. Deliguoro. 1995. Intersection and Union Types: Syntax and Semantics. *Information and Computation* 119, 2 (1995), 202–230. https://doi.org/10.1006/inco.1995.1086

Henk Barendregt, Mario Coppo, and Mariangiola Dezani-Ciancaglini. 1983. A filter lambda model and the completeness of type assignment. *Journal of Symbolic Logic* 48, 4 (1983), 931–940. https://doi.org/10.2307/2273659

David Binder, Ingo Skupin, David Läwen, and Klaus Ostermann. 2022. Structural Refinement Types. In *Proceedings of the 7th ACM SIGPLAN International Workshop on Type-Driven Development (TyDe '22)*. Association for Computing Machinery, New York, NY, USA. https://doi.org/10.1145/3546196.3550163

Michael Brandt and Fritz Henglein. 1998. Coinductive axiomatization of recursive type equality and subtyping. *Fundamenta Informaticae* 33, 4 (1998), 309–338.

Giuseppe Castagna. 2012. *Object-Oriented Programming A Unified Foundation*. Springer Science & Business Media.

Giuseppe Castagna, Victor Lanvin, Mickaël Laurent, and Kim Nguyen. 2021. Revisiting Occurrence Typing. arXiv:1907.05590 [cs.PL]

Giuseppe Castagna, Mickaël Laurent, Kim Nguyundefinedn, and Matthew Lutze. 2022. On Type-Cases, Union Elimination, and Occurrence Typing. *Proc. ACM Program. Lang.* 6, POPL, Article 13 (jan 2022), 31 pages. https://doi.org/10.1145/3498674

Giuseppe Castagna, Tommaso Petrucciani, and Kim Nguyen. 2016. Set-theoretic types for polymorphic variants. In *Proceedings of the 21st ACM SIGPLAN International Conference on Functional Programming (ICFP 2016)*. Association for Computing Machinery, Nara, Japan, 378–391. https://doi.org/10.1145/2951913.2951928

M. Coppo and M. Dezani-Ciancaglini. 1980. An extension of the basic functionality theory for the $\lambda$-calculus. *Notre Dame Journal of Formal Logic* 21, 4 (1980), 685 – 693. https://doi.org/10.1305/ndjfl/1093883253

Pavel Curtis. 1990. *Constrained Qualification in Polymorphic Type Analysis*. Ph.D. Dissertation. USA. UMI Order No. GAX90-26980.

Bruno C. d. S. Oliveira, Cui Shaobo, and Baber Rehman. 2020. The Duality of Subtyping. In *34th European Conference on Object-Oriented Programming (ECOOP 2020) (Leibniz International Proceedings in Informatics (LIPIcs), Vol. 166)*, Robert Hirschfeld and Tobias Pape (Eds.). Schloss Dagstuhl–Leibniz-Zentrum für Informatik, Dagstuhl, Germany, 29:1–29:29. https://doi.org/10.4230/LIPIcs.ECOOP.2020.29

Luis Damas and Robin Milner. 1982. Principal type-schemes for functional programs. In *Proceedings of the 9th ACM SIGPLAN-SIGACT symposium on Principles of programming languages (POPL '82)*. Association for Computing Machinery, Albuquerque, New Mexico, 207–212. https://doi.org/10.1145/582153.582176

Rowan Davies and Frank Pfenning. 2000. Intersection Types and Computational Effects. In *Proceedings of the Fifth ACM SIGPLAN International Conference on Functional Programming (ICFP '00)*. Association for Computing Machinery, New York, NY, USA, 198–208. https://doi.org/10.1145/351240.351259

Van Bakel Dezani-Ciancaglini, S. Van Bakel, M. Dezani-ciancaglini, and Y. Motohama. 1998. *The Minimal Relevant Logic and the Call-by-Value Lambda Calculus*. Technical Report.

Stephen Dolan. 2017. *Algebraic subtyping*. Ph.D. Dissertation.

Stephen Dolan and Alan Mycroft. 2017. Polymorphism, subtyping, and type inference in MLsub. *ACM SIGPLAN Notices* 52, 1 (Jan. 2017), 60–72. https://doi.org/10.1145/3093333.3009882

Jana Dunfield. 2012. Elaborating Intersection and Union Types. In *Proceedings of the 17th ACM SIGPLAN International Conference on Functional Programming* (Copenhagen, Denmark) *(ICFP '12)*. Association for Computing Machinery, New York, NY, USA, 17–28. https://doi.org/10.1145/2364527.2364534

Jonathan Eifrig, Scott Smith, and Valery Trifonov. 1995. Sound Polymorphic Type Inference for Objects. In *Proceedings of the Tenth Annual Conference on Object-Oriented Programming Systems, Languages, and Applications* (Austin, Texas, USA) *(OOPSLA '95)*. Association for Computing Machinery, New York, NY, USA, 169–184. https://doi.org/10.1145/217838.217858

Tim Freeman and Frank Pfenning. 1991. Refinement Types for ML. In *Proceedings of the ACM SIGPLAN 1991 Conference on Programming Language Design and Implementation (PLDI '91)*. ACM, New York, NY, USA, 268–277. https://doi.org/10.1145/113445.113468 event-place: Toronto, Ontario, Canada.

A. Frisch, G. Castagna, and V. Benzaken. 2002. Semantic subtyping. In *Proceedings 17th Annual IEEE Symposium on Logic in Computer Science*. 137–146. https://doi.org/10.1109/LICS.2002.1029823

Alain Frisch, Giuseppe Castagna, and Véronique Benzaken. 2008. Semantic Subtyping: Dealing Set-Theoretically with Function, Union, Intersection, and Negation Types. *J. ACM* 55, 4, Article 19 (Sept. 2008), 64 pages. https://doi.org/10.1145/1391289.1391293

You-Chin Fuh and Prateek Mishra. 1989. Polymorphic subtype inference: Closing the theory-practice gap. In *TAPSOFT '89*, J. Díaz and F. Orejas (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 167–183.

You-Chin Fuh and Prateek Mishra. 1990. Type inference with subtypes. *Theoretical Computer Science* 73, 2 (1990), 155–175. https://doi.org/10.1016/0304-3975(90)90144-7

Vladimir Gapeyev, Michael Y Levin, and Benjamin C Pierce. 2002. Recursive subtyping revealed. *Journal of Functional Programming* 12, 6 (2002), 511–548.

Jacques Garrigue. 1998. Programming with polymorphic variants. In *ML Workshop*, Vol. 13. Baltimore, 7.

Jacques Garrigue. 2001. Simple Type Inference for Structural Polymorphism.. In *APLAS*. 329–343.

Benedict R. Gaster and Mark P. Jones. 1996. A Polymorphic Type System for Extensible Records and Variants.

Silvia Ghilezan. 1996. Strong Normalization and Typability with Intersection Types. *Notre Dame Journal of Formal Logic* 37, 1 (1996), 44 – 52. https://doi.org/10.1305/ndjfl/1040067315

Roger Hindley. 1969. The Principal Type-Scheme of an Object in Combinatory Logic. *Trans. Amer. Math. Soc.* 146 (1969), 29–60. https://doi.org/10.2307/1995158 Publisher: American Mathematical Society.

Haruo Hosoya, Jérôme Vouillon, and Benjamin C. Pierce. 2005. Regular Expression Types for XML. *ACM Trans. Program. Lang. Syst.* 27, 1 (Jan. 2005), 46–90. https://doi.org/10.1145/1053468.1053470

Xuejing Huang and Bruno C. d. S. Oliveira. 2021. Distributing Intersection and Union Types with Splits and Duality (Functional Pearl). *Proc. ACM Program. Lang.* 5, ICFP, Article 89 (aug 2021), 24 pages. https://doi.org/10.1145/3473594

Edward V. Huntington. 1904. Sets of independent postulates for the algebra of logic. *Trans. Amer. Math. Soc.* 5, 3 (1904), 288–309. https://doi.org/10.1090/s0002-9947-1904-1500675-4

Trevor Jim and Jens Palsberg. 1999. Type Inference in Systems of Recursive Types With Subtyping.

Timothy Jones, Michael Homer, and James Noble. 2015. Brand Objects for Nominal Typing. In *29th European Conference on Object-Oriented Programming (ECOOP 2015) (Leibniz International Proceedings in Informatics (LIPIcs), Vol. 37)*, John Tang Boyland (Ed.). Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl, Germany, 198–221. https://doi.org/10.4230/LIPIcs.ECOOP.2015.198

Dexter Kozen, Jens Palsberg, and Michael I. Schwartzbach. 1994. Efficient inference of partial types. *J. Comput. System Sci.* 49, 2 (1994), 306–324. https://doi.org/10.1016/S0022-0000(05)80051-0

Joseph Lee, Jonathan Aldrich, Troy Shaw, and Alex Potanin. 2015. A Theory of Tagged Objects. In *29th European Conference on Object-Oriented Programming (ECOOP 2015) (Leibniz International Proceedings in Informatics (LIPIcs), Vol. 37)*, John Tang Boyland (Ed.). Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl, Germany, 174–197. https://doi.org/10.4230/LIPIcs.ECOOP.2015.174

David MacQueen, Gordon Plotkin, and Ravi Sethi. 1986. An ideal model for recursive polymorphic types. *Information and Control* 71, 1 (1986), 95–130. https://doi.org/10.1016/S0019-9958(86)80019-5

Robin Milner. 1978. A theory of type polymorphism in programming. *J. Comput. System Sci.* 17, 3 (Dec. 1978), 348–375. https://doi.org/10.1016/0022-0000(78)90014-4

John C. Mitchell. 1984. Coercion and Type Inference. In *Proceedings of the 11th ACM SIGACT-SIGPLAN Symposium on Principles of Programming Languages* (Salt Lake City, Utah, USA) *(POPL '84)*. Association for Computing Machinery, New York, NY, USA, 175–185. https://doi.org/10.1145/800017.800529

James Hiram Morris. 1969. *Lambda-calculus models of programming languages.* Ph.D. Dissertation. Massachusetts Institute of Technology.

Fabian Muehlboeck and Ross Tate. 2018. Empowering Union and Intersection Types with Integrated Subtyping. *Proc. ACM Program. Lang.* 2, OOPSLA, Article 112 (Oct. 2018), 29 pages. https://doi.org/10.1145/3276482

Martin Odersky, Philippe Altherr, Vincent Cremet, Burak Emir, Sebastian Maneth, Stéphane Micheloud, Nikolay Mihaylov, Michel Schinz, Erik Stenman, and Matthias Zenger. 2004. An overview of the Scala programming language. (2004).

Martin Odersky, Martin Sulzmann, and Martin Wehr. 1999. Type inference with constrained types. *Theory and Practice of Object Systems* 5, 1 (1999), 35–55.

Atsushi Ohori. 1995. A Polymorphic Record Calculus and Its Compilation. *ACM Trans. Program. Lang. Syst.* 17, 6 (nov 1995), 844–895. https://doi.org/10.1145/218570.218572

Jens Palsberg and Christina Pavlopoulou. 1998. From Polyvariant Flow Information to Intersection and Union Types. In *Proceedings of the 25th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages* (San Diego, California, USA) *(POPL '98)*. Association for Computing Machinery, New York, NY, USA, 197–208. https://doi.org/10.1145/268946.268963

Jens Palsberg, Mitchell Wand, and Patrick O'Keefe. 1997. Type inference with non-structural subtyping. *Formal Aspects of Computing* 9, 1 (Jan. 1997), 49–67. https://doi.org/10.1007/BF01212524

Lionel Parreaux. 2020. The Simple Essence of Algebraic Subtyping: Principal Type Inference with Subtyping Made Easy (Functional Pearl). *Proc. ACM Program. Lang.* 4, ICFP, Article 124 (Aug. 2020), 28 pages. https://doi.org/10.1145/3409006

Lionel Parreaux, Aleksander Boruch-Gruszecki, Andong Fan, and Chun Yin Chau. 2024. When Subtyping Constraints Liberate: A Novel Type Inference Approach for First-Class Polymorphism. *Proc. ACM Program. Lang.* 8, POPL, Article 48 (jan 2024), 33 pages. https://doi.org/10.1145/3632890

Lionel Parreaux and Chun Yin Chau. 2022. *MLstruct: Principal Type Inference in a Boolean Algebra of Structural Types (Extended Version)*. Technical Report. The Hong Kong University of Science and Technology. https://lptk.github.io/mlscript-paper

Lionel Parreaux, Luyu Cheng, Tony Chau, Ishan Bhanuka, Andong Fan, Malcolm Law, Ali Mahzoun, and Elise Rouillé. 2022. *MLstruct: Principal Type Inference in a Boolean Algebra of Structural Types (Artifact)*. https://doi.org/10.5281/zenodo.7121838

David J. Pearce. 2013. Sound and Complete Flow Typing with Unions, Intersections and Negations. In *Verification, Model Checking, and Abstract Interpretation (Lecture Notes in Computer Science)*, Roberto Giacobazzi, Josh Berdine, and Isabella Mastroeni (Eds.). Springer, Berlin, Heidelberg, 335–354. https://doi.org/10.1007/978-3-642-35873-9_21

Tommaso Petrucciani. 2019. *Polymorphic set-theoretic types for functional languages*. Ph.D. Dissertation. Università di Genova; Université Sorbonne Paris Cité – Université Paris Diderot.

Benjamin C Pierce. 1991. *Programming with intersection types and bounded polymorphism*. Ph.D. Dissertation. Citeseer.

Benjamin C. Pierce. 2002. *Types and programming languages*. MIT press.

François Pottier. 1996. Simplifying Subtyping Constraints. In *Proceedings of the First ACM SIGPLAN International Conference on Functional Programming* (Philadelphia, Pennsylvania, USA) *(ICFP '96)*. Association for Computing Machinery, New York, NY, USA, 122–133. https://doi.org/10.1145/232627.232642

François Pottier. 1998a. A Framework for Type Inference with Subtyping. In *Proceedings of the Third ACM SIGPLAN International Conference on Functional Programming* (Baltimore, Maryland, USA) *(ICFP '98)*. Association for Computing Machinery, New York, NY, USA, 228–238. https://doi.org/10.1145/289423.289448

François Pottier. 1998b. *Type Inference in the Presence of Subtyping: from Theory to Practice*. Research Report RR-3483. INRIA. https://hal.inria.fr/inria-00073205

François Pottier. 2001. Simplifying Subtyping Constraints: A Theory. *Information and Computation* 170, 2 (2001), 153–183. https://doi.org/10.1006/inco.2001.2963

François Pottier. 2003. A Constraint-Based Presentation and Generalization of Rows. In *IEEE Symposium on Logic In Computer Science (LICS)*. Ottawa, Canada, 331–340. http://cambium.inria.fr/~fpottier/publis/fpottier-lics03.pdf

Baber Rehman, Xuejing Huang, Ningning Xie, and Bruno C. d. S. Oliveira. 2022. Union Types with Disjoint Switches. In *36th European Conference on Object-Oriented Programming (ECOOP 2022) (Leibniz International Proceedings in Informatics (LIPIcs), Vol. 222)*, Karim Ali and Jan Vitek (Eds.). Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl, Germany, 25:1–25:31. https://doi.org/10.4230/LIPIcs.ECOOP.2022.25

Didier Rémy. 1994. *Type Inference for Records in Natural Extension of ML*. MIT Press, Cambridge, MA, USA, 67–95.

John C. Reynolds. 1997. *Design of the Programming Language Forsythe*. Birkhäuser Boston, Boston, MA, 173–233. https://doi.org/10.1007/978-1-4612-4118-8_9

Nathanael Schärli, Stéphane Ducasse, Oscar Nierstrasz, and Andrew P Black. 2003. Traits: Composable units of behaviour. In *European Conference on Object-Oriented Programming*. Springer, 248–274.

Vincent Simonet. 2003. Type Inference with Structural Subtyping: A Faithful Formalization of an Efficient Constraint Solver. In *Programming Languages and Systems*, Atsushi Ohori (Ed.). Springer Berlin Heidelberg, Berlin, Heidelberg, 283–302.

Geoffrey Seward Smith. 1991. *Polymorphic type inference for languages with overloading and subtyping*. Ph.D. Dissertation. Cornell University.

Scott F. Smith and Tiejun Wang. 2000. Polyvariant Flow Analysis with Constrained Types. In *Programming Languages and Systems*, Gert Smolka (Ed.). Springer Berlin Heidelberg, Berlin, Heidelberg, 382–396.

R. Stansifer. 1988. Type Inference with Subtypes. In *Proceedings of the 15th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages* (San Diego, California, USA) *(POPL '88)*. Association for Computing Machinery, New York, NY, USA, 88–97. https://doi.org/10.1145/73560.73568

Sam Tobin-Hochstadt and Matthias Felleisen. 2008. The Design and Implementation of Typed Scheme. In *Proceedings of the 35th Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages* (San Francisco, California, USA) *(POPL '08)*. Association for Computing Machinery, New York, NY, USA, 395–406. https://doi.org/10.1145/1328438.1328486

Valery Trifonov and Scott Smith. 1996. Subtyping constrained types. In *Static Analysis*, Radhia Cousot and David A. Schmidt (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 349–365.

Leo White. 2015. Row polymorphism. https://www.cl.cam.ac.uk/teaching/1415/L28/rows.pdf

# Appendix

## A Proofs and Auxiliary Definitions on Subtyping

### *A.1 Subtyping Derivation Shapes*

**Proof** [Lemma 3.10] Consider a derivation $D$ whose last applied rule is S-Assum. This rule application introduces a hypothesis $\triangleright H$ into the context of its premise derivation $D'$. In $D'$, $\triangleright H$ is kept unusable (because of the $\triangleright$) until applications of rules S-FunDepth or S-RcdDepth, within the premise derivations of which $H$ may be used, through applications $D_i^H$ of the S-Hyp rule Therefore, $H$ is never used at the bottom level of $D'$. Moreover, each $D_i^H$ will have a premise of the form $\Sigma{\cdot}H{\cdot}\Sigma_i$. So we can substitute all $D_i^H$ in $D$ with a weakened form (Lemma A.23) of the derivation $D$ *itself*. After this substitution, the main application of S-Assum becomes useless (the $H$ it introduces is no longer used in any subderivation), and it can therefore be removed, leaving the updated derivation $D'$.
It is easy to show that we can perform this S-Assum-elimination on bottom-level subderivations of any given derivation until that derivation becomes unassuming. ■

**Proof** [Lemma 3.12] By induction on the number of bottom-level applications of T-Subs. The result is immediate for derivations with zero or one bottom-level applications of T-Subs.
For derivations with $n \geqslant 2$ bottom-level applications of T-Subs, we first observe that the last two typing rules applied must be T-Subs (indeed, if the last rule applied was not T-Subs, then by definition the derivation would have no bottom-level applications of T-Subs; and the same reasoning goes for the second last application). The premises of the last application of T-Subs are $t : \tau'$ and $\tau' \leqslant \tau$ for some $\tau'$, where the subderivation for $t : \tau'$ has $n-1$ bottom-level applications of T-Subs. The premises of the second last application of T-Subs are $t : \tau''$ and $\tau'' \leqslant \tau'$ for some $\tau''$, where the subderivation for $t : \tau''$ has $n-2$ bottom-level applications of T-Subs. The subderivations of $\tau'' \leqslant \tau'$ and $\tau' \leqslant \tau$ can be merged by S-Trans into a derivations for $\tau'' \leqslant \tau$. We can then apply T-Subs to the subderivation for $t : \tau''$ and the new derivation for $\tau'' \leqslant \tau$ to obtain a new derivation for $t : \tau$ with $n-1$ bottom-level applications of T-Subs. By IH, such a derivation can be rewritten to an equivalent subsumption-normalized derivation. ■

### *A.2 Bounds Context Cleanup*

Bounds context cleanup removes occurrences of a type variable from the top level of its bounds, resulting in an equivalent guarded constraining context.

**Definition A.1** (Bounds context cleanup)**.** *The* constraining context cleanup *function is defined as follows:*

$$\begin{aligned}
\mathit{cleanup}(\epsilon) &= \epsilon \\
\mathit{cleanup}(\Xi{\cdot}(\alpha \leqslant \tau)) &= \mathit{cleanup}(\Xi){\cdot}\mathit{cleanup}'(\alpha \leqslant \mathrm{cdn}(\tau)) \\
\mathit{cleanup}(\Xi{\cdot}(\tau \leqslant \alpha)) &= \mathit{cleanup}(\Xi){\cdot}\mathit{cleanup}'(\mathrm{dcn}(\tau) \leqslant \alpha)
\end{aligned}$$

$$\begin{aligned}
\mathit{cleanup}'(\alpha \leqslant \textstyle\bigwedge_i \tau_i^{\mathrm{dn}}) &= (\alpha \leqslant \textstyle\bigwedge_j \pi_j^{\mathrm{dn}}) \quad \text{where } \overline{\mathit{cleanup}''(\alpha \leqslant \tau_i^{\mathrm{dn}})}^{\,i} = \overline{(\alpha \leqslant \pi_j^{\mathrm{dn}})}^{\,j} \\
\mathit{cleanup}'(\textstyle\bigvee_i \tau_i^{\mathrm{cn}} \leqslant \alpha) &= (\textstyle\bigvee_j \pi_j^{\mathrm{cn}} \leqslant \alpha) \quad \text{where } \overline{\mathit{cleanup}''(\tau_i^{\mathrm{cn}} \leqslant \alpha)}^{\,i} = \overline{(\pi_j^{\mathrm{cn}} \leqslant \alpha)}^{\,j}
\end{aligned}$$

$$\mathit{cleanup}''(\alpha \leqslant \textstyle\bigvee_i \tau_i^{\mathrm{n}}) = \begin{cases} \epsilon & \text{when } \alpha \in \{\, \overline{\tau_i^{\mathrm{n}}}^{\,i} \,\} \\ (\alpha \leqslant \bigvee_{i \mid \tau_i^{\mathrm{n}} \neq \neg\alpha} \tau_i^{\mathrm{n}}) & \text{otherwise} \end{cases}$$

$$\mathit{cleanup}''(\textstyle\bigwedge_i \tau_i^{\mathrm{n}} \leqslant \alpha) = \begin{cases} \epsilon & \text{when } \alpha \in \{\, \overline{\tau_i^{\mathrm{n}}}^{\,i} \,\} \\ (\bigwedge_{i \mid \tau_i^{\mathrm{n}} \neq \neg\alpha} \tau_i^{\mathrm{n}} \leqslant \alpha) & \text{otherwise} \end{cases}$$

**Lemma A.2** (Equivalence of constraining context cleanup)**.** $\overline{H \vDash \mathit{cleanup}(H)}^{\,H \in \Xi}$ *for all* $\Xi$.

**Lemma A.3** (Guardedness of constraining context cleanup)**.** $\mathit{cleanup}(\Xi)$ ***guard.*** *for all* $\Xi$.

**Lemma A.4** (Equivalence of bounds under constraining context cleanup)**.** $\alpha \wedge \mathit{ub}_{\Xi}(\alpha) \vee \mathit{lb}_{\Xi}(\alpha) \equiv \alpha \wedge \mathit{ub}_{\mathit{cleanup}(\Xi)}(\alpha) \vee \mathit{lb}_{\mathit{cleanup}(\Xi)}(\alpha)$ *for all* $\Xi$ *and* $\alpha$.

### *A.3 Some Useful Subtyping Relationships*

**Lemma A.5** (Identity Element)**.**

$$\frac{\Sigma \vdash \top^{\diamond} \wedge^{\diamond} \tau \leqslant^{\diamond} \pi}{\Sigma \vdash \tau \leqslant^{\diamond} \pi}\ \textsc{Identity}$$

**Proof**

$$\dfrac{\dfrac{\text{S-ToB}\diamond\ \dfrac{}{\tau \leqslant^{\diamond} \top^{\diamond}} \qquad \text{S-Refl}\ \dfrac{}{\tau \leqslant^{\diamond} \tau}}{\text{S-AndOr2}\overline{\diamond}\ \dfrac{}{\tau \leqslant^{\diamond} \top^{\diamond} \wedge^{\diamond} \tau}} \qquad \top^{\diamond} \wedge^{\diamond} \tau \leqslant^{\diamond} \pi}{\text{S-Trans}\ \dfrac{}{\tau \leqslant^{\diamond} \pi}}$$

■

**Theorem A.6** (Duality of Extrema)**.** $\top^{\diamond} \equiv \neg\bot^{\diamond}$

**Proof**

**Case** $\cdot$**.** We have $\neg\bot \leqslant \top$ by S-ToB$\cdot$. For $\top \leqslant \neg\bot$: We have $\top \leqslant \bot \vee \neg\bot$ by S-Compl$\cdot$, which implies $\top \leqslant \neg\bot$ by Lemma A.5$\supset$.

**Case** $\curvearrowright$**.** We have $\bot \leqslant \neg\top$ by S-ToB$\curvearrowright$. For $\neg\top \leqslant \bot$: We have $\top \wedge \neg\top \leqslant \bot$ by S-Compl$\curvearrowright$, which implies $\neg\top \leqslant \bot$ by Lemma A.5$\cdot$.

■

**Lemma A.7** (Covariance of unions and intersections)**.**

$$\dfrac{\Sigma \vdash \tau_1 \leqslant^{\diamond} \tau_2 \qquad \Sigma \vdash \tau_3 \leqslant^{\diamond} \tau_4}{\Sigma \vdash \tau_1 \vee^{\diamond} \tau_2 \leqslant^{\diamond} \tau_3 \vee^{\diamond} \tau_4}\ \textit{S-Covariance}$$

**Proof**

$$\dfrac{\dfrac{\dfrac{\tau_1 \leqslant^{\diamond} \tau_2}{\tau_1 \leqslant^{\diamond} \tau_2 \vee^{\diamond} \tau_4}\ \text{S-AndOr11}\diamond}{\tau_1 \leqslant^{\diamond} \tau_2 \vee^{\diamond} \tau_4}\ \text{S-Trans} \qquad \dfrac{\dfrac{\tau_3 \leqslant^{\diamond} \tau_4}{\tau_4 \leqslant^{\diamond} \tau_2 \vee^{\diamond} \tau_4}\ \text{S-AndOr12}\diamond}{\tau_3 \leqslant^{\diamond} \tau_2 \vee^{\diamond} \tau_4}\ \text{S-Trans}}{\tau_1 \vee^{\diamond} \tau_2 \leqslant^{\diamond} \tau_3 \vee^{\diamond} \tau_4}\ \text{S-AndOr2}\diamond$$

■

**Lemma A.8** (Associativity and Commutativity)**.**

$$\dfrac{}{\Sigma \vdash (\tau_1 \vee^{\diamond} \tau_2) \vee^{\diamond} \tau_3 \leqslant^{\diamond} (\tau_1 \vee^{\diamond} \tau_3) \vee^{\diamond} \tau_2}$$

**Proof**

$$\dfrac{\dfrac{\dfrac{}{\tau_1 \leqslant^{\diamond} \tau_1}\ \text{S-Refl} \qquad \dfrac{}{\tau_2 \vee^{\diamond} \tau_3 \leqslant^{\diamond} \tau_3 \vee^{\diamond} \tau_2}\ \text{S-Commut}\diamond}{\textbf{(1)}\ \tau_1 \vee^{\diamond} (\tau_2 \vee^{\diamond} \tau_3) \leqslant^{\diamond} \tau_1 \vee^{\diamond} (\tau_3 \vee^{\diamond} \tau_2)}\ \text{Lemma A.7}\diamond$$

$$\dfrac{\dfrac{}{(\tau_1 \vee^{\diamond} \tau_2) \vee^{\diamond} \tau_3 \leqslant^{\diamond} \tau_1 \vee^{\diamond} (\tau_2 \vee^{\diamond} \tau_3)}\ \text{S-Assoc}\diamond \qquad \dfrac{\textbf{(1)} \qquad \dfrac{}{\tau_1 \vee^{\diamond} (\tau_3 \vee^{\diamond} \tau_2) \leqslant^{\diamond} (\tau_1 \vee^{\diamond} \tau_3) \vee^{\diamond} \tau_2}\ \text{S-Assoc}\diamond}{\tau_1 \vee^{\diamond} (\tau_2 \vee^{\diamond} \tau_3) \leqslant^{\diamond} (\tau_1 \vee^{\diamond} \tau_3) \vee^{\diamond} \tau_2}\ \text{S-Trans}}{(\tau_1 \vee^{\diamond} \tau_2) \vee^{\diamond} \tau_3 \leqslant^{\diamond} (\tau_1 \vee^{\diamond} \tau_3) \vee^{\diamond} \tau_2}\ \text{S-Trans}$$

■

**Proof** [Proof of Lemma 3.1]

**Case** $\cdot, \Rightarrow$**.** Given **(1)**$\cdot$ $\Sigma \vdash \tau_1 \vee \tau_2 \leqslant \tau_3$, derive **(2)**$\cdot$ $\Sigma \vdash \tau_1 \leqslant \tau_3$ and **(3)**$\cdot$ $\Sigma \vdash \tau_2 \leqslant \tau_3$:

$$\dfrac{\dfrac{}{\tau_1 \leqslant \tau_1 \vee \tau_2}\ \text{S-AndOr11}\cdot \qquad \textbf{(1)}\cdot\ \tau_1 \vee \tau_2 \leqslant \tau_3}{\textbf{(2)}\cdot\ \tau_1 \leqslant \tau_3}\ \text{S-Trans}$$

Similar derivation for concluding **(3)**$\cdot$.

**Case** $\cdot, \Leftarrow$**.** Given **(2)**$\cdot$ and **(3)**$\cdot$, derive **(1)**$\cdot$:

$$\dfrac{\dfrac{\textbf{(2)}\cdot\ \tau_1 \leqslant \tau_3 \qquad \textbf{(3)}\cdot\ \tau_2 \leqslant \tau_3}{\tau_1 \vee \tau_2 \leqslant \tau_3 \vee \tau_3}\ \text{Lemma A.7}\cdot \qquad \dfrac{\dfrac{}{\tau_3 \leqslant \tau_3}\ \text{S-Refl} \qquad \dfrac{}{\tau_3 \leqslant \tau_3}\ \text{S-Refl}}{\tau_3 \vee \tau_3 \leqslant \tau_3}\ \text{S-AndOr2}\cdot}{\textbf{(1)}\cdot\ \tau_1 \vee \tau_2 \leqslant \tau_3}\ \text{S-Trans}$$

**Case** $\wr$**,** $\Rightarrow$**.** Given **(1)**$\wr$ $\Sigma \vdash \tau_3 \leqslant \tau_1 \wedge \tau_2$, derive **(2)**$\wr$ $\Sigma \vdash \tau_3 \leqslant \tau_1$ and **(3)**$\wr$ $\Sigma \vdash \tau_3 \leqslant \tau_2$:

$$\textsc{S-Trans}\ \dfrac{\textbf{(1)}\wr\ \tau_3 \leqslant \tau_1 \wedge \tau_2 \qquad \textsc{S-AndOr11}\wr\ \dfrac{}{\tau_1 \wedge \tau_2 \leqslant \tau_1}}{\textbf{(2)}\wr\ \tau_3 \leqslant \tau_1}$$

Similar derivation for concluding **(3)**$\wr$.

**Case** $\wr$**,** $\Leftarrow$**.** Given **(2)**$\wr$ and **(3)**$\wr$, derive **(1)**$\wr$:

$$\textsc{S-Trans}\ \dfrac{\textsc{S-AndOr2}\wr\ \dfrac{\textsc{S-Refl}\ \dfrac{}{\tau_3 \leqslant \tau_3} \qquad \textsc{S-Refl}\ \dfrac{}{\tau_3 \leqslant \tau_3}}{\tau_3 \leqslant \tau_3 \wedge \tau_3} \qquad \textsc{Lemma A.7}\cdot\ \dfrac{\textbf{(2)}\wr\ \tau_3 \leqslant \tau_1 \quad \textbf{(3)}\wr\ \tau_3 \leqslant \tau_2}{\tau_3 \vee \tau_3 \leqslant \tau_1 \vee \tau_2}}{\textbf{(1)}\wr\ \tau_3 \leqslant \tau_1 \vee \tau_2}$$

■

**Theorem A.9** (Swapping)**.**

$$\textit{S-Swap}\ \dfrac{\Sigma \vdash \tau_1 \vee^{\diamond} \tau_2 \geqslant^{\diamond} \tau_3}{\Sigma \vdash \tau_1 \geqslant^{\diamond} \tau_3 \wedge^{\diamond} \neg\tau_2}$$

**Proof**

**Case** $\wr$**.** Given **(1)** $\Sigma \vdash \tau_1 \wedge \tau_2 \leqslant \tau_3$, derive **(2)** $\Sigma \vdash \tau_1 \leqslant \tau_3 \vee \neg\tau_2$:

$$\textsc{S-Trans}\ \dfrac{\textsc{S-ToB}\cdot\ \dfrac{}{\tau_1 \leqslant \top} \qquad \textsc{S-Trans}\ \dfrac{\textsc{S-Compl}\cdot\ \dfrac{}{\top \leqslant \tau_2 \vee \neg\tau_2} \qquad \textsc{S-Commut}\cdot\ \dfrac{}{\tau_2 \vee \neg\tau_2 \leqslant \neg\tau_2 \vee \tau_2}}{\top \leqslant \neg\tau_2 \vee \tau_2}}{(\mathbf{1})\ \tau_1 \leqslant \neg\tau_2 \vee \tau_2}$$

$$\textsc{S-Trans}\ \dfrac{\textsc{S-AndOr2}\wr\ \dfrac{\textsc{S-AndOr12}\cdot\ \dfrac{}{\tau_1 \leqslant \neg\tau_2 \vee \tau_1} \quad (\mathbf{1})}{\tau_1 \leqslant (\neg\tau_2 \vee \tau_1) \wedge (\neg\tau_2 \vee \tau_2)} \qquad \textsc{S-Distrib}\wr\ \dfrac{}{(\neg\tau_2 \vee \tau_1) \wedge (\neg\tau_2 \vee \tau_2) \leqslant \neg\tau_2 \vee (\tau_1 \wedge \tau_2)}}{(\mathbf{2})\ \tau_1 \leqslant \neg\tau_2 \vee (\tau_1 \wedge \tau_2)}$$

$$\textsc{S-Trans}\ \dfrac{(\mathbf{2}) \qquad \textsc{S-Trans}\ \dfrac{\textsc{Lemma A.7}\cdot\ \dfrac{\textsc{S-Refl}\ \dfrac{}{\neg\tau_2 \leqslant \neg\tau_2} \quad \tau_1 \wedge \tau_2 \leqslant \tau_3}{\neg\tau_2 \vee (\tau_1 \wedge \tau_2) \leqslant \neg\tau_2 \vee \tau_3} \qquad \textsc{S-Commut}\ \dfrac{}{\neg\tau_2 \vee \tau_3 \leqslant \tau_3 \vee \neg\tau_2}}{\neg\tau_2 \vee (\tau_1 \wedge \tau_2) \leqslant \tau_3 \vee \neg\tau_2}}{\tau_1 \leqslant \tau_3 \vee \neg\tau_2}$$

**Case** $\cdot$**.** Symmetric.

■

**Theorem A.10** (Double Negation Introduction)**.**

$$\textit{S-Neg2}\ \dfrac{}{\tau \leqslant \neg\neg\tau}$$

**Proof**

$$\dfrac{\dfrac{\dfrac{}{\tau \wedge \neg\tau \leqslant \bot}\,\text{S-Compl}\supset}{\tau \leqslant \bot \vee \neg\neg\tau}\,\text{Theorem A.9}\supset \qquad \dfrac{\dfrac{\dfrac{}{\bot \leqslant \neg\neg\tau}\,\text{S-ToB}\supset \qquad \dfrac{}{\neg\neg\tau \leqslant \neg\neg\tau}\,\text{S-Refl}}{\bot \vee \neg\neg\tau \leqslant \neg\neg\tau}\,\text{S-AndOr2}\cdot}{\tau \leqslant \neg\neg\tau}\,\text{S-Trans}$$

■

**Theorem A.11** (Double Negation Elimination)**.**

$$\dfrac{}{\neg\neg\tau \leqslant \tau}\,\textit{S-Neg1}$$

**Proof**

$$\dfrac{\dfrac{\dfrac{\dfrac{}{\neg\neg\tau \leqslant \top}\,\text{S-ToB}\cdot \qquad \dfrac{}{\neg\neg\tau \leqslant \neg\neg\tau}\,\text{S-Refl}}{\neg\neg\tau \leqslant \top \wedge \neg\neg\tau}\,\text{S-AndOr2}\supset \qquad \dfrac{\dfrac{}{\top \leqslant \tau \vee \neg\tau}\,\text{S-Compl}\cdot}{\top \wedge \neg\neg\tau \leqslant \tau}\,\text{Theorem A.9}\cdot}{\neg\neg\tau \leqslant \tau}\,\text{S-Trans}$$

■

**Theorem A.12** (Associativity)**.**

$$\dfrac{}{(\tau_1 \vee^{\diamond} \tau_2) \vee^{\diamond} \tau_3 \equiv \tau_1 \vee^{\diamond} (\tau_2 \vee^{\diamond} \tau_3)}\,\textit{S-Assoc}\diamond$$

**Proof**

$$\dfrac{}{\textbf{(1)}\ (\tau_1 \vee^{\diamond} \tau_2) \vee^{\diamond} \tau_3 \geqslant^{\diamond} \tau_1 \vee^{\diamond} \tau_2}\,\text{S-AndOr11}\diamond$$

$$\dfrac{\dfrac{\dfrac{\dfrac{}{\tau_1 \vee^{\diamond} \tau_2 \geqslant^{\diamond} \tau_2}\,\text{S-AndOr12}\diamond \qquad \textbf{(1)}}{(\tau_1 \vee^{\diamond} \tau_2) \vee^{\diamond} \tau_3 \geqslant^{\diamond} \tau_2}\,\text{S-Trans} \qquad \dfrac{}{(\tau_1 \vee^{\diamond} \tau_2) \vee^{\diamond} \tau_3 \geqslant^{\diamond} \tau_3}\,\text{S-AndOr12}\diamond}{\textbf{(2)}\ (\tau_1 \vee^{\diamond} \tau_2) \vee^{\diamond} \tau_3 \geqslant^{\diamond} \tau_2 \vee^{\diamond} \tau_3}\,\text{S-AndOr2}\diamond$$

$$\dfrac{\dfrac{\dfrac{\dfrac{}{\tau_1 \vee^{\diamond} \tau_2 \geqslant^{\diamond} \tau_1}\,\text{S-AndOr11}\diamond \qquad \textbf{(1)}}{(\tau_1 \vee^{\diamond} \tau_2) \vee^{\diamond} \tau_3 \geqslant^{\diamond} \tau_1}\,\text{S-Trans} \qquad \textbf{(2)}}{(\tau_1 \vee^{\diamond} \tau_2) \vee^{\diamond} \tau_3 \geqslant^{\diamond} \tau_1 \vee^{\diamond} (\tau_2 \vee^{\diamond} \tau_3)}\,\text{S-AndOr2}\diamond$$

The other direction follows from S-Commut$\diamond$ (Theorem A.13 below). ■

**Theorem A.13** (Commutativity)**.**

$$\dfrac{}{\tau_1 \vee^{\diamond} \tau_2 \equiv \tau_2 \vee^{\diamond} \tau_1}\,\textit{S-Commut}\diamond$$

**Proof**

$$\dfrac{\text{S-AndOr12}\diamond\ \dfrac{}{\tau_1 \vee^\diamond \tau_2 \geqslant^\diamond \tau_2} \qquad \text{S-AndOr11}\diamond\ \dfrac{}{\tau_1 \vee^\diamond \tau_2 \geqslant^\diamond \tau_1}}{\text{S-AndOr2}\diamond\ \tau_1 \vee^\diamond \tau_2 \geqslant^\diamond \tau_2 \vee^\diamond \tau_1}$$

■

**Theorem A.14** (Distributivity)**.**

$$\dfrac{}{\tau_1 \vee^\diamond (\tau_2 \wedge^\diamond \tau_3) \equiv (\tau_1 \vee^\diamond \tau_2) \wedge^\diamond (\tau_1 \vee^\diamond \tau_3)}\ \textit{S-Distr}$$

**Proof**

**Case $\diamond$, $\geqslant^\diamond$ direction.** By S-Distrib$\diamond$.
**Case $\cdot$, $\leqslant$ direction.**

$$\dfrac{\text{Lemma A.7}\cdot\ \dfrac{\text{S-Refl}\ \dfrac{}{\tau_1 \leqslant \tau_1} \quad \text{S-AndOr11}\cdot\ \dfrac{}{\tau_2 \wedge \tau_3 \leqslant \tau_2}}{\tau_1 \vee (\tau_2 \wedge \tau_3) \leqslant \tau_1 \vee \tau_2} \qquad \text{Lemma A.7}\cdot\ \dfrac{\text{S-Refl}\ \dfrac{}{\tau_1 \leqslant \tau_1} \quad \text{S-AndOr12}\cdot\ \dfrac{}{\tau_2 \wedge \tau_3 \leqslant \tau_3}}{\tau_1 \vee (\tau_2 \wedge \tau_3) \leqslant \tau_1 \vee \tau_3}}{\text{S-AndOr2}\wr\ \tau_1 \vee (\tau_2 \wedge \tau_3) \leqslant (\tau_1 \vee \tau_2) \wedge (\tau_1 \vee \tau_3)}$$

**Case $\wr$, $\geqslant$ direction.** Symmetric.

■

**Theorem A.15** (Absorption)**.**

$$\dfrac{}{\tau_1 \vee^\diamond (\tau_1 \wedge^\diamond \tau_2) \equiv \tau_1}\ \textit{S-Absorp}$$

**Proof**

**Case $\diamond$, $\geqslant^\diamond$ direction.** By S-AndOr11$\diamond$.
**Case $\cdot$, $\leqslant$ direction.**

$$\dfrac{\text{Lemma A.7}\ \dfrac{\text{S-AndOr2}\cdot\ \dfrac{\text{S-Refl}\ \dfrac{}{\tau_1 \leqslant \tau_1} \quad \text{S-ToB}\cdot\ \dfrac{}{\tau_1 \leqslant \top}}{\tau_1 \leqslant \tau_1 \wedge \top} \qquad \text{S-Refl}\ \dfrac{}{\tau_1 \wedge \tau_2 \leqslant \tau_1 \wedge \tau_2}}{\textbf{(1)}\ \tau_1 \vee (\tau_1 \wedge \tau_2) \leqslant (\tau_1 \wedge \top) \vee (\tau_1 \wedge \tau_2)}}{\dfrac{\text{S-Distr}\wr\ \dfrac{\textbf{(1)}}{(\tau_1 \wedge \top) \vee (\tau_1 \wedge \tau_2) \leqslant \tau_1 \wedge (\top \vee \tau_2)} \qquad \text{S-AndOr11}\wr\ \dfrac{}{\tau_1 \wedge (\top \vee \tau_2) \leqslant \tau_1}}{\text{S-Trans}\ (\tau_1 \wedge \top) \vee (\tau_1 \wedge \tau_2) \leqslant \tau_1}}{\text{S-Trans}\ \tau_1 \vee (\tau_1 \wedge \tau_2) \leqslant \tau_1}$$

**Case $\wr$, $\geqslant$ direction.** Symmetric.

■

**Theorem A.16** (Negation contravariance)**.**

$$\dfrac{\Sigma \vdash \tau_1 \leqslant \tau_2}{\Sigma \vdash \neg\tau_2 \leqslant \neg\tau_1}\ \textit{S-NegInv}$$

**Proof**

$$\dfrac{\dfrac{\dfrac{}{\neg\tau_2 \wedge \tau_1 \leqslant \tau_1 \wedge \neg\tau_2}\ \text{S-Commut} \qquad \dfrac{\dfrac{\tau_1 \leqslant \tau_2}{\tau_1 \leqslant \bot \vee \tau_2}\ \text{S-Trans} \qquad \dfrac{}{\tau_2 \leqslant \bot \vee \tau_2}\ \text{S-AndOr12}\cdot}{\tau_1 \wedge \neg\tau_2 \leqslant \bot}\ \text{Theorem A.9}\cdot}{\neg\tau_2 \wedge \tau_1 \leqslant \bot}\ \text{S-Trans}}{(\mathbf{1})\ \neg\tau_2 \leqslant \bot \vee \neg\tau_1}\ \text{Theorem A.9}\curvearrowright$$

$$\dfrac{(\mathbf{1}) \qquad \dfrac{\dfrac{}{\bot \leqslant \neg\tau_1}\ \text{S-ToB}\curvearrowright \qquad \dfrac{}{\neg\tau_1 \leqslant \neg\tau_1}\ \text{S-Refl}}{\bot \vee \neg\tau_1 \leqslant \tau_1}\ \text{S-AndOr2}\cdot}{\neg\tau_2 \leqslant \neg\tau_1}\ \text{S-Trans}$$

■

**Proof** [Proof of Theorem 3.2]

$$\dfrac{\dfrac{}{\tau_m \leqslant \neg\neg\tau_m}\ \text{S-Neg2} \qquad \dfrac{\dfrac{\neg\tau_n \leqslant \neg\tau_m}{\neg\neg\tau_m \leqslant \neg\neg\tau_n}\ \text{S-NegInv} \qquad \dfrac{}{\neg\neg\tau_n \leqslant \tau_n}\ \text{S-Neg1}}{\neg\neg\tau_m \leqslant \tau_n}\ \text{S-Trans}}{\tau_m \leqslant \tau_n}\ \text{S-Trans}$$

Taking $(n, m) = (1, 2)$ and $(n, m) = (2, 1)$ yields the desired results. ■

**Theorem A.17** (De Morgan's Laws)**.**

$$\dfrac{}{\neg(\tau_1 \vee^{\diamond} \tau_2) \equiv \neg\tau_1 \wedge^{\diamond} \neg\tau_2}\ \textit{S-DeMorgan}$$

**Proof**

$$\dfrac{\dfrac{\dfrac{}{\top \leqslant \tau \vee \neg\tau}\,\text{S-Compl}\cdot \quad \dfrac{}{\pi \leqslant \pi}\,\text{S-Refl}}{\top \vee \pi \leqslant (\tau \vee \neg\tau) \vee \pi}\,\text{Lemma A.7}\cdot \quad \dfrac{}{(\tau \vee \neg\tau) \vee \pi \leqslant (\tau \vee \pi) \vee \neg\tau}\,\text{Lemma A.8}\cdot}{\textbf{(1)}\ \top \vee \pi \leqslant (\tau \vee \pi) \vee \neg\tau}\,\text{S-Trans}$$

$$\dfrac{\dfrac{\dfrac{}{\tau \leqslant \tau}\,\text{S-Refl} \quad \dfrac{}{\top \leqslant \pi \vee \neg\pi}\,\text{S-Compl}\cdot}{\tau \vee \top \leqslant \tau \vee (\pi \vee \neg\pi)}\,\text{Lemma A.7}\cdot \quad \dfrac{}{\tau \vee (\pi \vee \neg\pi) \leqslant (\tau \vee \pi) \vee \neg\pi}\,\text{S-Assoc}\cdot}{\textbf{(2)}\ \tau \vee \top \leqslant (\tau \vee \pi) \vee \neg\pi}\,\text{S-Trans}$$

$$\dfrac{\dfrac{\dfrac{\dfrac{}{\top \leqslant \top \vee \pi}\,\text{S-AndOr11}\cdot}{\top \leqslant (\tau \vee \pi) \vee \neg\tau}\,\textbf{(1)}\,\text{S-Trans} \quad \dfrac{\dfrac{}{\top \leqslant \pi \vee \top}\,\text{S-AndOr12}\cdot}{\top \leqslant (\tau \vee \pi) \vee \neg\pi}\,\textbf{(2)}\,\text{S-Trans}}{\textbf{(3)}\ \top \wedge \top \leqslant ((\tau \vee \pi) \vee \neg\tau) \wedge ((\tau \vee \pi) \vee \neg\pi)}\,\text{Lemma A.7}\rhd}$$

$$\dfrac{\dfrac{\dfrac{}{\top \leqslant \top}\,\text{S-Refl} \quad \dfrac{}{\top \leqslant \top}\,\text{S-Refl}}{\top \leqslant \top \wedge \top}\,\text{S-AndOr2}\rhd \quad \textbf{(3)}}{\textbf{(4)}\ \top \leqslant ((\tau \vee \pi) \vee \neg\tau) \wedge ((\tau \vee \pi) \vee \neg\pi)}\,\text{S-Trans}$$

$$\dfrac{\textbf{(4)} \quad \dfrac{}{((\tau \vee \pi) \vee \neg\tau) \wedge ((\tau \vee \pi) \vee \neg\pi) \leqslant (\tau \vee \pi) \vee (\neg\tau \wedge \neg\pi)}\,\text{S-Distrib}\rhd}{\textbf{(5)}\ \top \leqslant (\tau \vee \pi) \vee (\neg\tau \wedge \neg\pi)}\,\text{S-Trans}$$

$$\dfrac{\dfrac{\dfrac{\textbf{(5)} \quad \dfrac{}{(\tau \vee \pi) \vee (\neg\tau \wedge \neg\pi) \leqslant (\neg\tau \wedge \neg\pi) \vee (\tau \vee \pi)}\,\text{S-Commut}\cdot}{\top \leqslant (\neg\tau \wedge \neg\pi) \vee (\tau \vee \pi)}\,\text{S-Trans}}{\top \wedge \neg(\tau \vee \pi) \leqslant \neg\tau \wedge \neg\pi}\,\text{Theorem A.9}}{\neg(\tau \vee \pi) \leqslant \neg\tau \wedge \neg\pi}\,\text{Lemma A.5}$$

$\neg\tau \wedge \neg\pi \leqslant \neg(\tau \vee \pi)$ can be derived by similar reasoning. ■

### *A.4 Lemmas on Subtyping Entailment*

**Lemma A.18** (Reflexivity and weakening)**.** $\Sigma \cdot \Sigma' \vDash (\rhd)\Sigma$ *for all* $\Sigma$ *and* $\Sigma'$.

**Proof** By repeated applications of S-Cons or S-Cons$\rhd$ on S-Hyp. ■

**Lemma A.19** (Transitivity)**.** *If* $\Sigma \vDash \Sigma'$ *and* $\Sigma' \vDash \Sigma''$*, then* $\Sigma \vDash \Sigma''$.

**Proof** By straightforward induction on subtyping entailment derivations, making use of Lemma A.23 for cases S-Cons and S-Cons$\rhd$. ■

**Lemma A.20** (Merging)**.** *If* $\Sigma_1 \vDash \Sigma_1'$ *and* $\Sigma_2 \vDash \Sigma_2'$*, then* $\Sigma_1 \cdot \Sigma_2 \vDash \Sigma_1' \cdot \Sigma_2'$.

**Proof** By straightforward induction on subtyping entailment derivations for $\Sigma_2 \vDash \Sigma_2'$, making use of Lemma A.18 and Lemma A.19 for case S-Empty, and Lemma A.23 for cases S-Cons and S-Cons$\triangleright$. ∎

**Lemma A.21** (Guarding)**.** *If* $\Sigma \vDash \Sigma'$*, then* $\triangleright\Sigma \vDash \triangleright\Sigma'$.

**Proof** By straightforward induction on subtyping entailment judgements. ∎

**Lemma A.22** (Unguarding)**.** *If* $\Sigma \vDash \Sigma'$*, then* $\triangleleft\Sigma \vDash \triangleleft\Sigma'$.

**Proof** By straightforward induction on subtyping entailment judgements. ∎

**Lemma A.23** (Weakening of subtyping contexts in subtyping judgements)**.** *If* $\Sigma \vdash \tau \leqslant \pi$ *and* $\Sigma' \vDash \Sigma$*, then* $\Sigma' \vDash \tau \leqslant \pi$.

**Proof** By induction on unassuming subtyping derivations. The only non-trivial cases are S-Hyp, S-FunDepth, and S-RcdDepth.

**Case S-Hyp.** Then the premise of the rule is $(\tau \leqslant \pi) \in \Sigma$. By straightforward induction on subtyping entailment judgements, $\Sigma' \vDash \Sigma$ and $(\tau \leqslant \pi) \in \Sigma$ implies $\Sigma' \vdash \tau \leqslant \pi$.

**Case S-FunDepth.** Then we have $\tau = \tau_1 \to \tau_2$ for some $\tau_1$ and $\tau_2$, and $\pi = \pi_1 \to \pi_2$ for some $\pi_1$ and $\pi_2$. The premises of the rule are $\triangleleft\Sigma \vdash \pi_1 \leqslant \tau_1$ and $\triangleleft\Sigma \vdash \tau_2 \leqslant \pi_2$. By Lemma A.22, $\Sigma' \vDash \Sigma$ implies $\triangleleft\Sigma' \vDash \triangleleft\Sigma$. Then by IH on the premises, we have $\triangleleft\Sigma' \vdash \pi_1 \leqslant \tau_1$ and $\triangleleft\Sigma' \vdash \tau_2 \leqslant \pi_2$. Then we have $\Sigma' \vdash \tau_1 \to \tau_2 \leqslant \pi_1 \to \pi_2$ by S-FunDepth.

**Case S-RcdDepth.** Then we have $\tau = \{\, x : \tau_1 \,\}$ for some $\tau_1$ and $x$, and $\pi = \{\, x : \pi_1 \,\}$ for some $\pi_1$. The premise of the rule is $\triangleleft\Sigma \vdash \tau_1 \leqslant \pi_1$. By Lemma A.22, $\Sigma' \vDash \Sigma$ implies $\triangleleft\Sigma' \vDash \triangleleft\Sigma$. Then by IH on the premise, we have $\triangleleft\Sigma' \vdash \tau_1 \leqslant \pi_1$. Then we have $\Sigma' \vdash \{\, x : \tau_1 \,\} \leqslant \{\, x : \pi_1 \,\}$ by S-RcdDepth.

∎

**Corollary A.24** (Weakening of guarded subtyping contexts in subtyping judgements)**.** *If* $\triangleright\Sigma \vdash \tau \leqslant \pi$ *and* $\Sigma' \vDash \Sigma$*, then* $\triangleright\Sigma' \vDash \tau \leqslant \pi$.

**Proof** By Lemma A.21 and Lemma A.23. ∎

**Lemma A.25** (Weakening of guarded constraining contexts in consistency judgements)**.** *If* $\Sigma \vdash \triangleright\Xi_{\triangleright} \cdot \Xi\ ;\ \rho$ ***cons.*** *and* $\triangleright\Xi'_{\triangleright} \vDash \triangleright\Xi_{\triangleright}$*, then* $\Sigma \vdash \triangleright\Xi'_{\triangleright} \cdot \Xi\ ;\ \rho$ ***cons.***.

**Proof** By induction on consistency derivations.

**Base case.** For the base case, we have $\Xi = \epsilon$. Then by the base case of the definition of consistency, we have:

$$\Sigma \vdash \triangleright\Xi'_{\triangleright}\ ;\ \rho \textbf{ \textit{cons.}} \tag{1}$$

**Inductive case.** For the inductive case, we have $\rho = \rho_2 \circ \rho_1$ for some $\rho_1$ and $\rho_2$, where $dom(\rho_1) = \{\, \alpha \,\}$ for some $\alpha$. The premises of the rule are:

$$\triangleright\Xi_{\triangleright} \cdot \triangleright\Xi_{\alpha} \cdot \rho_1\Xi_{\not\alpha} \cdot \rho_1\Sigma \vDash \rho_1\Xi_{\alpha} \tag{2}$$

$$\rho_1\Sigma \vdash \triangleright\Xi_{\triangleright} \cdot \triangleright\Xi_{\alpha} \cdot \rho_1\Xi_{\not\alpha} \;;\; \rho_2 \; \textbf{\textit{cons.}} \tag{3}$$

where $split_{\alpha}(\Xi,\ dom(\rho_2)) = (\Xi_{\alpha},\ \Xi_{\not\alpha})$. From the assumption, we have:

$$\triangleright\Xi'_{\triangleright} \vDash \triangleright\Xi_{\triangleright} \tag{4}$$

By Lemma A.23 with (4), (2) implies:

$$\triangleright\Xi'_{\triangleright} \cdot \triangleright\Xi_{\alpha} \cdot \rho_1\Xi_{\not\alpha} \cdot \rho_1\Sigma \vDash \rho_1\Xi_{\alpha} \tag{5}$$

By IH on (3), we have:

$$\rho_1\Sigma \vdash \triangleright\Xi'_{\triangleright} \cdot \triangleright\Xi_{\alpha} \cdot \rho_1\Xi_{\not\alpha} \;;\; \rho_2 \; \textbf{\textit{cons.}} \tag{6}$$

Then by the inductive case of the definition of consistency, (5) and (6) imply:

$$\Sigma \vdash \triangleright\Xi'_{\triangleright} \cdot \Xi \;;\; \rho \; \textbf{\textit{cons.}} \tag{7}$$

■

**Lemma A.26** (Weakening of subtyping contexts in consistency judgements)**.** *If* $\Sigma \vdash \triangleright\Xi_{\triangleright} \cdot \Xi \;;\; \rho$ ***cons.*** *and* $\triangleright\Xi_{\triangleright} \cdot \Xi \cdot \Sigma' \vDash \Sigma$*, then* $\Sigma' \vdash \triangleright\Xi_{\triangleright} \cdot \Xi \;;\; \rho$ ***cons.***.

**Proof** By induction on consistency derivations.

**Base case.** For the base case, we have $\Xi = \epsilon$. Then by the base case of the definition of consistency, we have:

$$\Sigma' \vdash \triangleright\Xi_{\triangleright} \;;\; \rho \; \textbf{\textit{cons.}} \tag{1}$$

**Inductive case.** For the inductive case, we have $\rho = \rho_2 \circ \rho_1$ for some $\rho_1$ and $\rho_2$, where $dom(\rho_1) = \{\, \alpha \,\}$ for some $\alpha$. The premises of the rule are:

$$\triangleright\Xi_{\triangleright} \cdot \triangleright\Xi_{\alpha} \cdot \rho_1\Xi_{\not\alpha} \cdot \rho_1\Sigma \vDash \rho_1\Xi_{\alpha} \tag{2}$$

$$\rho_1\Sigma \vdash \triangleright\Xi_{\triangleright} \cdot \triangleright\Xi_{\alpha} \cdot \rho_1\Xi_{\not\alpha} \;;\; \rho_2 \; \textbf{\textit{cons.}} \tag{3}$$

where $split_{\alpha}(\Xi,\ dom(\rho_2)) = (\Xi_{\alpha},\ \Xi_{\not\alpha})$. From the assumption, we have:

$$\triangleright\Xi_{\triangleright} \cdot \Xi \cdot \Sigma' \vDash \Sigma \tag{4}$$

By Lemma A.38, (4) implies:

$$\triangleright\Xi_{\triangleright} \cdot \triangleright\Xi_{\alpha} \cdot \rho_1\Xi_{\not\alpha} \cdot \rho_1\Sigma' \vDash \rho_1\Sigma \tag{5}$$

By Lemma A.23 with (5), (2) implies:

$$\triangleright\Xi_{\triangleright} \cdot \triangleright\Xi_{\alpha} \cdot \rho_1\Xi_{\not\alpha} \cdot \rho_1\Sigma' \vDash \rho_1\Xi_{\alpha} \tag{6}$$

By IH on (3) and (5), we have:

$$\rho_1\Sigma' \vdash \triangleright\Xi_{\triangleright} \cdot \triangleright\Xi_{\alpha} \cdot \rho_1\Xi_{\not\alpha} \;;\; \rho_2 \; \textbf{\textit{cons.}} \tag{7}$$

Then by the inductive case of the definition of consistency, (6) and (7) imply:

$$\Sigma' \vdash \triangleright \Xi_{\triangleright} \cdot \Xi \,;\, \rho \ \textbf{\textit{cons.}} \tag{8}$$

■

**Lemma A.27** (Weakening of constraining contexts in typing judgements)**.** *If* $\Xi, \Gamma \vdash t : \tau$ *and* $\Xi' \models \Xi$*, then* $\Xi', \Gamma \vdash t : \tau$.

**Proof** By straightforward induction on typing derivations. The only non-trivial vases are T-SUBS and T-VAR2.

**Case T-SUBS.** By IH on the first premise, Lemma A.23 on the second premise, followed by T-SUBS.

**Case T-VAR2.** $\Gamma(x) = \forall \Xi''.\, \tau'$

We first notice that the subtyping entailment judgement is transitive by straightforward induction on subtyping entailment judgements, applying Lemma A.23 to the second premise of S-CONS. The first premise of S-ALL is $\Xi \models \rho(\Xi'')$, which implies $\Xi' \models \rho(\Xi'')$ by transitivity with the assumption $\Xi' \models \Xi$. The result then follows from Lemma A.23 on the second premise S-ALL, followed by S-ALL and T-VAR2.

■

### *A.5 Lemmas on Substitutions*

**Lemma A.28** (Preservation of typing under substitution)**.** *If* $\Xi, \Gamma \vdash t : \tau$ *and* $\mathcal{D}$ ***wf****, then* $\rho(\Xi), \rho(\Gamma) \vdash \rho(t) : \rho(\tau)$.

**Proof** By induction on typing derivations of $\Xi, \Gamma \vdash t : \tau$.

**Case T-SUBS.** By IH on the first premise, we have $\rho(\Xi), \rho(\Gamma) \vdash \rho(t) : \rho(\tau_1)$. By preservation of subtyping under substitution (Lemma A.29) on the second premise, $\rho(\Xi) \vdash \rho(\tau_1) \leqslant \rho(\tau_2)$. The result then follows from T-SUBS.

**Case T-OBJ.** By the definition of type substitution, $\rho(\#C \wedge \{\, \overline{x : \tau} \,\}) = \#C \wedge \{\, \overline{x : \rho(\tau)} \,\}$. By the definition of term substitution, $\rho(C\ \{\, \overline{x = t} \,\}) = C\ \{\, \overline{x = \rho(t)} \,\}$. By IH on the premises, we have $\overline{\rho(\Xi), \rho(\Gamma) \vdash \rho(t) : \rho(\tau)}$. Then $\rho(\Xi), \rho(\Gamma) \vdash C\ \{\, \overline{x = \rho(t)} \,\} : \#C \wedge \{\, \overline{x : \rho(\tau)} \,\}$ by T-OBJ, i.e., $\rho(\Xi), \rho(\Gamma) \vdash \rho(C\ \{\, \overline{x = t} \,\}) : \#C \wedge \rho(\{\, \overline{x : \tau} \,\})$.

**Case T-PROJ.** By the definition of term substitution, $\rho(t.x) = \rho(t).x$ By IH on the premise, we have $\rho(\Xi), \rho(\Gamma) \vdash t : \rho(\{\, x : \tau \,\})$, i.e., $\rho(\Xi), \rho(\Gamma) \vdash \rho(t) : \{\, x : \rho(\tau) \,\}$ by the definition of type substitution. Then $\rho(\Xi), \rho(\Gamma) \vdash \rho(t).x : \rho(\tau)$ by T-PROJ, i.e., $\rho(\Xi), \rho(\Gamma) \vdash \rho(t.x) : \rho(\tau)$.

**Case T-VAR1.** Then $t = x$. By the definition of term substitution, $\rho(x) = x$. From the premise and the definition of typing context substitution, we have $\rho(\Gamma)(x) = \rho(\tau)$. Then $\rho(\Xi), \rho(\Gamma) \vdash x : \rho(\tau)$ by T-VAR1, i.e., $\rho(\Xi), \rho(\Gamma) \vdash \rho(x) : \rho(\tau)$.

**Case T-VAR2.** Then $t = x$. By the definition of term substitution, $\rho(x) = x$. From the premise, we have $\Xi \vdash \Gamma(x) \leqslant^{\forall} \forall \epsilon.\, \tau$, where $\Gamma(x) = \forall \Xi'.\, \tau'$. Note that the judgement $\leqslant^{\forall}$ can only be derived by S-ALL, then from the premises of S-ALL, we have $\Xi \vDash \rho'(\Xi')$ and $\Xi \vdash \rho'(\tau') \leqslant \tau$. By preservation of subtyping under substitution (Lemma A.29), we have $\rho(\Xi) \vDash \rho(\rho'(\Xi'))$ and $\rho(\Xi) \vdash \rho(\rho'(\tau')) \leqslant \rho(\tau)$. Then $\rho(\Xi) \vdash \forall \Xi'.\, \tau' \leqslant^{\forall} \forall \epsilon.\, \rho(\tau)$ by S-ALL. Note that by the definition of typing context substitution, $\Gamma(x) = \forall \Xi'.\, \tau'$ implies $\rho(\Gamma)(x) = \forall \Xi'.\, \tau'$, then $\rho(\Xi), \rho(\Gamma) \vdash x : \rho(\tau)$ by T-VAR, i.e., $\rho(\Xi), \rho(\Gamma) \vdash \rho(x) : \rho(\tau)$.

**Case T-ABS.** By the definition of type substitution, $\rho(\tau_1 \to \tau_2) = \rho(\tau_1) \to \rho(\tau_2)$. By IH on the premise, we have $\rho(\Xi), \rho(\Gamma \cdot (x : \tau_1)) \vdash t : \rho(\tau_2)$, i.e., $\rho(\Xi), \rho(\Gamma) \cdot (x : \rho(\tau_1)) \vdash t : \rho(\tau_2)$ by the definition of typing context substitution. Then $\rho(\Xi), \rho(\Gamma) \vdash \lambda x.\, t : \rho(\tau_1) \to \rho(\tau_2)$ by T-ABS, i.e., $\rho(\Xi), \rho(\Gamma) \vdash \lambda x.\, t : \rho(\tau_1 \to \tau_2)$.

**Case T-APP.** By IH on the premise, we have $\rho(\Xi), \rho(\Gamma) \vdash t_1 : \rho(\tau_1)$ and $\rho(\Xi), \rho(\Gamma) \vdash t_0 : \rho(\tau_1 \to \tau_2)$, i.e., $\rho(\Xi), \rho(\Gamma) \vdash t_0 : \rho(\tau_1) \to \rho(\tau_2)$ by the definition of type substitution. Then $\rho(\Xi), \rho(\Gamma) \vdash t_0\ t_1 : \rho(\tau_2)$ by T-APP.

**Case T-ASC.** By the definition of term substitution, $\rho(t : \tau) = \rho(t) : \rho(\tau)$. By IH on the premise, we have $\rho(\Xi), \rho(\Gamma) \vdash \rho(t) : \rho(\tau)$. Then $\rho(\Xi), \rho(\Gamma) \vdash (\rho(t) : \rho(\tau)) : \rho(\tau)$, i.e., $\rho(\Xi), \rho(\Gamma) \vdash \rho(t : \tau) : \rho(\tau)$.

**Case T-CASE1.** By the definition of type substitution, $\rho(\bot) = \bot$. By the definition of term substitution, $\rho(\textbf{case}\ x = t_1\ \textbf{of}\ \epsilon) = (\textbf{case}\ x = \rho(t_1)\ \textbf{of}\ \epsilon)$. By IH on the premise, we have $\rho(\Xi), \rho(\Gamma) \vdash \rho(t_1) : \rho(\bot)$, i.e., $\rho(\Xi), \rho(\Gamma) \vdash \rho(t_1) : \bot$. Then $\rho(\Xi), \rho(\Gamma) \vdash \textbf{case}\ x = \rho(t_1)\ \textbf{of}\ \epsilon : \bot$, i.e., $\rho(\Xi), \rho(\Gamma) \vdash \rho(\textbf{case}\ x = t_1\ \textbf{of}\ \epsilon) : \rho(\bot)$.

**Case T-CASE2.** By the definition of term substitution, $\rho(\textbf{case}\ x = t_1\ \textbf{of}\ \_ \to t_2) = (\textbf{case}\ x = \rho(t_1)\ \textbf{of}\ \_ \to \rho(t_2))$. By IH on the premises, we have $\rho(\Xi), \rho(\Gamma) \vdash \rho(t_1) : \rho(\tau_1)$ and $\rho(\Xi), \rho(\Gamma \cdot (x : \tau_1)) \vdash \rho(t_2) : \rho(\tau)$, i.e., $\rho(\Xi), \rho(\Gamma) \cdot (x : \rho(\tau_1)) \vdash \rho(t_2) : \rho(\tau)$ by the definition of typing context substitution. Then $\rho(\Xi), \rho(\Gamma) \vdash \textbf{case}\ x = \rho(t_1)\ \textbf{of}\ \_ \to \rho(t_2) : \rho(\tau)$, i.e., $\rho(\Xi), \rho(\Gamma) \vdash \rho(\textbf{case}\ x = t_1\ \textbf{of}\ \_ \to t_2) : \rho(\tau)$.

**Case T-CASE3.** By the definition of term substitution, $\rho(\textbf{case}\ x = t_1\ \textbf{of}\ C \to t_2,\ M) = (\textbf{case}\ x = \rho(t_1)\ \textbf{of}\ C \to \rho(t_2),\ \rho(M))$. By IH on the first premise, we have $\rho(\Xi), \rho(\Gamma) \vdash \rho(t_1) : \rho(\#C \wedge \tau_1 \vee \neg\#C \wedge \tau_2)$, i.e., $\rho(\Xi), \rho(\Gamma) \vdash \rho(t_1) : \#C \wedge \rho(\tau_1) \vee \neg\#C \wedge \rho(\tau_2)$ by the definition of type substitution. By IH on the second premise, we have $\rho(\Xi), \rho(\Gamma \cdot (x : \tau_1)) \vdash \rho(t_2) : \rho(\tau)$, i.e., $\rho(\Xi), \rho(\Gamma) \cdot (x : \rho(\tau_1)) \vdash \rho(t_2) : \rho(\tau)$. By IH on the third premise, we have $\rho(\Xi), \rho(\Gamma \cdot (x : \tau_2)) \vdash \rho(\textbf{case}\ x = x\ \textbf{of}\ M) : \rho(\tau)$, i.e., $\rho(\Xi), \rho(\Gamma) \cdot (x : \rho(\tau_2)) \vdash \textbf{case}\ x = x\ \textbf{of}\ \rho(M) : \rho(\tau)$ by the definition of term substitution. Then $\rho(\Xi), \rho(\Gamma) \vdash \textbf{case}\ x = \rho(t_1)\ \textbf{of}\ C \to \rho(t_2),\ \rho(M) : \rho(\tau)$ by T-CASE3, i.e., $\rho(\Xi), \rho(\Gamma) \vdash \rho(\textbf{case}\ x = t_1\ \textbf{of}\ C \to t_2,\ M) : \rho(\tau)$.

■

**Lemma A.29** (Preservation of subtyping under substitution)**.** *If* $\Sigma \vdash \tau_1 \leqslant \tau_2$ *and* $\mathcal{D}$ ***wf****, then* $\rho(\Sigma) \vdash \rho(\tau_1) \leqslant \rho(\tau_2)$.

**Proof** By induction on subtyping derivations of $\Sigma \vdash \tau_1 \leqslant \tau_2$.

**Case S-REFL.** The result $\rho(\tau) \leqslant \rho(\tau)$ follows immediately from S-REFL.

**Case S-TOB$\diamond$.** By the definition of type substitution, $\rho(\top^\diamond) = \top^\diamond$. By S-TOB$\diamond$, $\rho(\tau) \leqslant^\diamond \top^\diamond$, i.e., $\rho(\tau) \leqslant^\diamond \rho(\top^\diamond)$.

**Case S-COMPL$\diamond$.** By the definition of type substitution, $\rho(\tau \vee^\diamond \neg\tau) = \rho(\tau) \vee^\diamond \rho(\neg\tau) = \rho(\tau) \vee^\diamond \neg\rho(\tau)$ and $\rho(\top^\diamond) = \top^\diamond$. By S-COMPL$\diamond$, $\rho(\tau) \vee^\diamond \neg\rho(\tau) \geqslant^\diamond \top^\diamond$, i.e., $\rho(\tau \vee^\diamond \neg\tau) \geqslant^\diamond \rho(\top^\diamond)$.

**Case S-ANDOR11$\diamond$.** By the definition of type substitution, $\rho(\tau_1 \vee^\diamond \tau_2) = \rho(\tau_1) \vee^\diamond \rho(\tau_2)$. By IH on the premise, we have $\rho(\Sigma) \vdash \rho(\tau_1) \geqslant^\diamond \rho(\tau)$. Then $\rho(\Sigma) \vdash \rho(\tau_1) \vee^\diamond \rho(\tau_2) \geqslant^\diamond \rho(\tau)$ by S-ANDOR11$\diamond$, i.e., $\rho(\Sigma) \vdash \rho(\tau_1 \vee \tau_2) \geqslant^\diamond \rho(\tau)$.

**Case S-ANDOR12$\diamond$.** Symmetric to the case above.

**Case S-ANDOR2$\diamond$.** By the definition of type substitution, $\rho(\tau_1 \vee^\diamond \tau_2) = \rho(\tau_1) \vee^\diamond \rho(\tau_2)$. By IH on the premises, we have $\rho(\Sigma) \vdash \rho(\tau) \geqslant^\diamond \rho(\tau_1)$ and $\rho(\Sigma) \vdash \rho(\tau) \geqslant^\diamond \rho(\tau_2)$. Then $\rho(\Sigma) \vdash \rho(\tau) \geqslant^\diamond \rho(\tau_1) \vee^\diamond \rho(\tau_2)$ by S-ANDOR2$\diamond$, i.e., $\rho(\Sigma) \vdash \rho(\tau) \geqslant^\diamond \rho(\tau_1 \vee^\diamond \tau_2)$.

**Case S-DISTRIB$\diamond$.** By the definition of type substitution, $\rho(\tau \wedge^\diamond (\tau_1 \vee^\diamond \tau_2)) = \rho(\tau) \wedge^\diamond \rho(\tau_1 \vee^\diamond \tau_2) = \rho(\tau) \wedge^\diamond (\rho(\tau_1) \vee^\diamond \rho(\tau_2))$ and $\rho((\tau \wedge^\diamond \tau_1) \vee^\diamond (\tau \wedge^\diamond \tau_2)) = \rho(\tau \wedge^\diamond \tau_1) \vee^\diamond \rho(\tau \wedge^\diamond \tau_2) = (\rho(\tau) \wedge^\diamond \rho(\tau_1)) \vee^\diamond (\rho(\tau) \wedge^\diamond \rho(\tau_2))$. By S-DISTRIB$\diamond$, $\rho(\tau) \wedge^\diamond (\rho(\tau_1) \vee^\diamond \rho(\tau_2)) \leqslant^\diamond (\rho(\tau) \wedge^\diamond \rho(\tau_1)) \vee^\diamond (\rho(\tau) \wedge^\diamond \rho(\tau_2))$, i.e., $\rho(\tau \wedge^\diamond (\tau_1 \vee^\diamond \tau_2)) \leqslant^\diamond \rho((\tau \wedge^\diamond \tau_1) \vee^\diamond (\tau \wedge^\diamond \tau_2))$.

**Case S-TRANS.** By IH on the premises, we have $\rho(\Sigma) \vdash \rho(\tau_0) \leqslant \rho(\tau_1)$ and $\rho(\Sigma) \vdash \rho(\tau_1) \leqslant \rho(\tau_2)$. Then $\rho(\Sigma) \vdash \rho(\tau_0) \leqslant \rho(\tau_2)$ by S-TRANS.

**Case S-WEAKEN.** By IH on the premise, we have $\rho(\tau_1) \leqslant \rho(\tau_2)$. Then $\rho(\Sigma) \vdash \rho(\tau_1) \leqslant \rho(\tau_2)$ by S-WEAKEN.

**Case S-ASSUM.** By the definition of subtyping context substitution, $\rho(\Xi \cdot \triangleright(\tau_1 \leqslant \tau_2)) = \rho(\Xi) \cdot \triangleright(\rho(\tau_1) \leqslant \rho(\tau_2))$. By IH on the premise, we have $\mathcal{D}{\cdot}\rho(\Xi \cdot \triangleright(\tau_1 \leqslant \tau_2)) \vdash \rho(\tau_1) \leqslant \rho(\tau_2)$, i.e., $\mathcal{D}{\cdot}\rho(\Xi) \cdot \triangleright(\rho(\tau_1) \leqslant \rho(\tau_2)) \vdash \rho(\tau_1) \leqslant \rho(\tau_2)$. Then $\mathcal{D}{\cdot}\rho(\Xi) \vdash \rho(\tau_1) \leqslant \rho(\tau_2)$ by S-ASSUM.

**Case S-HYP.** By the definition of subtyping context substitution and the $H \in \Sigma$ judgement, it is straightforward to show that if $(\tau \leqslant \tau') \in \Sigma$, then $(\rho(\tau) \leqslant \rho(\tau')) \in \rho(\Sigma)$ by induction on the size of $\Sigma$. Applying to the premise $(\tau_1 \leqslant \tau_2) \in \Sigma$, we have $(\rho(\tau_1) \leqslant \rho(\tau_2)) \in \rho(\Sigma)$. Then $\rho(\Sigma) \vdash \rho(\tau_1) \leqslant \rho(\tau_2)$ by S-HYP.

**Case S-FUNDEPTH.** By the definition of type substitution, $\rho(\tau \to \tau') = \rho(\tau) \to \rho(\tau')$. By IH on the premises, we have $\triangleleft\rho(\Sigma) \vdash \rho(\tau_0) \leqslant \rho(\tau_1)$ and $\triangleleft\rho(\Sigma) \vdash \rho(\tau_2) \leqslant \rho(\tau_3)$. Then $\triangleleft\rho(\Sigma) \vdash \rho(\tau_1) \to \rho(\tau_2) \leqslant \rho(\tau_0) \to \rho(\tau_3)$ by S-FUNDEPTH, i.e., $\triangleleft\rho(\Sigma) \vdash \rho(\tau_1 \to \tau_2) \leqslant \rho(\tau_0 \to \tau_3)$.

**Case S-FUNMRG$\diamond$.** By the definition of type substitution, $\rho((\tau_1 \vee^\diamond \tau_3) \to (\tau_2 \wedge^\diamond \tau_4)) = \rho(\tau_1 \vee^\diamond \tau_3) \to \rho(\tau_2 \wedge^\diamond \tau_4) = (\rho(\tau_1) \vee^\diamond \rho(\tau_3)) \to (\rho(\tau_2) \wedge^\diamond \rho(\tau_4))$. and $\rho(\tau_1 \to \tau_2 \wedge \tau_3 \to \tau_4) = \rho(\tau_1 \to \tau_2) \wedge \rho(\tau_3 \to \tau_4) = \rho(\tau_1) \to \rho(\tau_2) \wedge \rho(\tau_3) \to \rho(\tau_4)$. By S-FUNMRG$\diamond$, $(\rho(\tau_1) \vee^\diamond \rho(\tau_3)) \to (\rho(\tau_2) \wedge^\diamond \rho(\tau_4)) \geqslant^\diamond \rho(\tau_1) \to \rho(\tau_2) \wedge \rho(\tau_3) \to \rho(\tau_4)$, i.e., $\rho((\tau_1 \vee^\diamond \tau_3) \to (\tau_2 \wedge^\diamond \tau_4)) \geqslant^\diamond \rho(\tau_1 \to \tau_2 \wedge \tau_3 \to \tau_4)$.

**Case S-RCDDEPTH.** By the definition of type substitution, $\rho(\{x : \tau\}) = \{x : \rho(\tau)\}$. By IH on the premise, we have $\triangleleft\rho(\Sigma) \vdash \rho(\tau_1) \leqslant \rho(\tau_2)$. Then $\triangleleft\rho(\Sigma) \vdash \{x : \rho(\tau_1)\} \leqslant \{x : \rho(\tau_2)\}$ by S-RCDDEPTH, i.e., $\triangleleft\rho(\Sigma) \vdash \rho(\{x : \tau_1\}) \leqslant \rho(\{x : \tau_2\})$.

**Case S-RCDMRG$\diamond$.** By the definition of type substitution, $\rho(\{x : \tau_1 \vee^\diamond \tau_2\}) = \{x : \rho(\tau_1 \vee^\diamond \tau_2)\} = \{x : \rho(\tau_1) \vee^\diamond \rho(\tau_2)\}$ and $\rho(\{x : \tau_1\} \vee^\diamond \{x : \tau_2\}) = \rho(\{x : \tau_1\}) \vee^\diamond \rho(\{x : \tau_2\}) = \{x : \rho(\tau_1)\} \vee^\diamond \{x : \rho(\tau_2)\}$. By S-RCDMRG$\diamond$, $\{x :$

$\rho(\tau_1) \vee^{\diamond} \rho(\tau_2)\,\} \leqslant^{\diamond} \{\, x : \rho(\tau_1)\,\} \vee^{\diamond} \{\, x : \rho(\tau_2)\,\}$, i.e., $\rho(\{\, x : \tau_1 \vee^{\diamond} \tau_2\,\}) \leqslant^{\diamond} \rho(\{\, x : \tau_1\,\} \vee^{\diamond} \{\, x : \tau_2\,\})$.

**Case S-RcdTop.** By the definition of type substitution, $\rho(\top) = \top$ and $\rho(\{\, x : \tau_1\,\} \vee \tau) = \rho(\{\, x : \tau_1\,\}) \vee \rho(\tau) = \{\, x : \rho(\tau_1)\,\} \vee \rho(\tau)$. From the premise, we have $\rho(\tau) \in \{\rho(\{\, y^{\neq x} : \tau_2\,\}),\ \rho(\tau_2 \to \tau_3)\}$, i.e., $\rho(\tau) \in \{\{\, y^{\neq x} : \rho(\tau_2)\,\}\, \rho(\tau_2) \to \rho(\tau_3)\}$ by the definition of type substitution. Then $\tau \leqslant \{\, x : \rho(\tau_1)\,\} \vee \rho(\tau)$ by S-RcdTop, i.e., $\rho(\tau) \leqslant \rho(\{\, x : \tau_1\,\} \vee \tau)$.

**Case S-ClsSub.** Note that the declaration context rooted in by the subtyping context contains all the information required to determine the superclass relation, i.e., $\mathcal{S}_{\mathcal{D}\cdot\Sigma} = \mathcal{S}_{\mathcal{D}\cdot\Sigma'}$. Then the premise $C_2 \in \mathcal{S}(C_1[\overline{\alpha}])$ implies $C_2 \in \mathcal{S}(C_1[\overline{\alpha}])$. By the definition of type substitution, $\rho(\#C) = \#C$. Then $\rho(\Sigma) \vdash \#C_1 \leqslant \#C_2$ by S-ClsSub, i.e., $\rho(\Sigma) \vdash \rho(\#C_1) \leqslant \rho(\#C_2)$.

**Case S-ClsBot.** As noted in the case above, $\mathcal{S}_{\mathcal{D}\cdot\Sigma} = \mathcal{S}_{\mathcal{D}\cdot\Sigma'}$. By the definition of type substitution, $\rho(\#C_1 \wedge \#C_2) = \rho(\#C_1) \wedge \rho(\#C_2) = \#C_1 \wedge \#C_2$ and $\rho(\bot) = \bot$. Then the premise $C_1 \notin \mathcal{S}(C_2[\overline{\alpha}])$ and $C_2 \notin \mathcal{S}(C_1[\overline{\beta}])$ imply $C_1 \notin \mathcal{S}(C_2[\overline{\alpha}])$ and $C_2 \notin \mathcal{S}(C_1[\overline{\beta}])$. Then $\rho(\Sigma) \vdash \#C_1 \wedge \#C_2 \leqslant \bot$ by S-ClsBot, i.e., $\rho(\Sigma) \vdash \rho(\#C_1 \wedge \#C_2) \leqslant \rho(\bot)$.

**Case S-Exp$\diamond$.** We show that if $\Sigma \vdash \tau$ ***exp.*** $\tau'$, where $\mathcal{D}$ ***wf***, then $\rho(\Sigma) \vdash \rho(\tau)$ ***exp.*** $\rho(\tau')$. We consider rules that can derive the judgement $\Sigma \vdash \tau$ ***exp.*** $\tau'$.

**Case S-AlsExp.** Note that the declaration context contains all declarations, i.e., $d \in \Sigma$ implies $d \in \mathcal{D}\cdot\Sigma'$. Then the premise implies $(\textbf{type}\ A[\overline{\alpha_i}^{\,i \in S}] = \tau) \in \rho(\Sigma)$. By the definition of type substitution, $\rho(A[\overline{\tau_i}^{\,i \in S}]) = A[\overline{\rho(\tau_i)}^{\,i \in S}]$. By the well-formedness of $\mathcal{D}$, $TV(\tau) \subseteq \{\, \overline{\alpha_i}^{\,i \in S}\,\}$, which implies that all type variables in $[\overline{\alpha_i \mapsto \tau_i}^{\,i \in S}]\tau$ are introduced by the substitution $\{\, \overline{\alpha_i \mapsto \tau_i}^{\,i \in S}\,\}$, and $\rho([\overline{\alpha_i \mapsto \tau_i}^{\,i \in S}]\tau) = [\overline{\alpha_i \mapsto \rho(\tau_i)}^{\,i \in S}]\tau$. Then $\rho(\Sigma) \vdash A[\overline{\rho(\tau_i)}^{\,i \in S}]$ ***exp.*** $[\overline{\alpha_i \mapsto \rho(\tau_i)}^{\,i \in S}]\tau$ by S-AlsExp, i.e., $\rho(\Sigma) \vdash \rho(A[\overline{\tau_i}^{\,i \in S}])$ ***exp.*** $\rho([\overline{\alpha_i \mapsto \tau_i}^{\,i \in S}]\tau)$.

**Case S-ClsExp.** Similar to the case above, noting that $\rho(\#C \wedge [\overline{\alpha_i \mapsto \tau_i}^{\,i \in S}]\tau) = \rho(\#C) \wedge \rho([\overline{\alpha_i \mapsto \tau_i}^{\,i \in S}]\tau) = \#C \wedge \rho([\overline{\alpha_i \mapsto \tau_i}^{\,i \in S}]\tau)$.

Then the premise $\Sigma \vdash \tau$ ***exp.*** $\tau'$ implies $\rho(\Sigma) \vdash \rho(\tau)$ ***exp.*** $\rho(\tau')$, and $\rho(\Sigma) \vdash \rho(\tau) \geqslant^{\diamond} \rho(\tau')$ follows from S-Exp$\diamond$.

■

**Corollary A.30** (Preservation of subtyping entailment under substitution)**.** *If* $\Sigma \vDash \Sigma'$ *and* $\mathcal{D}$ ***wf****, then* $\rho(\Sigma) \vDash \rho(\Sigma')$.

**Proof** By induction on the derivation of subtyping entailment judgement $\Sigma \vDash \Sigma'$.

**Case S-Empty.** Immediate.

**Case S-Cons.** By the definition of subtyping context substitution, $\rho(\Sigma'\cdot(\tau_1 \leqslant \tau_2)) = \rho(\Sigma')\cdot(\rho(\tau_1) \leqslant \rho(\tau_2))$. By IH on the premise $\Sigma \vDash \Sigma'$, we have $\rho(\Sigma) \vDash \rho(\Sigma')$. By preservation of subtyping under substitution (Lemma A.29) on the premise $\Sigma \vdash \tau_1 \leqslant \tau_2$, we have $\rho(\Sigma) \vdash \rho(\tau_1) \leqslant \rho(\tau_2)$. Then $\rho(\Sigma) \vDash \rho(\Sigma')\cdot(\rho(\tau_1) \leqslant \rho(\tau_2))$ follows from S-Cons, i.e., $\rho(\Sigma) \vDash \rho(\Sigma'\cdot(\tau_1 \leqslant \tau_2))$.

■

**Lemma A.31** (Congruence of substitution on types)**.** *If* $\Sigma \vdash \pi \equiv \pi'$*, then* $\Sigma \vdash [\alpha \mapsto \pi]\tau \equiv [\alpha \mapsto \pi']\tau$ *for all* $\tau$*.*

**Proof** By straightforward induction on the syntax of $\tau$. The only non-trivial cases are:

**Case** $\tau = \tau_1 \to \tau_2$**.** From the assumption, we have:

$$\Sigma \vdash \pi \equiv \pi' \tag{1}$$

By Lemma A.23 with Lemma A.18, (1) implies:

$$\triangleleft\Sigma \vdash \pi \equiv \pi' \tag{2}$$

By IH on (2), we have:

$$\triangleleft\Sigma \vdash [\alpha \mapsto \pi]\tau_1 \equiv [\alpha \mapsto \pi']\tau_1 \tag{3}$$
$$\triangleleft\Sigma \vdash [\alpha \mapsto \pi]\tau_2 \equiv [\alpha \mapsto \pi']\tau_2 \tag{4}$$

Then by S-FunDepth on (3) and (4), we have:

$$\Sigma \vdash [\alpha \mapsto \pi](\tau_1 \to \tau_2) \equiv [\alpha \mapsto \pi'](\tau_1 \to \tau_2) \tag{5}$$

**Case** $\tau = \{\, x : \tau_1 \,\}$**.** From the assumption, we have:

$$\Sigma \vdash \pi \equiv \pi' \tag{6}$$

By Lemma A.23 with Lemma A.18, (6) implies:

$$\triangleleft\Sigma \vdash \pi \equiv \pi' \tag{7}$$

By IH on (7), we have:

$$\triangleleft\Sigma \vdash [\alpha \mapsto \pi]\tau_1 \equiv [\alpha \mapsto \pi']\tau_1 \tag{8}$$

Then by S-RcdDepth on (8) and (4), we have:

$$\Sigma \vdash [\alpha \mapsto \pi]\{\, x : \tau_1 \,\} \equiv [\alpha \mapsto \pi']\{\, x : \tau_1 \,\} \tag{9}$$

**Case** $\tau = \alpha$**.** From the assumption, we have:

$$\Sigma \vdash \pi \equiv \pi'$$
$$\text{i.e.,} \quad \Sigma \vdash [\alpha \mapsto \pi]\alpha \equiv [\alpha \mapsto \pi']\alpha \tag{10}$$

■

**Lemma A.32** (Congruence of substitution on guarded types)**.** *If* $\Sigma \vdash \pi \equiv \pi'$ *and* $\alpha \notin TTV(\tau)$*, then* $\triangleright\Sigma \vdash [\alpha \mapsto \pi]\tau \equiv [\alpha \mapsto \pi']\tau$*.*

**Proof** By straightforward induction on the syntax of $\tau$. The only non-trivial cases are:

**Case** $\tau = \tau_1 \to \tau_2$**.** From the assumption, we have:

$$\Sigma \vdash \pi \equiv \pi' \tag{1}$$

By Lemma A.23 with Lemma A.18, (1) implies:

$$\triangleleft\Sigma \vdash \pi \equiv \pi' \tag{2}$$

By Lemma A.31 on (2), we have:

$$\triangleleft\Sigma \vdash [\alpha \mapsto \pi]\tau_1 \equiv [\alpha \mapsto \pi']\tau_1 \tag{3}$$
$$\triangleleft\Sigma \vdash [\alpha \mapsto \pi]\tau_2 \equiv [\alpha \mapsto \pi']\tau_2 \tag{4}$$

Then by S-FunDepth on (3) and (4), we have:

$$\triangleright\Sigma \vdash [\alpha \mapsto \pi](\tau_1 \to \tau_2) \equiv [\alpha \mapsto \pi'](\tau_1 \to \tau_2) \tag{5}$$

**Case** $\tau = \{\, x : \tau_1 \,\}$**.** From the assumption, we have:

$$\Sigma \vdash \pi \equiv \pi' \tag{6}$$

By Lemma A.23 with Lemma A.18, (6) implies:

$$\triangleleft\Sigma \vdash \pi \equiv \pi' \tag{7}$$

By Lemma A.31 on (7), we have:

$$\triangleleft\Sigma \vdash [\alpha \mapsto \pi]\tau_1 \equiv [\alpha \mapsto \pi']\tau_1 \tag{8}$$

Then by S-RcdDepth on (8) and (4), we have:

$$\triangleright\Sigma \vdash [\alpha \mapsto \pi]\{\, x : \tau_1 \,\} \equiv [\alpha \mapsto \pi']\{\, x : \tau_1 \,\} \tag{9}$$

**Case** $\tau = \alpha$**.** Impossible since $\alpha \notin \mathit{TTV}(\tau)$.

■

**Corollary A.33.** $\Sigma \vdash \tau \equiv [\alpha \mapsto \alpha \wedge \mathit{ub}_\Sigma(\alpha) \vee \mathit{lb}_\Sigma(\alpha)]\tau$ *for all* $\tau$.

**Proof** By Lemma A.31 on $\Sigma \vdash \alpha \equiv \alpha \wedge \mathit{ub}_\Sigma(\alpha) \vee \mathit{lb}_\Sigma(\alpha)$. ■

**Corollary A.34.** *If* $\alpha \notin \mathit{TTV}(\tau)$*, then* $\triangleright\Sigma \vdash \tau \equiv [\alpha \mapsto \alpha \wedge \mathit{ub}_\Sigma(\alpha) \vee \mathit{lb}_\Sigma(\alpha)]\tau$.

**Proof** By Lemma A.32 on $\Sigma \vdash \alpha \equiv \alpha \wedge \mathit{ub}_\Sigma(\alpha) \vee \mathit{lb}_\Sigma(\alpha)$. ■

**Lemma A.35** (Inlining of bound)**.** *If* $\Sigma{\cdot}(\alpha \leqslant^\diamond \pi) \vdash \tau \leqslant \tau'$*, then* $\rho\Sigma{\cdot}\triangleright(\alpha \leqslant^\diamond \pi) \vdash \rho\tau \leqslant \rho\tau'$*, where* $\rho = [\alpha \mapsto \alpha \wedge^\diamond \pi]$.

**Proof** By straightforward induction on unassuming subtyping derivations. The only non-trivial case is S-Hyp when $(\tau \leqslant \tau') = (\alpha \leqslant^\diamond \pi)$.

**Case S-Hyp when** $(\tau \leqslant \tau') = (\alpha \leqslant^{\diamond} \pi)$**.** Let $\mathit{cleanup}((\alpha \leqslant^{\diamond} \pi)) = (\alpha \leqslant^{\diamond} \pi')$. By Lemma A.2, Lemma A.3, and Lemma A.4, we have:

$$(\alpha \leqslant^{\diamond} \pi) \rightrightarrows (\alpha \leqslant^{\diamond} \pi') \tag{1}$$

$$(\alpha \leqslant^{\diamond} \pi') \ \textbf{\textit{guard.}} \tag{2}$$

$$\alpha \wedge^{\diamond} \pi \equiv \alpha \wedge^{\diamond} \pi' \tag{3}$$

By S-Trans on $(\alpha \leqslant^{\diamond} \pi') \vdash \alpha \equiv \alpha \wedge^{\diamond} \pi'$ and (3), we have:

$$(\alpha \leqslant^{\diamond} \pi') \vdash \alpha \equiv \alpha \wedge^{\diamond} \pi \tag{4}$$

By Lemma A.32 on (2) and (4), we have:

$$\triangleright(\alpha \leqslant^{\diamond} \pi') \vdash \pi' \equiv \rho\pi' \tag{5}$$

By Lemma A.7 on (4) and (5), we have:

$$\triangleright(\alpha \leqslant^{\diamond} \pi') \vdash \alpha \wedge^{\diamond} \pi' \equiv (\alpha \wedge^{\diamond} \pi) \wedge^{\diamond} \rho\pi'$$

$$\text{i.e.,} \quad \triangleright(\alpha \leqslant^{\diamond} \pi') \vdash \alpha \wedge^{\diamond} \pi' \equiv \rho(\alpha \wedge^{\diamond} \pi') \tag{6}$$

By S-Trans on (3) and S-AndOr12$\diamond$, we have:

$$\alpha \wedge^{\diamond} \pi' \leqslant^{\diamond} \pi \tag{7}$$

By Lemma A.29, (7) implies:

$$\rho(\alpha \wedge^{\diamond} \pi) \leqslant^{\diamond} \rho\pi \tag{8}$$

Then by S-Trans on (3), (6), and (8), we have:

$$\triangleright(\alpha \leqslant^{\diamond} \pi') \vdash \alpha \wedge^{\diamond} \pi \leqslant^{\diamond} \rho\pi \tag{9}$$

Then by Lemma A.23 with (1), (9) implies:

$$\triangleright(\alpha \leqslant^{\diamond} \pi) \vdash \alpha \wedge^{\diamond} \pi \leqslant^{\diamond} \rho\pi$$

$$\text{i.e.,} \quad \triangleright(\alpha \leqslant^{\diamond} \pi) \vdash \rho\alpha \leqslant^{\diamond} \rho\pi \tag{10}$$

■

### *A.6 Lemmas on Consistency*

**Proof** [Lemma 3.3] From the assumptions, we have:

$$\Sigma \vdash \Xi \ \textbf{\textit{cons.}} \tag{1}$$

$$\Sigma \cdot \Xi \vdash \tau \leqslant \tau' \tag{2}$$

From the definition of weak consistecy, (1) implies:

$$\rho\Sigma \cdot \overline{\triangleright(\alpha \equiv \tau)}^{(\alpha \mapsto \tau) \in \rho} \models \rho\Xi \tag{3}$$

for some $\rho$. By Lemma A.29, on (2) implies:

$$\rho\Sigma \cdot \rho\Xi \vdash \rho\tau \leqslant \rho\tau' \tag{4}$$

Then by Lemma A.23 with (3), (4) implies:

$$\rho\Sigma{\cdot}\overline{\triangleright(\alpha \equiv \tau)}^{(\alpha \mapsto \tau) \in \rho} \vdash \rho\tau \leqslant \rho\tau' \tag{5}$$

■

**Proof** [Lemma 3.4] From the assumptions, we have:

$$\Sigma \vdash \Xi \ \textit{cons.} \tag{1}$$
$$\Sigma{\cdot}\Xi \vdash \tau \leqslant \tau' \tag{2}$$

By Lemma 3.3 on (1) and (2), we have:

$$\rho\Sigma{\cdot}\overline{\triangleright(\alpha \equiv \pi)}^{(\alpha \mapsto \pi) \in \rho} \vdash \rho\tau \leqslant \rho\tau' \tag{3}$$

for some $\rho$. By S-Hyp, we have:

$$\frac{}{\overline{(\alpha \equiv \pi)}^{(\alpha \mapsto \pi) \in \rho} \vdash \beta \equiv \pi'}^{(\beta \mapsto \pi') \in \rho} \tag{4}$$

By Lemma A.32 on (4), we have

$$\overline{\triangleright(\alpha \equiv \pi)}^{(\alpha \mapsto \pi) \in \rho} \vdash \tau \equiv \rho\tau \tag{5}$$
$$\overline{\triangleright(\alpha \equiv \pi)}^{(\alpha \mapsto \pi) \in \rho} \vdash \tau' \equiv \rho\tau' \tag{6}$$

Then by S-Trans on (5), (4), and (6), we have:

$$\rho\Sigma{\cdot}\overline{\triangleright(\alpha \equiv \pi)}^{(\alpha \mapsto \pi) \in \rho} \vdash \tau \leqslant \tau' \tag{7}$$

■

**Proof** [Lemma 3.6] By induction on consistency derivationsfor the following statement: if $\Sigma \vdash \triangleright\Xi_\triangleright{\cdot}\Xi\,;\ \rho\ \textit{cons.}$, then $\triangleright\Xi_\triangleright{\cdot}\overline{\triangleright(\alpha \equiv \tau)}^{(\alpha \mapsto \tau) \in \rho}{\cdot}\rho\Sigma \models \rho\Xi$.

If $\Xi$ is not guarded, we can replace it with $\mathit{cleanup}(\Xi)$ before applying the lemma, and restore it back to $\Xi$ in the conclusion. Therefore we can assume $\Xi$ *guard.*.

**Base case.** The base case is trivial since $\Xi = \epsilon$.

**Inductive case.** For the inductive case, we have $\rho = \rho_2 \circ \rho_1$ for some $\rho_1 = [\alpha \mapsto \alpha \wedge \mathit{ub}_\Xi(\alpha) \vee \mathit{lb}_\Xi(\alpha)]$, $\rho_2$, and $\alpha$. The premises of the rule are:

$$\triangleright\Xi_\triangleright{\cdot}\triangleright\Xi_\alpha{\cdot}\rho_1\Xi_{\not\alpha}{\cdot}\rho_1\Sigma \models \rho_1\Xi_\alpha \tag{1}$$
$$\rho_1\Sigma \vdash \triangleright\Xi_\triangleright{\cdot}\triangleright\Xi_\alpha{\cdot}\rho_1\Xi_{\not\alpha}\ ;\ \rho_2\ \textit{cons.} \tag{2}$$

where $\mathit{split}_\alpha(\Xi,\ \mathit{dom}(\rho_2)) = (\Xi_\alpha,\ \Xi_{\not\alpha})$. By IH on (2), we have:

$$\triangleright\Xi_\triangleright{\cdot}\overline{\triangleright(\beta \equiv \tau)}^{(\beta \mapsto \tau) \in \rho_2}{\cdot}\rho_2\rho_1\Sigma \models \rho_2\rho_1\Xi_{\not\alpha}$$
$$\text{i.e.,}\quad \triangleright\Xi_\triangleright{\cdot}\overline{\triangleright(\beta \equiv \tau)}^{(\beta \mapsto \tau) \in \rho_2}{\cdot}\rho\Sigma \models \rho\Xi_{\not\alpha} \tag{3}$$

By Lemma A.23 with Lemma A.18, (3) implies:

$$\triangleright\Xi_{\triangleright}\cdot\triangleright\overline{(\beta\equiv\tau)}^{(\beta\mapsto\tau)\in\rho_2}\cdot\triangleright\rho_2(\alpha\equiv\alpha\wedge ub_{\Xi}(\alpha)\vee lb_{\Xi}(\alpha))\cdot\rho\Sigma\vDash\rho\Xi_{\mathscr{A}}$$
$$\text{i.e.,}\quad \triangleright\Xi_{\triangleright}\cdot\triangleright\overline{(\beta\equiv\tau)}^{(\beta\mapsto\tau)\in\rho}\cdot\rho\Sigma\vDash\rho\Xi_{\mathscr{A}} \tag{4}$$

By S-AndOr2$\supset$ on (1), we have:

$$\triangleright\Xi_{\triangleright}\cdot\triangleright\Xi_{\alpha}\cdot\rho_1\Xi_{\mathscr{A}}\cdot\rho_1\Sigma\vDash\rho_1\alpha\leqslant\rho_1 ub_{\Xi_{\alpha}}(\alpha)$$
$$\text{i.e.,}\quad \triangleright\Xi_{\triangleright}\cdot\triangleright\Xi_{\alpha}\cdot\rho_1\Xi_{\mathscr{A}}\cdot\rho_1\Sigma\vDash\alpha\wedge ub_{\Xi}(\alpha)\vee lb_{\Xi}(\alpha)\leqslant\rho_1 ub_{\Xi}(\alpha) \tag{5}$$

By S-Trans on S-AndOr12$\cdot$ and (5), we have:

$$\triangleright\Xi_{\triangleright}\cdot\triangleright\Xi_{\alpha}\cdot\rho_1\Xi_{\mathscr{A}}\cdot\rho_1\Sigma\vDash lb_{\Xi}(\alpha)\leqslant\rho_1 ub_{\Xi}(\alpha) \tag{6}$$

By Corollary A.34, we have:

$$\triangleright\Xi_{\alpha}\vdash\tau\equiv[\alpha\mapsto\alpha\wedge ub_{\Xi_{\alpha}}(\alpha)\vee lb_{\Xi_{\alpha}}(\alpha)]\tau$$
$$\text{i.e.,}\quad \triangleright\Xi_{\alpha}\vdash\tau\equiv[\alpha\mapsto\alpha\wedge ub_{\Xi}(\alpha)\vee lb_{\Xi}(\alpha)]\tau$$
$$\text{i.e.,}\quad \triangleright\Xi_{\alpha}\vdash\tau\equiv\rho_1\tau \tag{7}$$

for all $\tau$ where $\alpha\notin TTV(\tau)$. Since $\Xi$ ***guard.***, we have $\alpha\notin TTV(ub_{\Xi}(\alpha))$. Then by S-Trans on (6) and (7), we have:

$$\triangleright\Xi_{\triangleright}\cdot\triangleright\Xi_{\alpha}\cdot\rho_1\Xi_{\mathscr{A}}\cdot\rho_1\Sigma\vDash lb_{\Xi}(\alpha)\leqslant ub_{\Xi}(\alpha) \tag{8}$$

By S-Trans on S-AndOr11$\cdot$/S-AndOr12$\cdot$ and S-Hyp, we have:

$$(\alpha\equiv\alpha\wedge ub_{\Xi}(\alpha)\vee lb_{\Xi}(\alpha))\cdot(lb_{\Xi}(\alpha)\leqslant ub_{\Xi}(\alpha))\vdash\tau\leqslant\alpha \tag{9}$$

for each $(\tau\leqslant\alpha)\in\Xi_{\alpha}$. By S-Trans on S-Hyp and Lemma A.7$\cdot$ on S-AndOr12$\supset$ and S-Refl, we have:

$$(\alpha\equiv\alpha\wedge ub_{\Xi}(\alpha)\vee lb_{\Xi}(\alpha))\cdot(lb_{\Xi}(\alpha)\leqslant ub_{\Xi}(\alpha))\vdash\alpha\leqslant ub_{\Xi}(\alpha)\vee lb_{\Xi}(\alpha) \tag{10}$$

By S-Trans on (10) and S-AndOr2$\cdot$ on S-Refl and S-Hyp, we have:

$$(\alpha\equiv\alpha\wedge ub_{\Xi}(\alpha)\vee lb_{\Xi}(\alpha))\cdot(lb_{\Xi}(\alpha)\leqslant ub_{\Xi}(\alpha))\vdash\alpha\leqslant ub_{\Xi}(\alpha) \tag{11}$$

By S-Trans on (11) and S-AndOr11$\supset$/S-AndOr12$\supset$, we have:

$$(\alpha\equiv\alpha\wedge ub_{\Xi}(\alpha)\vee lb_{\Xi}(\alpha))\cdot(lb_{\Xi}(\alpha)\leqslant ub_{\Xi}(\alpha))\vdash\alpha\leqslant\tau \tag{12}$$

for each $(\alpha\leqslant\tau)\in\Xi_{\alpha}$. Then (9) and (12) imply:

$$(\alpha\equiv\alpha\wedge ub_{\Xi}(\alpha)\vee lb_{\Xi}(\alpha))\cdot(lb_{\Xi}(\alpha)\leqslant ub_{\Xi}(\alpha))\vDash\Xi_{\alpha} \tag{13}$$

Then by Lemma A.23 with Lemma A.21 on (13), (8) implies:

$$\triangleright\Xi_{\triangleright}\cdot\triangleright(\alpha\equiv\alpha\wedge ub_{\Xi}(\alpha)\vee lb_{\Xi}(\alpha))\cdot lb_{\Xi}(\alpha)\leqslant ub_{\Xi}(\alpha)\cdot\rho_1\Xi_{\mathscr{A}}\cdot\rho_1\Sigma\vDash lb_{\Xi}(\alpha)\leqslant ub_{\Xi}(\alpha) \tag{14}$$

By S-Assum on (14), we have:

$$\triangleright\Xi_{\triangleright}\cdot\triangleright(\alpha\equiv\alpha\wedge ub_{\Xi}(\alpha)\vee lb_{\Xi}(\alpha))\cdot\rho_1\Xi_{\mathscr{A}}\cdot\rho_1\Sigma\vDash lb_{\Xi}(\alpha)\leqslant ub_{\Xi}(\alpha) \tag{15}$$

By Lemma A.23 with (15), (13) implies:

$$\triangleright\Xi_{\triangleright}\cdot(\alpha \equiv \alpha \wedge ub_{\Xi}(\alpha) \vee lb_{\Xi}(\alpha))\cdot\rho_1\Xi_{\not\alpha}\cdot\rho_1\Sigma \vDash \Xi_{\alpha} \tag{16}$$

By Lemma A.23 with Lemma A.21 on (16), (1) implies:

$$\triangleright\Xi_{\triangleright}\cdot\triangleright(\alpha \equiv \alpha \wedge ub_{\Xi}(\alpha) \vee lb_{\Xi}(\alpha))\cdot\rho_1\Xi_{\not\alpha}\cdot\rho_1\Sigma \vDash \rho_1\Xi_{\alpha} \tag{17}$$

By Lemma A.29, (17) implies:

$$\triangleright\rho_2\Xi_{\triangleright}\cdot\triangleright\rho_2(\alpha \equiv \alpha \wedge ub_{\Xi}(\alpha) \vee lb_{\Xi}(\alpha))\cdot\rho_2\rho_1\Xi_{\not\alpha}\cdot\rho_2\rho_1\Sigma \vDash \rho_2\rho_1\Xi_{\alpha}$$

$$\text{i.e.,}\quad \triangleright\rho_2\Xi_{\triangleright}\cdot\triangleright\rho_2(\alpha \equiv \alpha \wedge ub_{\Xi}(\alpha) \vee lb_{\Xi}(\alpha))\cdot\rho\Xi_{\not\alpha}\cdot\rho\Sigma \vDash \rho\Xi_{\alpha} \tag{18}$$

By Lemma A.23 with (4), (18) implies:

$$\triangleright\rho_2\Xi_{\triangleright}\cdot\triangleright\Xi_{\triangleright}\cdot\overline{\triangleright(\beta \equiv \tau)}^{(\beta\mapsto\tau)\in\rho}\cdot\rho\Sigma \vDash \rho\Xi_{\alpha} \tag{19}$$

By Lemma A.31 on S-Hyp, we have:

$$\overline{\triangleright(\beta \equiv \tau)}^{(\beta\mapsto\tau)\in\rho_2} \vdash \pi \equiv \rho_2\pi \tag{20}$$

for all $\pi$. By S-Trans on S-Hyp and (20), we have:

$$\Xi_{\triangleright}\cdot\overline{\triangleright(\beta \equiv \tau)}^{(\beta\mapsto\tau)\in\rho_2} \vDash \rho_2\Xi_{\triangleright} \tag{21}$$

By Lemma A.23 with Lemma A.21 on (21), (19) implies:

$$\triangleright\Xi_{\triangleright}\cdot\overline{\triangleright(\beta \equiv \tau)}^{(\beta\mapsto\tau)\in\rho}\cdot\rho\Sigma \vDash \rho\Xi_{\alpha} \tag{22}$$

Then by Lemma A.20, (22) and (4) imply:

$$\triangleright\Xi_{\triangleright}\cdot\overline{\triangleright(\beta \equiv \tau)}^{(\beta\mapsto\tau)\in\rho}\cdot\rho\Sigma \vDash \rho\Xi_{\alpha}\cdot\rho\Xi_{\not\alpha}$$

$$\text{i.e.,}\quad \triangleright\Xi_{\triangleright}\cdot\overline{\triangleright(\beta \equiv \tau)}^{(\beta\mapsto\tau)\in\rho}\cdot\rho\Sigma \vDash \rho\Xi \tag{23}$$

■

**Lemma A.36** (Congruence of substitution on consistency)**.** *If* $[\alpha \mapsto \tau]\Sigma \vdash \triangleright\Xi_{\triangleright}\cdot[\alpha \mapsto \tau]\Xi$ ; $\rho$ ***cons.*** *and* $\triangleright\Xi_{\triangleright} \vdash \tau \equiv \tau'$*, where* $\tau$ *and* $\tau'$ *are not type variables, then* $[\alpha \mapsto \tau']\Sigma \vdash \triangleright\Xi_{\triangleright}\cdot[\alpha \mapsto \tau']\Xi$ ; $\rho'$ ***cons.*** *for some* $\rho'$*, where* $dom(\rho') = dom(\rho)$*.*

**Proof** By induction on consistency derivations for the statement: if $\rho''[\alpha \mapsto \tau]\Sigma \vdash \triangleright\Xi_{\triangleright}\cdot\rho''[\alpha \mapsto \tau]\Xi$ ; $\rho$ ***cons.*** and $\triangleright\Xi_{\triangleright} \vdash \tau \equiv \tau'$ and $\overline{\Xi_{\triangleright} \vdash \gamma \equiv \tau_{\gamma}}^{(\gamma \mapsto \tau_{\gamma})\in\rho''}$, where $\tau$ and $\tau'$ are not type variables and $\overline{\gamma = \gamma'}^{(\gamma \mapsto \gamma')\in\rho''}$ and $dom(\rho) \cap dom(\rho'') = \varnothing$, then $\rho''[\alpha \mapsto \tau']\Sigma \vdash \triangleright\Xi_{\triangleright}\cdot\rho''[\alpha \mapsto \tau']\Xi$ ; $\rho'$ ***cons.*** for some $\rho'$, where $dom(\rho') = dom(\rho)$.

**Base case.** For the base case, we have $\Xi = \epsilon$. Then by the base case of the definition of consistency, we have:

$$\rho''[\alpha \mapsto \tau']\Sigma \vdash \triangleright\Xi_{\triangleright}\cdot\rho''[\alpha \mapsto \tau']\Xi\,;\ id\ \textbf{\textit{cons.}} \tag{1}$$

**Inductive case on $\alpha$.** For the inductive case on $\alpha$, i.e., where $\rho = \rho_2 \circ \rho_1$ for some $\rho_1$ and $\rho_2$, where $dom(\rho_1) = \{ \alpha \}$, the preimses of the rule are:

$$\triangleright\Xi_\triangleright \cdot \triangleright\Xi'_\alpha \cdot \rho_1\Xi'_{\not\alpha} \cdot \rho_1\rho''[\alpha \mapsto \tau]\Sigma \vDash \rho_1\Xi'_\alpha \tag{2}$$

$$\rho_1\rho''[\alpha \mapsto \tau]\Sigma \vdash \triangleright\Xi_\triangleright \cdot \triangleright\Xi'_\alpha \cdot \rho_1\Xi'_{\not\alpha}\ ;\ \rho_2\ \textbf{\textit{cons}}. \tag{3}$$

where $split_\alpha(\rho''[\alpha \mapsto \tau]\Xi,\ dom(\rho_2)) = (\Xi'_\alpha,\ \Xi'_{\not\alpha})$ and $\rho_1 = [\alpha \mapsto \alpha \wedge ub_{\rho''[\alpha \mapsto \tau]\Xi}(\alpha) \vee lb_{\rho''[\alpha \mapsto \tau]\Xi}(\alpha)]$. Since $\tau$ is not a type varialbe, we have:

$$\Xi'_\alpha = \epsilon \tag{4}$$

$$\Xi'_{\not\alpha} = \rho''[\alpha \mapsto \tau]\Xi \tag{5}$$

$$\rho_1 = [\alpha \mapsto \alpha] \tag{6}$$

Then (3) implies:

$$\rho''[\alpha \mapsto \tau]\Sigma \vdash \triangleright\Xi_\triangleright \cdot \rho''[\alpha \mapsto \tau]\Xi\ ;\ \rho_2\ \textbf{\textit{cons}}. \tag{7}$$

Then by IH on (7), we have:

$$\rho''[\alpha \mapsto \tau']\Sigma \vdash \triangleright\Xi_\triangleright \cdot \rho''[\alpha \mapsto \tau']\Xi\ ;\ \rho_2\ \textbf{\textit{cons}}.$$

$$\text{i.e.,}\quad \rho''[\alpha \mapsto \tau']\Sigma \vdash \triangleright\Xi_\triangleright \cdot \rho''[\alpha \mapsto \tau']\Xi\ ;\ \rho_2 \circ \rho_1\ \textbf{\textit{cons}}. \tag{8}$$

for some $\rho'_2$, where $dom(\rho'_2) = dom(\rho_2)$.

**Inductive case not on $\alpha$.** For the inductive case not on $\alpha$, i.e., where $\rho = \rho_2 \circ \rho_1$ for some $\rho_1$ and $\rho_2$ and $\beta \neq \alpha$, where $dom(\rho_1) = \{ \beta \}$, the premises of the rule are:

$$\triangleright\Xi_\triangleright \cdot \triangleright\Xi'_\beta \cdot \rho_1\Xi'_{\not\beta} \cdot \rho_1\rho''[\alpha \mapsto \tau]\Sigma \vDash \rho_1\Xi'_\beta \tag{9}$$

$$\rho_1\rho''[\alpha \mapsto \tau]\Sigma \vdash \triangleright\Xi_\triangleright \cdot \triangleright\Xi'_\beta \cdot \rho_1\Xi'_{\not\beta}\ ;\ \rho_2\ \textbf{\textit{cons}}. \tag{10}$$

where $split_\beta(\rho''[\alpha \mapsto \tau]\Xi,\ dom(\rho_2)) = (\Xi'_\beta,\ \Xi'_{\not\beta})$ and $\rho_1 = [\beta \mapsto \beta \wedge ub_{\rho''[\alpha \mapsto \tau]\Xi}(\beta) \vee lb_{\rho''[\alpha \mapsto \tau]\Xi}(\beta)]$. Let $split_\beta(\Xi,\ dom(\rho_2)) = (\Xi_\beta,\ \Xi_{\not\beta})$. Since $\tau$ is not a type variable and $\overline{\gamma = \gamma'}^{(\gamma \mapsto \gamma') \in \rho''}$, we have $\Xi'_\beta = \rho''[\alpha \mapsto \tau]\Xi_\beta$ and $\Xi'_{\not\beta} = \rho''[\alpha \mapsto \tau]\Xi_{\not\beta}$. Then (9) and (10) imply:

$$\triangleright\Xi_\triangleright \cdot \triangleright\rho''[\alpha \mapsto \tau]\Xi_\beta \cdot \rho_1\rho''[\alpha \mapsto \tau]\Xi_{\not\beta} \cdot \rho_1\rho''[\alpha \mapsto \tau]\Sigma \vDash \rho_1\rho''[\alpha \mapsto \tau]\Xi_\beta \tag{11}$$

$$\rho_1\rho''[\alpha \mapsto \tau]\Sigma \vdash \triangleright\Xi_\triangleright \cdot \triangleright\rho''[\alpha \mapsto \tau]\Xi_\beta \cdot \rho_1\rho''[\alpha \mapsto \tau]\Xi_{\not\beta}\ ;\ \rho_2\ \textbf{\textit{cons}}. \tag{12}$$

Expanding the composition, we have:

$$\rho_1 \circ \rho'' = [\overline{\gamma \mapsto \rho_1\tau_\gamma}^{(\gamma \mapsto \tau_\gamma) \in \rho''},\ \beta \mapsto \beta \wedge ub_{\rho''[\alpha \mapsto \tau]\Xi}(\beta) \vee lb_{\rho''[\alpha \mapsto \tau]\Xi}(\beta)] \tag{13}$$

From the assumption, we have:

$$\overline{\triangleright\Xi_\triangleright \vdash \gamma \equiv \tau_\gamma}^{(\gamma \mapsto \tau_\gamma) \in \rho''} \tag{14}$$

By Corollary A.33, we have:

$$\rho''[\alpha \mapsto \tau]\Xi_\beta \vdash \pi \equiv [\beta \mapsto \beta \wedge ub_{\rho''[\alpha \mapsto \tau]\Xi_\beta}(\beta) \vee lb_{\rho''[\alpha \mapsto \tau]\Xi_\beta}(\beta)]\pi \quad \textit{for all } \pi$$

$$\text{i.e.,}\quad \rho''[\alpha \mapsto \tau]\Xi_\beta \vdash \pi \equiv [\beta \mapsto \beta \wedge ub_{\rho''[\alpha \mapsto \tau]\Xi}(\beta) \vee lb_{\rho''[\alpha \mapsto \tau]\Xi}(\beta)]\pi \quad \textit{for all } \pi$$

$$\text{i.e.,}\quad \rho''[\alpha \mapsto \tau]\Xi_\beta \vdash \pi \equiv \rho_1\pi \quad \textit{for all } \pi \tag{15}$$

By S-TRANS on (14) and (15), we have:

$$\frac{}{\triangleright\Xi_{\triangleright}\cdot\rho''[\alpha\mapsto\tau]\Xi_{\beta}\vdash\gamma\equiv\rho_1\tau_{\gamma}}\,(\gamma\mapsto\tau_{\gamma})\in\rho'' \tag{16}$$

Taking $\pi=\beta$, (15) implies:

$$\rho''[\alpha\mapsto\tau]\Xi_{\beta}\vdash\beta\equiv\beta\wedge ub_{\rho''[\alpha\mapsto\tau]\Xi}(\beta)\vee lb_{\rho''[\alpha\mapsto\tau]\Xi}(\beta) \tag{17}$$

Then (16) and (17) imply:

$$\frac{}{\triangleright\Xi_{\triangleright}\cdot\rho''[\alpha\mapsto\tau]\Xi_{\beta}\vdash\gamma\equiv\tau_{\gamma}}\,(\gamma\mapsto\tau_{\gamma})\in\rho_1\circ\rho'' \tag{18}$$

Then by IH on (12) and (18), we have:

$$\rho_1\rho''[\alpha\mapsto\tau']\Sigma\vdash\triangleright\Xi_{\triangleright}\cdot\triangleright\rho''[\alpha\mapsto\tau]\Xi_{\beta}\cdot\rho_1\rho''[\alpha\mapsto\tau']\Xi_{\not\beta}\;;\;\rho_2'\;\textbf{\textit{cons.}} \tag{19}$$

for some $\rho_2'$, where $dom(\rho_2')=dom(\rho_2)$.
From the assumptions, we have:

$$\triangleright\Xi_{\triangleright}\vdash\tau\equiv\tau' \tag{20}$$

By Lemma A.31, (20) implies:

$$\triangleright\Xi_{\triangleright}\vdash[\alpha\mapsto\tau]\pi\equiv[\alpha\mapsto\tau']\pi\quad\textit{for all }\pi \tag{21}$$

By S-TRANS on Lemma A.18 and (21), we have:

$$\triangleright\Xi_{\triangleright}\cdot[\alpha\mapsto\tau']\Xi_{\beta}\models[\alpha\mapsto\tau]\Xi_{\beta} \tag{22}$$

By Lemma A.29, (22) implies:

$$\triangleright\rho''\Xi_{\triangleright}\cdot\rho''[\alpha\mapsto\tau']\Xi_{\beta}\models\rho''[\alpha\mapsto\tau]\Xi_{\beta} \tag{23}$$

By Lemma A.31, (14) implies:

$$\Xi_{\triangleright}\vdash\pi\equiv\rho''\pi\quad\textit{for all }\pi \tag{24}$$

By S-TRANS on Lemma A.23 and (24), we have

$$\Xi_{\triangleright}\models\rho''\Xi_{\triangleright} \tag{25}$$

Then by Lemma A.23 with (25), (23) implies:

$$\triangleright\Xi_{\triangleright}\cdot\rho''[\alpha\mapsto\tau']\Xi_{\beta}\models\rho''[\alpha\mapsto\tau]\Xi_{\beta} \tag{26}$$

Then by Lemma A.23 with (26), (19) implies:

$$\rho_1\rho''[\alpha\mapsto\tau']\Sigma\vdash\triangleright\Xi_{\triangleright}\cdot\triangleright\rho''[\alpha\mapsto\tau']\Xi_{\beta}\cdot\rho_1\rho''[\alpha\mapsto\tau']\Xi_{\not\beta}\;;\;\rho_2'\;\textbf{\textit{cons.}} \tag{27}$$

Similarly, we have:

$$\triangleright\Xi_{\triangleright}\cdot\rho''[\alpha\mapsto\tau']\Xi_{\not\beta}\cdot\rho''[\alpha\mapsto\tau']\Sigma\models\rho''[\alpha\mapsto\tau]\Xi_{\not\beta}\cdot\rho''[\alpha\mapsto\tau]\Sigma \tag{28}$$

By Lemma A.29, (28) implies:

$$\triangleright\rho_1\Xi_{\triangleright}\cdot\rho_1\rho''[\alpha\mapsto\tau']\Xi_{\not\beta}\cdot\rho_1\rho''[\alpha\mapsto\tau']\Sigma\models\rho_1\rho''[\alpha\mapsto\tau]\Xi_{\not\beta}\cdot\rho_1\rho''[\alpha\mapsto\tau]\Sigma \tag{29}$$

By S-TRANS on Lemma A.18 and (15), we have:

$$\Xi_{\triangleright}\cdot\rho''[\alpha\mapsto\tau]\Xi_{\beta}\models\rho_1\Xi_{\triangleright} \tag{30}$$

Then by Lemma A.23 with (30), (29) implies:

$$\triangleright\Xi_{\triangleright}\cdot\triangleright\rho''[\alpha\mapsto\tau]\Xi_{\beta}\cdot\rho_1\rho''[\alpha\mapsto\tau']\Xi_{\not\beta}\cdot\rho_1\rho''[\alpha\mapsto\tau']\Sigma \vDash \rho_1\rho''[\alpha\mapsto\tau]\Xi_{\not\beta}\cdot\rho_1\rho''[\alpha\mapsto\tau]\Sigma \tag{31}$$

Then by Lemma A.23 with (31), (11) implies:

$$\triangleright\Xi_{\triangleright}\cdot\triangleright\rho''[\alpha\mapsto\tau]\Xi_{\beta}\cdot\rho_1\rho''[\alpha\mapsto\tau']\Xi_{\not\beta}\cdot\rho_1\rho''[\alpha\mapsto\tau']\Sigma \vDash \rho_1\rho''[\alpha\mapsto\tau]\Xi_{\beta} \tag{32}$$

Similarly, we have:

$$\triangleright\Xi_{\triangleright}\cdot\triangleright\rho''[\alpha\mapsto\tau]\Xi_{\beta}\cdot\rho_1\rho''[\alpha\mapsto\tau]\Xi_{\beta} \vDash \rho_1\rho''[\alpha\mapsto\tau']\Xi_{\beta} \tag{33}$$

Then by Lemma A.19 on (32) and (33), we have:

$$\triangleright\Xi_{\triangleright}\cdot\triangleright\rho''[\alpha\mapsto\tau]\Xi_{\beta}\cdot\rho_1\rho''[\alpha\mapsto\tau']\Xi_{\not\beta}\cdot\rho_1\rho''[\alpha\mapsto\tau']\Sigma \vDash \rho_1\rho''[\alpha\mapsto\tau']\Xi_{\beta} \tag{34}$$

Then by Lemma A.23 with (26), (34) implies:

$$\triangleright\Xi_{\triangleright}\cdot\triangleright\rho''[\alpha\mapsto\tau']\Xi_{\beta}\cdot\rho_1\rho''[\alpha\mapsto\tau']\Xi_{\not\beta}\cdot\rho_1\rho''[\alpha\mapsto\tau']\Sigma \vDash \rho_1\rho''[\alpha\mapsto\tau']\Xi_{\beta} \tag{35}$$

Let $\rho_1' = [\beta\mapsto\beta\wedge ub_{\rho''[\alpha\mapsto\tau']\Xi}(\beta)\vee lb_{\rho''[\alpha\mapsto\tau']\Xi}(\beta)]$. Since $\tau$ and $\tau'$ are not type variables and $\overline{\gamma=\gamma'}^{(\gamma\mapsto\gamma')\in\rho''}$, we have:

$$\rho_1 = [\beta\mapsto\beta\wedge\rho''[\alpha\mapsto\tau]ub_{\Xi}(\beta)\vee\rho''[\alpha\mapsto\tau]lb_{\Xi}(\beta)] \tag{36}$$
$$\rho_1' = [\beta\mapsto\beta\wedge\rho''[\alpha\mapsto\tau']ub_{\Xi}(\beta)\vee\rho''[\alpha\mapsto\tau']lb_{\Xi}(\beta)] \tag{37}$$

By Lemma A.29, (21) implies:

$$\triangleright\rho''\Xi_{\triangleright}\vdash\rho''[\alpha\mapsto\tau]ub_{\Xi}(\beta)\equiv\rho''[\alpha\mapsto\tau']ub_{\Xi}(\beta) \tag{38}$$
$$\triangleright\rho''\Xi_{\triangleright}\vdash\rho''[\alpha\mapsto\tau]lb_{\Xi}(\beta)\equiv\rho''[\alpha\mapsto\tau']lb_{\Xi}(\beta) \tag{39}$$

By Lemma A.23 with (25), (38) and (39) imply:

$$\triangleright\Xi_{\triangleright}\vdash\rho''[\alpha\mapsto\tau]ub_{\Xi}(\beta)\equiv\rho''[\alpha\mapsto\tau']ub_{\Xi}(\beta) \tag{40}$$
$$\triangleright\Xi_{\triangleright}\vdash\rho''[\alpha\mapsto\tau]lb_{\Xi}(\beta)\equiv\rho''[\alpha\mapsto\tau']lb_{\Xi}(\beta) \tag{41}$$

Then by Lemma A.7 on S-Refl, (40), and (41), we have:

$$\begin{aligned}\triangleright\Xi_{\triangleright}\vdash\beta\wedge\rho''[\alpha\mapsto\tau]ub_{\Xi}(\beta)\vee\rho''[\alpha\mapsto\tau]lb_{\Xi}(\beta)\\ \equiv\beta\wedge\rho''[\alpha\mapsto\tau']ub_{\Xi}(\beta)\vee\rho''[\alpha\mapsto\tau']lb_{\Xi}(\beta)\end{aligned} \tag{42}$$

Then by IH on (27) and (42), we have:

$$\rho_1'\rho''[\alpha\mapsto\tau']\Sigma\vdash\triangleright\Xi_{\triangleright}\cdot\triangleright\rho''[\alpha\mapsto\tau']\Xi_{\beta}\cdot\rho_1'\rho''[\alpha\mapsto\tau']\Xi_{\not\beta}\,;\,\rho_2''\ \textbf{\textit{cons.}} \tag{43}$$

for some $\rho_2''$, where $dom(\rho_2'') = dom(\rho_2')$.
By Lemma A.31, (42) implies:

$$\triangleright\Xi_{\triangleright}\vdash\rho_1\pi\equiv\rho_1'\pi\quad \textit{for all } \pi \tag{44}$$

By S-Trans on Lemma A.18 and (44), we have:

$$\triangleright\Xi_{\triangleright}\cdot\rho_1'\rho''[\alpha\mapsto\tau']\Xi_{\not\beta}\cdot\rho_1'\rho''[\alpha\mapsto\tau']\Sigma\vDash\rho_1\rho''[\alpha\mapsto\tau']\Xi_{\not\beta}\cdot\rho_1\rho''[\alpha\mapsto\tau']\Sigma \tag{45}$$
$$\triangleright\Xi_{\triangleright}\cdot\rho_1\rho''[\alpha\mapsto\tau']\Xi_{\beta}\vDash\rho_1'\rho''[\alpha\mapsto\tau']\Xi_{\beta} \tag{46}$$

Then by Lemma A.23 with (45), (35) implies:

$$\triangleright\Xi_{\triangleright}\cdot\triangleright\rho''[\alpha\mapsto\tau']\Xi_{\beta}\cdot\rho_1'\rho''[\alpha\mapsto\tau']\Xi_{\not\beta}\cdot\rho_1'\rho''[\alpha\mapsto\tau']\Sigma\vDash\rho_1\rho''[\alpha\mapsto\tau']\Xi_{\beta}\quad(47)$$

Then by Lemma A.19 on (47) and (46), we have:

$$\triangleright\Xi_{\triangleright}\cdot\triangleright\rho''[\alpha\mapsto\tau']\Xi_{\beta}\cdot\rho_1'\rho''[\alpha\mapsto\tau']\Xi_{\not\beta}\cdot\rho_1'\rho''[\alpha\mapsto\tau']\Sigma\vDash\rho_1'\rho''[\alpha\mapsto\tau']\Xi_{\beta}\quad(48)$$

Since $\tau'$ is not a type variable and $\overline{\gamma=\gamma'}^{(\gamma\mapsto\gamma')\in\rho''}$, we have $split_{\beta}(\rho''[\alpha\mapsto\tau']\Xi,\ dom(\rho_2''))=(\rho''[\alpha\mapsto\tau']\Xi_{\beta},\ \rho''[\alpha\mapsto\tau']\Xi_{\not\beta})$. Then by the inductive case of the definition of consistency, (43) and (48) imply:

$$\rho''[\alpha\mapsto\tau']\Sigma\vdash\triangleright\Xi_{\triangleright}\cdot\rho''[\alpha\mapsto\tau']\Xi\ ;\ \rho_2''\circ\rho_1'\ \textbf{\textit{cons.}}\quad(49)$$

■

**Lemma A.37** (Inversion of consistency)**.** *If* $\Sigma\vdash\triangleright\Xi_{\triangleright}\cdot\Xi\ ;\ \rho$ ***cons.****, then for all* $\alpha$*, we have* $\triangleright\Xi_{\triangleright}\cdot\triangleright\Xi_{\alpha}\cdot\rho_{\alpha}\Xi_{\not\alpha}\cdot\rho_{\alpha}\Sigma\vDash\rho_{\alpha}\Xi_{\alpha}$ *and* $\rho_{\alpha}\Sigma\vdash\triangleright\Xi_{\triangleright}\cdot\triangleright\Xi_{\alpha}\cdot\rho_{\alpha}\Xi_{\not\alpha}\ ;\ \rho'$ ***cons.*** *for some* $\rho'$*, where* $split_{\alpha}(\Xi,\ dom(\rho'))=(\Xi_{\alpha},\ \Xi_{\not\alpha})$*,* $\rho_{\alpha}=[\alpha\mapsto\alpha\wedge ub_{\Xi}(\alpha)\vee lb_{\Xi}(\alpha)]$*, and* $dom(\rho')=dom(\rho)\backslash\{\alpha\}$*.*

**Proof** By induction on consistency derivations. If $\Xi$ is not guarded, we can replace it with $cleanup(\Xi)$ before applying the lemma, and restore it back to $\Xi$ in the conclusion. Therefore we can assume $\Xi$ ***guard.***.

**Base case.** For the base case, we have $\Xi=\epsilon$. Then we have $\Xi_{\alpha}=\epsilon$, $\Xi_{\not\alpha}=\epsilon$, and $\rho_{\alpha}=id$. By S-Empty, we have:

$$\triangleright\Xi_{\triangleright}\cdot\Sigma\vDash\epsilon$$

$$\text{i.e.,}\quad\triangleright\Xi_{\triangleright}\cdot\triangleright\Xi_{\alpha}\cdot\rho_{\alpha}\Xi_{\not\alpha}\cdot\rho_{\alpha}\Sigma\vDash\rho_{\alpha}\Xi_{\alpha}\quad(1)$$

By the base case of the definition of consistency, we have:

$$\Sigma\vdash\triangleright\Xi_{\triangleright}\ ;\ id\ \textbf{\textit{cons.}}$$

$$\text{i.e.,}\quad\rho_{\alpha}\Sigma\vdash\triangleright\Xi_{\triangleright}\cdot\triangleright\Xi_{\alpha}\cdot\rho_{\alpha}\Xi_{\not\alpha}\ ;\ id\ \textbf{\textit{cons.}}\quad(2)$$

**Inductive case on** $\alpha$**.** For the inductive case on $\alpha$, i.e., where $\rho=\rho_2\circ\rho_1$ for some $\rho_1$ and $\rho_2$, where $dom(\rho_1)=\{\alpha\}$, we have the result immediately from the premises.

**Inductive case not on** $\alpha$**.** For the inductive case not on $\alpha$, i.e., where $\rho=\rho_2\circ\rho_1$ for some $\rho_1$ and $\rho_2$, where $dom(\rho_1)=\{\beta\}$ for some $\beta\neq\alpha$, the premises of the rule are:

$$\triangleright\Xi_{\triangleright}\cdot\triangleright\Xi_{\beta}\cdot\rho_1\Xi_{\not\beta}\cdot\rho_1\Sigma\vDash\rho_1\Xi_{\beta}\quad(3)$$

$$\rho_1\Sigma\vdash\triangleright\Xi_{\triangleright}\cdot\triangleright\Xi_{\beta}\cdot\rho_1\Xi_{\not\beta}\ ;\ \rho_2\ \textbf{\textit{cons.}}\quad(4)$$

where $split_{\beta}(\Xi,\ dom(\rho_2))=(\Xi_{\beta},\ \Xi_{\not\beta})$ and $\rho_1=[\beta\mapsto\beta\wedge ub_{\Xi}(\beta)\vee lb_{\Xi}(\beta)]$. By IH on (4), we have:

$$\triangleright\Xi_{\triangleright}\cdot\triangleright\Xi_{\beta}\cdot\triangleright\Xi'_{\alpha}\cdot\rho'_{\alpha}\Xi'_{\not\alpha}\cdot\rho'_{\alpha}\rho_1\Sigma\vDash\rho'_{\alpha}\Xi'_{\alpha}\quad(5)$$

$$\rho'_{\alpha}\rho_1\Sigma\vdash\triangleright\Xi_{\triangleright}\cdot\triangleright\Xi_{\beta}\cdot\triangleright\Xi'_{\alpha}\cdot\rho'_{\alpha}\Xi'_{\not\alpha}\ ;\ \rho_3\ \textbf{\textit{cons.}}\quad(6)$$

for some $\rho_3$, where $split_\alpha(\rho_1\Xi_{\not\beta}, dom(\rho_3)) = (\Xi'_\alpha, \Xi'_{\not\alpha})$ and $\rho'_\alpha = [\alpha \mapsto \alpha \wedge ub_{\rho_1\Xi_{\not\beta}}(\alpha) \vee lb_{\rho_1\Xi_{\not\beta}}(\alpha)]$ and $dom(\rho_3) = dom(\rho_2)\backslash\{\alpha\}$. It is easy to see that $\Xi'_\alpha = \rho_1\Xi_\alpha$ and $\Xi'_{\not\alpha} = \rho_1\Xi_{\not\beta\not\alpha}$, where $split_\alpha(\Xi_{\not\beta}, dom(\rho_3)) = (\Xi_\alpha, \Xi_{\not\beta\not\alpha})$. Then (5) and (6) imply:

$$\triangleright\Xi_\triangleright \cdot \triangleright\Xi_\beta \cdot \triangleright\rho_1\Xi_\alpha \cdot \rho'_\alpha\rho_1\Xi_{\not\beta\not\alpha} \cdot \rho'_\alpha\rho_1\Sigma \vDash \rho'_\alpha\rho_1\Xi_\alpha \tag{7}$$

$$\rho'_\alpha\rho_1\Sigma \vdash \triangleright\Xi_\triangleright \cdot \triangleright\Xi_\beta \cdot \triangleright\rho_1\Xi_\alpha \cdot \rho'_\alpha\rho_1\Xi_{\not\beta\not\alpha}\ ;\ \rho_3\ \textbf{\textit{cons.}} \tag{8}$$

Since $(\alpha \leqslant^\diamond \rho_1\pi) \in \rho_1\Xi_{\not\beta}$ only if $(\alpha \leqslant^\diamond \pi) \in \Xi$, we have $ub_{\rho_1\Xi_{\not\beta}}(\beta) = \rho_1 ub_\Xi(\beta)$ and $lb_{\rho_1\Xi_{\not\beta}}(\beta) = \rho_1 lb_\Xi(\beta)$. Then we have:

$$\rho'_\alpha = [\alpha \mapsto \alpha \wedge \rho_1 ub_\Xi(\alpha) \vee \rho_1 lb_\Xi(\alpha)] \tag{9}$$

Expanding the composition, we have:

$$\rho'_\alpha \circ \rho_1 = [\alpha \mapsto \alpha \wedge \rho_1 ub_\Xi(\alpha) \vee \rho_1 lb_\Xi(\alpha),\ \beta \mapsto \beta \wedge \rho'_\alpha ub_\Xi(\beta) \vee \rho'_\alpha lb_\Xi(\alpha)] \tag{10}$$

By Corollary A.33, we have:

$$\begin{gathered} \Xi_\beta \vdash \beta \equiv [\beta \mapsto ub_{\Xi_\beta}(\beta) \vee lb_{\Xi_\beta}(\beta)]\beta \\ \text{i.e.,} \quad \Xi_\beta \vdash \beta \equiv [\beta \mapsto ub_\Xi(\beta) \vee lb_\Xi(\beta)]\beta \\ \text{i.e.,} \quad \Xi_\beta \vdash \beta \equiv \rho_1\beta \end{gathered} \tag{11}$$

Then by Lemma A.31, (11) implies:

$$\begin{gathered} \Xi_\beta \vdash [\beta \mapsto \beta](\alpha \wedge ub_\Xi(\alpha) \vee lb_\Xi(\alpha)) \equiv [\beta \mapsto \rho_1\beta](\alpha \wedge ub_\Xi(\alpha) \vee lb_\Xi(\alpha)) \\ \text{i.e.,} \quad \Xi_\beta \vdash \alpha \wedge ub_\Xi(\alpha) \vee lb_\Xi(\alpha) \equiv \alpha \wedge \rho_1 ub_\Xi(\alpha) \vee \rho_1 lb_\Xi(\alpha) \\ \text{i.e.,} \quad \Xi_\beta \vdash \alpha \wedge ub_\Xi(\alpha) \vee lb_\Xi(\alpha) \equiv \alpha \wedge ub_{\rho_1\Xi_{\not\beta}}(\alpha) \vee lb_{\rho_1\Xi_{\not\beta}}(\alpha) \end{gathered} \tag{12}$$

Then by Lemma A.32, (12) implies:

$$\triangleright\Xi_\beta \vdash \rho_\alpha ub_\Xi(\beta) \equiv \rho'_\alpha ub_\Xi(\beta) \tag{13}$$

$$\triangleright\Xi_\beta \vdash \rho_\alpha lb_\Xi(\beta) \equiv \rho'_\alpha lb_\Xi(\beta) \tag{14}$$

By Lemma A.7 on S-Refl, (13) and (14), we have:

$$\triangleright\Xi_\beta \vdash \beta \wedge \rho_\alpha ub_\Xi(\beta) \vee \rho_\alpha lb_\Xi(\beta) \equiv \beta \wedge \rho'_\alpha ub_\Xi(\beta) \vee \rho'_\alpha lb_\Xi(\beta) \tag{15}$$

Let $\rho'_1 = [\beta \mapsto \beta \wedge ub_{\rho_\alpha\Sigma_{\not\alpha}}(\beta) \vee lb_{\rho_\alpha\Sigma_{\not\alpha}}(\beta)]$. By the same reasoning, we have:

$$\rho'_1 \circ \rho_\alpha = [\alpha \mapsto \alpha \wedge \rho'_1 ub_\Xi(\alpha) \vee \rho'_1 lb_\Xi(\alpha),\ \beta \mapsto \beta \wedge \rho_\alpha ub_\Xi(\beta) \vee \rho_\alpha lb_\Xi(\beta)] \tag{16}$$

$$\triangleright\Xi_\alpha \vdash \alpha \wedge \rho_1 ub_\Xi(\alpha) \vee \rho_1 lb_\Xi(\alpha) \equiv \alpha \wedge \rho'_1 ub_\Xi(\alpha) \vee \rho'_1 lb_\Xi(\alpha) \tag{17}$$

Then by Lemma A.31 on (15) and (17), we have:

$$\triangleright\Xi_\alpha \cdot \triangleright\Xi_\beta \vdash \rho'_\alpha\rho_1\pi \equiv \rho'_1\rho_\alpha\pi \quad \textit{for all } \pi \tag{18}$$

By S-Trans on Lemma A.18 and (18), we have:

$$\triangleright\Xi_\alpha \cdot \triangleright\Xi_\beta \cdot \rho'_1\rho_\alpha\Delta \vDash \rho'_\alpha\rho_1\Delta \quad \textit{for all } \Delta \tag{19}$$

By Corollary A.33, we have

$$\Xi_\beta \vdash \pi \equiv [\beta \mapsto \beta \wedge ub_{\Xi_\beta}(\beta) \vee lb_{\Xi_\beta}(\beta)]\pi \quad \textit{for all } \pi$$
$$\text{i.e.,} \quad \Xi_\beta \vdash \pi \equiv [\beta \mapsto \beta \wedge ub_{\Xi}(\beta) \vee lb_{\Xi}(\beta)]\pi \quad \textit{for all } \pi$$
$$\text{i.e.,} \quad \Xi_\beta \vdash \pi \equiv \rho_1\pi \quad \textit{for all } \pi \tag{20}$$

By S-Trans on Lemma A.18 and (20), we have:

$$\Xi_\alpha \cdot \Xi_\beta \vDash \rho_1 \Xi_\alpha \tag{21}$$

By Lemma A.21, (21) implies:

$$\triangleright\Xi_\alpha \cdot \triangleright\Xi_\beta \vDash \triangleright\rho_1\Xi_\alpha \tag{22}$$

By the same reasoning, we have:

$$\triangleright\rho_1\Xi_\alpha \cdot \triangleright\Xi_\triangleright \cdot \triangleright\Xi_\beta \vDash \triangleright\rho'_\alpha\Xi_\triangleright \cdot \triangleright\rho'_\alpha\Xi_\beta \tag{23}$$
$$\triangleright\Xi_\alpha \cdot \triangleright\rho_\alpha\Xi_\beta \vDash \triangleright\Xi_\beta \tag{24}$$

By Lemma A.29, (3) implies:

$$\triangleright\rho'_\alpha\Xi_\triangleright \cdot \triangleright\rho'_\alpha\Xi_\beta \cdot \rho'_\alpha\rho_1\Xi_{\not\beta} \cdot \rho'_\alpha\rho_1\Sigma \vDash \rho'_\alpha\rho_1\Xi_\beta$$
$$\text{i.e.,} \quad \triangleright\rho'_\alpha\Xi_\triangleright \cdot \triangleright\rho'_\alpha\Xi_\beta \cdot \rho'_\alpha\rho_1\Xi_\alpha \cdot \rho'_\alpha\rho_1\Xi_{\not\beta\not\alpha} \cdot \rho'_\alpha\rho_1\Sigma \vDash \rho'_\alpha\rho_1\Xi_\beta \tag{25}$$

By Lemma A.23 with (7), (25) implies:

$$\triangleright\rho'_\alpha\Xi_\triangleright \cdot \triangleright\rho'_\alpha\Xi_\beta \cdot \triangleright\Xi_\triangleright \cdot \triangleright\Xi_\beta \cdot \triangleright\rho_1\Xi_\alpha \cdot \rho'_\alpha\rho_1\Xi_{\not\beta\not\alpha} \cdot \rho'_\alpha\rho_1\Sigma \vDash \rho'_\alpha\rho_1\Xi_\beta \tag{26}$$

Let $split_\beta(\Xi_{\not\alpha},\ dom(\rho_3)) = (\Xi_\beta,\ \Xi_{\not\alpha\not\beta})$. It is easy to see that $\Xi_{\not\alpha\not\beta} = \Xi_{\not\beta\not\alpha}$. Then (26) and (8) imply:

$$\triangleright\rho'_\alpha\Xi_\triangleright \cdot \triangleright\rho'_\alpha\Xi_\beta \cdot \triangleright\Xi_\triangleright \cdot \triangleright\Xi_\beta \cdot \triangleright\rho_1\Xi_\alpha \cdot \rho'_\alpha\rho_1\Xi_{\not\alpha\not\beta} \cdot \rho'_\alpha\rho_1\Sigma \vDash \rho'_\alpha\rho_1\Xi_\beta \tag{27}$$
$$\rho'_\alpha\rho_1\Sigma \vdash \triangleright\Xi_\triangleright \cdot \triangleright\Xi_\beta \cdot \triangleright\rho_1\Xi_\alpha \cdot \rho'_\alpha\rho_1\Xi_{\not\alpha\not\beta}\ ;\ \rho_3\ \textbf{\textit{cons.}} \tag{28}$$

By Lemma A.23 with (23), (27) implies:

$$\triangleright\Xi_\triangleright \cdot \triangleright\Xi_\beta \cdot \triangleright\rho_1\Xi_\alpha \cdot \rho'_\alpha\rho_1\Xi_{\not\alpha\not\beta} \cdot \rho'_\alpha\rho_1\Sigma \vDash \rho'_\alpha\rho_1\Xi_\beta \tag{29}$$

By Lemma A.23 and Lemma A.25 with (22), (29) and (28) imply:

$$\triangleright\Xi_\triangleright \cdot \triangleright\Xi_\alpha \cdot \triangleright\Xi_\beta \cdot \rho'_\alpha\rho_1\Xi_{\not\alpha\not\beta} \cdot \rho'_\alpha\rho_1\Sigma \vDash \rho'_\alpha\rho_1\Xi_\beta \tag{30}$$
$$\rho'_\alpha\rho_1\Sigma \vdash \triangleright\Xi_\triangleright \cdot \triangleright\Xi_\alpha \cdot \triangleright\Xi_\beta \cdot \rho'_\alpha\rho_1\Xi_{\not\alpha\not\beta}\ ;\ \rho_3\ \textbf{\textit{cons.}} \tag{31}$$

By Lemma A.23 and Lemma A.19 with (19), (30) implies:

$$\triangleright\Xi_\triangleright \cdot \triangleright\Xi_\alpha \cdot \triangleright\Xi_\beta \cdot \rho'_1\rho_\alpha\Xi_{\not\alpha\not\beta} \cdot \rho'_1\rho_\alpha\Sigma \vDash \rho'_1\rho_\alpha\Xi_\beta \tag{32}$$

By Lemma A.36 with (15) and (17), (31) implies:

$$\rho'_1\rho_\alpha\Sigma \vdash \triangleright\Xi_\triangleright \cdot \triangleright\Xi_\alpha \cdot \triangleright\Xi_\beta \cdot \rho'_1\rho_\alpha\Xi_{\not\alpha\not\beta}\ ;\ \rho'_3\ \textbf{\textit{cons.}} \tag{33}$$

for some $\rho'_3$, where $dom(\rho'_3) = dom(\rho_3)$. By Lemma A.23 and Lemma A.25 with (24), (32) and (33) imply:

$$\triangleright\Xi_\triangleright \cdot \triangleright\Xi_\alpha \cdot \triangleright\rho_\alpha\Xi_\beta \cdot \rho'_1\rho_\alpha\Xi_{\not\alpha\not\beta} \cdot \rho'_1\rho_\alpha\Sigma \vDash \rho'_1\rho_\alpha\Xi_\beta \tag{34}$$

$$\rho'_1\rho_\alpha\Sigma \vdash \triangleright\Xi_\triangleright \cdot \triangleright\Xi_\alpha \cdot \triangleright\rho_\alpha\Xi_\beta \cdot \rho'_1\rho_\alpha\Xi_{\not\alpha\not\beta} \;;\; \rho'_3 \;\textbf{\textit{cons.}} \tag{35}$$

It is easy to see that $split_\beta(\rho_\alpha\Xi_{\not\alpha}, dom(\rho'_3)) = (\rho_\alpha\Xi_\beta, \rho_\alpha\Xi_{\not\alpha\not\beta})$. Then by the inductive case of the definition of consistency, (34) and (35) imply:

$$\rho_\alpha\Sigma \vdash \triangleright\Xi_\triangleright \cdot \triangleright\Xi_\alpha \cdot \rho_\alpha\Xi_{\not\alpha} \;;\; \rho'_3 \circ \rho'_1 \;\textbf{\textit{cons.}} \tag{36}$$

By Lemma A.23 with (22), (7) implies:

$$\triangleright\Xi_\triangleright \cdot \triangleright\Xi_\alpha \cdot \triangleright\Xi_\beta \cdot \rho'_\alpha\rho_1\Xi_{\not\beta\not\alpha} \cdot \rho'_\alpha\rho_1\Sigma \vDash \rho'_\alpha\rho_1\Xi_\alpha \tag{37}$$

By Lemma A.23 and Lemma A.19 with (19), (37) implies:

$$\triangleright\Xi_\triangleright \cdot \triangleright\Xi_\alpha \cdot \triangleright\Xi_\beta \cdot \rho'_1\rho_\alpha\Xi_{\not\beta\not\alpha} \cdot \rho'_1\rho_\alpha\Sigma \vDash \rho'_1\rho_\alpha\Xi_\alpha \tag{38}$$

By Lemma A.23 with (24), (38) implies:

$$\triangleright\Xi_\triangleright \cdot \triangleright\Xi_\alpha \cdot \triangleright\rho_\alpha\Xi_\beta \cdot \rho'_1\rho_\alpha\Xi_{\not\beta\not\alpha} \cdot \rho'_1\rho_\alpha\Sigma \vDash \rho'_1\rho_\alpha\Xi_\alpha \tag{39}$$

By Lemma A.23 with Lemma A.18, (39) implies:

$$\triangleright\Xi_\triangleright \cdot \triangleright\Xi_\alpha \cdot \rho_\alpha\Xi_\beta \cdot \rho'_1\rho_\alpha\Xi_{\not\beta\not\alpha} \cdot \rho'_1\rho_\alpha\Sigma \vDash \rho'_1\rho_\alpha\Xi_\alpha \tag{40}$$

By Corollary A.33, we have

$$\begin{gathered}
\rho_\alpha\Xi_\beta \vdash \pi \equiv [\beta \mapsto \beta \wedge ub_{\rho_\alpha\Xi_\beta}(\beta) \vee lb_{\rho_\alpha\Xi_\beta}(\beta)]\pi \quad \textit{for all } \pi \\
\text{i.e.,} \quad \rho_\alpha\Xi_\beta \vdash \pi \equiv [\beta \mapsto \beta \wedge ub_{\rho_\alpha\Xi_{\not\alpha}}(\beta) \vee lb_{\rho_\alpha\Xi_{\not\alpha}}(\beta)]\pi \quad \textit{for all } \pi \\
\text{i.e.,} \quad \rho_\alpha\Xi_\beta \vdash \pi \equiv \rho'_1\pi \quad \textit{for all } \pi
\end{gathered} \tag{41}$$

By S-TRANS on Lemma A.18 and (41), we have:

$$\rho_\alpha\Xi_\beta \cdot \Delta \vDash \rho'_1\Delta \quad \textit{for all } \Delta \tag{42}$$

Then by Lemma A.23 and Lemma A.19 with (42), (40) implies:

$$\begin{gathered}
\triangleright\Xi_\triangleright \cdot \triangleright\Xi_\alpha \cdot \rho_\alpha\Xi_\beta \cdot \rho_\alpha\Xi_{\not\beta\not\alpha} \cdot \rho_\alpha\Sigma \vDash \rho_\alpha\Xi_\alpha \\
\text{i.e.,} \quad \triangleright\Xi_\triangleright \cdot \triangleright\Xi_\alpha \cdot \rho_\alpha\Xi_\beta \cdot \rho_\alpha\Xi_{\not\alpha\not\beta} \cdot \rho_\alpha\Sigma \vDash \rho_\alpha\Xi_\alpha \\
\text{i.e.,} \quad \triangleright\Xi_\triangleright \cdot \triangleright\Xi_\alpha \cdot \rho_\alpha\Xi_{\not\alpha} \cdot \rho_\alpha\Sigma \vDash \rho_\alpha\Xi_\alpha
\end{gathered} \tag{43}$$

■

**Lemma A.38** (Inlining of consistent bounds)**.** *If* $\Sigma \vdash \Xi \;;\; \rho$ ***cons.*** *and* $\Xi\cdot\Sigma \vdash \tau \leqslant \tau'$*, then* $\triangleright\Xi\cdot\rho\Sigma \vdash \rho\tau \leqslant \rho\tau'$.

**Proof** By induction on consistency derivations for the statement: if $\Sigma \vdash \triangleright\Xi_\triangleright\cdot\Xi \;;\; \rho$ ***cons.*** and $\triangleright\Xi_\triangleright\cdot\Xi\cdot\Sigma \vdash \tau \leqslant \tau'$, then $\triangleright\Xi_\triangleright\cdot\triangleright\Xi\cdot\rho\Sigma \vdash \rho\tau \leqslant \rho\tau'$.

**Base case.** The base case is trivial since we have $\Xi = \epsilon$ and $\rho = id$.

**Inductive case.** For the inductive case, we have $\rho = \rho_2 \circ \rho_1$ for some $\rho_1 = [\alpha \mapsto \alpha \wedge ub_\Xi(\alpha) \vee lb_\Xi(\alpha)]$ and $\rho_2$ and $\alpha$. The premises of the rule are:

$$\triangleright\Xi_\triangleright \cdot \triangleright\Xi_\alpha \cdot \rho_1\Xi_{\not\alpha} \cdot \rho_1\Sigma \vDash \rho_1\Xi_\alpha \tag{1}$$

$$\rho_1\Sigma \vdash \triangleright\Xi_\triangleright \cdot \triangleright\Xi_\alpha \cdot \rho_1\Xi_{\not\alpha} \;;\; \rho_2 \;\textbf{\textit{cons.}} \tag{2}$$

where $split_\alpha(\Xi,\ dom(\rho_2)) = (\Xi_\alpha,\ \Xi_{\not\alpha})$. From the assumption, we have:

$$\triangleright\Xi_\triangleright \cdot \Xi \cdot \Sigma \vdash \tau \leqslant \tau' \tag{3}$$

By Lemma A.29, (3) implies:

$$\triangleright\rho_1\Xi_\triangleright \cdot \rho_1\Xi \cdot \rho_1\Sigma \vdash \rho_1\tau \leqslant \rho_1\tau'$$

$$\text{i.e.,}\quad \triangleright\rho_1\Xi_\triangleright \cdot \rho_1\Xi_\alpha \cdot \rho_1\Xi_{\not\alpha} \cdot \rho_1\Sigma \vdash \rho_1\tau \leqslant \rho_1\tau' \tag{4}$$

By Lemma A.23 with (1), (4) implies:

$$\triangleright\rho_1\Xi_\triangleright \cdot \triangleright\Xi_\triangleright \cdot \triangleright\Xi_\alpha \cdot \rho_1\Xi_{\not\alpha} \cdot \rho_1\Sigma \vdash \rho_1\tau \leqslant \rho_1\tau' \tag{5}$$

By Corollary A.33, we have:

$$\Xi_\alpha \vdash \pi \equiv [\alpha \mapsto \alpha \wedge ub_{\Xi_\alpha}(\alpha) \vee lb_{\Xi_\alpha}(\alpha)]\pi \quad \textit{for all } \pi$$

$$\text{i.e.,}\quad \Xi_\alpha \vdash \pi \equiv [\alpha \mapsto \alpha \wedge ub_\Xi(\alpha) \vee lb_\Xi(\alpha)]\pi \quad \textit{for all } \pi$$

$$\text{i.e.,}\quad \Xi_\alpha \vdash \pi \equiv \rho_1\pi \quad \textit{for all } \pi \tag{6}$$

By S-TRANS on Lemma A.18 and (6), we have:

$$\Xi_\triangleright \cdot \Xi_\alpha \vDash \rho_1\Xi_\triangleright \tag{7}$$

$$\Xi_\alpha \cdot \Xi_{\not\alpha} \vDash \rho_1\Xi_{\not\alpha} \tag{8}$$

Then by Lemma A.23 with (7), (5) implies:

$$\triangleright\Xi_\triangleright \cdot \triangleright\Xi_\alpha \cdot \rho_1\Xi_{\not\alpha} \cdot \rho_1\Sigma \vdash \rho_1\tau \leqslant \rho_1\tau' \tag{9}$$

Then by IH on (2) and (9), we have:

$$\triangleright\Xi_\triangleright \cdot \triangleright\Xi_\alpha \cdot \triangleright\rho_1\Xi_{\not\alpha} \cdot \rho_2\rho_1\Sigma \vdash \rho_2\rho_1\tau \leqslant \rho_2\rho_1\tau'$$

$$\text{i.e.,}\quad \triangleright\Xi_\triangleright \cdot \triangleright\Xi_\alpha \cdot \triangleright\rho_1\Xi_{\not\alpha} \cdot \rho\Sigma \vdash \rho\tau \leqslant \rho\tau' \tag{10}$$

Then by Lemma A.23 with (8), (10) implies:

$$\triangleright\Xi_\triangleright \cdot \triangleright\Xi_\alpha \cdot \triangleright\Xi_{\not\alpha} \cdot \rho\Sigma \vdash \rho\tau \leqslant \rho\tau'$$

$$\text{i.e.,}\quad \triangleright\Xi_\triangleright \cdot \triangleright\Xi \cdot \rho\Sigma \vdash \rho\tau \leqslant \rho\tau' \tag{11}$$

■

**Lemma A.39** (Equivalence of inlining of consistent bounds)**.** *If* $\Sigma \vdash \Xi \;;\; \rho$ ***cons.****, then* $\overline{\Xi \cdot \Sigma \vdash \alpha \equiv \tau}^{(\alpha \mapsto \tau) \in \rho}$.

**Proof** By induction on consistency derivations for the statement: if $\Sigma \vdash \triangleright\Xi_\triangleright \cdot \Xi \;;\; \rho$ ***cons.***, then $\overline{\triangleright\Xi_\triangleright \cdot \Xi \cdot \Sigma \vdash \alpha \equiv \tau}^{(\alpha \mapsto \tau) \in \rho}$.

**Base case.** The base case holds vacuously since we have $\rho = id$.

**Inductive case.** For the inductive case, we have $\rho = \rho_2 \circ \rho_1$ for some $\rho_1 = [\alpha \mapsto \alpha \wedge ub_\Xi(\alpha) \vee lb_\Xi(\alpha)]$ and $\rho_2$ and $\alpha$. The premises of the rule are:

$$\triangleright\Xi_\triangleright \cdot \triangleright\Xi_\alpha \cdot \rho_1\Xi_{\not\alpha} \cdot \rho_1\Sigma \vDash \rho_1\Xi_\alpha \tag{1}$$

$$\rho_1\Sigma \vdash \triangleright\Xi_\triangleright \cdot \triangleright\Xi_\alpha \cdot \rho_1\Xi_{\not\alpha} \;;\; \rho_2 \textit{ cons.} \tag{2}$$

where $split_\alpha(\Xi,\, dom(\rho_2)) = (\Xi_\alpha,\, \Xi_{\not\alpha})$. Let $\rho_2 = [\overline{\alpha_i \mapsto \tau_i}^{\,i}]$ for some $\overline{\alpha_i}^{\,i}$ and $\overline{\tau_i}^{\,i}$. Expanding the composition, we have:

$$\rho = [\overline{\alpha_i \mapsto \tau_i}^{\,i},\; \alpha \mapsto \rho_2(\alpha \wedge ub_\Xi(\alpha) \vee lb_\Xi(\alpha))] \tag{3}$$

By IH on (2), we have:

$$\overline{\triangleright\Xi_\triangleright \cdot \triangleright\Xi_\alpha \cdot \rho_1\Xi_{\not\alpha} \cdot \rho_1\Sigma \vdash \alpha_i \equiv \tau_i}^{\,i} \tag{4}$$

By Corollary A.33, we have:

$$\begin{gathered} \Xi_\alpha \vdash \pi \equiv [\alpha \mapsto \alpha \wedge ub_{\Xi_\alpha}(\alpha) \vee lb_{\Xi_\alpha}(\alpha)]\pi \quad \textit{for all } \pi \\ \text{i.e.,} \quad \Xi_\alpha \vdash \pi \equiv [\alpha \mapsto \alpha \wedge ub_\Xi(\alpha) \vee lb_\Xi(\alpha)]\pi \quad \textit{for all } \pi \\ \text{i.e.,} \quad \Xi_\alpha \vdash \pi \equiv \rho_1\pi \quad \textit{for all } \pi \end{gathered} \tag{5}$$

By S-Trans on Lemma A.18 and (5), we have:

$$\Xi_\alpha \cdot \Xi_{\not\alpha} \cdot \Sigma \vDash \rho_1\Xi_{\not\alpha} \cdot \rho_1\Sigma \tag{6}$$

Then by Lemma A.23 with (6), (4) implies:

$$\begin{gathered} \overline{\triangleright\Xi_\triangleright \cdot \Xi_\alpha \cdot \Xi_{\not\alpha} \cdot \Sigma \vdash \alpha_i \equiv \tau_i}^{\,i} \\ \text{i.e.,} \quad \overline{\triangleright\Xi_\triangleright \cdot \Xi \cdot \Sigma \vdash \alpha_i \equiv \tau_i}^{\,i} \end{gathered} \tag{7}$$

By Lemma A.31 on (7), we have:

$$\triangleright\Xi_\triangleright \cdot \Xi \cdot \Sigma \vdash \pi \equiv \rho_2\pi \quad \textit{for all } \pi \tag{8}$$

Then by S-Trans on (5) and (8), we have:

$$\triangleright\Xi_\triangleright \cdot \Xi \cdot \Sigma \vdash \alpha \equiv \rho_2(\alpha \wedge ub_\Xi(\alpha) \vee lb_\Xi(\alpha)) \tag{9}$$

Then (7) and (9) imply:

$$\overline{\triangleright\Xi_\triangleright \cdot \Xi \cdot \Sigma \vdash \alpha \equiv \tau}^{\,(\alpha \mapsto \tau) \in \rho} \tag{10}$$

■

**Lemma A.40** (Congruence of inlining of consistent bounds on types)**.** *If* $\Sigma \vdash \Xi \;;\; \rho$ ***cons.****, then* $\Xi \cdot \Sigma \vdash \tau \equiv \rho\tau$ *for all* $\tau$*.*

**Proof** By induction on the syntax of $\tau$.

**Case** $\tau = \tau_1 \to \tau_2$**.** By IH, we have:

$$\Xi \cdot \Sigma \vdash \tau_1 \equiv \rho\tau_1 \tag{1}$$

$$\Xi \cdot \Sigma \vdash \tau_2 \equiv \rho\tau_2 \tag{2}$$

By Lemma A.23 with Lemma A.18, (1) and (2) imply:

$$\lhd\Xi\cdot\lhd\Sigma \vdash \tau_1 \equiv \rho\tau_1 \tag{3}$$
$$\lhd\Xi\cdot\lhd\Sigma \vdash \tau_2 \equiv \rho\tau_2 \tag{4}$$

Then by S-FunDepth on (3) and (4), we have:

$$\Xi\cdot\Sigma \vdash \tau_1 \to \tau_2 \equiv \rho\tau_1 \to \rho\tau_2$$
$$\text{i.e.,}\quad \Xi\cdot\Sigma \vdash \tau_1 \to \tau_2 \equiv \rho(\tau_1 \to \tau_2) \tag{5}$$

**Case** $\tau = \{\, x : \tau_1 \,\}$**.** By IH, we have:

$$\Xi\cdot\Sigma \vdash \tau_1 \equiv \rho\tau_1 \tag{6}$$

By Lemma A.23 with Lemma A.18, (6) implies:

$$\lhd\Xi\cdot\lhd\Sigma \vdash \tau_1 \equiv \rho\tau_1 \tag{7}$$

Then by S-RcdDepth on (7), we have:

$$\Xi\cdot\Sigma \vdash \{\, x : \tau_1 \,\} \equiv \{\, x : \rho\tau_1 \,\}$$
$$\text{i.e.,}\quad \Xi\cdot\Sigma \vdash \{\, x : \tau_1 \,\} \equiv \rho\{\, x : \tau_1 \,\} \tag{8}$$

**Cases** $\tau = \#C$**,** $\tau = \top^\diamond$**,** $\tau = \alpha \notin dom(\rho)$**.** Then $\tau = \rho\tau$. By S-Refl, we have:

$$\tau \equiv \rho\tau \tag{9}$$

**Case** $\tau = \alpha \in dom(\rho)$**.** From the assumption, we have:

$$\Sigma \vdash \Xi\,;\ \rho\ \textbf{\textit{cons.}} \tag{10}$$

By Lemma A.39 on (10), we have:

$$\Xi\cdot\Sigma \vdash \alpha \equiv \rho\alpha \tag{11}$$

**Case** $\tau = \tau_1 \vee^\diamond \tau_2$**.** By IH, we have:

$$\Xi\cdot\Sigma \vdash \tau_1 \equiv \rho\tau_1 \tag{12}$$
$$\Xi\cdot\Sigma \vdash \tau_2 \equiv \rho\tau_2 \tag{13}$$

Then by Lemma A.7$\diamond$ on (12) and (13), we have:

$$\Xi\cdot\Sigma \vdash \tau_1 \vee^\diamond \tau_2 \equiv \rho\tau_1 \vee^\diamond \rho\tau_2$$
$$\text{i.e.,}\quad \Xi\cdot\Sigma \vdash \tau_1 \vee^\diamond \tau_2 \equiv \rho(\tau_1 \vee^\diamond \tau_2) \tag{14}$$

**Case** $\tau = \neg\tau_1$**.** By IH, we have:

$$\Xi\cdot\Sigma \vdash \tau_1 \equiv \rho\tau_1 \tag{15}$$

Then by S-NegInv on (15), we have:

$$\Xi\cdot\Sigma \vdash \neg\tau_1 \equiv \neg\rho\tau_1$$
$$\text{i.e.,}\quad \Xi\cdot\Sigma \vdash \neg\tau_1 \equiv \rho\neg\tau_1 \tag{16}$$

■

**Lemma A.41** (Congruence of inlining of consistent bounds on guarded types)**.** *If* $\Sigma \vdash \Xi$ ; $\rho$ ***cons.*** *and* $TTV(\tau) = \varnothing$*, then* $\triangleright\Xi{\cdot}\triangleright\Sigma \vdash \tau \equiv \rho\tau$.

**Proof** By induction on the syntax of $\tau$.

**Case** $\tau = \tau_1 \to \tau_2$**.** By Lemma A.40, we have:

$$\Xi{\cdot}\Sigma \vdash \tau_1 \equiv \rho\tau_1 \tag{1}$$
$$\Xi{\cdot}\Sigma \vdash \tau_2 \equiv \rho\tau_2 \tag{2}$$

By Lemma A.23 with Lemma A.18, (1) and (2) imply:

$$\triangleleft\Xi{\cdot}\triangleleft\Sigma \vdash \tau_1 \equiv \rho\tau_1 \tag{3}$$
$$\triangleleft\Xi{\cdot}\triangleleft\Sigma \vdash \tau_2 \equiv \rho\tau_2 \tag{4}$$

Then by S-FunDepth on (3) and (4), we have:

$$\triangleright\Xi{\cdot}\triangleright\Sigma \vdash \tau_1 \to \tau_2 \equiv \rho\tau_1 \to \rho\tau_2$$
$$\text{i.e.,} \quad \triangleright\Xi{\cdot}\triangleright\Sigma \vdash \tau_1 \to \tau_2 \equiv \rho(\tau_1 \to \tau_2) \tag{5}$$

**Case** $\tau = \{\, x : \tau_1 \,\}$**.** By Lemma A.40, we have:

$$\Xi{\cdot}\Sigma \vdash \tau_1 \equiv \rho\tau_1 \tag{6}$$

By Lemma A.23 with Lemma A.18, (6) implies:

$$\triangleleft\Xi{\cdot}\triangleleft\Sigma \vdash \tau_1 \equiv \rho\tau_1 \tag{7}$$

Then by S-RcdDepth on (7), we have:

$$\triangleright\Xi{\cdot}\triangleright\Sigma \vdash \{\, x : \tau_1 \,\} \equiv \{\, x : \rho\tau_1 \,\}$$
$$\text{i.e.,} \quad \triangleright\Xi{\cdot}\triangleright\Sigma \vdash \{\, x : \tau_1 \,\} \equiv \rho\{\, x : \tau_1 \,\} \tag{8}$$

**Cases** $\tau = \#C$**,** $\tau = \top^\diamond$**.** Then $\tau = \rho\tau$. By S-Refl, we have:

$$\tau \equiv \rho\tau \tag{9}$$

**Case** $\tau = \alpha$**.** Impossible since $TTV(\tau) = \varnothing$.
**Case** $\tau = \tau_1 \vee^\diamond \tau_2$**.** By IH, we have:

$$\triangleright\Xi{\cdot}\triangleright\Sigma \vdash \tau_1 \equiv \rho\tau_1 \tag{10}$$
$$\triangleright\Xi{\cdot}\triangleright\Sigma \vdash \tau_2 \equiv \rho\tau_2 \tag{11}$$

Then by Lemma A.7$\diamond$ on (10) and (11), we have:

$$\triangleright\Xi{\cdot}\triangleright\Sigma \vdash \tau_1 \vee^\diamond \tau_2 \equiv \rho\tau_1 \vee^\diamond \rho\tau_2$$
$$\text{i.e.,} \quad \triangleright\Xi{\cdot}\triangleright\Sigma \vdash \tau_1 \vee^\diamond \tau_2 \equiv \rho(\tau_1 \vee^\diamond \tau_2) \tag{12}$$

**Case** $\tau = \neg\tau_1$**.** By IH, we have:

$$\triangleright\Xi{\cdot}\triangleright\Sigma \vdash \tau_1 \equiv \rho\tau_1 \tag{13}$$

Then by S-NegInv on (13), we have:

$$\triangleright\Xi\cdot\triangleright\Sigma \vdash \neg\tau_1 \equiv \neg\rho\tau_1$$
$$\text{i.e.,}\quad \triangleright\Xi\cdot\triangleright\Sigma \vdash \neg\tau_1 \equiv \rho\neg\tau_1 \tag{14}$$

■

**Lemma A.42** (Inlining of consistent bounds on guarded derivations)**.** *If* $\Sigma \vdash \Xi\,;\ \rho$ ***cons.*** *and* $\Xi\cdot\Sigma \vdash \tau \leqslant \tau'$ *and* $TTV(\tau) \cup TTV(\tau') = \varnothing$*, then* $\triangleright\Xi\cdot\triangleright\Sigma\cdot\rho\Sigma \vdash \tau \leqslant \tau'$*.*

**Proof** From the assumptions, we have:

$$\Sigma \vdash \Xi\,;\ \rho\ \textbf{\textit{cons.}} \tag{1}$$
$$\Xi\cdot\Sigma \vdash \tau \leqslant \tau' \tag{2}$$

By Lemma A.38 on (1) and (2), we have:

$$\triangleright\Xi\cdot\rho\Sigma \vdash \rho\tau \leqslant \rho\tau' \tag{3}$$

By Lemma A.41 on (1), we have:

$$\triangleright\Xi\cdot\triangleright\Sigma \vdash \tau \equiv \rho\tau \tag{4}$$
$$\triangleright\Xi\cdot\triangleright\Sigma \vdash \tau' \equiv \rho\tau' \tag{5}$$

Then by S-Trans on (4), (3), and (5), we have:

$$\triangleright\Xi\cdot\triangleright\Sigma\cdot\rho\Sigma \vdash \tau \leqslant \tau' \tag{6}$$

■

**Lemma A.43** (Inlining of bound in consistency)**.** *If* $\Sigma\cdot(\alpha \leqslant^{\diamond} \tau) \vdash \triangleright\Xi_{\triangleright}\cdot\Xi\,;\ \rho$ ***cons.****, where* $\alpha \notin dom(\rho)$*, then* $\rho_\alpha\Sigma \vdash \triangleright\Xi_{\triangleright}\cdot\triangleright(\alpha \leqslant^{\diamond} \tau)\cdot\rho_\alpha\Xi\,;\ \rho'$ ***cons.*** *for some* $\rho'$*, where* $\rho_\alpha = [\alpha \mapsto \alpha \wedge^{\diamond} \tau]$ *and* $dom(\rho') = dom(\rho)$*.*

**Proof** By induction on consistency derivations. If $\Xi$ is not guarded, we can replace it with $cleanup(\Xi)$ before applying the lemma, and restore it back to $\Xi$ in the conclusion. Therefore we can assume $\Xi$ ***guard.***.

**Base case.** For the base case, we have $\Xi = \epsilon$. Then by the base case of the definition of consistency, we have:

$$\rho_\alpha\Sigma \vdash \triangleright\Xi_{\triangleright}\cdot\triangleright(\alpha \leqslant^{\diamond} \tau)\,;\ id\ \textbf{\textit{cons.}} \tag{1}$$

**Inductive case.** For the inductive case, we have $\rho = \rho_2 \circ \rho_1$ for some $\rho_1 = [\beta \mapsto \beta \wedge ub_\Xi(\beta) \vee lb_\Xi(\beta)]$ and $\rho_2$ and $\beta \neq \alpha$. The premises of the rule are:

$$\triangleright\Xi_{\triangleright}\cdot\triangleright\Xi_\beta\cdot\rho_1\Xi_{\not\beta}\cdot\rho_1\Sigma\cdot\rho_1(\alpha \leqslant^{\diamond} \tau) \vDash \rho_1\Xi_\beta \tag{2}$$
$$\rho_1\Sigma\cdot\rho_1(\alpha \leqslant^{\diamond} \tau) \vdash \triangleright\Xi_{\triangleright}\cdot\triangleright\Xi_\beta\cdot\rho_1\Xi_{\not\beta}\,;\ \rho_2\ \textbf{\textit{cons.}} \tag{3}$$

where $split_\beta(\Xi,\ dom(\rho_2)) = (\Xi_\beta,\ \Xi_{\not\beta})$. By IH on (3), we have:

$$\rho'_\alpha \rho_1 \Sigma \vdash \triangleright\Xi_\triangleright \cdot \triangleright\Xi_\beta \cdot \triangleright(\alpha \leqslant^\diamond \rho_1\tau) \cdot \rho'_\alpha \rho_1 \Xi_{\not\beta}\ ;\ \rho'_2\ \textbf{\textit{cons.}} \tag{4}$$

for some $\rho'_2$, where $\rho'_\alpha = [\alpha \mapsto \alpha \wedge^\diamond \rho_1\tau]$ and $dom(\rho'_2) = dom(\rho_2)$. Expanding the composition, we have:

$$\rho'_\alpha \circ \rho_1 = [\alpha \mapsto \alpha \wedge^\diamond \rho_1\tau,\ \beta \mapsto \beta \wedge \rho'_\alpha ub_\Xi(\beta) \vee \rho'_\alpha lb_\Xi(\beta)] \tag{5}$$

By Corollary A.33, we have:

$$\begin{gathered}(\alpha \leqslant^\diamond \tau) \vDash \beta \wedge ub_\Xi(\beta) \vee lb_\Xi(\beta) \equiv [\alpha \mapsto \alpha \wedge^\diamond \tau](\beta \wedge ub_\Xi(\beta) \vee lb_\Xi(\beta)) \\ \text{i.e.,}\quad (\alpha \leqslant^\diamond \tau) \vDash \beta \wedge ub_\Xi(\beta) \vee lb_\Xi(\beta) \equiv \beta \wedge \rho_\alpha ub_\Xi(\beta) \vee \rho_\alpha lb_\Xi(\beta) \\ \text{i.e.,}\quad (\alpha \leqslant^\diamond \tau) \vDash \beta \wedge ub_\Xi(\beta) \vee lb_\Xi(\beta) \equiv \beta \wedge ub_{\rho_\alpha\Xi}(\beta) \vee lb_{\rho_\alpha\Xi}(\beta)\end{gathered} \tag{6}$$

Then by Lemma A.32, (6) implies:

$$\triangleright(\alpha \leqslant^\diamond \tau) \vDash \rho_1\tau \equiv \rho'_1\tau \tag{7}$$

Then by Lemma A.7 on S-Refl and (7), we have:

$$\triangleright(\alpha \leqslant^\diamond \tau) \vDash \alpha \wedge^\diamond \rho_1\tau \equiv \alpha \wedge^\diamond \rho'_1\tau \tag{8}$$

Let $\rho'_1 = [\beta \mapsto \beta \wedge ub_{\rho_\alpha\Xi}(\beta) \vee lb_{\rho_\alpha\Xi}(\beta)]$. By the same reasoning, we have:

$$\begin{aligned}\rho'_1 \circ \rho_\alpha &= [\alpha \mapsto \alpha \wedge^\diamond \rho'_1\tau,\ \beta \mapsto \beta \wedge ub_{\rho_\alpha\Xi}(\beta) \vee lb_{\rho_\alpha\Xi}(\beta)] \\ &= [\alpha \mapsto \alpha \wedge^\diamond \rho'_1\tau,\ \beta \mapsto \beta \wedge \rho_\alpha ub_\Xi(\beta) \vee \rho_\alpha lb_\Xi(\beta)]\end{aligned} \tag{9}$$

$$\triangleright\Xi_\beta \vDash \beta \wedge \rho_\alpha ub_\Xi(\beta) \vee \rho_\alpha lb_\Xi(\beta) \equiv \beta \wedge \rho'_\alpha ub_\Xi(\beta) \vee \rho'_\alpha lb_\Xi(\beta) \tag{10}$$

Then by Lemma A.31 on (8) and (10), we have:

$$\triangleright(\alpha \wedge^\diamond \tau) \cdot \triangleright\Xi_\beta \vDash \rho'_\alpha \rho_1 \pi \equiv \rho'_1 \rho_\alpha \pi \quad \textit{for all } \pi \tag{11}$$

By S-Trans on Lemma A.18 and (11), we have:

$$\triangleright(\alpha \wedge^\diamond \tau) \cdot \triangleright\Xi_\beta \cdot \rho'_1 \rho_\alpha \Delta \vDash \rho'_\alpha \rho_1 \Delta \quad \textit{for all } \Delta \tag{12}$$

By Corollary A.33, we have

$$\begin{gathered}\Xi_\beta \vdash \pi \equiv [\beta \mapsto \beta \wedge ub_{\Xi_\beta}(\beta) \vee lb_{\Xi_\beta}(\beta)]\pi \quad \textit{for all } \pi \\ \text{i.e.,}\quad \Xi_\beta \vdash \pi \equiv [\beta \mapsto \beta \wedge ub_\Xi(\beta) \vee lb_\Xi(\beta)]\pi \quad \textit{for all } \pi \\ \text{i.e.,}\quad \Xi_\beta \vdash \pi \equiv \rho_1\pi \quad \textit{for all } \pi\end{gathered} \tag{13}$$

By S-Trans on Lemma A.18 and (13), we have:

$$(\alpha \leqslant^\diamond \tau) \cdot \Xi_\beta \vDash (\alpha \leqslant^\diamond \rho_1\tau) \tag{14}$$

By Lemma A.21, (14) implies:

$$\triangleright(\alpha \leqslant^\diamond \tau) \cdot \triangleright\Xi_\beta \vDash \triangleright(\alpha \leqslant^\diamond \rho_1\tau) \tag{15}$$

By the same reasoning, we have:

$$\triangleright(\alpha \leqslant^{\diamond} \rho_1\tau)\cdot\triangleright\Xi_{\triangleright}\cdot\triangleright\Xi_{\beta} \vDash \triangleright\rho'_{\alpha}\Xi_{\triangleright}\cdot\triangleright\rho'_{\alpha}\Xi_{\beta} \tag{16}$$

$$\triangleright\Xi_{\beta}\cdot\triangleright(\alpha \leqslant^{\diamond} \tau) \vDash \triangleright(\alpha \leqslant^{\diamond} \rho_1\tau) \tag{17}$$

$$\triangleright(\alpha \leqslant^{\diamond} \tau)\cdot\triangleright\rho_{\alpha}\Xi_{\beta} \vDash \triangleright\Xi_{\beta} \tag{18}$$

By Lemma A.35, (2) implies:

$$\triangleright\rho'_{\alpha}\Xi_{\triangleright}\cdot\triangleright\rho'_{\alpha}\Xi_{\beta}\cdot\rho'_{\alpha}\rho_1\Xi_{\not\beta}\cdot\rho'_{\alpha}\rho_1\Sigma\cdot\triangleright(\alpha \leqslant^{\diamond} \rho_1\tau) \vDash \rho'_{\alpha}\rho_1\Xi_{\beta} \tag{19}$$

By Lemma A.23 with (16), (19) implies:

$$\triangleright\Xi_{\triangleright}\cdot\triangleright\Xi_{\beta}\cdot\rho'_{\alpha}\rho_1\Xi_{\not\beta}\cdot\rho'_{\alpha}\rho_1\Sigma\cdot\triangleright(\alpha \leqslant^{\diamond} \rho_1\tau) \vDash \rho'_{\alpha}\rho_1\Xi_{\beta} \tag{20}$$

By Lemma A.23 with (17), (20) and (4) implies:

$$\triangleright\Xi_{\triangleright}\cdot\triangleright\Xi_{\beta}\cdot\triangleright(\alpha \leqslant^{\diamond} \tau)\cdot\rho'_{\alpha}\rho_1\Xi_{\not\beta}\cdot\rho'_{\alpha}\rho_1\Sigma \vDash \rho'_{\alpha}\rho_1\Xi_{\beta} \tag{21}$$

$$\rho'_{\alpha}\rho_1\Sigma \vdash \triangleright\Xi_{\triangleright}\cdot\triangleright\Xi_{\beta}\cdot\triangleright(\alpha \leqslant^{\diamond} \rho_1\tau)\cdot\rho'_{\alpha}\rho_1\Xi_{\not\beta}\ ;\ \rho'_2\ \textbf{\textit{cons.}} \tag{22}$$

By Lemma A.23 and Lemma A.19 with (12), (21) implies:

$$\triangleright\Xi_{\triangleright}\cdot\triangleright\Xi_{\beta}\cdot\triangleright(\alpha \leqslant^{\diamond} \tau)\cdot\rho'_1\rho_{\alpha}\Xi_{\not\beta}\cdot\rho'_1\rho_{\alpha}\Sigma \vDash \rho'_1\rho_{\alpha}\Xi_{\beta} \tag{23}$$

By Lemma A.36 with (8) and (10), (22) implies:

$$\rho'_1\rho_{\alpha}\Sigma \vdash \triangleright\Xi_{\triangleright}\cdot\triangleright\Xi_{\beta}\cdot\triangleright(\alpha \leqslant^{\diamond} \rho_1\tau)\cdot\rho'_1\rho_{\alpha}\Xi_{\not\beta}\ ;\ \rho''_2\ \textbf{\textit{cons.}} \tag{24}$$

for some $\rho''_2$, where $dom(\rho''_2) = dom(\rho'_2)$. By Lemma A.23 with (18), (23) and (24) implies:

$$\triangleright\Xi_{\triangleright}\cdot\triangleright(\alpha \leqslant^{\diamond} \tau)\cdot\triangleright\rho_{\alpha}\Xi_{\beta}\cdot\rho'_1\rho_{\alpha}\Xi_{\not\beta}\cdot\rho'_1\rho_{\alpha}\Sigma \vDash \rho'_1\rho_{\alpha}\Xi_{\beta} \tag{25}$$

$$\rho'_1\rho_{\alpha}\Sigma \vdash \triangleright\Xi_{\triangleright}\cdot\triangleright(\alpha \leqslant^{\diamond} \rho_1\tau)\cdot\triangleright\rho_{\alpha}\Xi_{\beta}\cdot\rho'_1\rho_{\alpha}\Xi_{\not\beta}\ ;\ \rho''_2\ \textbf{\textit{cons.}} \tag{26}$$

It is easy to see that $split_{\beta}(\rho_{\alpha}\Xi,\ dom(\rho''_2)) = (\rho_{\alpha}\Xi_{\beta},\ \rho_{\alpha}\Xi_{\not\beta})$. Then by the inductive case of the definition of consistency, (25) and (26) imply:

$$\rho_{\alpha}\Sigma \vdash \triangleright\Xi_{\triangleright}\cdot\triangleright(\alpha \leqslant^{\diamond} \rho_1\tau)\cdot\rho_{\alpha}\Xi\ ;\ \rho''_2 \circ \rho'_1\ \textbf{\textit{cons.}} \tag{27}$$

■

### *A.7 Pure Boolean-Algebraic Subtyping*

**Lemma A.44.** *If $\bigvee_i \tau_i \subseteq \bigwedge_j \pi_j$, then $\overline{\tau_i \subseteq \pi_j}^{i,j}$. Additionally, if $\bigvee_i \tau_i = \tau_1$ where $\tau_1$ is not an intersection; or if $\bigwedge_j \pi_j = \pi_1$ where $\pi_1$ is not a union, then the derivation for $\overline{\tau_i \subseteq \pi_j}^{i,j}$ has a size not larger than that of the assumption $\bigvee_i \tau_i \subseteq \bigwedge_j \pi_j$.*

**Proof** By induction on right-leaning $\subseteq$ derivations.

**Case S-Refl.**

**Case** $\bigwedge_j \pi_j = \pi_1 = \bigvee_i \tau_i$**.** By repeated applications of S-TRANS with S-ANDOR11$\cdot$, followed by an application of S-TRANS with S-ANDOR12$\cdot$, we have $\overline{\tau_i \subseteq \bigvee_i \tau_i}^{\,i}$, i.e., $\overline{\tau_i \subseteq \pi_j}^{\,i,j}$.

If $\bigvee_i \tau_i = \tau_1$ where $\tau_1$ is not an intersection, then $\bigvee_i \tau_i = \tau_1$. Then $\overline{\tau_i \subseteq \pi_j}^{\,i,j}$ is just $\tau_1 \subseteq \pi_1$, which is the assumption itself.

If $\bigwedge_j \pi_j = \pi_1$ where $\pi_1$ is not a union, then $\pi_1 = \bigvee_i \tau_i$ is not a union, i.e., $\bigvee_i \tau_i = \tau_1$. Then $\overline{\tau_i \subseteq \pi_j}^{\,i,j}$ is just $\tau_1 \subseteq \pi_1$, which is the assumption itself.

**Case** $\bigvee_i \tau_i = \tau_1 = \bigwedge_j \pi_j$**.** By repeated applications of S-TRANS with S-ANDOR11$\supset$, followed by an application of S-TRANS with S-ANDOR12$\supset$, we have $\overline{\bigwedge_j \pi_j \subseteq \pi_j}^{\,j}$, i.e., $\overline{\tau_i \subseteq \pi_j}^{\,i,j}$.

If $\bigvee_i \tau_i = \tau_1$ where $\tau_1$ is not an intersection, then $\tau_1 = \bigwedge_j \pi_j$ is not an intersection, i.e., $\bigwedge_j \pi_j = \pi_1$. Then $\overline{\tau_i \subseteq \pi_j}^{\,i,j}$ is just $\tau_1 \subseteq \pi_1$, which is the assumption itself.

If $\bigwedge_j \pi_j = \pi_1$ where $\pi_1$ is not a union, then $\bigwedge_j \pi_j = \pi_1$. Then $\overline{\tau_i \subseteq \pi_j}^{\,i,j}$ is just $\tau_1 \subseteq \pi_1$, which is the assumption itself.

**Case S-TOB$\cdot$.** $\bigwedge_j \pi_j = \top$. The result follows from S-TOB$\cdot$ on each of $\overline{\tau_i}^{\,i}$.

**Case S-TOB$\supset$.** $\bigvee_i \tau_i = \bot$. The result follows from S-TOB$\supset$ on each of $\overline{\pi_j}^{\,j}$.

**Case S-COMPL$\cdot$.** $\bigvee_i \tau_i = \top$ and $\bigwedge_j \pi_j = \pi_1 = \pi' \vee \neg\pi'$ for some $\pi'$. The result follows immediately.

**Case S-COMPL$\supset$.** $\bigwedge_j \pi_j = \bot$ and $\bigvee_i \tau_i = \tau_1 = \tau' \wedge \neg\tau'$ for some $\tau'$. The result follows immediately.

**Case S-ANDOR11$\cdot$.** $\bigwedge_j \pi_j = \pi_1 = \bigvee_i \tau_i \vee \pi'$ for some $\pi'$. By repeated applications of S-TRANS with S-ANDOR11$\cdot$, followed by an application of S-TRANS with S-ANDOR12$\cdot$, we have $\overline{\tau_i \subseteq \bigvee_i \tau_i \vee \pi'}^{\,i}$, i.e., $\overline{\tau_i \subseteq \pi_j}^{\,i,j}$.

If $\bigvee_i \tau_i = \tau_1$ where $\tau_1$ is not an intersection, then $\overline{\tau_i \subseteq \pi_j}^{\,i,j}$ is just $\tau \subseteq \tau \vee \pi'$, which is the assumption itself.

It is impossible to have $\bigwedge_j \pi_j = \pi_1$ where $\pi_1$ is not a union since $\pi_1 = \bigvee_i \tau_i \vee \pi'$.

**Case S-ANDOR11$\supset$.** $\bigvee_i \tau_i = \tau_1 = \bigwedge_j \pi_j \wedge \tau'$ for some $\tau'$. By repeated applications of S-TRANS with S-ANDOR11$\supset$, followed by an application of S-TRANS with S-ANDOR12$\supset$, we have $\overline{\bigwedge_j \pi_j \wedge \tau' \subseteq \pi_j}^{\,j}$, i.e., $\overline{\tau_i \subseteq \pi_j}^{\,i,j}$.

It is impossible to have $\bigvee_i \tau_i = \tau_1$ where $\tau_1$ is not an intersection since $\pi_1 = \bigvee_i \tau_i \vee \pi'$.

If $\bigwedge_j \pi_j = \pi_1$ where $\pi_1$ is not a union, then $\overline{\tau_i \subseteq \pi_j}^{\,i,j}$ is just $\tau \subseteq \tau \vee \pi'$, which is the assumption itself.

**Cases S-ANDOR12$\diamond$.** Similar to the cases S-ANDOR11$\diamond$.

**Case S-ANDOR2$\cdot$.** Let the range of $i$ be $1..m$. We have $\bigvee_i \tau_i = \bigvee_{i \in 1..m-1} \tau_i \vee \tau_m$. The premises of the rule are $\bigvee_{i \in 1..m-1} \tau_i \subseteq \bigwedge_j \pi_j$ and $\tau_m \subseteq \bigwedge_j \pi_j$. By IH on the first premise, we have $\overline{\tau_i \subseteq \pi_j}^{\,i \in 1..m-1, j}$. By IH on the second premise, we have $\overline{\tau_m \subseteq \pi_j}^{\,j}$. Then we have $\overline{\tau_i \subseteq \pi_j}^{\,i,j}$.

**Case S-ANDOR2$\supset$.** Let the range of $j$ be $1..n$. We have $\bigwedge_j \pi_j = \bigwedge_{j \in 1..n-1} \pi_j \wedge \pi_j$. The premises of the rule are $\bigvee_i \tau_i \subseteq \bigwedge_{j \in 1..n-1} \pi_j$ and $\bigvee_i \tau_i \subseteq \pi_n$. By IH on the first

premise, we have $\overline{\tau_i \subseteq \pi_j}^{\,i,j \in 1..n-1}$. By IH on the second premise, we have $\overline{\tau_i \subseteq \pi_n}^{\,i}$. Then we have $\overline{\tau_i \subseteq \pi_j}^{\,i,j}$.

**Case S-Distrib$\cdot$.** $\bigvee_i \tau_i = \tau_1 = \tau' \wedge (\tau'_1 \vee \tau'_2)$ and $\bigwedge_j \pi_j = \pi_1 = (\tau' \wedge \tau'_1) \vee (\tau' \wedge \tau'_2)$ for some $\tau'$ and $\tau'_1$ and $\tau'_2$. The result follows immediately.

**Case S-Distrib$\wr$.** $\bigwedge_j \pi_j = \pi_1 = \tau' \vee (\tau'_1 \wedge \tau'_2)$ and $\bigvee_i \tau_i = \tau_1 = (\tau' \vee \tau'_1) \wedge (\tau' \vee \tau'_2)$ for some $\tau'$ and $\tau'_1$ and $\tau'_2$. The result follows immediately.

**Case S-Trans.** The premises of the rule are $\bigvee_i \tau_i \subseteq \tau'$ and $\tau' \subseteq \bigwedge_j \pi_j$ for some $\tau'$. By IH on the former premise, we have $\overline{\tau_i \subseteq \tau'}^{\,i}$. By IH on the latter premise, we have $\overline{\tau' \subseteq \pi_j}^{\,j}$. The result follows from S-Trans on each of $\overline{\tau_i \subseteq \tau'}^{\,i}$ with each of $\overline{\tau_i \subseteq \tau'}^{\,i}$.

■

**Proof** [Lemma 4.9] By straightforward induction on $\preceq$ rules. ■

**Proof** [Lemma 4.10]

(A) By induction on right-leaning $\subseteq$ derivations. We only consider rules that can syntactically apply. Denote the size of the current derivation as $n$.

**Case S-Refl.** Immediate.

**Case S-AndOr2$\cdot$.** $U^C = U_1^{C_1} \vee U_2^{C_2}$ for some $U_1^{C_1}$ and $U_2^{C_2}$, where $U_2^{C_2}$ is not a union. The premises of the rule are $U_1^{C_1} \subseteq \tau$ and $U_2^{C_2} \subseteq \tau$. By IH, we have $U_1^{C_1} = \bigvee_k \tau$ and $U_2^{C_2} = \bigvee_l \tau$. Since $U_2^{C_2}$ is not a union, $U_2^{C_2} = \tau$. Then $U^C = U_1^{C_1} \vee U_2^{C_2} = \bigvee_k \tau \vee \tau$.

**Case S-Trans.** Then the premises are $U^C \subseteq \tau'$ and $\tau' \subseteq \tau$ for some $\tau'$, both of size $n-1$. By induction on the size of the subderivation for the former premise, denoted by $m$. Denote the inner induction hypothesis as IH$'$.

**Cases (S-Refl, $*$), ($*$, S-Refl).** By IH on the other premise.

**Cases (S-ToB$\cdot$, $*$).** Then $\tau' = \top$. The latter premise is $\top \subseteq \tau$, which is impossible by Lemma A.62. Therefore this case is impossible.

**Cases (S-Compl$\cdot$, $*$).** Then $U^C = \top$. The conclusion is $\top \subseteq \tau$, which is impossible by Lemma A.62. Therefore this case is impossible.

**Cases (S-AndOr11$\cdot$, $*$).** Then $\tau' = U^C \vee \tau'_1$ for some $\tau'_1$. By Lemma A.44 on the latter premise, we have $U^C \subseteq \tau$ with a derivation of size at most $n-1$. The result then follows from IH.

**Cases (S-AndOr12$\cdot$, $*$).** Then $\tau' = \tau'_1 \vee U^C$ for some $\tau'_1$. By Lemma A.44 on the latter premise, we have $U^C \subseteq \tau$ with a derivation of size at most $n-1$. The result then follows from IH.

**Cases (S-AndOr2$\cdot$, $*$).** Then $U^C = U_1^{C_1} \vee U_2^{C_2}$ for some $U_1^{C_1}$ and $U_2^{C_2}$, where $U_2^{C_2}$ is not a union. The premises of the former rule are $U_1^{C_1} \subseteq \tau'$ and $U_2^{C_2} \subseteq \tau'$, both of size $m-1$. By S-Trans with $\tau' \subseteq \tau$, we have $U_1^{C_1} \subseteq \tau$ and $U_2^{C_2} \subseteq \tau$, both of size $n$ with a former premise of size $m-1$. Then by IH$'$, we have $U_1^{C_1} \cong \tau$ and $U_2^{C_2} \cong \tau$, which imply $U_1^{C_1} \vee U_2^{C_2} \cong \tau$.

**Cases (S-AndOr2$\wr$, $*$).** Then $\tau' = \tau'_1 \wedge \tau'_2$ for some $\tau'_1$ and $\tau'_2$. The premises of the former rule are $U^C \subseteq \tau'_1$ and $U^C \subseteq \tau'_2$, both of size $m-1$. By

Lemma A.57 on the latter premise, we have $\tau'_l \subseteq \tau$ of size at most $n-1$ for some $l \in \{1,2\}$. By S-Trans on $U^C \subseteq \tau'_l$ and $\tau'_l \subseteq \tau$, we have $U^C \subseteq \tau$ of size $n$ with a former premise of size $m-1$. The result then follows from IH$'$.

(B) By induction on right-leaning $\subseteq$ derivations. We only consider rules that can syntactically apply. Denote the size of the current derivation as $n$.

**Case S-Refl.** Immediate.

**Case S-AndOr2↷.** $X^C = X_1^{C_1} \wedge X_2^{C_2}$ for some $X_1^{C_1}$ and $X_2^{C_2}$, where $X_2^{C_2}$ is not a intersection. The premises of the rule are $\tau \subseteq X_1^{C_1}$ and $\tau \subseteq X_2^{C_2}$. By IH, we have $X_1^{C_1} = \bigwedge_k \tau$ and $X_2^{C_2} = \bigwedge_l \tau$. Since $X_2^{C_2}$ is not a intersection, $X_2^{C_2} = \tau$. Then $X^C = X_1^{C_1} \wedge X_2^{C_2} = \bigwedge_k \tau \wedge \tau$.

**Case S-Trans.** Then the premises are $\tau \subseteq \tau'$ and $\tau' \subseteq X^C$ for some $\tau'$, both of size $n-1$. By induction on the size of the subderivation of the former premise, denoted by $m$. Denote the inner induction hypothesis as IH$'$.

**Cases (S-Refl, $*$), ($*$, S-Refl).** By IH on the other premise.

**Cases (S-ToB$\cdot$, $*$).** Then $\tau' = \top$. The latter premise is $\top \subseteq X^C$, which implies $\top \subseteq X_2^{C_2}$ for some $X_2^{C_2} \in \{\pi_1 \to \pi_2, \{x' : \pi_1\}, \#C'\}$ by Lemma A.44, where $X^C = X_1^{C_1} \wedge X_2^{C_2}$, which is impossible by Lemma A.62. Therefore this case is impossible.

**Cases (S-AndOr11$\cdot$, $*$).** Then $\tau' = \tau \vee \tau'_1$ for some $\tau'_1$. By Lemma A.44 on the latter premise, we have $\tau \subseteq X^C$ with a derivation of size at most $n-1$. The result then follows from IH.

**Cases (S-AndOr12$\cdot$, $*$).** Then $\tau' = \tau'_1 \vee \tau$ for some $\tau'_1$. By Lemma A.44 on the latter premise, we have $\tau \subseteq X^C$ with a derivation of size at most $n-1$. The result then follows from IH.

**Cases (S-AndOr2↷, $*$).** Then $\tau' = \tau'_1 \wedge \tau'_2$ for some $\tau'_1$ and $\tau'_2$. The premises of the former rule are $\tau \subseteq \tau'_1$ and $\tau \subseteq \tau'_2$, both of size $m-1$. By Lemma A.44 on the latter premise, we have $\overline{\tau'_1 \wedge \tau'_2 \subseteq X_i^{C_i}}^{\,i}$, where $X^C = \bigwedge_i X_i^{C_i}$ and $\overline{X_i^{C_i}}^{\,i}$ are not intersections, each of size at most $n-1$. Then by Lemma A.57, we have $\overline{\tau'_{l_i} \subseteq X_i^{C_i}}^{\,i}$ for some $\overline{l_i \in \{1,2\}}^{\,i}$, each of size at most $n-1$. By S-Trans on $\tau \subseteq \tau'_l$ and $\tau'_l \subseteq X_i^{C_i}$, we have $\overline{\tau \subseteq X_i^{C_i}}^{\,i}$, each of size $n$ with a former premise of size $m-1$. Then $\overline{X_i^{C_i} = \tau}^{\,i}$ by IH$'$ (note that $\overline{X_i^{C_i}}^{\,i}$ are not intersections), i.e., $X^C = \bigwedge_i \tau$.

■

**Lemma A.45.**

*(A) If $\top \leqslant \tau$, then $U^C \subseteq \tau$ for some $U^C$ and $C \in \{\top, \not\perp\}$.*
*(B) If $\tau \leqslant \bot$, then $\tau \subseteq X^C$ for some $X^C$ and $C \in \{\bot, \not\top\}$.*

**Proof** By straightforward induction on subtyping derivations. ■

### A.8 CDN- and DCN-normalized type forms and derivations

**Proof** [Lemma 4.21] By induction on unassuming subtyping derivations.

**Case S-Refl.** Then $\tau = \pi$, which implies $\mathrm{cdn}(\tau) = \mathrm{cdn}(\pi)$. Then we have $\mathrm{cdn}(\tau) \leqslant^{\mathrm{cdn}} \mathrm{cdn}(\pi)$ by S-CDN.

**Case S-ToB$\cdot$.** Then $\pi = \top$ and $\mathrm{cdn}(\pi) = \top$. Then we have $\mathrm{cdn}(\tau) \leqslant^{\mathrm{cdn}} \top$ by S-ToB$\cdot$.

**Case S-ToB$\curvearrowright$.** Then $\tau = \bot$ and $\mathrm{cdn}(\tau) = \neg\top$. Then we have $\neg\top \leqslant^{\mathrm{cdn}} \mathrm{cdn}(\pi)$ by S-ToB$\curvearrowright$.

**Case S-Compl$\cdot$.** Then $\tau = \top$ and $\pi = \pi' \vee \neg\pi'$ for some $\pi'$. Let $\mathrm{cdn}(\pi') = \bigwedge_{i \in 1..m} \bigvee_{j_i \in 1..n_i} \pi^{\mathrm{n}}_{ij_i}$. Then $\mathrm{cdn}(\neg\pi') = \mathrm{neg}(\mathrm{cdn}(\pi')) = \bigwedge_{\overline{j_{i'} \in 1..n_{i'}}^{\,i' \in 1..m}} \bigvee_{i \in 1..m} \mathrm{neg}(\pi^{\mathrm{n}}_{ij_i})$. Then $\mathrm{cdn}(\pi' \vee \neg\pi') = \mathrm{dis}(\mathrm{cdn}(\pi), \mathrm{cdn}(\pi')) = \bigwedge_{i \in 1..m,\, \overline{j_{i'} \in 1..n_{i'}}^{\,i' \in 1..m}} \left( \bigvee_{j'_i \in 1..n_i} \pi^{\mathrm{n}}_{ij'_i} \vee \bigvee_{i' \in 1..m} \mathrm{neg}(\pi^{\mathrm{n}}_{i'j_{i'}}) \right)$. For each $i$, $\overline{j_{i'}}^{\,i' \in 1..m}$, $\bigvee_{j'_i \in 1..n_i} \pi^{\mathrm{n}}_{ij'_i}$ contains the disjunct $\pi^{\mathrm{n}}_{ij_i}$, and $\bigvee_{i' \in 1..m} \mathrm{neg}(\pi^{\mathrm{n}}_{i'j_{i'}})$ contains the disjunct $\mathrm{neg}(\pi^{\mathrm{n}}_{ij_i})$. Then by commutativity, we have $\bigvee_{j'_i \in 1..n_i} \pi^{\mathrm{n}}_{ij'_i} \vee \bigvee_{i' \in 1..m} \mathrm{neg}(\pi^{\mathrm{n}}_{i'j_{i'}}) \geqslant^{\mathrm{cdn}} \bigvee_{j'_i \in 1..n_i \setminus \{ j_i \}} \pi^{\mathrm{n}}_{ij'_i} \vee \bigvee_{i' \in 1..m \setminus \{ i \}} \mathrm{neg}(\pi^{\mathrm{n}}_{i'j_{i'}}) \vee \pi^{\mathrm{n}}_{ij_i} \vee \mathrm{neg}(\pi^{\mathrm{n}}_{ij_i})$, which implies $\top \leqslant^{\mathrm{cdn}} \bigvee_{j'_i \in 1..n_i} \pi^{\mathrm{n}}_{ij'_i} \vee \bigvee_{i' \in 1..m} \mathrm{neg}(\pi^{\mathrm{n}}_{i'j_{i'}})$. Finally by S-AndOr2$\curvearrowright$, we have $\top \leqslant^{\mathrm{cdn}} \mathrm{cdn}(\pi' \vee \neg\pi')$.

**Case S-Compl$\curvearrowright$.** Then $\tau = \tau' \wedge \neg\tau'$ and $\pi = \bot$ for some $\tau'$. Let $\mathrm{cdn}(\tau') = \bigwedge_{i \in 1..m} \bigvee_{j_i \in 1..n_i} \tau^{\mathrm{n}}_{ij_i}$. Then $\mathrm{cdn}(\neg\tau') = \mathrm{neg}(\mathrm{cdn}(\tau')) = \bigwedge_{\overline{j_i \in 1..n_i}^{\,i \in 1..m}} \bigvee_{i \in 1..m} \mathrm{neg}(\tau^{\mathrm{n}}_{ij_i})$. We want to show $\bigwedge_{i \in 1..m} \bigvee_{j_i \in 1..n_i} \tau^{\mathrm{n}}_{ij_i} \wedge \bigwedge_{\overline{j_i \in 1..n_i}^{\,i \in 1..m}} \bigvee_{i \in 1..m} \mathrm{neg}(\tau^{\mathrm{n}}_{ij_i}) \leqslant \neg\top$. By S-DistribCdn$\cdot$, it suffices to show $\overline{\tau^{\mathrm{n}}_{1j'_1} \wedge \bigwedge_{i \in 2..m} \bigvee_{j_i \in 1..n_i} \tau^{\mathrm{n}}_{ij_i} \wedge \bigwedge_{\overline{j_i \in 1..n_i}^{\,i \in 1..m}} \bigvee_{i \in 1..m} \mathrm{neg}(\tau^{\mathrm{n}}_{ij_i}) \leqslant \neg\top}^{\,j'_1 \in 1..n_1}$, i.e., $\overline{\tau^{\mathrm{n}}_{1j'_1} \wedge \bigwedge_{i \in 2..m} \bigvee_{j_i \in 1..n_i} \tau^{\mathrm{n}}_{ij_i} \wedge \bigwedge_{\overline{j_i \in 1..n_i}^{\,i \in 1..m}} \bigvee_{i \in 1..m \setminus \{ i' \in 1..1 \mid j_{i'} = j'_{i'} \}} \mathrm{neg}(\tau^{\mathrm{n}}_{ij_i}) \leqslant \neg\top}^{\,j'_1 \in 1..n_1}$ since $\tau^{\mathrm{n}}_{1j'_1} \wedge (\mathrm{neg}(\tau^{\mathrm{n}}_{1j'_1}) \vee \tau'') \leqslant \tau^{\mathrm{n}}_{1j'_1} \wedge \tau''$ for any $\tau''$. Repeating the process, it suffices to show $\overline{\tau^{\mathrm{n}}_{1j'_1} \wedge \tau^{\mathrm{n}}_{2j'_2} \wedge \bigwedge_{i \in 3..m} \bigvee_{j_i \in 1..n_i} \tau^{\mathrm{n}}_{ij_i} \wedge \bigwedge_{\overline{j_i \in 1..n_i}^{\,i \in 1..m}} \bigvee_{i \in 1..m \setminus \{ i' \in 1..2 \mid j_{i'} = j'_{i'} \}} \mathrm{neg}(\tau^{\mathrm{n}}_{ij_i}) \leqslant \neg\top}^{\,j'_1 \in 1..n_1,\, j'_2 \in 1..n_2}$. Repeating the process $m$ times, it suffices to show $\overline{\bigwedge_{i \in 1..m} \tau^{\mathrm{n}}_{ij'_i} \wedge \bigwedge_{\overline{j_i \in 1..n_i}^{\,i \in 1..m}} \bigvee_{i \in 1..m \setminus \{ i' \in 1..m \mid j_{i'} = j'_{i'} \}} \mathrm{neg}(\tau^{\mathrm{n}}_{ij_i}) \leqslant \neg\top}^{\,\overline{j_i \in 1..n_i}^{\,i \in 1..m}}$, which is indeed true since one of the conjuncts is an empty union, i.e., $\neg\top$, when $\overline{j_i = j'_i}^{\,i \in 1..m}$.

**Case S-AndOr11$\cdot$.** Then $\pi = \tau \vee \pi'$ for some $\pi'$. Then $\mathrm{cdn}(\pi) = \mathrm{dis}(\mathrm{cdn}(\tau), \mathrm{cdn}(\pi'))$. Let $\mathrm{cdn}(\tau) = \bigwedge_i \tau^{\mathrm{dn}}_i$ and $\mathrm{cdn}(\pi') = \bigwedge_j \pi^{\mathrm{dn}}_j$. Then $\mathrm{dis}(\mathrm{cdn}(\tau), \mathrm{cdn}(\pi')) = \bigwedge_{i,j} (\tau^{\mathrm{dn}}_i \vee \pi^{\mathrm{dn}}_j)$. By S-AndOr1$\cdot$, we have $\overline{\tau^{\mathrm{dn}}_i \leqslant^{\mathrm{cdn}} \tau^{\mathrm{dn}}_i \vee \pi^{\mathrm{dn}}_j}^{\,i,j}$, which imply $\overline{\tau^{\mathrm{dn}}_i \leqslant^{\mathrm{cdn}} \bigwedge_j (\tau^{\mathrm{dn}}_i \vee \pi^{\mathrm{dn}}_j)}^{\,i}$ by S-AndOr2$\curvearrowright$, which imply $\bigwedge_i \tau^{\mathrm{dn}}_i \leqslant^{\mathrm{cdn}} \bigwedge_{i,j} (\tau^{\mathrm{dn}}_i \vee \pi^{\mathrm{dn}}_j)$ by Lemma A.7$\curvearrowright$, i.e., $\mathrm{cdn}(\tau) \leqslant^{\mathrm{cdn}} \mathrm{cdn}(\pi)$.

**Case S-AndOr11$\curvearrowright$.** Then $\tau = \pi \wedge \tau'$ for some $\tau'$. Then $\mathrm{cdn}(\tau) = \mathrm{con}(\mathrm{cdn}(\pi), \mathrm{cdn}(\tau')) = \mathrm{cdn}(\pi) \wedge \mathrm{cdn}(\tau')$. Let $\mathrm{cdn}(\tau') = \bigwedge_i \tau^{\mathrm{dn}}_i$ and

$\mathrm{cdn}(\pi) = \bigwedge_j \pi_j^{\mathrm{dn}}$. By S-AndOr1$\curvearrowright$, we have $\bigwedge_j \pi_j^{\mathrm{dn}} \wedge \bigwedge_i \tau_i^{\mathrm{dn}} \leqslant^{\mathrm{cdn}} \bigwedge_j \pi_j^{\mathrm{dn}}$, i.e., $\mathrm{cdn}(\tau) \leqslant^{\mathrm{cdn}} \mathrm{cdn}(\pi)$.

**Cases S-AndOr12$\diamond$.** Similar to the cases above.

**Case S-AndOr2$\cdot$.** Then $\tau = \tau_1 \vee \tau_2$ for some $\tau_1$ and $\tau_2$. By IH on the premises, we have $\mathrm{cdn}(\Sigma) \vdash \mathrm{cdn}(\tau_1) \leqslant^{\mathrm{cdn}} \mathrm{cdn}(\pi)$ and $\mathrm{cdn}(\Sigma) \vdash \mathrm{cdn}(\tau_2) \leqslant^{\mathrm{cdn}} \mathrm{cdn}(\pi)$. Then by Corollary A.47, we have $\mathrm{cdn}(\Sigma) \vdash \mathrm{dis}(\mathrm{cdn}(\tau_1), \mathrm{cdn}(\tau_2)) \leqslant^{\mathrm{cdn}} \mathrm{cdn}(\pi)$, i.e., $\mathrm{cdn}(\Sigma) \vdash \mathrm{cdn}(\tau_1 \vee \tau_2) \leqslant^{\mathrm{cdn}} \mathrm{cdn}(\pi)$.

**Case S-AndOr2$\curvearrowright$.** Then $\pi = \pi_1 \wedge \pi_2$ for some $\pi_1$ and $\pi_2$. By IH on the premises, we have $\mathrm{cdn}(\Sigma) \vdash \mathrm{cdn}(\tau) \leqslant^{\mathrm{cdn}} \mathrm{cdn}(\pi_1)$ and $\mathrm{cdn}(\Sigma) \vdash \mathrm{cdn}(\tau) \leqslant^{\mathrm{cdn}} \mathrm{cdn}(\pi_2)$. Let $\mathrm{cdn}(\pi_1) = \bigwedge_i \pi_{1i}^{\mathrm{dn}}$ and $\mathrm{cdn}(\pi_2) = \bigwedge_j \pi_{2j}^{\mathrm{dn}}$. By Lemma 3.1, we have $\overline{\mathrm{cdn}(\Sigma) \vdash \mathrm{cdn}(\tau) \leqslant^{\mathrm{cdn}} \pi_{1i}^{\mathrm{dn}}}^{\,i}$ and $\overline{\mathrm{cdn}(\Sigma) \vdash \mathrm{cdn}(\tau) \leqslant^{\mathrm{cdn}} \pi_{2j}^{\mathrm{dn}}}^{\,j}$. Then by S-AndOr2$\curvearrowright$, we have $\mathrm{cdn}(\Sigma) \vdash \mathrm{cdn}(\tau) \leqslant^{\mathrm{cdn}} \bigwedge_i \pi_{1i}^{\mathrm{dn}} \wedge \bigwedge_j \pi_{2j}^{\mathrm{dn}} = \mathrm{cdn}(\pi_1 \wedge \pi_2)$.

**Case S-Distrib$\cdot$.** Then $\tau = \tau_0 \wedge (\tau_1 \vee \tau_2)$ and $\pi = (\tau_0 \wedge \tau_1) \vee (\tau_0 \wedge \tau_2)$ for some $\tau_0$ and $\tau_1$ and $\tau_2$. Let $\mathrm{cdn}(\tau_0) = \bigwedge_k \tau_{0k}^{\mathrm{dn}}$, $\mathrm{cdn}(\tau_1) = \bigwedge_i \tau_{1i}^{\mathrm{dn}}$, and $\mathrm{cdn}(\tau_2) = \bigwedge_j \tau_{2j}^{\mathrm{dn}}$. Then we have:

$$\begin{aligned}\mathrm{cdn}(\tau) &= \mathrm{con}(\mathrm{cdn}(\tau_0), \mathrm{dis}(\mathrm{cdn}(\tau_1), \mathrm{cdn}(\tau_2))) \\ &= \textstyle\bigwedge_k \tau_{0k}^{\mathrm{dn}} \wedge \bigwedge_{i,j} (\tau_{1i}^{\mathrm{dn}} \vee \tau_{2j}^{\mathrm{dn}})\end{aligned}$$

$$\begin{aligned}\mathrm{cdn}(\pi) &= \mathrm{dis}(\mathrm{con}(\mathrm{cdn}(\tau_0), \mathrm{cdn}(\tau_1)), \mathrm{con}(\mathrm{cdn}(\tau_0), \mathrm{cdn}(\tau_2))) \\ &= \mathrm{dis}(\textstyle\bigwedge_k \tau_{0k}^{\mathrm{dn}} \wedge \bigwedge_i \tau_{1i}^{\mathrm{dn}}, \bigwedge_k \tau_{0k}^{\mathrm{dn}} \wedge \bigwedge_j \tau_{2j}^{\mathrm{dn}}) \\ &= \mathrm{dis}(\textstyle\bigwedge_{i' \in \{\overline{0k}^k, \overline{1i}^i\}} \tau_{i'}^{\mathrm{dn}}, \bigwedge_{j' \in \{\overline{0k}^k, \overline{2j}^j\}} \tau_{j'}^{\mathrm{dn}}) \\ &= \textstyle\bigwedge_{i' \in \{\overline{0k}^k, \overline{1i}^i\},\, j' \in \{\overline{0k}^k, \overline{2j}^j\}} (\tau_{i'}^{\mathrm{dn}} \vee \tau_{j'}^{\mathrm{dn}})\end{aligned}$$

For each $i' \in \{\overline{0k}^k, \overline{1i}^i\}$, $j' \in \{\overline{0k}^k, \overline{2j}^j\}$, we have the following: If $i' = 0k_1$ for some $k_1$, then we have $\tau_{0k_1}^{\mathrm{dn}} \leqslant^{\mathrm{cdn}} \tau_{0k_1}^{\mathrm{dn}} \vee \tau_{j'}^{\mathrm{dn}}$ by S-AndOr1$\cdot$. If $j' = 0k_2$ for some $k_2$, then we have $\tau_{0k_2}^{\mathrm{dn}} \leqslant^{\mathrm{cdn}} \tau_{i'}^{\mathrm{dn}} \vee \tau_{0k_2}^{\mathrm{dn}}$ by S-AndOr1$\cdot$. Otherwise, we have $\tau_{1i}^{\mathrm{dn}} \vee \tau_{2j}^{\mathrm{dn}} \leqslant^{\mathrm{cdn}} \tau_{1i}^{\mathrm{dn}} \vee \tau_{2j}^{\mathrm{dn}}$ by S-Refl.

Then we have $\bigwedge_k \tau_{0k}^{\mathrm{dn}} \wedge \bigwedge_{i,j} (\tau_{1i}^{\mathrm{dn}} \vee \tau_{2j}^{\mathrm{dn}}) \leqslant^{\mathrm{cdn}} \bigwedge_{i' \in \{\overline{0k}^k, \overline{1i}^i\},\, j' \in \{\overline{0k}^k, \overline{2j}^j\}} (\tau_{i'}^{\mathrm{dn}} \vee \tau_{j'}^{\mathrm{dn}})$ by Lemma A.7$\curvearrowright$, commutativity, and idempotence, i.e., $\mathrm{cdn}(\tau_0 \wedge (\tau_1 \vee \tau_2)) \leqslant^{\mathrm{cdn}} \mathrm{cdn}((\tau_0 \wedge \tau_1) \vee (\tau_0 \wedge \tau_2))$.

**Case S-Distrib$\curvearrowright$.** Then $\tau = (\tau_0 \vee \tau_1) \wedge (\tau_0 \vee \tau_2)$ and $\pi = \tau_0 \vee (\tau_1 \wedge \tau_2)$ for some $\tau_0$ and $\tau_1$ and $\tau_2$. Let $\mathrm{cdn}(\tau_0) = \bigwedge_k \tau_{0k}^{\mathrm{dn}}$, $\mathrm{cdn}(\tau_1) = \bigwedge_i \tau_{1i}^{\mathrm{dn}}$, and $\mathrm{cdn}(\tau_2) = \bigwedge_j \tau_{2j}^{\mathrm{dn}}$. Then we have $\mathrm{cdn}((\tau_0 \vee \tau_1) \wedge (\tau_0 \vee \tau_2)) = \bigwedge_{k,i} (\tau_{0k}^{\mathrm{dn}} \vee \tau_{1i}^{\mathrm{cdn}}) \wedge (\tau_{0k}^{\mathrm{dn}} \vee \tau_{2j}^{\mathrm{dn}})$ and $\mathrm{cdn}(\tau_0 \vee (\tau_1 \wedge \tau_2)) = \bigwedge_{k,i} (\tau_{0k}^{\mathrm{dn}} \vee \tau_{1i}^{\mathrm{cdn}}) \wedge (\tau_{0k}^{\mathrm{dn}} \vee \tau_{2j}^{\mathrm{dn}})$. Then we have $\mathrm{cdn}((\tau_0 \vee \tau_1) \wedge (\tau_0 \vee \tau_2)) \leqslant^{\mathrm{cdn}} \mathrm{cdn}(\tau_0 \vee (\tau_1 \wedge \tau_2))$ by S-Refl.

**Case S-Trans.** By IH on the premises, followed by S-Trans.

**Case S-Hyp.** Then the premise of the rule is $(\tau \leqslant \pi) \in \Sigma$. Let $\mathrm{cdn}(\neg\tau \vee \pi) = \bigwedge_i \bigvee_{j_i} \tau_{ij_i}^{\mathrm{n}}$. Then we have $\overline{\mathrm{cdn}(\top \leqslant \bigvee_{j_i} \tau_{ij_i}^{\mathrm{n}}) \subseteq \mathrm{cdn}(\Sigma)}^{\,i}$. For each $i$, we have:

**Case** $\exists\alpha.\,\{\,\alpha,\ \neg\alpha\,\} \subseteq \{\,\overline{\tau^{\mathrm{n}}_{ij_i}}^{\,j_i}\,\}$**.** Then we have $\top \leqslant^{\mathrm{cdn}} \alpha \vee \neg\alpha$ by S-Compl$\cdot$ and $\alpha \vee \neg\alpha \leqslant^{\mathrm{cdn}} \bigvee_{j_i} \tau^{\mathrm{n}}_{ij_i}$ by S-AndOr1$\cdot$ for some $\alpha$, which imply $\top \leqslant^{\mathrm{cdn}} \bigvee_{j_i} \tau^{\mathrm{n}}_{ij_i}$ by S-Trans.

**Case** $(\exists\alpha.\,\alpha \in \{\,\overline{\tau^{\mathrm{n}}_{ij_i}}^{\,j_i}\,\})$ **and** $(\forall\alpha \in \{\,\overline{\tau^{\mathrm{n}}_{ij_i}}^{\,j_i}\,\}.\,\neg\alpha \notin \{\,\overline{\tau^{\mathrm{n}}_{ij_i}}^{\,j_i}\,\})$**.** Then $(\bigwedge_{j_i \mid \tau^{\mathrm{n}}_{ij_i} \neq \alpha} \mathrm{neg}(\tau^{\mathrm{n}}_{ij_i}) \leqslant \alpha) \in \mathrm{cdn}(\Sigma)$ for some $\alpha$ and we have $\mathrm{cdn}(\Sigma) \vdash \bigwedge_{j_i \mid \tau^{\mathrm{n}}_{ij_i} \neq \alpha} \mathrm{neg}(\tau^{\mathrm{n}}_{ij_i}) \leqslant^{\mathrm{cdn}} \alpha$ by S-Hyp, which implies $\mathrm{cdn}(\Sigma) \vdash \top \leqslant^{\mathrm{cdn}} \bigvee_{j_i} \tau^{\mathrm{n}}_{ij_i}$ by Theorem A.9.

**Case** $(\exists\alpha.\,\neg\alpha \in \{\,\overline{\tau^{\mathrm{n}}_{ij_i}}^{\,j_i}\,\})$ **and** $(\forall\alpha \in \{\,\overline{\tau^{\mathrm{n}}_{ij_i}}^{\,j_i}\,\}.\,\neg\alpha \notin \{\,\overline{\tau^{\mathrm{n}}_{ij_i}}^{\,j_i}\,\})$**.** Then $(\alpha \leqslant \bigvee_{j_i \mid \tau^{\mathrm{n}}_{ij_i} \neq \neg\alpha} \tau^{\mathrm{n}}_{ij_i}) \in \mathrm{cdn}(\Sigma)$ for some $\alpha$ and we have $\mathrm{cdn}(\Sigma) \vdash \alpha \leqslant^{\mathrm{cdn}} \bigvee_{j_i \mid \tau^{\mathrm{n}}_{ij_i} \neq \neg\alpha} \tau^{\mathrm{n}}_{ij_i}$ by S-Hyp, which implies $\mathrm{cdn}(\Sigma) \vdash \top \leqslant^{\mathrm{cdn}} \bigvee_{j_i} \tau^{\mathrm{n}}_{ij_i}$ by Theorem A.9.

**Case** $\forall\alpha.\,\{\,\alpha,\ \neg\alpha\,\} \cap \{\,\overline{\tau^{\mathrm{n}}_{ij_i}}^{\,j_i}\,\} = \varnothing$**.** Then $(\top \leqslant \bigvee_{j_i} \tau^{\mathrm{n}}_{ij_i}) \in \mathrm{cdn}(\Sigma)$ and we have $\mathrm{cdn}(\Sigma) \vdash \top \leqslant^{\mathrm{cdn}} \bigvee_{j_i} \tau^{\mathrm{n}}_{ij_i}$ by S-Hyp.

Then $\overline{\mathrm{cdn}(\Sigma) \vdash \top \leqslant^{\mathrm{cdn}} \bigvee_{j_i} \tau^{\mathrm{n}}_{ij_i}}^{\,i}$ imply $\mathrm{cdn}(\Sigma) \vdash \top \leqslant^{\mathrm{cdn}} \bigwedge_i \bigvee_{j_i} \tau^{\mathrm{n}}_{ij_i}$ by S-AndOr2$\wr$, i.e., $\mathrm{cdn}(\Sigma) \vdash \top \leqslant^{\mathrm{cdn}} \mathrm{cdn}(\neg\tau \vee \pi)$. Let $\mathrm{cdn}(\tau) = \bigwedge_p \bigvee_{q_p} \tau'^{\mathrm{n}}_{pq_p}$ and $\mathrm{cdn}(\pi) = \bigwedge_r \bigvee_{s_r} \pi'^{\mathrm{n}}_{rs_r}$. Then by definition, $\mathrm{cdn}(\neg\tau \vee \pi) = \bigwedge_{\overline{q_p}^{\,p},\, r} (\bigvee_p \mathrm{neg}(\tau'^{\mathrm{n}}_{pq_p}) \vee \bigvee_{s_r} \pi'^{\mathrm{n}}_{rs_r})$. $\mathrm{cdn}(\Sigma) \vdash \top \leqslant^{\mathrm{cdn}} \bigwedge_{\overline{q_p}^{\,p},\, r} (\bigvee_p \mathrm{neg}(\tau'^{\mathrm{n}}_{pq_p}) \vee \bigvee_{s_r} \pi'^{\mathrm{n}}_{rs_r})$ implies $\overline{\mathrm{cdn}(\Sigma) \vdash \top \leqslant^{\mathrm{cdn}} \bigvee_p \mathrm{neg}(\tau'^{\mathrm{n}}_{pq_p}) \vee \bigvee_{s_r} \pi'^{\mathrm{n}}_{rs_r}}^{\,\overline{q_p}^{\,p},\, r}$ by Lemma 3.1, which imply $\overline{\mathrm{cdn}(\Sigma) \vdash \bigwedge_p \tau'^{\mathrm{n}}_{pq_p} \leqslant^{\mathrm{cdn}} \bigvee_{s_r} \pi'^{\mathrm{n}}_{rs_r}}^{\,\overline{q_p}^{\,p},\, r}$ by Theorem A.9, which imply $\overline{\mathrm{cdn}(\Sigma) \vdash \bigwedge_p \tau'^{\mathrm{n}}_{pq_p} \leqslant^{\mathrm{cdn}} \bigwedge_r \bigvee_{s_r} \pi'^{\mathrm{n}}_{rs_r}}^{\,\overline{q_p}^{\,p}}$ by S-AndOr2$\wr$, which imply $\mathrm{cdn}(\Sigma) \vdash \bigwedge_p \bigvee_{q_p} \tau'^{\mathrm{n}}_{pq_p} \leqslant^{\mathrm{cdn}} \bigwedge_r \bigvee_{s_r} \pi'^{\mathrm{n}}_{rs_r}$ by repeated applications of S-DistribCdn$\cdot$ and commutativity i.e., $\mathrm{cdn}(\Sigma) \vdash \mathrm{cdn}(\tau) \leqslant^{\mathrm{cdn}} \mathrm{cdn}(\pi)$.

**Case S-TMrg$\diamond$.** Immediate since it is already in the desired form.

**Case S-TDepth.** By Lemma 4.18, we have $\mathrm{cdn}(\Sigma') \models \Sigma'$, which implies $\triangleleft\mathrm{cdn}(\Sigma') \models \triangleleft\Sigma'$ by Lemma A.22. Then for each premise $\triangleleft\Sigma' \vdash \tau' \leqslant \pi'$, we have $\triangleleft\mathrm{cdn}(\Sigma') \vdash \tau' \leqslant \pi'$ by Lemma A.23. Then we have $\mathrm{cdn}(\Sigma) \vdash \tau \leqslant^{\mathrm{cdn}} \pi$ by S-TDepth.

**Cases** $R \in \mathcal{R}$**.** For each premise $\Sigma' \vdash \tau' \leqslant \pi'$:

- If $\max(\mathit{depth}(\tau'), \mathit{depth}(\pi')) < \max(\mathit{depth}(\tau), \mathit{depth}(\pi))$, then by Lemma 4.18, we have $\mathrm{cdn}(\Sigma') \models \Sigma'$, which implies $\mathrm{cdn}(\Sigma') \vdash \tau' \leqslant \pi'$ by Lemma A.23 on $\Sigma' \vdash \tau' \leqslant \pi'$.
- If $\max(\mathit{depth}(\tau'), \mathit{depth}(\pi')) = \max(\mathit{depth}(\tau), \mathit{depth}(\pi))$, Then we have $\mathrm{cdn}(\Sigma') \vdash \mathrm{cdn}(\tau') \leqslant^{\mathrm{cdn}} \mathrm{cdn}(\pi')$ by IH on $\Sigma' \vdash \tau' \leqslant \pi'$.

Note that we are not allowed to apply $R$ at this point, since the premises of $R$ may have contexts with $\triangleleft$, and $\triangleleft$ does not commute with $\mathrm{cdn}(\cdot)$. $\triangleleft$ and $\mathrm{cdn}(\cdot)$ do commute up to the $\rightleftharpoons$ equivalence relation, so by Lemma A.23 on $\mathrm{cdn}(\Sigma') \vdash \tau' \leqslant \pi'$, we have the appropriate premises with the $\leqslant$ relation. For the remaining premises with the $\leqslant^{\mathrm{cdn}}$ relation, note that we have the restriction that $\Sigma$ ***cons.*** implies $\Sigma'$ ***cons.***, so $\Sigma'$ cannot have additional $\triangleleft$ over $\Sigma$, and the non-commutativity between $\triangleleft$ and $\mathrm{cdn}(\cdot)$ does not prevent us from applying $R$. Then we have $\mathrm{cdn}(\Sigma) \vdash \tau \leqslant^{\mathrm{cdn}} \pi$ by $R$.

■

**Lemma A.46.** *If* $\Sigma \vdash \bigwedge_i \tau_{1i}^{\mathrm{dn}} \leqslant^{\mathrm{cdn}} \tau^{\mathrm{dn}}$ *and* $\Sigma \vdash \bigwedge_j \tau_{2j}^{\mathrm{dn}} \leqslant^{\mathrm{cdn}} \tau^{\mathrm{dn}}$, *then* $\Sigma \vdash \bigwedge_{i,j} (\tau_{1i}^{\mathrm{dn}} \vee \tau_{2j}^{\mathrm{dn}}) \leqslant^{\mathrm{cdn}} \tau^{\mathrm{dn}}$.

**Proof** For each $i$, we have

$$\dfrac{\text{S-DistribCdn}\curlyvee \dfrac{\text{S-AndOr12}\cdot \dfrac{}{\tau_{1i}^{\mathrm{dn}} \leqslant^{\mathrm{cdn}} \tau^{\mathrm{dn}} \vee \tau_{1i}^{\mathrm{dn}}} \qquad \text{S-Trans} \dfrac{\bigwedge_j \tau_{2j}^{\mathrm{dn}} \leqslant^{\mathrm{cdn}} \tau^{\mathrm{dn}} \qquad \text{S-AndOr11}\cdot \dfrac{}{\tau^{\mathrm{dn}} \leqslant^{\mathrm{cdn}} \tau^{\mathrm{dn}} \vee \tau_{1i}^{\mathrm{dn}}}}{\bigwedge_j \tau_{2j}^{\mathrm{dn}} \leqslant^{\mathrm{cdn}} \tau^{\mathrm{dn}} \vee \tau_{1i}^{\mathrm{dn}}}}{\textbf{(i)}\ \bigwedge_j (\tau_{1i}^{\mathrm{dn}} \vee \tau_{2j}^{\mathrm{dn}}) \leqslant^{\mathrm{cdn}} \tau^{\mathrm{dn}} \vee \tau_{1i}^{\mathrm{dn}}}$$

Then we have

$$\dfrac{\text{S-Trans} \dfrac{\text{Lemma A.7}\curlyvee \dfrac{\overline{\textbf{(i)}}^{\,i}}{\bigwedge_{i,j} (\tau_{1i}^{\mathrm{dn}} \vee \tau_{2j}^{\mathrm{dn}}) \leqslant^{\mathrm{cdn}} \bigwedge_i (\tau^{\mathrm{dn}} \vee \tau_{1i}^{\mathrm{dn}})} \qquad \text{S-DistribCdn}\curlyvee \dfrac{\text{S-Refl} \dfrac{}{\tau^{\mathrm{dn}} \leqslant^{\mathrm{cdn}} \tau^{\mathrm{dn}}} \quad \bigwedge_i \tau_{1i}^{\mathrm{dn}} \leqslant^{\mathrm{cdn}} \tau^{\mathrm{dn}}}{\bigwedge_i (\tau^{\mathrm{dn}} \vee \tau_{1i}^{\mathrm{dn}}) \leqslant^{\mathrm{cdn}} \tau^{\mathrm{dn}}}}{\bigwedge_{i,j} (\tau_{1i}^{\mathrm{dn}} \vee \tau_{2j}^{\mathrm{dn}}) \leqslant^{\mathrm{cdn}} \tau^{\mathrm{cdn}}}$$

■

**Corollary A.47.** *If* $\Sigma \vdash \bigwedge_i \tau_{1i}^{\mathrm{dn}} \leqslant^{\mathrm{cdn}} \tau^{\mathrm{cdn}}$ *and* $\Sigma \vdash \bigwedge_j \tau_{2j}^{\mathrm{dn}} \leqslant^{\mathrm{cdn}} \tau^{\mathrm{cdn}}$, *then* $\Sigma \vdash \bigwedge_{i,j} (\tau_{1i}^{\mathrm{dn}} \vee \tau_{2j}^{\mathrm{dn}}) \leqslant^{\mathrm{cdn}} \tau^{\mathrm{cdn}}$. *In other words, if* $\Sigma \vdash \tau_1^{\mathrm{cdn}} \leqslant^{\mathrm{cdn}} \tau^{\mathrm{cdn}}$ *and* $\Sigma \vdash \tau_2^{\mathrm{cdn}} \leqslant^{\mathrm{cdn}} \tau^{\mathrm{cdn}}$, *then* $\Sigma \vdash \mathrm{dis}(\tau_1^{\mathrm{cdn}}, \tau_2^{\mathrm{cdn}}) \leqslant^{\mathrm{cdn}} \tau^{\mathrm{cdn}}$.

**Proof** We have $\tau^{\mathrm{cdn}} = \bigwedge_k \tau_{0k}^{\mathrm{dn}}$ for some $\overline{\tau_{0k}^{\mathrm{dn}}}^{\,k}$. By Lemma 3.1, we have $\overline{\Sigma \vdash \bigwedge_i \tau_{1i}^{\mathrm{dn}} \leqslant^{\mathrm{cdn}} \tau_{0k}^{\mathrm{dn}}}^{\,k}$ and $\overline{\Sigma \vdash \bigwedge_j \tau_{2j}^{\mathrm{dn}} \leqslant^{\mathrm{cdn}} \tau_{0k}^{\mathrm{dn}}}^{\,k}$, which imply $\overline{\Sigma \vdash \bigwedge_{i,j} (\tau_{1i}^{\mathrm{dn}} \vee \tau_{2j}^{\mathrm{dn}}) \leqslant^{\mathrm{cdn}} \tau_{0k}^{\mathrm{dn}}}^{\,k}$ by Lemma A.46, which imply $\Sigma \vdash \bigwedge_{i,j} (\tau_{1i}^{\mathrm{dn}} \vee \tau_{2j}^{\mathrm{dn}}) \leqslant^{\mathrm{cdn}} \bigwedge_k \tau_{0k}^{\mathrm{dn}} = \tau^{\mathrm{cdn}}$ by S-AndOr2$\curlyvee$. ■

### *A.8.1 DCN-normalized type forms*

The contents of section is symmetric to that of Section 4.4.1 and thus wholly unsurprising, which is why we develop it in appendix.

**Definition A.48** (DCN-normalized form)**.** *The syntax of DCN-normalized (disjunction-conjunction-negation) form is presented in Figure 21. We say that a DCN-normalized form* $\tau^{\mathrm{dcn}}$ *is* complement-free *if* $\tau^{\mathrm{dcn}} = \bigvee_i \bigwedge_{j \in 1..n_i} \tau_{ij}^{\mathrm{n}}$, *where* $\forall \overline{j_i \in 1..n_i}^{\,i}. \top \notin \bigvee_i \tau_{ij_i}^{\mathrm{n}}$.

In the proofs below, we sometimes abuse the notations $\tau_1^{\mathrm{cn}} \wedge \tau_2^{\mathrm{cn}}$ and $\tau_1^{\mathrm{dcn}} \vee \tau_2^{\mathrm{dcn}}$ to mean their properly associated versions, i.e., $\mathrm{con}(\tau_1^{\mathrm{cn}}, \tau_2^{\mathrm{cn}})$ and $\mathrm{dis}(\tau_1^{\mathrm{dcn}}, \tau_2^{\mathrm{dcn}})$ in Figure 22 respectively.

**Definition A.49** (DCN-normalized form translation)**.** *The translation from arbitrary types into DCN-normalized types* $\mathrm{DCN}(\cdot)$ *is defined in Figure 22.*

$$
\begin{aligned}
\tau^0 &::= T\ \overline{\tau^+}\ \overline{\tau^-}\ \overline{\tau^0} \mid \alpha \mid \bot \\
\tau^{\mathrm{n}} &::= \tau^0 \mid \neg\tau^0 \\
\tau^{\mathrm{cn}} &::= \tau^{\mathrm{n}} \mid \tau^{\mathrm{n}} \wedge \tau^{\mathrm{cn}} \\
\tau^{\mathrm{dcn}} &::= \tau^{\mathrm{cn}} \mid \tau^{\mathrm{cn}} \vee \tau^{\mathrm{dcn}}
\end{aligned}
$$

Fig. 21. Syntax of DCN-normalized form.

**Definition A.50** (DCN-normalized subtyping context)**.** $\Sigma$ *is DCN-normalized if for all* $H \in \Sigma$*, either one of the following is true:*

1. $H = (\bigwedge_i \tau_i^{\mathrm{n}} \leqslant \bot)$*, where* $\forall\alpha.\ \{\ \alpha,\ \neg\alpha\ \} \cap \{\ \overline{\tau_i^{\mathrm{n}}}^{\,i}\ \} = \varnothing$*;*
2. $H = (\alpha \leqslant \bigvee_i \tau_i^{\mathrm{n}})$*, where the following are true:*
   - $\{\ \alpha,\ \neg\alpha\ \} \cap \{\ \overline{\tau_i^{\mathrm{n}}}^{\,i}\ \} = \varnothing$*;*
   - $\forall\beta \in \{\ \overline{\tau_i^{\mathrm{n}}}^{\,i}\ \}.\ \neg\beta \notin \{\ \overline{\tau_i^{\mathrm{n}}}^{\,i}\ \}$*;*
   - $\forall\beta \in \{\ \overline{\tau_i^{\mathrm{n}}}^{\,i}\ \}.\ \exists(\bigwedge_j \pi_j^{\mathrm{n}} \leqslant \beta) \in \Sigma.\ \{\ \overline{\pi_j^{\mathrm{n}}}^{\,j}\ \} = \{\ \overline{\mathrm{neg}(\tau_i^{\mathrm{n}})}^{\,i \mid \tau_i^{\mathrm{n}} \neq \beta},\ \alpha\ \}$*;*
   - $\forall\neg\beta \in \{\ \overline{\tau_i^{\mathrm{n}}}^{\,i}\ \}.\ \exists(\beta \leqslant \bigvee_j \pi_j^{\mathrm{n}}) \in \Sigma.\ \{\ \overline{\pi_j^{\mathrm{n}}}^{\,j}\ \} = \{\ \overline{\tau_i^{\mathrm{n}}}^{\,i \mid \tau_i^{\mathrm{n}} \neq \neg\beta},\ \neg\alpha\ \}$*;*
3. $H = (\bigwedge_i \tau_i^{\mathrm{n}} \leqslant \alpha)$*, where the following are true:*
   - $\{\ \alpha,\ \neg\alpha\ \} \cap \{\ \overline{\tau_i^{\mathrm{n}}}^{\,i}\ \} = \varnothing$*;*
   - $\forall\beta \in \{\ \overline{\tau_i^{\mathrm{n}}}^{\,i}\ \}.\ \neg\beta \notin \{\ \overline{\tau_i^{\mathrm{n}}}^{\,i}\ \}$*;*
   - $\forall\beta \in \{\ \overline{\tau_i^{\mathrm{n}}}^{\,i}\ \}.\ \exists(\beta \leqslant \bigvee_j \pi_j^{\mathrm{n}}) \in \Sigma.\ \{\ \overline{\pi_j^{\mathrm{n}}}^{\,j}\ \} = \{\ \overline{\mathrm{neg}(\tau_i^{\mathrm{n}})}^{\,i \mid \tau_i^{\mathrm{n}} \neq \beta},\ \alpha\ \}$*;*
   - $\forall\neg\beta \in \{\ \overline{\tau_i^{\mathrm{n}}}^{\,i}\ \}.\ \exists(\bigwedge_j \pi_j^{\mathrm{n}} \leqslant \beta) \in \Sigma.\ \{\ \overline{\pi_j^{\mathrm{n}}}^{\,j}\ \} = \{\ \overline{\tau_i^{\mathrm{n}}}^{\,i \mid \tau_i^{\mathrm{n}} \neq \neg\beta},\ \neg\alpha\ \}$*;*

**Lemma A.51.** *For any* $\tau$*,* $\mathrm{dcn}(\tau) \cong \tau$*.*

**Proof** By straightforward induction. ∎

**Definition A.52** (DCN-normalized subtyping context translation)**.** *The translation from arbitrary subtyping contexts into DCN-normalized subtyping contexts* $\mathrm{dcn}(\cdot)$ *is defined in Figure 23.*

**Lemma A.53.** *For any* $\Sigma$*, we have* $\Sigma \vDash \mathrm{dcn}(\Sigma)$ *and* $\mathrm{dcn}(\Sigma) \vDash \Sigma$*.*

**Proof** Straightforward, notably making use of Theorem A.9 and Lemma A.51. ∎

### *A.8.2 DCN-normalized derivations*

For each rule in $\mathcal{R}$ with conclusion $\Sigma \vdash \tau \leqslant \pi$, we assume without loss of generality that $\mathrm{dnc}(\tau \wedge \neg\pi) = \tau^{\mathrm{cn}}$ for some $\pi^{\mathrm{cn}}$, since we can otherwise split the rule into multiple simpler rules while keeping the original rule admissible.

$$\boxed{\mathrm{dcn}(\tau)} : \tau^{\mathrm{dcn}}$$
$$\mathrm{dcn}(\tau^0) = \tau^0$$
$$\mathrm{dcn}(\top) = \neg\bot$$
$$\mathrm{dcn}(\neg\tau) = \mathrm{neg}(\mathrm{dcn}(\tau))$$
$$\mathrm{dcn}(\tau_1 \wedge \tau_2) = \mathrm{con}(\mathrm{dcn}(\tau_1),\ \mathrm{dcn}(\tau_2))$$
$$\mathrm{dcn}(\tau_1 \vee \tau_2) = \mathrm{dis}(\mathrm{dcn}(\tau_1),\ \mathrm{dcn}(\tau_2))$$

$$\boxed{\mathrm{neg}(\tau^{\mathrm{dcn}})} : \tau^{\mathrm{dcn}}$$
$$\mathrm{neg}(\tau^0) = \neg\tau^0$$
$$\mathrm{neg}(\neg\tau^0) = \tau^0$$
$$\mathrm{neg}(\tau_1^{\mathrm{n}} \wedge \tau_2^{\mathrm{cn}}) = \mathrm{dis}(\mathrm{neg}(\tau_1^{\mathrm{n}}),\ \mathrm{neg}(\tau_2^{\mathrm{cn}}))$$
$$\mathrm{neg}(\tau_1^{\mathrm{cn}} \vee \tau_2^{\mathrm{dcn}}) = \mathrm{con}(\mathrm{neg}(\tau_1^{\mathrm{cn}}),\ \mathrm{neg}(\tau_2^{\mathrm{dcn}}))$$

$$\boxed{\mathrm{con}(\tau^{\mathrm{dcn}},\ \tau^{\mathrm{dcn}})} : \tau^{\mathrm{dcn}}$$
$$\mathrm{con}(\tau_{11}^{\mathrm{cn}} \vee \tau_{12}^{\mathrm{dcn}},\ \tau_2^{\mathrm{dcn}}) = \mathrm{dis}(\mathrm{con}(\tau_{11}^{\mathrm{cn}},\ \tau_2^{\mathrm{dcn}}),\ \mathrm{con}(\tau_{12}^{\mathrm{dcn}},\ \tau_2^{\mathrm{dcn}}))$$
$$\mathrm{con}(\tau_{11}^{\mathrm{n}} \wedge \tau_{12}^{\mathrm{cn}},\ \tau_2^{\mathrm{dcn}}) = \mathrm{con}(\tau_{11}^{\mathrm{n}},\ \mathrm{con}(\tau_{12}^{\mathrm{cn}},\ \tau_2^{\mathrm{dcn}}))$$
$$\mathrm{con}(\tau_1^{\mathrm{n}},\ \tau_{21}^{\mathrm{cn}} \vee \tau_{22}^{\mathrm{dcn}}) = \mathrm{dis}(\mathrm{con}(\tau_1^{\mathrm{n}},\ \tau_{21}^{\mathrm{cn}}),\ \mathrm{con}(\tau_1^{\mathrm{n}},\ \tau_{22}^{\mathrm{dcn}}))$$
$$\mathrm{con}(\tau_1^{\mathrm{n}},\ \tau_2^{\mathrm{cn}}) = \tau_1^{\mathrm{n}} \wedge \tau_2^{\mathrm{cn}}$$
$$\mathrm{Con}_{i \in m..n}\, \tau_i^{\mathrm{dcn}} = \mathrm{con}(\tau_m^{\mathrm{dcn}},\ \mathrm{Con}_{i \in m+1..n}\, \tau_i^{\mathrm{dcn}})$$
$$\mathrm{Con}_{i \in n..n}\, \tau_i^{\mathrm{dcn}} = \tau_n^{\mathrm{dcn}}$$

$$\boxed{\mathrm{dis}(\tau^{\mathrm{dcn}},\ \tau^{\mathrm{dcn}})} : \tau^{\mathrm{dcn}}$$
$$\mathrm{dis}(\tau_{11}^{\mathrm{cn}} \vee \tau_{12}^{\mathrm{dcn}},\ \tau_2^{\mathrm{dcn}}) = \mathrm{dis}(\tau_{11}^{\mathrm{cn}},\ \mathrm{dis}(\tau_{12}^{\mathrm{dcn}},\ \tau_2^{\mathrm{dcn}}))$$
$$\mathrm{dis}(\tau_1^{\mathrm{cn}},\ \tau_2^{\mathrm{dcn}}) = \tau_1^{\mathrm{cn}} \vee \tau_2^{\mathrm{dcn}}$$
$$\mathrm{Dis}_{i \in m..n}\, \tau_i^{\mathrm{dcn}} = \mathrm{dis}(\tau_m^{\mathrm{dcn}},\ \mathrm{Dis}_{i \in m+1..n}\, \tau_i^{\mathrm{dcn}})$$
$$\mathrm{Dis}_{i \in n..n}\, \tau_i^{\mathrm{dcn}} = \tau_n^{\mathrm{dcn}}$$

Fig. 22. DCN-normalized form translation

**Definition A.54** (DCN-normalized derivations)**.** *The DCN-normalized subtyping relation* $\leqslant^{\mathrm{dcn}}$ *is defined in Figure 24. The following are the difference compared to the full subtyping relation* $\leqslant$ *in Figure 16:*

- *On the top level, the relation is restricted to* $\Sigma \vdash \tau^{\mathrm{dcn}} \leqslant \tau^{\mathrm{dcn}}$.
- *On the top level, all occurrences of* $\top$ *are replaced with* $\neg\bot$.
- *The rule S-*Distrib$\diamond$ *is replaced by S-*DistribCdn$\diamond$*, which requires an application of S-*Distrib$\diamond$ *to be followed immediately by an application of S-*AndOr2$\supset$ *in a transitivity chain by merging the two rules into one.*

$$\boxed{\mathrm{dcn}(\Sigma)} : \Sigma$$

$$\mathrm{dcn}(\Sigma) = \overline{\mathrm{dcn}(\mathrm{dcn}(\tau \wedge \neg\pi) \leqslant \bot)}^{(\tau \leqslant \pi) \in \Sigma} \cdot \overline{\triangleright H}^{\triangleright H \in \Sigma}$$

$$\boxed{\mathrm{dcn}(\tau^{\mathrm{dcn}} \leqslant \bot)} : \Sigma$$

$$\mathrm{dcn}(\textstyle\bigvee_i \bigwedge_{j_i} \tau^{\mathrm{n}}_{ij_i} \leqslant \bot) = \overline{\mathrm{dcn}(\bigwedge_{j_i} \tau^{\mathrm{n}}_{ij_i} \leqslant \bot)}^{i}$$

$$\mathrm{dcn}(\textstyle\bigwedge_i \tau^{\mathrm{n}}_i \leqslant \bot) = \begin{cases} \epsilon & \text{if } \exists\alpha.\, \{\, \alpha,\ \neg\alpha \,\} \subseteq \{\, \overline{\tau^{\mathrm{n}}_i}^{\,i} \,\} \\ \overline{(\alpha \leqslant \bigvee_{i \mid \tau^{\mathrm{n}}_i \neq \alpha} \mathrm{neg}(\tau^{\mathrm{n}}_i))}^{\alpha \in \{\, \overline{\tau^{\mathrm{n}}_i}^{\,i} \,\}} \cdot \overline{(\bigwedge_{i \mid \tau^{\mathrm{n}}_i \neq \neg\alpha} \tau^{\mathrm{n}}_i \leqslant \alpha)}^{\alpha \mid \neg\alpha \in \{\, \overline{\tau^{\mathrm{n}}_i}^{\,i} \,\}} & \\ \qquad \text{if } (\exists\alpha.\, \{\, \alpha,\ \neg\alpha \,\} \cap \{\, \overline{\tau^{\mathrm{n}}_i}^{\,i} \,\} \neq \varnothing) \text{ and } (\forall\alpha \in \{\, \overline{\tau^{\mathrm{n}}_i}^{\,i} \,\}.\, \neg\alpha \notin \{\, \overline{\tau^{\mathrm{n}}_i}^{\,i} \,\}) & \\ (\bigwedge_i \tau^{\mathrm{n}}_i \leqslant \bot) \quad \text{if } \forall\alpha.\, \{\, \alpha,\ \neg\alpha \,\} \cap \{\, \overline{\tau^{\mathrm{n}}_i}^{\,i} \,\} = \varnothing \end{cases}$$

Fig. 23. DCN-normalized subtyping context translation

- *For each rule in $\mathcal{R}$ with conclusion $\Sigma \vdash \tau \leqslant \pi$ and premises $\Sigma' \vdash \tau' \leqslant \pi'$, we transform them into the equivalent DCN-normalized derivation rule in $\mathcal{R}^{\mathrm{dcn}}$ by performing the following:*
  - *Transform the conclusion into $\Sigma \vdash \mathrm{dcn}(\tau) \leqslant^{\mathrm{dcn}} \mathrm{dcn}(\pi)$*
  - *If $\max(\mathit{depth}(\tau'), \mathit{depth}(\pi')) < \max(\mathit{depth}(\tau), \mathit{depth}(\pi))$, keep the premises as is*
  - *If $\max(\mathit{depth}(\tau'), \mathit{depth}(\pi')) = \max(\mathit{depth}(\tau), \mathit{depth}(\pi))$, then transform the premises into $\Sigma' \vdash \mathrm{dcn}(\tau') \leqslant^{\mathrm{dcn}} \mathrm{dcn}(\pi')$*

*Notice that S-TDEPTH is treated the same way as rules in $\mathcal{R}$, so its premises still refer to the full $\leqslant$ relation, even though its conclusion is about the $\leqslant^{\mathrm{dcn}}$ relation.*

*The DCN-normalized boolean subtyping relation $\subseteq^{\mathrm{dcn}}$ is defined similarly.*

Notice that Lemma A.7 and Lemma 3.1 extend to DCN-normalized derivations. In the proofs below, we also make use of extended versions of commutativity ($\tau_1 \vee^{\diamond} \tau_2 (\vee^{\diamond} \tau_3) \leqslant^{\mathrm{dcn}} \tau_2 \vee^{\diamond} \tau_1 (\vee^{\diamond} \tau_3)$) and idempotence ($\tau_1 \vee^{\diamond} \tau_1 (\vee^{\diamond} \tau_2) \leqslant^{\mathrm{dcn}} \tau_1 (\vee^{\diamond} \tau_2)$).

**Lemma A.55.** *$\Sigma \vdash \tau_1^{\mathrm{dcn}} \leqslant \tau_2^{\mathrm{dcn}}$ if $\Sigma \vdash \tau_1^{\mathrm{dcn}} \leqslant^{\mathrm{dcn}} \tau_2^{\mathrm{dcn}}$. Similarly, $\tau_1^{\mathrm{dcn}} \subseteq \tau_2^{\mathrm{dcn}}$ if $\tau_1^{\mathrm{dcn}} \subseteq^{\mathrm{dcn}} \tau_2^{\mathrm{dcn}}$.*

**Proof** It is easy to see that every rule of $\leqslant^{\mathrm{dcn}}$ is admissible in $\leqslant$. ∎

**Lemma A.56.** *If $\Sigma \vdash \tau \leqslant \pi$, then $\mathrm{dcn}(\Sigma) \vdash \mathrm{dcn}(\tau) \leqslant^{\mathrm{dcn}} \mathrm{dcn}(\pi)$. Similarly, if $\tau \subseteq \pi$, then $\mathrm{dcn}(\tau) \subseteq^{\mathrm{dcn}} \mathrm{dcn}(\pi)$.*

**Proof** Symmetric to Lemma 4.21. ∎

### *A.8.3 Some useful lemmas*

**Lemma A.57.**

$$\boxed{\Sigma \vdash \tau^{\mathrm{dcn}} \leqslant^{\mathrm{dcn}} \tau^{\mathrm{dcn}}} \qquad \boxed{\tau^{\mathrm{dcn}} \leqslant^{\mathrm{dcn}} \tau^{\mathrm{dcn}}}$$

$$\triangleleft \Xi = \Xi \qquad \triangleleft(\Sigma \cdot H) = \triangleleft\Sigma \cdot H \qquad \triangleleft(\Sigma \cdot \triangleright H) = \triangleleft\Sigma \cdot H$$

S-Refl
$$\frac{}{\tau^{\mathrm{dcn}} \leqslant^{\mathrm{dcn}} \tau^{\mathrm{dcn}}}$$

S-ToB$\cdot$
$$\frac{}{\tau^{\mathrm{dcn}} \leqslant^{\mathrm{dcn}} \neg\bot}$$

S-ToB$\rotatebox{180}{$\cdot$}$
$$\frac{}{\bot \leqslant^{\mathrm{dcn}} \tau^{\mathrm{dcn}}}$$

S-Compl$\cdot$
$$\frac{}{\neg\bot \leqslant^{\mathrm{dcn}} \tau^0 \vee \neg\tau^0}$$

S-Compl$\rotatebox{180}{$\cdot$}$
$$\frac{}{\tau^0 \wedge \neg\tau^0 \leqslant^{\mathrm{dcn}} \bot}$$

S-AndOr1$\cdot$
$$\frac{S \subseteq \{\, \overline{i} \,\}}{\bigvee_{i' \in S} \tau^{\mathrm{cn}}_{i'} \leqslant^{\mathrm{dcn}} \bigvee_i \tau^{\mathrm{cn}}_i}$$

S-AndOr1$\rotatebox{180}{$\cdot$}$
$$\frac{S \subseteq \{\, \overline{i} \,\}}{\bigwedge_i \tau^{\mathrm{n}}_i \leqslant^{\mathrm{dcn}} \bigwedge_{i' \in S} \tau^{\mathrm{n}}_{i'}}$$

S-AndOr2$\cdot$
$$\frac{\dfrac{}{\Sigma \vdash \tau^{\mathrm{cd}}_i \leqslant^{\mathrm{dcn}} \tau^{\mathrm{dcn}}}{}^{i}}{\Sigma \vdash \bigvee_i \tau^{\mathrm{cd}}_i \leqslant^{\mathrm{dcn}} \tau^{\mathrm{dcn}}}$$

S-AndOr2$\rotatebox{180}{$\cdot$}$
$$\frac{\dfrac{}{\Sigma \vdash \tau^{\mathrm{dcn}} \leqslant^{\mathrm{dcn}} \tau^{\mathrm{n}}_i}{}^{i}}{\Sigma \vdash \tau^{\mathrm{dcn}} \leqslant^{\mathrm{dcn}} \bigwedge_i \tau^{\mathrm{n}}_i}$$

S-DistribDcn$\cdot$
$$\frac{\Sigma \vdash \pi^{\mathrm{dcn}} \leqslant^{\mathrm{dcn}} \tau^{\mathrm{n}} \quad \Sigma \vdash \pi^{\mathrm{dcn}} \leqslant^{\mathrm{dcn}} \bigvee_i \tau^{\mathrm{cn}}_i}{\Sigma \vdash \pi^{\mathrm{dcn}} \leqslant^{\mathrm{dcn}} \bigvee_i (\tau^{\mathrm{n}} \wedge \tau^{\mathrm{cn}}_i)}$$

S-DistribDcn$\rotatebox{180}{$\cdot$}$
$$\frac{\dfrac{}{\Sigma \vdash \pi^{\mathrm{dcn}} \leqslant^{\mathrm{dcn}} \tau^{\mathrm{n}}_i \vee \tau^{\mathrm{dcn}}}{}^{i}}{\Sigma \vdash \pi^{\mathrm{dcn}} \leqslant^{\mathrm{dcn}} (\bigwedge_i \tau^{\mathrm{n}}_i) \vee \tau^{\mathrm{dcn}}}$$

S-Trans
$$\frac{\Sigma \vdash \tau^{\mathrm{dcn}}_0 \leqslant^{\mathrm{dcn}} \tau^{\mathrm{dcn}}_1 \quad \Sigma \vdash \tau^{\mathrm{dcn}}_1 \leqslant^{\mathrm{dcn}} \tau^{\mathrm{dcn}}_2}{\Sigma \vdash \tau^{\mathrm{dcn}}_0 \leqslant^{\mathrm{dcn}} \tau^{\mathrm{dcn}}_2}$$

S-Weaken
$$\frac{H}{\Sigma \vdash H}$$

S-Assum
$$\frac{\Sigma \cdot \triangleright H \vdash H}{\Sigma \vdash H}$$

S-Hyp
$$\frac{H \in \Sigma}{\Sigma \vdash H}$$

S-TMrg$\diamond$
$$\frac{}{T\ \overline{(\tau^+_i \vee^{\diamond} \pi^+_i)}^{i}\ \overline{(\tau^-_j \wedge^{\diamond} \pi^-_j)}^{j}\ \overline{\tau^0_k}^{k} \leqslant^{\diamond\mathrm{dcn}} T\ \overline{\tau^+_i}^{i}\ \overline{\tau^-_j}^{j}\ \overline{\tau^0_k}^{k}\ \vee^{\diamond} T\ \overline{\pi^+_i}^{i}\ \overline{\pi^-_j}^{j}\ \overline{\tau^0_k}^{k}}$$

S-TDepth
$$\frac{\dfrac{}{\triangleleft\Sigma \vdash \tau^+_i \leqslant \pi^+_i}^{i} \quad \dfrac{}{\triangleleft\Sigma \vdash \pi^-_j \leqslant \tau^-_j}^{j} \quad \dfrac{}{\triangleleft\Sigma \vdash \tau^0_k \equiv \pi^0_k}^{k}}{\Sigma \vdash T\ \overline{\tau^+_i}^{i}\ \overline{\tau^-_j}^{j}\ \overline{\tau^0_k}^{k} \leqslant^{\mathrm{dcn}} T\ \overline{\pi^+_i}^{i}\ \overline{\pi^-_j}^{j}\ \overline{\pi^0_k}^{k}} \qquad \mathcal{R}^{\mathrm{dcn}}$$

Fig. 24. DCN-normalized subtyping rules for $\mathfrak{S}(\mathcal{T}, \mathcal{R})$.

*(A) For $\tau \in \{\, \top, T\ \overline{\tau^+}\ \overline{\tau^-}\ \overline{\tau^0} \,\}$ and $\bigwedge_i \pi^{\mathrm{dn}}_i$ in complement-free CDN-normalized form, if $\bigwedge_i \pi^{\mathrm{dn}}_i \subseteq \tau$ with a derivation of size n, then $\pi^{\mathrm{dn}}_k \subseteq \tau$ for some k with a derivation of size n.*

*(B) For $\tau \in \{\, \bot, T\ \overline{\tau^+}\ \overline{\tau^-}\ \overline{\tau^0} \,\}$ and $\bigvee_i \pi^{\mathrm{cn}}_i$ in complement-free DCN-normalized form, if $\tau \subseteq \bigvee_i \pi^{\mathrm{cn}}_i$ with a derivation of size n, then either $\tau \subseteq \pi^{\mathrm{cn}}_k$ for some k with a derivation of size n.*

Only the proof for (A) is shown below. The proof for (B) is symmetric.

**Proof** By induction on right-leaning $\subseteq$ derivations.

**Case S-Refl.** Immediate.

**Case S-ToB$\cdot$.** Then $\tau = \top$ and we have $\dfrac{}{\pi_i^{\mathrm{dn}} \subseteq \top}\,i$ by S-ToB$\cdot$, with a derivation of size 1.

**Case S-ToB$\supset$.** Then $\bigwedge_i \pi_i^{\mathrm{dn}} = \pi_1^{\mathrm{dn}} = \bot$. The result is immediate.

**Case S-Compl$\cdot$.** Impossible since $\tau$ is not a union.

**Case S-Compl$\supset$.** Impossible since $\tau \neq \bot$.

**Case S-AndOr11$\cdot$.** Impossible since $\tau$ is not a union.

**Case S-AndOr11$\supset$.** Then $\pi_1^{\mathrm{dn}} = \tau$ and we have $\pi_1^{\mathrm{dn}} \subseteq \tau$ by S-Refl, with a derivation of size 1.

**Cases S-AndOr12$\cdot$.** Impossible since $\tau$ is not a union.

**Cases S-AndOr12$\supset$.** Then $\bigwedge_{i>1} \pi_i^{\mathrm{dn}} = \pi_2^{\mathrm{dn}} = \tau$ and we have $\pi_2^{\mathrm{dn}} \subseteq \tau$ by S-Refl, with a derivation of size 1.

**Case S-AndOr2$\cdot$.** Then $\bigwedge_i \pi_i^{\mathrm{dn}} = \pi_1^{\mathrm{dn}} = \pi_{11}^{\mathrm{n}} \vee \pi_{12}^{\mathrm{dn}}$ for some $\pi_{11}^{\mathrm{n}}$ and $\pi_{12}^{\mathrm{dn}}$. The result is immediate.

**Case S-AndOr2$\supset$.** Impossible since $\tau$ is not an intersection.

**Case S-Trans.** Then the premises are $\bigwedge_i \pi_i^{\mathrm{dn}} \subseteq \tau'$ and $\tau' \subseteq \tau$ for some $\tau'$, both with a derivation of size $n-1$. By induction on the size of the subderivation for the former premise, denoted by $m$. Denote the inner induction hypothesis as IH$'$.

- **Cases (S-Refl, $*$), ($*$, S-Refl).** By IH on the other premise.
- **Cases (S-ToB$\cdot$, $*$).** Then $\tau' = \top$. By S-ToB$\cdot$, we have $\dfrac{}{\pi_i^{\mathrm{dn}} \subseteq \top}\,i$. By S-Trans with $\top \subseteq \tau$, we have $\dfrac{}{\pi_i^{\mathrm{dn}} \subseteq \tau}\,i$ with a derivation of size $n$.
- **Cases (S-ToB$\supset$, $*$).** Then $\bigwedge_i \pi_i^{\mathrm{dn}} = \pi_1^{\mathrm{dn}} = \bot$. The result is immediate.
- **Cases (S-Compl$\cdot$, $*$).** Then $\bigwedge_i \pi_i^{\mathrm{dn}} = \pi_1^{\mathrm{dn}} = \top$. The result is immediate.
- **Cases (S-Compl$\supset$, $*$).** Impossible since $\bigwedge_i \pi_i^{\mathrm{dn}}$ is a complement-free CDN-normalized form.
- **Cases (S-AndOr11$\cdot$, $*$).** Then $\tau' = \bigwedge_i \pi_i^{\mathrm{dn}} \vee \tau_1'$ for some $\tau_1'$. By Lemma A.44 on the latter premise, we have $\bigwedge_i \pi_i^{\mathrm{dn}} \subseteq \tau$ with a derivation of size $n-1$. The result then follows from IH.
- **Cases (S-AndOr11$\supset$, $*$).** Then $\tau' = \pi_1^{\mathrm{dn}}$. The result is immediate from the latter premise.
- **Cases (S-AndOr12$\cdot$, $*$).** Then $\tau' = \tau_1' \vee \bigwedge_i \pi_i^{\mathrm{dn}}$ for some $\tau_1'$. By Lemma A.44 on the latter premise, we have $\bigwedge_i \pi_i^{\mathrm{dn}} \subseteq \tau$ with a derivation of size $n-1$. The result then follows from IH.
- **Cases (S-AndOr12$\supset$, $*$).** Then $\tau' = \bigwedge_{i>1} \pi_i^{\mathrm{dn}}$. By IH on the latter rule, we have $\pi_k^{\mathrm{dn}} \subseteq \tau$ for some $k > 1$.
- **Cases (S-AndOr2$\cdot$, $*$).** Then $\bigwedge_i \pi_i^{\mathrm{dn}} = \pi_1^{\mathrm{dn}} = \pi_{11}^{\mathrm{n}} \vee \pi_{12}^{\mathrm{dn}}$ for some $\pi_{11}^{\mathrm{n}}$ and $\pi_{12}^{\mathrm{dn}}$. The result is immediate.
- **Cases (S-AndOr2$\supset$, $*$).** Then $\tau' = \tau_1' \wedge \tau_2'$ for some $\tau_1'$ and $\tau_2'$. Since $\tau$ is not an intersection, it is easy to see that the intersection must be consumed by an application of S-AndOr11$\supset$, S-AndOr12$\supset$, or S-Distrib$\diamond$ in the transitivity chain. Then it is possible to rewrite the derivation into a smaller one by dropping the application of S-AndOr2$\supset$. The result then follows from IH.
- **Cases (S-Distrib$\cdot$, $*$).** Then $\pi_2^{\mathrm{dn}} = \pi_{21}^{\mathrm{n}} \vee \pi_{22}^{\mathrm{dn}}$ and $\tau' = (\pi_1^{\mathrm{dn}} \wedge \pi_{21}^{\mathrm{n}}) \vee (\pi_1^{\mathrm{dn}} \wedge \pi_{22}^{\mathrm{dn}})$ for some $\pi_{21}^{\mathrm{n}}$ and $\pi_{22}^{\mathrm{dn}}$. By Lemma A.44 on the latter rule, we have $\pi_1^{\mathrm{dn}} \wedge \pi_{21}^{\mathrm{n}} \subseteq \tau$ and $\pi_1^{\mathrm{dn}} \wedge \pi_{22}^{\mathrm{dn}} \subseteq \tau$, both with a derivation of size $n-1$. By IH on $\pi_1^{\mathrm{dn}} \wedge \pi_{21}^{\mathrm{n}} \subseteq \tau$, we have $\pi_1^{\mathrm{dn}} \subseteq \tau$ or $\pi_{21}^{\mathrm{n}} \subseteq \tau$, both with a derivation of size $n-1$. By IH

on $\pi_1^{\mathrm{dn}} \wedge \pi_{22}^{\mathrm{dn}} \subseteq \tau$, we have $\pi_1^{\mathrm{dn}} \subseteq \tau$ or $\pi_{22}^{\mathrm{dn}} \subseteq \tau$, both with a derivation of size $n-1$. If $\pi_1^{\mathrm{dn}} \subseteq \tau$, then we have the result immediately. Otherwise, we have $\pi_{21}^{\mathrm{n}} \subseteq \tau$ and $\pi_{22}^{\mathrm{dn}} \subseteq \tau$, which imply $\pi_2^{\mathrm{dn}} = \pi_{21}^{\mathrm{n}} \vee \pi_{22}^{\mathrm{dn}} \subseteq \tau$ by S-AndOr2·, with a derivation of size $n$.

**Cases (S-Distrib⤴, ∗).** Then $\pi_1^{\mathrm{dn}} = \pi_0^{\mathrm{n}} \vee \pi_{12}^{\mathrm{dn}}$ and $\pi_2^{\mathrm{dn}} = \pi_0^{\mathrm{n}} \vee \pi_{22}^{\mathrm{dn}}$ for some $\pi_0^{\mathrm{n}}$ and $\pi_{12}^{\mathrm{dn}}$ and $\pi_{22}^{\mathrm{dn}}$, and $\tau' = \pi_0^{\mathrm{n}} \vee (\tau_1' \wedge \tau_2')$. By Lemma A.44 on the latter rule, we have $\pi_{12}^{\mathrm{dn}} \wedge \pi_{22}^{\mathrm{dn}} \subseteq \tau$ and $\pi_0^{\mathrm{n}} \subseteq \tau$, both with a derivation of size $n-1$. By IH, we have $\pi_{12}^{\mathrm{dn}} \subseteq \tau$ or $\pi_{22}^{\mathrm{dn}} \subseteq \tau$, which implies $\pi_1^{\mathrm{dn}} = \pi_0^{\mathrm{n}} \vee \pi_{12}^{\mathrm{dn}} \subseteq \tau$ or $\pi_2^{\mathrm{dn}} = \pi_0^{\mathrm{n}} \vee \pi_{22}^{\mathrm{dn}} \subseteq \tau$ with a derivation of size $n$ by S-AndOr2· with $\pi_0^{\mathrm{n}} \subseteq \tau$.

■

**Lemma A.58.**

*(A) For* $\tau \in \{\, \top, T\ \overline{\tau^+}\ \overline{\tau^-}\ \overline{\tau^0} \,\}$*, if* $\pi_1^{\mathrm{n}} \wedge \pi_2^{\mathrm{cn}} \subseteq \tau$*, then either* $\pi_1^{\mathrm{n}} \subseteq \tau$ *or* $\pi_2^{\mathrm{cn}} \subseteq \tau$ *or* $\pi_1^{\mathrm{n}} \wedge \pi_2^{\mathrm{cn}} \subseteq \bot$.

*(B) For* $\tau \in \{\, \bot, T\ \overline{\tau^+}\ \overline{\tau^-}\ \overline{\tau^0} \,\}$*, if* $\tau \subseteq \pi_1^{\mathrm{n}} \vee \pi_2^{\mathrm{dn}}$*, then either* $\tau \subseteq \pi_1^{\mathrm{n}}$ *or* $\tau \subseteq \pi_2^{\mathrm{dn}}$ *or* $\top \subseteq \pi_1^{\mathrm{n}} \vee \pi_2^{\mathrm{dn}}$.

Only the proof for (A) is shown below. The proof for (B) is symmetric.

**Proof** By induction on right-leaning $\subseteq^{\mathrm{dcn}}$ derivations for the following statements, where S-AndOr2· does not occur as the first premise of S-Trans in any of the judgements (in both the assumptions and conclusions). It is easy to see that we can rewrite any subderivations with S-AndOr2· as the first premise of S-Trans into an equivalent one by applying S-Trans to the premises of S-AndOr2· and the second premise of S-Trans, followed by an application of S-AndOr2·.

1. For $\tau \in \{\, \neg\bot, T\ \overline{\tau^+}\ \overline{\tau^-}\ \overline{\tau^0} \,\}$, if $\pi_1^{\mathrm{n}} \wedge \pi_2^{\mathrm{cn}} \subseteq \tau$ with a derivation of size $n$, then either $\pi_1^{\mathrm{n}} \subseteq \tau$ or $\pi_2^{\mathrm{cn}} \subseteq \tau$ with a derivation of size $n$, or $\pi_1^{\mathrm{n}} \wedge \pi_2^{\mathrm{cn}} \subseteq \bot$.
2. For $\tau_c \in \{\, T\ \overline{\tau^+}\ \overline{\tau^-}\ \overline{\tau^0} \,\}$, if $\bigvee_i \pi_i^{\mathrm{cn}} \subseteq \tau_c$ with a derivation of size $n$, then $\overline{\pi_i^{\mathrm{cn}} \subseteq \tau_c}^{\,i}$, all with a derivation of size $n-1$.

In the remainder of this proof, we abbreviate $\subseteq^{\mathrm{dcn}}$ as $\subseteq$.

**Case S-Refl.** Impossible

**Case S-ToB·.**

1. Then $\tau = \neg\bot$ and we have both $\pi_1^{\mathrm{n}} \subseteq \tau$ and $\pi_2^{\mathrm{cn}} \subseteq \tau$ by S-ToB·.
2. Impossible.

**Cases S-ToB⤴, S-Compl◇,S-AndOr1·.** Impossible.

**Case S-AndOr1⤴.**

1. Then $\tau = \pi_k^{\mathrm{n}}$ for some $k$, where $\pi_2^{\mathrm{cn}} = \bigwedge_{i>1} \pi_i^{\mathrm{n}}$ for some $\overline{\pi_i^{\mathrm{n}}}^{\,i>1}$. If $k = 1$, then we have $\pi_1^{\mathrm{n}} \subseteq \tau$ by S-Refl. Otherwise, we have $\pi_2^{\mathrm{cn}} \subseteq \tau$ by S-AndOr1⤴.
2. Impossible.

**Case S-AndOr2·.**

1. Impossible.
2. The premises of the rule are $\overline{\pi_i^{\mathrm{cn}} \subseteq \tau_c}^{\,i}$, all of size $n-1$.

**Cases S-AndOr2$\wr$, S-DistribDcn$\diamond$.** Impossible.

**Case S-Trans.**

1. Then the premises of the rule are $\pi_1^{\mathrm{n}} \wedge \pi_2^{\mathrm{cn}} \subseteq \tau^{\mathrm{dcn}}$ and $\tau^{\mathrm{dcn}} \subseteq \tau$ for some $\tau^{\mathrm{dcn}}$, both of size $n-1$.
2. Then the premises of the rule are $\bigvee_i \pi_i^{\mathrm{cn}} \subseteq \tau^{\mathrm{dcn}}$ and $\tau^{\mathrm{dcn}} \subseteq \tau_c$ for some $\tau^{\mathrm{dcn}}$, both of size $n-1$.

By induction on the size of the former premise of S-Trans, denoted by $m$. Denote the inner induction hypothesis as IH$'$.

**Cases (S-Refl, $*$).** By IH on the latter premise.

**Cases (S-ToB$\cdot$, $*$).**

1. Then $\tau^{\mathrm{dcn}} = \neg\bot$. We have both $\pi_1^{\mathrm{n}} \subseteq \neg\bot$ and $\pi_2^{\mathrm{cn}} \subseteq \neg\bot$ by S-ToB$\cdot$. Then we have both $\pi_1^{\mathrm{n}} \subseteq \tau$ and $\pi_2^{\mathrm{cn}} \subseteq \tau$ by S-Trans with the latter premise, both with a derivation of size $n$.
2. Impossible since $\neg\bot \subseteq \tau_c$ cannot be derived (Lemma A.62).

**Cases (S-ToB$\wr$, $*$), (S-Compl$\cdot$, $*$).** Impossible.

**Cases (S-Compl$\wr$, $*$).**

1. Then $\tau^{\mathrm{dcn}} = \bot$. $\pi_1^{\mathrm{n}} \wedge \pi_2^{\mathrm{cn}} \subseteq \bot$ is immediate from the former premise.
2. Impossible.

**Cases (S-AndOr1$\cdot$, $*$).**

1. Then $\tau^{\mathrm{dcn}} = (\pi_1^{\mathrm{n}} \wedge \pi_2^{\mathrm{cn}}) \vee \tau_2^{\mathrm{dcn}}$ for some $\tau_2^{\mathrm{dcn}}$. If $\tau = \top$, then we have both $\pi_1^{\mathrm{n}} \subseteq \tau$ and $\pi_2^{\mathrm{cn}} \subseteq \tau$ with a derivation of size 1 by S-ToB$\cdot$. Otherwise, the latter premise $(\pi_1^{\mathrm{n}} \wedge \pi_2^{\mathrm{cn}}) \vee \tau_2^{\mathrm{dcn}} \subseteq \tau$ implies $\pi_1^{\mathrm{n}} \wedge \pi_2^{\mathrm{cn}} \subseteq \tau$ with a derivation of size $n-2$ by IH (2). The result then follows from IH (1).
2. Then $\tau^{\mathrm{dcn}} = (\bigvee_i \pi_i^{\mathrm{cn}}) \vee \tau_2^{\mathrm{dcn}}$ for some $\tau_2^{\mathrm{dcn}}$. The latter premise $(\bigvee_i \pi_i^{\mathrm{cn}}) \vee \tau_2^{\mathrm{dcn}} \subseteq \tau_c$ implies $\bigvee_i \pi_i^{\mathrm{cn}} \subseteq \tau_c$ with a derivation of size $n-2$ by IH (2), which implies $\overline{\pi_i^{\mathrm{cn}} \subseteq \tau_c}^{\,i}$, all with a derivation of size $n-3$ by IH (2).

**Cases (S-AndOr1$\wr$, $*$).**

1. Then $\tau^{\mathrm{dcn}} = \bigwedge_{i' \in S} \pi_{i'}^{\mathrm{n}}$ for some $S \subseteq \{\, \bar{i} \,\}$, where $\pi_2^{\mathrm{cn}} = \bigwedge_{i>1} \pi_i^{\mathrm{n}}$ for some $\overline{\pi_i^{\mathrm{n}}}^{\,i>1}$.

   **Case** $1 \in S$**.** By IH (1) on the latter premise, we have either $\pi_1^{\mathrm{n}} \subseteq \tau$ or $\bigwedge_{i' \in S \setminus \{\,1\,\}} \pi_{i'}^{\mathrm{n}} \subseteq \tau$ with a derivation of size $n-1$, or $\bigwedge_{i' \in S} \pi_{i'}^{\mathrm{n}} \subseteq \bot$. If $\pi_1^{\mathrm{n}} \subseteq \tau$, the result is immediate. If $\bigwedge_{i' \in S \setminus \{\,1\,\}} \pi_{i'}^{\mathrm{n}} \subseteq \tau$, then we have $\pi_2^{\mathrm{cn}} \subseteq \tau$ with a derivation of size $n$ by S-Trans with S-AndOr1$\wr$. If $\bigwedge_{i' \in S} \pi_{i'}^{\mathrm{n}} \subseteq \bot$, then we have $\pi_1^{\mathrm{n}} \wedge \pi_2^{\mathrm{cn}} \subseteq \bot$ by S-Trans with S-AndOr1$\wr$.

   **Case** $1 \notin S$**.** Then $\pi_2^{\mathrm{cn}} \subseteq \tau$ follows by IH (1) on the latter premise, followed by S-Trans with S-AndOr1$\wr$, with a derivation of size $n$.

2. Impossible.

**Cases (S-AndOr2$\cdot$, $*$).** Impossible by assumption.

**Cases (S-AndOr2$\supset$, $*$).**

1. Then $\tau^{\mathrm{dcn}} = \bigwedge_j \tau_j^{\mathrm{n}}$ for some $\overline{\tau_j^{\mathrm{n}}}^{j}$. The premises of the former rule are $\overline{\pi_1^{\mathrm{n}} \wedge \pi_2^{\mathrm{cn}} \subseteq \tau_j^{\mathrm{n}}}^{j}$, all of size $m-1$. By repeated applications of IH (1), the latter premise $\bigwedge_j \tau_j^{\mathrm{n}} \subseteq \tau$ implies $\tau_k^{\mathrm{n}} \subseteq \tau$ for some $k$ with a derivation of size $n-1$, or $\bigwedge_j \tau_j^{\mathrm{n}} \subseteq \bot$.

   **Case** $\tau_k^{\mathrm{n}} \subseteq \tau$**.** Then by S-Trans with one of the premises of the former rule, we have $\pi_1^{\mathrm{n}} \wedge \pi_2^{\mathrm{cn}} \subseteq \tau$ with a derivation of size $n$ and a former premise of size $m-1$. The result then follows from IH′ (1).

   **Case** $\bigwedge_j \tau_j^{\mathrm{n}} \subseteq \bot$**.** Then we have $\pi_1^{\mathrm{n}} \wedge \pi_2^{\mathrm{cn}} \subseteq \bot$ by S-Trans with the former premise.

2. Then $\tau^{\mathrm{dcn}} = \bigwedge_j \tau_j^{\mathrm{n}}$ for some $\overline{\tau_j^{\mathrm{n}}}^{j}$. The premises of the former rule are $\overline{\bigvee_i \pi_i^{\mathrm{cn}} \subseteq \tau_j^{\mathrm{n}}}^{j}$, all of size $m-1$. By IH (1), the latter premise $\bigwedge_j \tau_j^{\mathrm{n}} \subseteq \tau_c$ implies $\tau_k^{\mathrm{n}} \subseteq \tau_c$ for some $k$ with a derivation of size $n-1$, or $\bigwedge_j \tau_j^{\mathrm{n}} \subseteq \bot$.

   **Case** $\tau_k^{\mathrm{n}} \subseteq \tau_c$**.** Then by S-Trans with one of the premises of the former rule, we have $\bigvee_i \pi_i^{\mathrm{cn}} \subseteq \tau_c$ with a derivation of size $n$ and a former premise of size $m-1$. The result then follows from IH′ (2).

   **Case** $\bigwedge_j \tau_j^{\mathrm{n}} \subseteq \bot$**.** Then it is easy to see that the transitivity chain in the derivation for one of $\overline{\bigvee_i \pi_i^{\mathrm{cn}} \subseteq \tau_j^{\mathrm{n}}}^{j}$ must pass through $\bot$, i.e., $\bigvee_i \pi_i^{\mathrm{cn}} \subseteq \bot$ can be derived with size $n-2$. Then we have $\bigvee_i \pi_i^{\mathrm{cn}} \subseteq \tau_c$ with a derivation of size $n-1$ by S-Trans with S-ToB$\supset$. The result then follows from IH (2).

**Cases (S-DistribDcn$\cdot$, $*$).**

1. Then $\tau^{\mathrm{dcn}} = \bigvee_j (\tau_0^{\mathrm{n}} \wedge \tau_j^{\mathrm{cn}})$ for some $\tau_0^{\mathrm{n}}$ and $\overline{\tau_j^{\mathrm{cn}}}^{j}$. The premises of the former rule are:

$$\pi_1^{\mathrm{n}} \wedge \pi_2^{\mathrm{cn}} \subseteq \tau_0^{\mathrm{n}} \tag{1}$$

$$\pi_1^{\mathrm{n}} \wedge \pi_2^{\mathrm{cn}} \subseteq \bigvee_j \tau_j^{\mathrm{cn}} \tag{2}$$

   both of size $m-1$. The latter premise is:

$$\bigvee_j (\tau_0^{\mathrm{n}} \wedge \tau_j^{\mathrm{cn}}) \subseteq \tau \tag{3}$$

   By IH (2), (3) implies:

$$\overline{\tau_0^{\mathrm{n}} \wedge \tau_j^{\mathrm{cn}} \subseteq \tau}^{j} \tag{4}$$

   all with derivations of size $n-2$. For each $j$, by IH (1), (4) implies $\tau_0^{\mathrm{n}} \subseteq \tau$ or $\tau_j^{\mathrm{cn}} \subseteq \tau$ with a derivation of size $n-2$, or $\tau_0^{\mathrm{n}} \wedge \tau_j^{\mathrm{cn}} \subseteq \bot$.

   **Case** $\tau_0^{\mathrm{n}} \subseteq \tau$**.** Then by S-Trans with (1), we have:

$$\pi_1^{\mathrm{n}} \wedge \pi_2^{\mathrm{cn}} \subseteq \tau \tag{5}$$

   with a derivation of size $n-1$. The result then follows from IH (1).

   **Case** $\tau_0^{\mathrm{n}} \nsubseteq \tau$**.** Then for each $j$, we have $\tau_j^{\mathrm{cn}} \subseteq \tau$ or $\tau_0^{\mathrm{n}} \wedge \tau_j^{\mathrm{cn}} \subseteq \bot$. Let $S = \{\, j \mid \tau_0^{\mathrm{n}} \wedge \tau_j^{\mathrm{cn}} \subseteq \bot \,\}$. By S-AndOr2$\cdot$, we have

$$\bigvee_{j \notin S} \tau_j^{\mathrm{cn}} \subseteq \tau \tag{6}$$

with a derivation of size $n-1$. From the definiton of $S$, we have:

$$\overline{\tau_0^{\mathrm{n}} \wedge \tau_j^{\mathrm{cn}} \subseteq \bot}^{\,j \in S} \tag{7}$$

By Theorem A.9, (7) implies:

$$\overline{\tau_j^{\mathrm{cn}} \subseteq \tau_{0'}^{\mathrm{n}}}^{\,j \in S} \tag{8}$$

where $\tau_{0'}^{\mathrm{n}} = \mathrm{neg}(\tau_0^{\mathrm{n}})$. By Lemma A.7· on (8) and S-Refl, we have:

$$\bigvee_j \tau_j^{\mathrm{cn}} \subseteq \bigvee_{j \notin S} \tau_j^{\mathrm{cn}} \vee \tau_{0'}^{\mathrm{n}} \tag{9}$$

Then by S-Trans on (2) and (9), we have:

$$\pi_1^{\mathrm{n}} \wedge \pi_2^{\mathrm{cn}} \subseteq \bigvee_{j \notin S} \tau_j^{\mathrm{cn}} \vee \tau_{0'}^{\mathrm{n}} \tag{10}$$

By Theorem A.9, (10) implies:

$$\tau_0^{\mathrm{n}} \wedge \pi_1^{\mathrm{n}} \wedge \pi_2^{\mathrm{cn}} \subseteq \bigvee_{j \notin S} \tau_j^{\mathrm{cn}} \tag{11}$$

By S-Trans with S-AndOr2 on (1) and S-Refl, (11) implies:

$$\pi_1^{\mathrm{n}} \wedge \pi_2^{\mathrm{cn}} \subseteq \bigvee_{j \notin S} \tau_j^{\mathrm{cn}} \tag{12}$$

Since we have (2) with a derivation of size $m-1$ and (12), it is easy to see that (12) can be derived with size $m-1$. Then by S-Trans with (6), we have:

$$\pi_1^{\mathrm{n}} \wedge \pi_2^{\mathrm{cn}} \subseteq \tau \tag{13}$$

with a derivation of size $n$ and a former premise of size $m-1$. The result then follows from IH′ (1).

2. Then $\tau^{\mathrm{dcn}} = \bigvee_j (\tau_0^{\mathrm{n}} \wedge \tau_j^{\mathrm{cn}})$ for some $\tau_0^{\mathrm{n}}$ and $\overline{\tau_j^{\mathrm{cn}}}^{\,j}$. The premises of the former rule are:

$$\bigvee_i \pi_i^{\mathrm{cn}} \subseteq \tau_0^{\mathrm{n}} \tag{14}$$

$$\bigvee_i \pi_i^{\mathrm{cn}} \subseteq \bigvee_j \tau_j^{\mathrm{cn}} \tag{15}$$

both with a derivation of size $m-1$. The latter premise is:

$$\bigvee_j (\tau_0^{\mathrm{n}} \wedge \tau_j^{\mathrm{cn}}) \subseteq \tau_c \tag{16}$$

By IH (2), (16) implies:

$$\overline{\tau_0^{\mathrm{n}} \wedge \tau_j^{\mathrm{cn}} \subseteq \tau_c}^{\,j} \tag{17}$$

all with a derivation of size $n-2$. For each $j$, by IH (1), (17) implies $\tau_0^{\mathrm{n}} \subseteq \tau_c$ or $\tau_j^{\mathrm{cn}} \subseteq \tau_c$ with a derivation of size $n-2$, or $\tau_0^{\mathrm{n}} \wedge \tau_j^{\mathrm{cn}} \subseteq \bot$.

**Case** $\tau_0^{\mathrm{n}} \subseteq \tau_c$**.** Then by S-Trans with (14), we have:

$$\bigvee_i \pi_i^{\mathrm{cn}} \subseteq \tau_c \tag{18}$$

with a derivation of size $n-1$. The result then follows from IH.

**Case** $\tau_0^{\mathrm{n}} \not\subseteq \tau_c$**.** Then for each $j$, we have $\tau_j^{\mathrm{cn}} \subseteq \tau_c$ or $\tau_0^{\mathrm{n}} \wedge \tau_j^{\mathrm{cn}} \subseteq \bot$. Let $S = \{\, j \mid \tau_0^{\mathrm{n}} \wedge \tau_j^{\mathrm{cn}} \subseteq \bot \,\}$. By S-AndOr2·, we have:

$$\bigvee_{j \notin S} \tau_j^{\mathrm{cn}} \subseteq \tau_c \tag{19}$$

with a derivation of size $n-1$. From the definiton of $S$, we have:

$$\overline{\tau_0^{\mathrm{n}} \wedge \tau_j^{\mathrm{cn}} \subseteq \bot}^{\,j \in S} \tag{20}$$

By Theorem A.9, (20) implies:

$$\overline{\tau_j^{\mathrm{cn}} \subseteq \tau_{0'}^{\mathrm{n}}}^{\,j \in S} \tag{21}$$

where $\tau_{0'}^{\mathrm{n}} = \mathrm{neg}(\tau_0^{\mathrm{n}})$. By Lemma A.7· on (21) and S-Refl, we have:

$$\bigvee_j \tau_j^{\mathrm{cn}} \subseteq \bigvee_{j \notin S} \tau_j^{\mathrm{cn}} \vee \tau_{0'}^{\mathrm{n}} \tag{22}$$

Then by S-Trans on (15) and (22), we have:

$$\bigvee_i \pi_i^{\mathrm{cn}} \subseteq \bigvee_{j \notin S} \tau_j^{\mathrm{cn}} \vee \tau_{0'}^{\mathrm{n}} \tag{23}$$

By Theorem A.9, (23) implies:

$$\tau_0^{\mathrm{n}} \wedge \bigvee_i \pi_i^{\mathrm{cn}} \subseteq \bigvee_{j \notin S} \tau_j^{\mathrm{cn}} \tag{24}$$

By S-Trans with S-AndOr2 on (14) and S-Refl, (24) implies:

$$\bigvee_i \pi_i^{\mathrm{cn}} \subseteq \bigvee_{j \notin S} \tau_j^{\mathrm{cn}} \tag{25}$$

Since we have (15) with a derivation of size $m-1$ and (25), it is easy to see that (25) can be derived with size $m-1$. Then by S-Trans with (19), we have:

$$\bigvee_i \pi_i^{\mathrm{cn}} \subseteq \tau_c \tag{26}$$

with a derivation of size $n$ and a former premise of size $m-1$. The result then follows from IH′ (2).

**Cases (S-DistribDcn⊃, ∗).**

1. Then $\tau^{\mathrm{dcn}} = (\bigwedge_j \tau_j^{\mathrm{n}}) \vee \bigvee_k \tau_k^{\mathrm{cn}}$ for some $\overline{\tau_j^{\mathrm{n}}}^{\,j}$ and $\overline{\tau_k^{\mathrm{cn}}}^{\,k}$. The premises of the former rule are:

$$\overline{\pi_1^{\mathrm{n}} \wedge \pi_2^{\mathrm{cn}} \subseteq \tau_j^{\mathrm{n}} \vee \bigvee_k \tau_k^{\mathrm{cn}}}^{\,j} \tag{27}$$

all with a derivation of size $m-1$. The latter premise is:

$$(\bigwedge_j \tau_j^{\mathrm{n}}) \vee \bigvee_k \tau_k^{\mathrm{cn}} \subseteq \tau \tag{28}$$

By IH (2), (28) implies:

$$\bigwedge_j \tau_j^{\mathrm{n}} \subseteq \tau \tag{29}$$

$$\overline{\tau_k^{\mathrm{cn}} \subseteq \tau}^{\,k} \tag{30}$$

all with a derivation of size $n-2$. By repeated applications of IH (1), (29) implies $\tau_l^{\mathrm{n}} \subseteq \tau$ for some $l \in \{\, \overline{j} \,\}$ with a derivation of size $n-2$, or $\bigwedge_j \tau_j^{\mathrm{n}} \subseteq \bot$.

**Case** $\tau_l^{\mathrm{n}} \subseteq \tau$**.** Then by S-AndOr2$\cdot$ with (30), we have:

$$\tau_l^{\mathrm{n}} \vee \bigvee_k \tau_k^{\mathrm{cn}} \subseteq \tau \tag{31}$$

with a derivation of size $n-1$. Then by S-Trans on (27) for $j=k$ and (31), we have:

$$\pi_1^{\mathrm{n}} \wedge \pi_2^{\mathrm{cn}} \subseteq \tau \tag{32}$$

with a derivation of size $n$ and a former premise of size $m-1$. The result then follows from IH′ (1).

**Case** $\bigwedge_j \tau_j^{\mathrm{n}} \subseteq \bot$**.** Then it is easy to see that the transitivity chain in the derivation for one of (27) must pass through either $\bigvee_k \tau_k^{\mathrm{cn}}$ or $\bot$, i.e., $\pi_1^{\mathrm{n}} \wedge \pi_2^{\mathrm{cn}} \subseteq \bigvee_k \tau_k^{\mathrm{cn}}$ or $\pi_1^{\mathrm{n}} \wedge \pi_2^{\mathrm{cn}} \subseteq \bot$ can be derived with size $m-1$.

**Case** $\pi_1^{\mathrm{n}} \wedge \pi_2^{\mathrm{cn}} \subseteq \bigvee_k \tau_k^{\mathrm{cn}}$**.** Then by S-Trans with S-AndOr2$\cdot$ on (30), we have (32) with a derivation of size $n$ and a former derivation of size $m-1$. The result then follows from IH′ (1).

**Case** $\pi_1^{\mathrm{n}} \wedge \pi_2^{\mathrm{cn}} \subseteq \bot$**.** then we have the result immediately.

2. Then $\tau^{\mathrm{dcn}} = (\bigwedge_j \tau_j^{\mathrm{n}}) \vee \bigvee_k \tau_k^{\mathrm{cn}}$ for some $\overline{\tau_j^{\mathrm{n}}}^{j}$ and $\overline{\tau_k^{\mathrm{cn}}}^{k}$. The premises of the former rule are:

$$\overline{\bigvee_i \pi_i^{\mathrm{cn}} \subseteq \tau_j^{\mathrm{n}} \vee \bigvee_k \tau_k^{\mathrm{cn}}}^{j} \tag{33}$$

all with a derivation of size $m-1$. The latter premise is:

$$(\bigwedge_j \tau_j^{\mathrm{n}}) \vee \bigvee_k \tau_k^{\mathrm{cn}} \subseteq \tau_c \tag{34}$$

By IH (2), (34) implies:

$$\bigwedge_j \tau_j^{\mathrm{n}} \subseteq \tau_c \tag{35}$$

$$\overline{\tau_k^{\mathrm{cn}} \subseteq \tau_c}^{k} \tag{36}$$

all with a derivation of size $n-2$. By repeated applications of IH (1), (35) implies $\tau_l^{\mathrm{n}} \subseteq \tau_c$ for some $l \in \{\overline{j}\}$ with a derivation of size $n-2$, or $\bigwedge_j \tau_j^{\mathrm{n}} \subseteq \bot$.

**Case** $\tau_l^{\mathrm{n}} \subseteq \tau_c$**.** Then by S-AndOr2$\cdot$ with (36), we have:

$$\tau_l^{\mathrm{n}} \vee \bigvee_k \tau_k^{\mathrm{cn}} \subseteq \tau_c \tag{37}$$

with a derivation of size $n-1$. Then by S-Trans on (33) for $j=l$ and (37), we have:

$$\bigvee_i \pi_i^{\mathrm{cn}} \subseteq \tau_c \tag{38}$$

with a derivation of size $n$ and a former premise of size $m-1$. The result then follows from IH′ (1).

**Case** $\bigwedge_j \tau_j^{\mathrm{n}} \subseteq \bot$**.** Then it is easy to see that the transitivity chain in the derivation for one of (33) must pass through either $\bigvee_k \tau_k^{\mathrm{cn}}$ or $\bot$, i.e., $\bigvee_i \pi_i^{\mathrm{cn}} \subseteq \bigvee_k \tau_k^{\mathrm{cn}}$ or $\bigvee_i \pi_i^{\mathrm{cn}} \subseteq \bot$ can be derived with size $m-1$.

**Case** $\bigvee_i \pi_i^{\mathrm{cn}} \subseteq \bigvee_k \tau_k^{\mathrm{cn}}$**.** Then by S-Trans with S-AndOr2$\cdot$ on (36), we have:

$$\bigvee_i \pi_i^{\mathrm{cn}} \subseteq \tau_c \tag{39}$$

with a derivation of size $n$ and a former derivation of size $m-1$. The result then follows from IH′ (2).

**Case** $\bigvee_i \pi_i^{\mathrm{cn}} \subseteq \bot$**.** Then by S-Trans with S-ToB$\supset$, we have:

$$\bigvee_i \pi_i^{\mathrm{cn}} \subseteq \tau_c \tag{40}$$

with a derivation of size $m \leqslant n-1$. The result then follows from IH (2).

■

**Corollary A.59.** *For* $\tau \in \{ \top^\diamond, T\ \overline{\tau^+}\ \overline{\tau^-}\ \overline{\tau^0} \}$*, if* $\bigwedge_i^\diamond \pi_i^{\mathrm{n}} \subseteq^\diamond \tau$*, then either* $\pi_k^{\mathrm{n}} \subseteq^\diamond \tau$ *for some* $k$ *or* $\bigwedge_i^\diamond \pi_i^{\mathrm{n}} \subseteq^\diamond \bot^\diamond$*.*

**Proof** By repeated applications of Lemma A.58. ■

**Lemma A.60.**

*(A) If* $\bigwedge_{i \in 1..n} \tau_i^{\mathrm{dn}} \subseteq \pi^{\mathrm{dn}}$ *with a derivation of size n, where* $\bigwedge_{i \in 1..n} \tau_i^{\mathrm{dn}}$ *is a complement-free CDN-normalized form, then either* $\tau_1^{\mathrm{dn}} \subseteq \pi^{\mathrm{dn}}$ *or* $\bigwedge_{i \in 2..n} \tau_i^{\mathrm{dn}} \subseteq \pi^{\mathrm{dn}}$ *with a derivation of size n.*

*(B) If* $\pi^{\mathrm{cn}} \subseteq \bigvee_{i \in 1..n} \tau_i^{\mathrm{cn}}$ *with a derivation of size n, where* $\bigvee_{i \in 1..n} \tau_i^{\mathrm{cn}}$ *is a complement-free DCN-normalized form, then either* $\pi^{\mathrm{cn}} \subseteq \tau_1^{\mathrm{cn}}$ *or* $\pi^{\mathrm{cn}} \subseteq \bigvee_{i \in 2..n} \tau_i^{\mathrm{cn}}$ *with a derivation of size n.*

Only the proof for (A) is shown below. The proof for (B) is symmetric.

**Proof** By induction on right-leaning $\subseteq^{\mathrm{cdn}}$ derivations, where S-DistribCDN$\diamond$ does not occur as the first premise of S-Trans in any of the judgements (in both the assumptions and conclusions). It is easy to see that we can rewrite any subderivations with S-DistribCDN$\diamond$ as the first premise of S-Trans into an equivalent one by applying S-Trans to the premises of S-DistribCDN$\diamond$ and the second premise of S-Trans, followed by an application of S-DistribCDN$\diamond$.

In the remainder of this proof, we abbreviate $\subseteq^{\mathrm{cdn}}$ as $\subseteq$.

**Case S-Refl.** Then $\bigwedge_{i \in 1..n} \tau_i^{\mathrm{dn}} = \tau_1^{\mathrm{dn}} = \pi^{\mathrm{dn}}$, i.e., we have $\tau_1^{\mathrm{dn}} \subseteq \pi^{\mathrm{dn}}$.
**Case S-ToB$\cdot$.** Then $\pi^{\mathrm{dn}} = \top$ and we have both $\tau_1^{\mathrm{dn}} \subseteq \top$ and $\bigwedge_{i \in 2..n} \tau_i^{\mathrm{dn}} \subseteq \top$ by S-ToB$\cdot$.
**Case S-ToB$\supset$.** Then $\bigwedge_{i \in 1..n} \tau_i^{\mathrm{dn}} = \tau_1^{\mathrm{dn}} = \neg\top$, i.e., we have $\tau_1^{\mathrm{dn}} \subseteq \pi^{\mathrm{dn}}$.
**Case S-Compl$\cdot$.** Then $\bigwedge_{i \in 1..n} \tau_i^{\mathrm{dn}} = \tau_1^{\mathrm{dn}} = \top$, i.e., we have $\tau_1^{\mathrm{dn}} \subseteq \pi^{\mathrm{dn}}$.
**Case S-Compl$\supset$.** Impossible since $\bigwedge_{i \in 1..n} \tau_i^{\mathrm{dn}}$ is a complement-free CDN-normalized form.
**Case S-AndOr1$\cdot$.** Then $\bigwedge_{i \in 1..n} \tau_i^{\mathrm{dn}}$ is not an intersection, i.e., $\bigwedge_{i \in 1..n} \tau_i^{\mathrm{dn}} = \tau_1^{\mathrm{dn}}$ and we have $\tau_1^{\mathrm{dn}} \subseteq \pi^{\mathrm{dn}}$.

**Case S-AndOr1⤸.** Then $\pi^{\mathrm{dn}} = \tau_k^{\mathrm{dn}}$ for some $k \in \{\, \bar{i} \,\}$. If $k = 1$, then we have $\tau_1^{\mathrm{dn}} \subseteq \pi^{\mathrm{dn}}$ by S-Refl. Otherwise, we have $\bigwedge_{i \in 2..n} \tau_i^{\mathrm{dn}} \subseteq \pi^{\mathrm{dn}}$ by S-AndOr1⤸.

**Case S-AndOr2·.** Then $\bigwedge_{i \in 1..n} \tau_i^{\mathrm{dn}} = \tau_1^{\mathrm{dn}}$, i.e., we have $\tau_1^{\mathrm{dn}} \subseteq \pi^{\mathrm{dn}}$.

**Case S-AndOr2⤸.** Impossible since $\pi^{\mathrm{dn}}$ is not an intersection.

**Case S-DistribCdn·.** Then $\tau_1^{\mathrm{dn}} = \bigvee_j \tau_j^{\mathrm{n}}$ for some $\overline{\tau_j^{\mathrm{n}}}^{\,j}$. The premises of the rule are $\overline{\tau_j^{\mathrm{n}} \wedge \bigwedge_{i \in 2..n} \tau_i^{\mathrm{dn}} \subseteq \pi^{\mathrm{dn}}}^{\,j}$, all with a derivation of size $n - 1$. By IH on the premises, we have $\overline{\tau_j^{\mathrm{n}} \subseteq \pi^{\mathrm{dn}} \text{ or } \bigwedge_{i \in 2..n} \tau_i^{\mathrm{dn}} \subseteq \pi^{\mathrm{dn}}}^{\,j}$. If $\bigwedge_{i \in 2..n} \tau_i^{\mathrm{dn}} \subseteq \pi^{\mathrm{dn}}$, then we have the result immediately. Otherwise, we have $\overline{\tau_j^{\mathrm{n}} \subseteq \pi^{\mathrm{dn}}}^{\,j}$, which imply $\bigvee_j \tau_j^{\mathrm{n}} \subseteq \pi^{\mathrm{dn}}$ with a derivation of size $n$ by S-AndOr2·, i.e., $\tau_1^{\mathrm{dn}} \subseteq \pi^{\mathrm{dn}}$.

**Case S-DistribCdn⤸.** Then $\overline{\tau_i^{\mathrm{dn}} = \tau^{\mathrm{n}} \vee \tau_{i'}^{\mathrm{dn}}}^{\,i \in 1..n}$ for some $\tau^{\mathrm{n}}$ and $\overline{\tau_{i'}^{\mathrm{dn}}}^{\,i \in 1..n}$. The premises of the rule are $\tau^{\mathrm{n}} \subseteq \pi^{\mathrm{dn}}$ and $\bigwedge_{i \in 1..n} \tau_{i'}^{\mathrm{dn}} \subseteq \pi^{\mathrm{dn}}$. By IH on the latter premise, we have $\tau_{1'}^{\mathrm{dn}} \subseteq \pi^{\mathrm{dn}}$ or $\bigwedge_{i \in 2..n} \tau_{i'}^{\mathrm{dn}} \subseteq \pi^{\mathrm{dn}}$ with a derivation of size $n - 1$. If $\tau_{1'}^{\mathrm{dn}} \subseteq \pi^{\mathrm{dn}}$, then by S-AndOr2· with $\tau^{\mathrm{n}} \subseteq \pi^{\mathrm{dn}}$, we have $\tau_1^{\mathrm{dn}} = \tau^{\mathrm{n}} \vee \tau_{1'}^{\mathrm{dn}} \subseteq \pi^{\mathrm{dn}}$ with a derivation of size $n$. If $\bigwedge_{i \in 2..n} \tau_{i'}^{\mathrm{dn}} \subseteq \pi^{\mathrm{dn}}$, then by S-DistribCdn⤸ with $\tau^{\mathrm{n}} \subseteq \pi^{\mathrm{dn}}$, we have $\bigwedge_{i \in 2..n} \tau_i^{\mathrm{dn}} = \bigwedge_{i \in 2..n} (\tau^{\mathrm{n}} \vee \tau_{i'}^{\mathrm{dn}}) \subseteq \pi^{\mathrm{dn}}$ with a derivation of size $n$.

**Case S-Trans.** Then the premises of the rule are $\bigwedge_{i \in 1..n} \tau_i^{\mathrm{dn}} \subseteq \tau^{\mathrm{cdn}}$ and $\tau^{\mathrm{cdn}} \subseteq \pi^{\mathrm{dn}}$ for some $\tau^{\mathrm{cdn}}$. By induction on the size of the former premise of S-Trans, denoted by $m$. Denote the inner induction hypothesis by IH′.

**Cases (S-Refl, $*$).** By IH on the latter premise.

**Cases (S-ToB·).** Then $\tau^{\mathrm{cdn}} = \top$. By S-ToB·, we have both $\tau_1^{\mathrm{dn}} \subseteq \top$ and $\bigwedge_{i \in 2..n} \tau_i^{\mathrm{dn}} \subseteq \top$. Then we have $\tau_1^{\mathrm{dn}} \subseteq \pi^{\mathrm{dn}}$ and $\bigwedge_{i \in 2..n} \tau_i^{\mathrm{dn}} \subseteq \pi^{\mathrm{dn}}$ by S-Trans with the latter premise $\top \subseteq \pi^{\mathrm{dn}}$.

**Cases (S-ToB⤸, $*$).** Then $\bigwedge_{i \in 1..n} \tau_i^{\mathrm{dn}} = \tau_1^{\mathrm{dn}} = \neg\top$, i.e., we have $\tau_1^{\mathrm{dn}} \subseteq \pi^{\mathrm{dn}}$.

**Cases (S-Compl·, $*$).** Then $\bigwedge_{i \in 1..n} \tau_i^{\mathrm{dn}} = \tau_1^{\mathrm{dn}} = \top$, i.e., we have $\tau_1^{\mathrm{dn}} \subseteq \pi^{\mathrm{dn}}$.

**Cases (S-Compl⤸, $*$).** Impossible since $\bigwedge_{i \in 1..n} \tau_i^{\mathrm{dn}}$ is a complement-free CDN-normalized form.

**Cases (S-AndOr1·, $*$).** Then $\bigwedge_{i \in 1..n} \tau_i^{\mathrm{dn}}$ is not an intersection, i.e., $\bigwedge_{i \in 1..n} \tau_i^{\mathrm{dn}} = \tau_1^{\mathrm{dn}}$ and we have $\tau_1^{\mathrm{dn}} \subseteq \pi^{\mathrm{dn}}$.

**Cases (S-AndOr1⤸, $*$).** Then $\tau^{\mathrm{cdn}} = \bigwedge_{i' \in S} \tau_{i'}^{\mathrm{dn}}$ for some $S \subseteq \{\, \bar{i} \,\}$. If $1 \in S$, by IH on the latter premise, we have $\tau_1^{\mathrm{dn}} \subseteq \pi^{\mathrm{dn}}$ or $\bigwedge_{i' \in S \setminus \{1\}} \subseteq \pi^{\mathrm{dn}}$ with a derivation of size $n - 1$. If $\tau_1^{\mathrm{dn}} \subseteq \pi^{\mathrm{dn}}$, the result is immediate. If $\bigwedge_{i' \in S \setminus \{1\}} \subseteq \pi^{\mathrm{dn}}$, then we have $\bigwedge_{i \in 2..n} \tau_i^{\mathrm{dn}} \subseteq \pi^{\mathrm{dn}}$ with a derivation of size $n$ by S-Trans with S-AndOr1⤸. If $1 \notin S$, then $\bigwedge_{i \in 2..n} \tau_i^{\mathrm{dn}} \subseteq \pi^{\mathrm{dn}}$ with a derivation of size $n$ follows from IH on the latter premise, followed by S-Trans with S-AndOr1⤸.

**Cases (S-AndOr2·, $*$).** Then $\bigwedge_{i \in 1..n} \tau_i^{\mathrm{dn}} = \tau_1^{\mathrm{dn}}$, i.e., we have $\tau_1^{\mathrm{dn}} \subseteq \pi^{\mathrm{dn}}$.

**Cases (S-AndOr2⤸, $*$).** Then $\tau^{\mathrm{cdn}} = \bigwedge_j \pi_j^{\mathrm{dn}}$ for some $\overline{\pi_j^{\mathrm{dn}}}^{\,j}$. The premises of the former rule are $\overline{\bigwedge_{i \in 1..n} \tau_i^{\mathrm{dn}} \subseteq \pi_j^{\mathrm{dn}}}^{\,j}$. The latter premise is $\bigwedge_j \pi_j^{\mathrm{dn}} \subseteq \pi^{\mathrm{dn}}$, which implies $\pi_k^{\mathrm{dn}} \subseteq \pi^{\mathrm{dn}}$ for some $k \in \{\, \bar{j} \,\}$ with a derivation of size $n - 1$ by repeated applications of IH, which implies $\bigwedge_{i \in 1..n} \tau_i^{\mathrm{dn}} \subseteq \pi^{\mathrm{dn}}$ with a derivation of size $n$ and a former premise of size $m - 1$ by S-Trans with $\bigwedge_{i \in 1..n} \tau_i^{\mathrm{dn}} \subseteq \pi_k^{\mathrm{dn}}$. The result then follows from IH′.

**Cases (S-DistribCdn◇, $*$).** Impossible by assumption.

■

**Corollary A.61.**

*(A) If* $\bigwedge_i \tau_i^{\mathrm{dn}} \subseteq \pi^{\mathrm{dn}}$*, where* $\bigwedge_i \tau_i^{\mathrm{dn}}$ *is a complement-free CDN-normalized form, then* $\tau_k^{\mathrm{dn}} \subseteq \pi^{\mathrm{dn}}$ *for some* $k \in \{\, \overline{i} \,\}$.
*(B) If* $\pi^{\mathrm{cn}} \subseteq \bigvee_i \tau_i^{\mathrm{cn}}$*, where* $\bigvee_i \tau_i^{\mathrm{cn}}$ *is a complement-free DCN-normalized form, then* $\pi^{\mathrm{cn}} \subseteq \tau_k^{\mathrm{cn}}$ *for some* $k \in \{\, \overline{i} \,\}$.

**Proof** By repeated application of Lemma A.60. ■

**Lemma A.62.** $\top^{\diamond} \subseteq^{\diamond} \tau$ *is not derivable for* $\tau \in \{\, T\ \overline{\tau^+}\ \overline{\tau^-}\ \overline{\tau^0} \,\}$.

**Proof** By induction on $\subseteq^{\mathrm{cdn}}$ and $\subseteq^{\mathrm{dcn}}$ derivations respectively. ■

### *A.9 Soundness of Subtyping*

**Theorem A.63** (Subtyping consistency)**.** *If* $\Xi$ ***cons.*** *and* $\Xi \vdash \tau \leqslant \pi$*, where:*

$$\tau \in \{\, \bot,\ \top,\ \#C,\ \tau_1 \to \tau_2,\ \{\, \overline{x_i : \tau_i}^{\,i} \,\} \,\}$$
$$\pi \in \{\, \bot,\ \top,\ \#C',\ \pi_1 \to \pi_2,\ \{\, x' : \pi_1 \,\} \,\}$$

*then exactly one of the following is true:*

*(a)* $\tau = \bot$ *or* $\pi = \top$*;*
*(b)* $\tau = \#C$ *and* $\pi = \#C'$ *and* $C' \in \mathcal{S}(\#C)$*;*
*(c)* $\tau = \tau_1 \to \tau_2$ *and* $\pi = \pi_1 \to \pi_2$ *and* $\Xi \vdash \pi_1 \leqslant \tau_1$ *and* $\Xi \vdash \tau_2 \leqslant \pi_2$*;*
*(d)* $\tau = \{\, \overline{x_i : \tau_i}^{\,i} \,\}$ *and* $\pi = \{\, x_k : \pi_1 \,\}$ *and* $\Xi \vdash \tau_k \leqslant \pi_1$ *for some* $k$.

**Proof** By Lemma 3.4 on the assumption, we have:

$$\triangleright\Xi \vdash \tau \leqslant \pi \tag{1}$$

Then proceed by case analysis on $\tau$.

**Case** $\tau = \bot$**.** Then (a) is true and (b), (c), (d) are false.
**Case** $\tau = \top$**.** Then (b), (c), (d) are false. Since $\tau \cong \bot \vee \top$, by Lemma 4.22 on (1), we have:

$$\pi \cong \bigwedge_j \left( \pi'_j \vee V_j^{D_j} \right) \tag{2}$$

$$\overline{\triangleright\Xi \vdash \top \preceq V_j^{D_j}}^{\,j} \tag{3}$$

for some $\overline{\pi'_j}^{\,j}$ and $\overline{D_j}^{\,j}$ and $\overline{V_j^{D_j}}^{\,j}$, where $\bigwedge_j V_j^{D_j}$ is complement-free. By Lemma 4.9, (3) implies:

$$\overline{D_j \in \{\, \top, \not\perp \,\}}^{\,j} \tag{4}$$

By Lemma A.7↷ on S-AndOr12·, we have:

$$\bigwedge_j V_j^{D_j} \subseteq \bigwedge_j \left(\pi'_j \vee V_j^{D_j}\right) \tag{5}$$

By S-Trans on (5) and (2), we have:

$$\bigwedge_j V_j^{D_j} \subseteq \pi \tag{6}$$

Since $\bigwedge_j V_j^{D_j}$ is complement-free, we have:

$$\bigwedge_j V_j^{D_j} \nsubseteq \bot \tag{7}$$

Then (6) and (7) imply:

$$\pi \nsubseteq \bot \tag{8}$$

By Lemma A.57, (6) implies:

$$V_k^{D_k} \subseteq \pi \tag{9}$$

for some $k$. By case analysis on the syntax of $V_k^{D_k}$ and the assumption on the form of $\pi$, (9) can only be derived when $\pi = \top$. Then we have $\pi = \top$, i.e., (a) is true.

**Case** $\tau = \#C$**.** Then (c), (d) are false. Since $\tau \cong \bot \vee \#C$, by Lemma 4.22 on (1), we have:

$$\pi \cong \bigwedge_j \left(\pi'_j \vee V_j^{D_j}\right) \tag{10}$$

$$\overline{\triangleright \Xi \vdash \#C \preceq V_j^{D_j}}^{\,j} \tag{11}$$

for some $\overline{\pi'_j}^{\,j}$ and $\overline{D_j}^{\,j}$ and $\overline{V_j^{D_j}}^{\,j}$, where $\bigwedge_j V_j^{D_j}$ is complement-free. By Lemma 4.9, (11) implies:

$$\overline{D_j \in \{\, \#C_1, \cancel{\#C_2}, \top, \cancel{\not\perp} \,\}}^{\,j} \tag{12}$$

where $C_1 \in \mathcal{S}(\#C)$ and $C_2 \notin \mathcal{S}(\#C)$ and $C \notin \mathcal{S}(\#C_2)$. By Lemma A.7↷ on S-AndOr12·, we have:

$$\bigwedge_j V_j^{D_j} \subseteq \bigwedge_j \left(\pi'_j \vee V_j^{D_j}\right) \tag{13}$$

By S-Trans on (13) and (10), we have:

$$\bigwedge_j V_j^{D_j} \subseteq \pi \tag{14}$$

Since $\bigwedge_j V_j^{D_j}$ is complement-free, we have:

$$\bigwedge_j V_j^{D_j} \nsubseteq \bot \tag{15}$$

Then (14) and (15) imply:

$$\pi \nsubseteq \bot \tag{16}$$

By Lemma A.57, (14) implies:

$$V_k^{D_k} \subseteq \pi \tag{17}$$

for some $k$. By Lemma 4.10, (17) implies either $\pi = \top$ or $V_k^{D_k} = \bigvee_l \pi$.

**Case** $\pi = \top$**.** Then (a) is true and (b) is false.
**Case** $\pi \neq \top$**.** Then we have:

$$\pi \cong \bigvee_l \pi = V_k^{D_k} \tag{18}$$

By the syntax of $U^{\top}$ and $U^{\not\perp}$, we have:

$$D_k \notin \{ \top, \not\perp \} \tag{19}$$

Then (12) and (19) imply:

$$D_k \in \{ \#C_1, \cancel{\#C_2} \} \tag{20}$$

By case analysis on the assumption on the form of $\pi$, we have:

$$\pi = \#C_1 \tag{21}$$

where $C_1 \in \mathcal{S}(\#C)$. Then (b) is true and (a) is false.

**Case** $\tau = \tau_1 \to \tau_2$**.** Then (b), (d) are false. Since $\tau \cong \bot \vee (\tau_1 \to \tau_2)$, by Lemma 4.22 on (1), we have:

$$\pi \cong \bigwedge_j \left( \pi'_j \vee V_j^{D_j} \right) \tag{22}$$

$$\rhd \Xi \vdash \tau_1 \to \tau_2 \preceq \overline{V_j^{D_j}}^{j} \tag{23}$$

for some $\overline{\pi'_j}^{j}$ and $\overline{D_j}^{j}$ and $\overline{V_j^{D_j}}^{j}$, where $\bigwedge_j V_j^{D_j}$ is complement-free. By Lemma 4.9, (23) implies:

$$\overline{D_j \in \{ \to, \top, \not\perp \}}^{j} \tag{24}$$

By Lemma A.7↷ on S-AndOr12·, we have:

$$\bigwedge_j V_j^{D_j} \subseteq \bigwedge_j \left( \pi'_j \vee V_j^{D_j} \right) \tag{25}$$

By S-Trans on (25) and (10), we have:

$$\bigwedge_j V_j^{D_j} \subseteq \pi \tag{26}$$

Since $\bigwedge_j V_j^{D_j}$ is complement-free, we have:

$$\bigwedge_j V_j^{D_j} \not\subseteq \bot \tag{27}$$

Then (26) and (27) imply:

$$\pi \not\subseteq \bot \tag{28}$$

By Lemma A.57, (26) implies:

$$V_k^{D_k} \subseteq \pi \tag{29}$$

for some $k$. By Lemma 4.10, (29) implies either $\pi = \top$ or $V_k^{D_k} = \bigvee_l \pi$.
**Case** $\pi = \top$**.** Then (a) is true and (c) is false.
**Case** $\pi \neq \top$**.** Then we have:

$$\pi \cong \bigvee_l \pi = V_k^{D_k} \tag{30}$$

By the syntax of $U^{\top}$ and $U^{\not\perp}$, we have:

$$D_k \notin \{ \top, \not\perp \} \tag{31}$$

Then (24) and (31) imply:

$$D_k = \rightarrow \tag{32}$$

By case analysis on the assumption on the form of $\pi$, we have:

$$\pi = \pi_1 \rightarrow \pi_2 \tag{33}$$

Then (23) implies:

$$\triangleright \Xi \vdash \tau_1 \rightarrow \tau_2 \preceq \bigvee_l \pi_1 \rightarrow \pi_2 \tag{34}$$

By case analysis on the $\preceq$ rules, (34) implies:

$$\triangleright \Xi \vdash \tau_1 \rightarrow \tau_2 \preceq (\bigwedge_l \pi_1) \rightarrow (\bigvee_l \pi_2) \tag{35}$$

Again by case analysis on the $\preceq$ rules, (35) implies:

$$\Xi \vdash \bigwedge_l \pi_1 \leqslant \tau_1 \tag{36}$$

$$\Xi \vdash \tau_2 \leqslant \bigvee_l \pi_2 \tag{37}$$

By S-Trans with S-AndOr2$\diamond$ on S-Refl, (36) and (37) imply:

$$\Xi \vdash \pi_1 \leqslant \tau_1 \tag{38}$$

$$\Xi \vdash \tau_2 \leqslant \pi_2 \tag{39}$$

Then (c) is true and (a) is false.

**Case** $\tau = \{ \overline{x_i : \tau_i}^{\,i} \}$**.** Then (b), (c) are false. Since $\tau \cong \bigwedge_i (\bot \vee \{ x_i : \tau_i \})$, by Lemma 4.22 on (1), we have:

$$\pi \cong \bigwedge_j \left( \pi'_j \vee V_j^{D_j} \right) \tag{40}$$

$$\overline{\triangleright \Xi \vdash \{ x_{k_j} : \tau_{k_j} \} \preceq V_j^{D_j}}^{\,j} \tag{41}$$

for some $\overline{\pi'_j}^{\,j}$ and $\overline{D_j}^{\,j}$ and $\overline{V_j^{D_j}}^{\,j}$ and $\overline{k_j}^{\,j}$, where $\bigwedge_j V_j^{D_j}$ is complement-free. By Lemma 4.9, (41) implies:

$$\overline{D_j \in \{ x_{k_j}, \top, \not\perp \}}^{\,j} \tag{42}$$

By Lemma A.7$\wr$ on S-AndOr12$\cdot$, we have:

$$\bigwedge_j V_j^{D_j} \subseteq \bigwedge_j \left( \pi'_j \vee V_j^{D_j} \right) \tag{43}$$

By S-Trans on (43) and (10), we have:

$$\bigwedge_j V_j^{D_j} \subseteq \pi \tag{44}$$

Since $\bigwedge_j V_j^{D_j}$ is complement-free, we have:

$$\bigwedge_j V_j^{D_j} \nsubseteq \bot \tag{45}$$

Then (44) and (45) imply:

$$\pi \not\leqq \bot \tag{46}$$

By Lemma A.57, (44) implies:

$$V_k^{D_k} \subseteq \pi \tag{47}$$

for some $k$. By Lemma 4.10, (47) implies either $\pi = \top$ or $V_k^{D_k} = \bigvee_l \pi$.

**Case** $\pi = \top$**.** Then (a) is true and (d) is false.

**Case** $\pi \neq \top$**.** Then we have:

$$\pi \cong \bigvee_l \pi = V_k^{D_k} \tag{48}$$

By the syntax of $U^{\top}$ and $U^{\not\perp}$, we have:

$$D_k \notin \{ \top, \not\perp \} \tag{49}$$

Then (42) and (49) imply:

$$D_k = x_{k_k} \tag{50}$$

By case analysis on the assumption on the form of $\pi$, we have:

$$\pi = \{ x_{k_k} : \pi_1 \} \tag{51}$$

Then (41) implies:

$$\triangleright\Xi \vdash \{ x_{k_k} : \tau_{k_k} \} \preceq \bigvee_l \{ x_{k_k} : \pi_1 \} \tag{52}$$

By case analysis on the $\preceq$ rules, (52) implies:

$$\triangleright\Xi \vdash \{ x_{k_k} : \tau_{k_k} \} \preceq \{ x_{k_k} : \bigvee_l \pi_1 \} \tag{53}$$

Again by case analysis on the $\preceq$ rules, (53) implies:

$$\Xi \vdash \tau_{k_k} \leqslant \bigvee_l \pi_1 \tag{54}$$

By S-Trans with S-AndOr2· on S-Refl, (54) implies:

$$\Xi \vdash \tau_{k_k} \leqslant \pi_1 \tag{55}$$

Then (d) is true and (a) is false.

■

Only the proof for (A) of Lemma 4.22 is shown below. The proof for (B) is *mostly* symmetric.

**Proof** [Lemma 4.22] By Lemma 4.14, there exists some $\tau^{\mathrm{cdn}}$ and $\pi^{\mathrm{cdn}}$ such that $\tau \cong \tau^{\mathrm{cdn}}$ and $\pi \cong \pi^{\mathrm{cdn}}$. Then by Lemma 4.21, we only need to consider CDN-normalized derivations for $\tau^{\mathrm{cdn}} \leqslant^{\mathrm{cdn}} \pi^{\mathrm{cdn}}$, and the result would also apply to the original derivation for $\tau \leqslant \pi$. By induction on unassuming CDN-normalized subtyping derivations.

**Case S-Refl.** Immediate since $\tau \cong \pi$. Pick $\overline{\pi_i' = \tau_i'}^{\,i}$ and $\overline{V_i^{D_i} = U_i^{C_i}}^{\,i}$. Then $\pi \cong \bigwedge_i \left( \pi_i' \vee V_i^{D_i} \right)$ and $\overline{U_i^{C_i} \preceq V_i^{D_i}}^{\,i}$.

**Case S-ToB$\cdot$.** Then $\pi = \top$. Pick $\pi'_1 = \bot$ and $V_1^{D_1} = V_1^{\top} = \top$. Then $\pi \cong \pi'_1 \vee V_1^{D_1}$ and $\overline{U_i^{C_i} \preceq V_1^{D_1}}^{i}$.

**Case S-ToB$\curvearrowright$.** Then $\tau = \neg\top$. So $\pi \cong \pi \vee \tau \cong \pi \vee \bigwedge_i \left(\tau'_i \vee U_i^{C_i}\right)$. By distributivity, we have $\pi \cong \bigwedge_i \left(\pi \vee \tau'_i \vee U_i^{C_i}\right)$. Pick $\overline{\pi'_i = \pi \vee \tau'_i}^{i}$ and $\overline{V_i^{D_i} = U_i^{C_i}}^{i}$. Then $\pi \cong \bigwedge_i \left(\pi'_i \vee V_i^{D_i}\right)$ and $\overline{U_i^{C_i} \preceq V_i^{D_i}}^{i}$.

**Cases S-Compl$\diamond$.** Immediate since $\tau \cong \pi$. Proceed with the same reasoning as case S-Refl.

**Case S-AndOr1$\cdot$.** $\pi = \tau \vee \pi'$ for some $\pi'$. Then $\pi \cong \bigwedge_i \left(\tau'_i \vee U_i^{C_i}\right) \vee \pi' \cong \bigwedge_i \left(\tau'_i \vee \pi' \vee U_i^{C_i}\right)$. Pick $\overline{\pi'_i = \tau'_i \vee \pi'}^{i}$ and $\overline{V_i^{D_i} = U_i^{C_i}}^{i}$. Then $\pi \cong \bigwedge_i \left(\pi'_i \vee V_i^{D_i}\right)$ and $\overline{U_i^{C_i} \preceq V_i^{D_i}}^{i}$.

**Case S-AndOr1$\curvearrowright$.** $\tau = \pi \wedge \tau'$ for some $\tau'$. Then from the assumption, we have:

$$\tau = \pi \wedge \tau' \cong \bigwedge_i \left(\tau'_i \vee U_i^{C_i}\right) \tag{1}$$

By Lemma A.7$\cdot$ on S-Refl and (1), we have:

$$(\pi \wedge \neg\tau') \vee (\pi \wedge \tau') \cong (\pi \wedge \neg\tau') \vee \bigwedge_i \left(\tau'_i \vee U_i^{C_i}\right)$$

$$\text{i.e.,} \quad \pi \cong \bigwedge_i \left((\pi \wedge \neg\tau') \vee \tau'_i \vee U_i^{C_i}\right) \tag{2}$$

Pick $\overline{\pi'_i = (\pi \wedge \neg\tau') \vee \tau'_i}^{i}$ and $\overline{V_i^{D_i} = U_i^{C_i}}^{i}$. Then $\pi \cong \bigwedge_i \left(\pi'_i \vee V_i^{D_i}\right)$ and $\overline{U_i^{C_i} \preceq V_i^{D_i}}^{i}$.

**Case S-AndOr2$\cdot$.** By induction on the number of premises. Denote the inner induction hypothesis as IH′. We have $\tau = \bigvee_{h \in 1..n} \tau_h^{\mathrm{n}}$ for some $\overline{\tau_h^{\mathrm{n}}}^{h \in 1..n}$. Let $\tau_2^{\mathrm{dn}} = \bigvee_{h \in 2..n} \tau_h^{\mathrm{n}}$, then $\tau = \tau_1^{\mathrm{n}} \vee \tau_2^{\mathrm{dn}}$. The premises of the rule are:

$$\overline{\tau_h^{\mathrm{n}} \leqslant \pi}^{h \in 1..n} \tag{3}$$

By S-AndOr2$\cdot$ on (3) for $h \in 2..n$, we have:

$$\tau_2^{\mathrm{dn}} \leqslant \pi \tag{4}$$

with the same size as the current derivation and one fewer premise. From the assumption, we have:

$$\bigwedge_i \left(\tau'_i \vee U_i^{C_i}\right) \subseteq \tau = \tau_1^{\mathrm{n}} \vee \tau_2^{\mathrm{dn}} \tag{5}$$

By S-Trans with Lemma A.7$\curvearrowright$ on S-AndOr12$\cdot$, (5) implies:

$$\bigwedge_i U_i^{C_i} \subseteq \tau_1^{\mathrm{n}} \vee \tau_2^{\mathrm{dn}} \tag{6}$$

By Corollary A.61, (6) implies:

$$U_k^{C_k} \subseteq \tau_1^{\mathrm{n}} \vee \tau_2^{\mathrm{dn}} \tag{7}$$

for some $k$.

**Case $C_k = B$.** If $C_k = B$ for some $B$, then by Lemma A.44, (7) implies:

$$\overline{{U'}_l^{C'_l} \subseteq \tau_1^{\mathrm{n}} \vee \tau_2^{\mathrm{dn}}}^{\,l} \tag{8}$$

where $U_k^{C_k} = \bigvee_l {U'}_l^{C'_l}$ and $\overline{{U'}_l^{C'_l}}^{\,l}$ are not unions. By Lemma A.58, (8) implies either $\overline{{U'}_l^{C'_l} \subseteq \tau_1^{\mathrm{n}}}^{\,l}$ or $\overline{{U'}_l^{C'_l} \subseteq \tau_2^{\mathrm{dn}}}^{\,l}$ or $\top \subseteq \tau_1^{\mathrm{n}} \vee \tau_2^{\mathrm{dn}}$.

**Case $\overline{{U'}_l^{C'_l} \subseteq \tau_1^{\mathrm{n}}}^{\,l}$ or $\overline{{U'}_l^{C'_l} \subseteq \tau_2^{\mathrm{dn}}}^{\,l}$.** By S-AndOr2·, we have:

$$U^{1\,C^1} := \bigvee_{l \mid {U'}_l^{C'_l} \subseteq \tau_1^{\mathrm{n}}} {U'}_l^{C'_l} \subseteq \tau_1^{\mathrm{n}} \tag{9}$$

$$U^{2\,C^2} := \bigvee_{l \mid {U'}_l^{C'_l} \subseteq \tau_2^{\mathrm{dn}}} {U'}_l^{C'_l} \subseteq \tau_2^{\mathrm{dn}} \tag{10}$$

By S-AndOr2· with S-Refl, (9) and (10) imply:

$$\tau_1^{\mathrm{n}} \vee U^{1\,C^1} \subseteq \tau_1^{\mathrm{n}} \tag{11}$$

$$\tau_2^{\mathrm{dn}} \vee U^{2\,C^2} \subseteq \tau_2^{\mathrm{dn}} \tag{12}$$

Since we have the other direction by S-AndOr11·, (11) and (12) imply:

$$\tau_1^{\mathrm{n}} \cong \tau_1^{\mathrm{n}} \vee U^{1\,C^1} \tag{13}$$

$$\tau_2^{\mathrm{dn}} \cong \tau_2^{\mathrm{dn}} \vee U^{2\,C^2} \tag{14}$$

Then by IH on the (3) for $h = 1$ and (13), we have:

$$\pi \cong \bigwedge_p \left(\pi_p^1 \vee {V^1}_p^{\,{D^1}_p}\right) \tag{15}$$

$$\overline{\triangleright\Sigma \vdash U^{1\,C^1} \preceq {V^1}_p^{\,{D^1}_p}}^{\,p} \tag{16}$$

By IH′ on (4) and (14), we have:

$$\pi \cong \bigwedge_q \left(\pi_q^2 \vee {V^2}_q^{\,{D^2}_q}\right) \tag{17}$$

$$\overline{\triangleright\Sigma \vdash U^{2\,C^2} \preceq {V^2}_q^{\,{D^2}_q}}^{\,q} \tag{18}$$

By distributivity, (15) and (17) imply:

$$\pi \cong \bigwedge_{p,q} \left(\pi_p^1 \vee \pi_q^2 \vee {V^1}_p^{\,{D^1}_p} \vee {V^2}_q^{\,{D^2}_q}\right) \tag{19}$$

For each pair $(p, q)$, we pick $\pi'_{pq}$ and $V_{pq}^{D_{pq}}$ as follows:

- If ${D^1}_p \in \{\top, \not\perp\}$, pick $\pi'_{pq} = \pi_p^1 \vee \pi_q^2 \vee {V^2}_q^{\,{D^2}_q}$ and $V_{pq}^{D_{pq}} = {V^1}_p^{\,{D^1}_p}$. Then $\triangleright\Sigma \vdash U_k^{C_k} \preceq V_{pq}^{D_{pq}}$.
- If ${D^2}_q \in \{\top, \not\perp\}$, pick $\pi'_{pq} = \pi_p^1 \vee \pi_q^2 \vee {V^1}_p^{\,{D^1}_p}$ and $V_{pq}^{D_{pq}} = {V^2}_q^{\,{D^2}_q}$. Then $\triangleright\Sigma \vdash U_k^{C_k} \preceq V_{pq}^{D_{pq}}$.
- If ${D^1}_p \notin \{\top, \not\perp\}$ and ${D^2}_q \notin \{\top, \not\perp\}$ and ${D^1}_p \neq {D^2}_q$, then we have at least one of the following by Lemma 4.9 (note that since $C_k = B$, we have $C^1 = B^1$ and $C^2 = B^2$ for some $B^1$ and $B^2$):

- $D^1{}_p = C^1$ and $D^2{}_q = C^2$, which implies $C^1 \neq C^2$. Since $U_k^{C_k} \cong U^{1^{C^1}} \vee U^{2^{C^2}}$, we have $C_k = \top$ and $(C^1, C^2) \in \{ (x, y^{\neq x}), (x, \to), (\to, x) \}$ for some $x$ and $y$. Then $V^{1^{D^1{}_p}}_p \vee V^{2^{D^2{}_q}}_q \cong \pi^3_{pq} \vee V^{\top}_{pq}$ for some $\pi^3_{pq}$ and $V^{\top}_{pq}$. Then we can pick $\pi'_{pq} = \pi^1_p \vee \pi^2_q \vee \pi^3_{pq}$ and $V^{D_{pq}}_{pq} = V^{\top}_{pq}$, where we have $\triangleright\Sigma \vdash U_k^{C_k} \preceq V_{pq}^{D_{pq}}$.
- $C^1 = \#C_1$ and $D^1{}_p = \#C_2$, where $C_2 \in \mathcal{S}(\#C_1)$. Since $U_k^{C_k} \cong U^{1^{C^1}} \vee U^{2^{C^2}}$, we have $C_k = C^1 = C^2 = \#C_1$ Then we can pick $\pi'_{pq} = \pi^1_p \vee \pi^2_q \vee V^{2^{D^2{}_q}}_q$ and $V^{D_{pq}}_{pq} = V^{1^{D^1{}_p}}_p$, where we have $\triangleright\Sigma \vdash U_k^{C_k} \preceq V_{pq}^{D_{pq}}$.
- $C^1 = \#C_1$ and $D^1{}_p = \cancel{\#C_2}$, where $C_1 \notin \mathcal{S}(\#C_2)$ and $C_2 \notin \mathcal{S}(\#C_1)$. Proceed similarly as above.
- $C^2 = \#C_1$ and $D^2{}_q = \#C_2$, where $C_2 \in \mathcal{S}(\#C_1)$. Since $U_k^{C_k} \cong U^{1^{C^1}} \vee U^{2^{C^2}}$, we have $C_k = C^1 = C^2 = \#C_1$ Then we can pick $\pi'_{pq} = \pi^1_p \vee \pi^2_q \vee V^{1^{D^1{}_p}}_p$ and $V^{D_{pq}}_{pq} = V^{2^{D^2{}_q}}_q$, where we have $\triangleright\Sigma \vdash U_k^{C_k} \preceq V_{pq}^{D_{pq}}$.
- $C^2 = \#C_1$ and $D^2{}_q = \cancel{\#C_2}$, where $C_1 \notin \mathcal{S}(\#C_2)$ and $C_2 \notin \mathcal{S}(\#C_1)$. Proceed similarly as above.

• If $D^1{}_p = D^2{}_q \notin \{ \top, \not\perp \}$, then we have $C^1 = C^2 = D^1{}_p = D^2{}_q$. Then $U_k^{C_k} \cong U^{1^{C^1}} \vee U^{2^{C^2}}$ and $U^{1^{C^1}} \preceq V^{1^{D^1{}_p}}_p$ and $U^{2^{C^2}} \preceq V^{2^{D^2{}_q}}_q$ imply $U_k^{C_k} \preceq V^{1^{D^1{}_p}}_p \vee V^{2^{D^2{}_q}}_q$, so we can pick $\pi'_{pq} = \pi^1_p \vee \pi^2_q$ and $V^{D_{pq}}_{pq} = V^{1^{D^1{}_p}}_p \vee V^{2^{D^2{}_q}}_q$.

Then we have:

$$\pi \cong \bigwedge\nolimits_{p,q} \left( \pi'_{pq} \vee V^{D_{pq}}_{pq} \right) \tag{20}$$

$$D_{pq} \in \{ \overline{C_i}^{\,i} \} \cup \{ \top, \not\perp \} \cup \{ \overline{x}^{x \notin \{ \overline{C_i}^{\,i} \}} \} \cup \{ \overline{\cancel{\#C}}^{\#C \notin \{ \overline{C_i}^{\,i} \}} \} \tag{21}$$

$$\triangleright\Sigma \vdash U_k^{C_k} \preceq V_{pq}^{D_{pq}} \tag{22}$$

The conditions on $D_{pq}$ in (21) ensures that we can rewrite $\bigwedge_{p,q} \left( \pi'_{pq} \vee V^{D_{pq}}_{pq} \right)$ to an equivalent complement-free form, where the $\preceq$ relation is still satisfyable.

**Case** $\top \subseteq \tau_1^{\mathrm{n}} \vee \tau_2^{\mathrm{dn}}$**.** By Lemma A.45, we have $V^D \subseteq \pi$ for some $V^D$ and $D \in \{ \top, \not\perp \}$. Then we can pick $\pi'_1 = \pi$ and $V_1^{D_1} = V^D$, which indeed satisfies $\pi \cong \pi_1 \vee V_1^{D_1}$ and $U_k^{C_k} \preceq V_1^{D_1}$.

**Case** $C_k = \cancel{B}$**.** If $C_k = \cancel{B}$ for some $B$, then we proceed symmetrically to the case above on the negation-inversion of $U_k^{C_k} \subseteq \tau_1^{\mathrm{n}} \vee \tau_2^{\mathrm{dn}}$, i.e., $\tau'^{\mathrm{n}}_1 \vee \tau'^{\mathrm{cn}}_2 \subseteq X_k^{\cancel{C_k}}$ for some $\tau'^{\mathrm{n}}_1$ and $\tau'^{\mathrm{cn}}_2$ and $X_k^{\cancel{C_k}}$, and finally apply negation-inversion again to obtain the desired result.

**Case S-AndOr2⊃.** Then $\pi = \bigwedge_h \pi_h^{\mathrm{dn}}$ for some $\overline{\pi_h^{\mathrm{dn}}}^h$. The premises are $\overline{\tau \leqslant \pi_h^{\mathrm{dn}}}^h$. By IH on each premise, we have $\pi_h^{\mathrm{dn}} \cong \bigwedge_{p_h} \left( \pi_{p_h}^h \vee V^h {}^{D^h_{p_h}}_{p_h} \right)$ and $\overline{\triangleright\Sigma \vdash U^{C_{k^h_{p_h}}}_{k^h_{p_h}} \preceq V^h {}^{D^h_{p_h}}_{p_h}}^{p_h}$ for some $\overline{\pi^h_{p_h}}^{p_h}$ and $\overline{V^h {}^{D^h_{p_h}}_{p_h}}^{p_h}$ and $\overline{k^h_{p_h}}^{p_h}$. Then we have $\pi \cong \bigwedge_h \bigwedge_{p_h} \left( \pi^h_{p_h} \vee V^h {}^{D^h_{p_h}}_{p_h} \right)$.

**Cases S-DistribCdn⋄.** Similar to case S-AndOr2⋅.

**Case S-RcdDepth.** Then $\tau = \{ x : \tau_1 \}$ and $\pi = \{ x : \pi_1 \}$ for some $\tau_1$ and $\pi_1$. From the assumption, we have:

$$\bigwedge_i \left( \tau_i' \vee U_i^{C_i} \right) \subseteq \tau = \{ x : \tau_1 \} \tag{23}$$

By S-Trans with Lemma A.7⊃ on S-AndOr12⋅, (23) implies:

$$\bigwedge_i U_i^{C_i} \subseteq \{ x : \tau_1 \} \tag{24}$$

By Lemma A.57, (24) implies:

$$U_k^{C_k} \subseteq \{ x : \tau_1 \} \tag{25}$$

for some $k$. By Lemma 4.10, (25) implies:

$$U_k^{C_k} = \bigvee_l \{ x : \tau_1 \} \tag{26}$$

The premise of the rule is:

$$\triangleright\Sigma \vdash \tau_1 \leqslant \pi_1 \tag{27}$$

By the definition of $\preceq$, (27) implies:

$$\triangleright\Sigma \vdash \{ x : \tau_1 \} \preceq \{ x : \pi_1 \}$$
$$\text{i.e.,} \quad \triangleright\Sigma \vdash U_k^{C_k} \preceq \{ x : \pi_1 \} \tag{28}$$

So we can pick $\pi_1' = \bot$ and $V_1^{D_1} = \{ x : \pi_1 \}$, which indeed yields $\pi = \{ x : \pi_1 \} \cong \pi_1' \vee V_1^{D_1}$.

**Case S-RcdMrg⋅.** Then $\tau = \{ x : \tau_1 \vee \tau_2 \}$ and $\pi = \{ x : \tau_1 \} \vee \{ x : \tau_2 \}$ for some $\tau_1$ and $\tau_2$. From the assumption, we have:

$$\bigwedge_i \left( \tau_i' \vee U_i^{C_i} \right) \subseteq \tau = \{ x : \tau_1 \vee \tau_2 \} \tag{29}$$

By S-Trans with Lemma A.7⊃ on S-AndOr12⋅, (29) implies:

$$\bigwedge_i U_i^{C_i} \subseteq \{ x : \tau_1 \vee \tau_2 \} \tag{30}$$

By Lemma A.57, (30) implies:

$$U_k^{C_k} \subseteq \{ x : \tau_1 \vee \tau_2 \} \tag{31}$$

for some $k$. By Lemma 4.10, (31) implies:

$$U_k^{C_k} = \bigvee_l \{ x : \tau_1 \vee \tau_2 \} \tag{32}$$

Pick $\pi'_1 = \bot$ and $V_1^{D_1} = \{ x : \tau_1 \} \vee \{ x : \tau_2 \}$, which indeed satisfies $\pi = \{ x : \tau_1 \} \vee \{ x : \tau_2 \} \cong \pi'_1 \vee V_1^{D_1}$ and $U_k^{C_k} \preceq V_1^{D_1}$.

**Case S-RcdMrg⤾.** Then $\tau = \{ x : \tau_1 \} \wedge \{ x : \tau_2 \}$ and $\pi = \{ x : \tau_1 \wedge \tau_2 \}$ for some $\tau_1$ and $\tau_2$. From the assumption, we have:

$$\bigwedge_i \left( \tau'_i \vee U_i^{C_i} \right) \subseteq \tau = \{ x : \tau_1 \} \wedge \{ x : \tau_2 \} \tag{33}$$

By S-Trans with Lemma A.7⤾ on S-AndOr12·, (33) implies:

$$\bigwedge_i U_i^{C_i} \subseteq \{ x : \tau_1 \} \wedge \{ x : \tau_2 \} \tag{34}$$

Let $l$ range from 1 to 2. By Lemma A.44, (34) implies:

$$\overline{\bigwedge_i U_i^{C_i} \subseteq \{ x : \tau_l \}}^{l} \tag{35}$$

By Lemma A.57, (35) implies:

$$\overline{U_{k_l}^{C_{k_l}} \subseteq \{ x : \tau_l \}}^{l} \tag{36}$$

for some $\overline{k_l}^{l}$. By Lemma 4.10, (36) implies:

$$\overline{U_{k_l}^{C_{k_l}} = \bigvee_{l_l} \{ x : \tau_l \}}^{l} \tag{37}$$

Pick $\pi'_1 = \bot$ and $V_1^{D_1} = \{ x : \tau_1 \wedge \tau_2 \}$, which indeed satisfies $\pi = \{ x : \tau_1 \wedge \tau_2 \} \cong \pi'_1 \vee V_1^{D_1}$ and $\bigwedge_l U_{k_l}^{C_{k_l}} \preceq V_1^{D_1}$.

**Case S-RcdTop.** Then $\tau = \top$ and $\pi = \{ x : \pi_1 \} \vee \pi_0$, where $\pi_0 \in \{ \{ y^{\neq x} : \tau_2 \},\ \tau_2 \to \tau_3 \}$. Pick $\pi'_1 = \bot$ and $D_1 = \top$ and $V_1^{D_1} = \{ x : \pi_1 \} \vee \pi_0$, which indeed satisfies $\pi = \{ x : \pi_1 \} \vee \pi_0 \cong \pi'_1 \vee V_1^{D_1}$ and $\overline{U_i^{C_i} \preceq V_1^{D_1}}^{i}$.

**Case S-FunDepth.** Then $\tau = \tau_1 \to \tau_2$ and $\pi = \tau_0 \to \tau_3$ for some $\overline{\tau_l}^{l \in 0..3}$. From the assumption, we have:

$$\bigwedge_i \left( \tau'_i \vee U_i^{C_i} \right) \subseteq \tau = \tau_1 \to \tau_2 \tag{38}$$

By S-Trans with Lemma A.7⤾ on S-AndOr12·, (38) implies:

$$\bigwedge_i U_i^{C_i} \subseteq \tau_1 \to \tau_2 \tag{39}$$

By Lemma A.57, (39) implies:

$$U_k^{C_k} \subseteq \tau_1 \to \tau_2 \tag{40}$$

for some $k$. By Lemma 4.10, (40) implies:

$$U_k^{C_k} = \bigvee_l \tau_1 \to \tau_2 \tag{41}$$

The premises of the rule are:

$$\triangleright \Sigma \vdash \tau_0 \leqslant \tau_1 \tag{42}$$

$$\triangleright \Sigma \vdash \tau_2 \leqslant \tau_3 \tag{43}$$

By the definition of $\preceq$, (42) and (43) imply:

$$\triangleright\Sigma \vdash \tau_1 \to \tau_2 \preceq \tau_0 \to \tau_3$$
$$\text{i.e.,} \quad \triangleright\Sigma \vdash U_k^{C_k} \preceq \tau_0 \to \tau_3 \tag{44}$$

So we can pick $\pi_1' = \bot$ and $V_1^{D_1} = \tau_0 \to \tau_3$, which indeed yields $\pi = \tau_0 \to \tau_3 \cong \pi_1' \vee V_1^{D_1}$.

**Case S-FunMrg·.** Then $\tau = \tau_{11} \to \tau_{12} \wedge \tau_{21} \to \tau_{22}$ and $\pi = (\tau_{11} \vee \tau_{21}) \to (\tau_{12} \wedge \tau_{22})$ for some $\tau_{11}$, $\tau_{12}$, $\tau_{21}$, and $\tau_{22}$. From the assumption, we have:

$$\bigwedge_i \left(\tau_i' \vee U_i^{C_i}\right) \subseteq \tau = \tau_{11} \to \tau_{12} \wedge \tau_{21} \to \tau_{22} \tag{45}$$

By S-Trans with Lemma A.7↷ on S-AndOr12·, (45) implies:

$$\bigwedge_i U_i^{C_i} \subseteq \tau_{11} \to \tau_{12} \wedge \tau_{21} \to \tau_{22} \tag{46}$$

Let $l$ range from 1 to 2. By Lemma A.44, (46) implies:

$$\overline{\bigwedge_i U_i^{C_i} \subseteq \tau_{l1} \to \tau_{l2}}^{\,l} \tag{47}$$

By Lemma A.57, (47) implies:

$$\overline{U_{k_l}^{C_{k_l}} \subseteq \tau_{l1} \to \tau_{l2}}^{\,l} \tag{48}$$

for some $\overline{k_l}^{l}$. By Lemma 4.10, (48) implies:

$$\overline{U_{k_l}^{C_{k_l}} = \bigvee_{l_l} \tau_{l1} \to \tau_{l2}}^{\,l} \tag{49}$$

Pick $\pi_1' = \bot$ and $V_1^{D_1} = (\tau_{11} \vee \tau_{21}) \to (\tau_{12} \wedge \tau_{22})$, which indeed satisfies $\pi = (\tau_{11} \vee \tau_{21}) \to (\tau_{12} \wedge \tau_{22}) \cong \pi_1' \vee V_1^{D_1}$ and $\bigwedge_l U_{k_l}^{C_{k_l}} \preceq V_1^{D_1}$.

**Case S-FunMrg↷.** Then $\tau = (\tau_1 \wedge \tau_3) \to (\tau_2 \vee \tau_4)$ and $\pi = \tau_1 \to \tau_2 \vee \tau_3 \to \tau_4$ for some $\overline{\tau_l}^{\,l \in 1..4}$. From the assumption, we have:

$$\bigwedge_i \left(\tau_i' \vee U_i^{C_i}\right) \subseteq \tau = (\tau_1 \wedge \tau_3) \to (\tau_2 \vee \tau_4) \tag{50}$$

By S-Trans with Lemma A.7↷ on S-AndOr12·, (50) implies:

$$\bigwedge_i U_i^{C_i} \subseteq (\tau_1 \wedge \tau_3) \to (\tau_2 \vee \tau_4) \tag{51}$$

By Lemma A.57, (51) implies:

$$U_k^{C_k} \subseteq (\tau_1 \wedge \tau_3) \to (\tau_2 \vee \tau_4) \tag{52}$$

for some $k$. By Lemma 4.10, (52) implies:

$$U_k^{C_k} = \bigvee_l (\tau_1 \wedge \tau_3) \to (\tau_2 \vee \tau_4) \tag{53}$$

Pick $\pi_1' = \bot$ and $V_1^{D_1} = \tau_1 \to \tau_2 \vee \tau_3 \to \tau_4$, which indeed satisfies $\pi = \tau_1 \to \tau_2 \vee \tau_3 \to \tau_4 \cong \pi_1' \vee V_1^{D_1}$ and $U_k^{C_k} \preceq V_1^{D_1}$.

**Case S-ClsSub.** Then $\tau = \#C_1$ and $\pi = \#C_2$ for some $\#C_1$ and $\#C_2$. From the assumption, we have:

$$\bigwedge_i \left(\tau_i' \vee U_i^{C_i}\right) \subseteq \tau = \#C_1 \tag{54}$$

By S-Trans with Lemma A.7↷ on S-AndOr12·, (54) implies:

$$\bigwedge_i U_i^{C_i} \subseteq \#C_1 \tag{55}$$

By Lemma A.57, (55) implies:

$$U_k^{C_k} \subseteq \#C_1 \tag{56}$$

By Lemma 4.10, (56) implies:

$$U_k^{C_k} = \bigvee_l \#C_1 \tag{57}$$

The premise of the rule is:

$$C_2 \in \mathcal{S}(\#C_1) \tag{58}$$

By the definition of $\preceq$, (58) implies:

$$\begin{aligned} &\#C_1 \preceq \#C_2 \\ \text{i.e.,}\quad &U_k^{C_k} \preceq \#C_2 \end{aligned} \tag{59}$$

So we can pick $\pi_1' = \bot$ and $V_1^{D_1} = \#C_2$, which indeed yields $\pi = \#C_2 \cong \pi_1' \vee V_1^{D_1}$.

**Case S-ClsBot.** Then $\tau = \#C_1 \wedge \#C_2$ and $\pi = \bot$ for some $\#C_1$ and $\#C_2$. From the assumption, we have:

$$\bigwedge_i \left(\tau_i' \vee U_i^{C_i}\right) \subseteq \tau = \#C_1 \wedge \#C_2 \tag{60}$$

By S-Trans with Lemma A.7↷ on S-AndOr12·, (60) implies:

$$\bigwedge_i U_i^{C_i} \subseteq \#C_1 \wedge \#C_2 \tag{61}$$

Let $l$ range from 1 to 2. By Lemma A.44, (61) implies:

$$\overline{\bigwedge_i U_i^{C_i} \subseteq \#C_l}^{\,l} \tag{62}$$

By Lemma A.57, (62) implies:

$$\overline{U_{k_l}^{C_{k_l}} \subseteq \#C_l}^{\,l} \tag{63}$$

for some $\overline{k_l}^{\,l}$. By Lemma 4.10, (63) implies:

$$\overline{U_{k_l}^{C_{k_l}} = \bigvee_{k_l} \#C_l}^{\,l} \tag{64}$$

Then (64) implies:

$$\overline{C_{k_l} = \#C_l}^{\,l} \tag{65}$$

The premises of the rule are:

$$C_1 \notin \mathcal{S}(\#C_2) \tag{66}$$
$$C_2 \notin \mathcal{S}(\#C_1) \tag{67}$$

which is impossible by the condition on $\overline{C_i}^i$.

■

## B Formalization of MLstruct, Continued

We now give the full details of MLstruct's formalization.

### *B.1 Declarative Typing Rules*

The declarative typing rules of $\lambda^{\neg}$ are presented in Figure 25.

Rule T-Body is used to type programs that happen to be simple terms, after having accumulated a set of declarations in the context $\mathcal{D}$, which is checked for well-formedness using the rules presented in Figure 26 and explained later (Section B.2).

In T-Def, we type the body of a **def** inside a constraining context $\Xi$ added on top of the current declarations context, and subsequently use $\Xi$ as part of the resulting polymorphic type of this **def**, which is placed into the typing context for use later in the program. Importantly, $\Xi$ has to be checked for consistency, which is done with the $\Xi$ ***cons.*** judgement, defined in Figure 25 — essentially, this makes sure that there is at least one assignment of variable that makes the constraints hold in the base declarations context. This is to forbid the use of inconsistent bounds on type variables, such as $(\mathrm{Bool} \leqslant \alpha) \cdot (\alpha \leqslant \mathrm{Int})$, which could lead to accepting ill-typed definitions.

As a concrete example for T-Def, consider a definition such as **def** $f = \lambda x.\ x + 1$ in a program where a type synonym **type** $A = \mathrm{Int}$ is defined. One hypothetical judgement used to type this definition could be '$(\textbf{type}\ A = \mathrm{Int}) \cdot (\alpha \leqslant A), \Gamma \vdash \lambda x.\ x + 1 : \alpha \to \mathrm{Int}$' where $\Xi = (\alpha \leqslant A)$ is the constraints part of the context. According to T-Def, because $\Xi$ is consistent (since $lb_\Xi(\alpha) = \bot \leqslant ub_\Xi(\alpha) = \mathrm{Int}$), we can type the definition $f$ as '$\forall (\alpha \leqslant A).\ \alpha \to \mathrm{Int}$'. As a side note, this type can be rewritten to $f : A \to A$, which is equivalent in the declarations context $(\textbf{type}\ A = \mathrm{Int})$.

Rule T-Var2 is an interesting counterpart to rule T-Def explained above. It *instantiates* a given polymorphic type through the $\leqslant^{\forall}$ relation defined by rule S-All.

Rule S-All uses a substitution $\rho$, a premise that the subtyping holds under this substitution, and the *entailment* judgement $\Sigma \cdot \Xi' \models \rho(\Xi)$, which simply makes sure that every subtyping constraint in $\rho(\Xi)$ holds in $\Sigma$ with $\Xi'$ (which is $\epsilon$ for T-Var2). Condition $dom(\rho) = TV(\Xi) \cup TV(\tau)$, where $TV(\cdot)$ is defined in Section B.3, is used to make sure that $\rho$ assigns a substitution to all the variables quantified by the polymorphic type.

$\boxed{\Xi, \Gamma \vdash^\star P : \tau}$

$$\textsc{T-Body}\ \frac{\Xi\ \textbf{\textit{cons.}} \quad \Xi, \Gamma \vdash t : \tau}{\Xi, \Gamma \vdash^\star t : \tau} \qquad \textsc{T-Def}\ \frac{\Xi'\ \textbf{\textit{cons.}} \quad \Xi', \Gamma \vdash t : \tau \quad \Xi, \Gamma \cdot (x : \forall \Xi'.\ \tau) \vdash^\star P : \tau_P}{\Xi, \Gamma \vdash^\star \textbf{def}\ x = t\ ;\ P : \tau_P}$$

$\boxed{\Xi, \Gamma \vdash t : \tau}$

$$\textsc{T-Subs}\ \frac{\Xi, \Gamma \vdash t : \tau_1 \quad \Xi \vdash \tau_1 \leqslant \tau_2}{\Xi, \Gamma \vdash t : \tau_2} \qquad \textsc{T-Obj}\ \frac{\Xi, \Gamma \vdash t : \tau \quad C\ \textbf{\textit{final}}}{\Xi, \Gamma \vdash C\ \{\, \overline{x = t}\, \} : \#C \wedge \{\, \overline{x : \tau}\, \}}$$

$$\textsc{T-Proj}\ \frac{\Xi, \Gamma \vdash t : \{\, x : \tau\, \}}{\Xi, \Gamma \vdash t.x : \tau} \qquad \textsc{T-Var1}\ \frac{\Gamma(x) = \tau}{\Xi, \Gamma \vdash x : \tau} \qquad \textsc{T-Var2}\ \frac{\Gamma(x) = \sigma \quad \Xi \vdash \sigma \leqslant^\forall \forall \epsilon.\ \tau}{\Xi, \Gamma \vdash x : \tau} \qquad \textsc{T-Abs}\ \frac{\Xi, \Gamma \cdot (x : \tau_1) \vdash t : \tau_2}{\Xi, \Gamma \vdash \lambda x.\ t : \tau_1 \to \tau_2}$$

$$\textsc{T-App}\ \frac{\Xi, \Gamma \vdash t_0 : \tau_1 \to \tau_2 \quad \Xi, \Gamma \vdash t_1 : \tau_1}{\Xi, \Gamma \vdash t_0\ t_1 : \tau_2} \qquad \textsc{T-Asc}\ \frac{\Xi, \Gamma \vdash t : \tau}{\Xi, \Gamma \vdash (t : \tau) : \tau}$$

$$\textsc{T-Case1}\ \frac{\Xi, \Gamma \vdash t_1 : \bot}{\Xi, \Gamma \vdash \textbf{case}\ x = t_1\ \textbf{of}\ \epsilon : \bot} \qquad \textsc{T-Case2}\ \frac{\Xi, \Gamma \vdash t_1 : \tau_1 \wedge \#C \quad \Xi, \Gamma \cdot (x : \tau_1) \vdash t_2 : \tau}{\Xi, \Gamma \vdash \textbf{case}\ x = t_1\ \textbf{of}\ \_ \to t_2 : \tau}$$

$$\textsc{T-Case3}\ \frac{\Xi, \Gamma \vdash t_1 : \#C \wedge \tau_1 \vee \neg \#C \wedge \tau_2 \quad \Xi, \Gamma \cdot (x : \tau_1) \vdash t_2 : \tau \quad \Xi, \Gamma \cdot (x : \tau_2) \vdash \textbf{case}\ x = x\ \textbf{of}\ M : \tau}{\Xi, \Gamma \vdash \textbf{case}\ x = t_1\ \textbf{of}\ C \to t_2,\ M : \tau}$$

$\boxed{\Sigma \vdash \triangleright\Xi \cdot \Xi\ ;\ \rho\ \textbf{\textit{cons.}}}$

*Assuming $\Sigma$ holds, then bounds $\triangleright\Xi \cdot \Xi$ are consistent, as witnessed by $\rho$.*

$$\Xi\ \textbf{\textit{cons.}} \equiv \exists \rho.\ \epsilon \vdash \Xi\ ;\ \rho\ \textbf{\textit{cons.}}$$

$$\frac{}{\Sigma \vdash \triangleright\Xi\ ;\ [\,]\ \textbf{\textit{cons.}}} \qquad \frac{\begin{array}{c} \mathit{split}_\alpha(\Xi,\ \mathit{dom}(\rho')) = (\Xi_\alpha,\ \Xi_{\not\alpha}) \quad \rho = [\alpha \mapsto \alpha \wedge \mathit{ub}_\Xi(\alpha) \vee \mathit{lb}_\Xi(\alpha)] \\ \triangleright\Xi_\triangleright \cdot \triangleright\Xi_\alpha \cdot \rho\Xi_{\not\alpha} \cdot \rho\Sigma \vDash \rho\Xi_\alpha \quad \rho\Sigma \vdash \triangleright\Xi_\triangleright \cdot \triangleright\Xi_\alpha \cdot \rho\Xi_{\not\alpha}\ ;\ \rho'\ \textbf{\textit{cons.}} \end{array}}{\Sigma \vdash \triangleright\Xi_\triangleright \cdot \Xi\ ;\ \rho' \circ \rho\ \textbf{\textit{cons.}}}$$

$$\mathit{split}_\alpha(\Xi,\ \{\, \overline{\gamma}\, \}) =$$

$$\Big(\overline{(\tau \leqslant \pi)}^{(\tau \leqslant \pi) \in \Xi \mid \alpha \in \{\, \tau,\, \pi\, \}},\ \overline{(\tau \leqslant \pi)}^{(\tau \leqslant \pi) \in \Xi \mid \alpha \notin \{\, \tau,\, \pi\, \}} \cdot \overline{(\alpha \leqslant^\diamond \beta)}^{(\alpha \leqslant^\diamond \beta) \in \Xi \mid \beta \in \{\, \overline{\gamma}\, \}}\Big)$$

$\boxed{\Sigma \vdash \sigma \leqslant^\forall \sigma}$

$$\textsc{S-All}\ \frac{\Xi' \cdot \Sigma \vDash \rho(\Xi) \quad \Xi' \cdot \Sigma \vdash \rho(\tau) \leqslant \tau' \quad \mathit{dom}(\rho) = \mathit{TV}(\Xi) \cup \mathit{TV}(\tau)}{\Sigma \vdash \forall \Xi.\ \tau \leqslant^\forall \forall \Xi'.\ \tau'}$$

$\boxed{\Sigma \vDash \Sigma}$

$$\textsc{S-Empty}\ \frac{}{\Sigma \vDash \epsilon} \qquad \textsc{S-Cons}\ \frac{\Sigma \vDash \Sigma' \quad \Sigma \vdash \tau_1 \leqslant \tau_2}{\Sigma \vDash \Sigma' \cdot (\tau_1 \leqslant \tau_2)} \qquad \textsc{S-Cons}\triangleright\ \frac{\Sigma \vDash \Sigma' \quad \triangleleft\Sigma \vdash \tau_1 \leqslant \tau_2}{\Sigma \vDash \Sigma' \cdot \triangleright(\tau_1 \leqslant \tau_2)}$$

Fig. 25. Full declarative typing, consistency, and subtyping entailment rules.

### B.1.1 Superclasses

**Definition B.1** (Superclasses)**.** *We define the superclasses $\mathcal{S}(\tau)$ of a type $\tau$ as the set of classes transitively inherited by type $\tau$, assuming $\tau$ is a class type or the expansion of a class type:*

$$\frac{}{C \in \mathcal{S}(\#C)} \qquad \frac{C \in \mathcal{S}(\#D)}{C \in \mathcal{S}(D[\overline{\tau}])} \qquad \frac{\tau \ \textbf{\textit{exp.}}\ \tau' \qquad C \in \mathcal{S}(\tau')}{C \in \mathcal{S}(\tau)} \qquad \frac{C \ \in \ \mathcal{S}(\tau_1) \cup \mathcal{S}(\tau_2)}{C \in \mathcal{S}(\tau_1 \wedge \tau_2)}$$

### B.1.2 Substitution

**Definition B.2** (Term substitution)**.** *A term substitution is a pair of variable and term $[x \mapsto t]$. Applying a term substitution to a term $t'$, denoted by $[x \mapsto t]t'$, replaces all free occurrences of $x$ in $t'$ with $t$, which is defined as follows:*

$$[x \mapsto t]y = \begin{cases} t & \text{if } y = x \\ y & \text{if } y \neq x \end{cases} \qquad [x \mapsto t](t_0\ t_1) = [x \mapsto t]t_0\ [x \mapsto t]t_1$$

$$[x \mapsto t](t' : \tau) = [x \mapsto t]t' : \tau \qquad [x \mapsto t]t'.x' = ([x \mapsto t]t').x'$$

$$[x \mapsto t]\lambda x'.\ t' = \begin{cases} \lambda x'.\ t' & \text{if } x' = x \\ \lambda x'.\ [x \mapsto t]t' & \text{if } x' \neq x \end{cases} \qquad [x \mapsto t](C\ \{\ \overline{x' = t'}\ \}) = C\ \{\ \overline{x' = [x \mapsto t]t'}\ \}$$

$$[x \mapsto t]\ \textbf{\textit{case}}\ x' = t'\ \textbf{\textit{of}}\ M = \begin{cases} \textbf{\textit{case}}\ x' = [x \mapsto t]t'\ \textbf{\textit{of}}\ M & \text{if } x' = x \\ \textbf{\textit{case}}\ x' = [x \mapsto t]t'\ \textbf{\textit{of}}\ [x \mapsto t]M & \text{if } x' \neq x \end{cases}$$

*Where case branches term substitution $[x \mapsto t]M$ is defined as:*

$$[x \mapsto t]\ \epsilon = \epsilon \qquad [x \mapsto t](\_ \to t') = \_ \to [x \mapsto t]t'$$

$$[x \mapsto t](C \to t',\ M) = C \to [x \mapsto t]t',\ [x \mapsto t]M$$

*Similarly, applying a term substitution to a program $P$, denoted by $[x \mapsto t]P$, replaces all free occurrences of $x$ in $P$ with $t$, which is defined as follows:*

$$[x \mapsto t]\ (\textbf{\textit{def}}\ x' = t';\ P) = \begin{cases} \textbf{\textit{def}}\ x' = t';\ P & \text{if } x' = x \\ \textbf{\textit{def}}\ x' = [x \mapsto t]t';\ [x \mapsto t]P & \text{if } x' \neq x \end{cases}$$

**Definition B.3** (Type substitution)**.** *A type substitution $\rho = \{\ \overline{\alpha \mapsto \tau}\ \}$ is a mapping from type variables to types.*

*We use the notation $(\alpha_1 \mapsto \tau_1) \in \rho$ to signify that $\alpha_1 \in dom(\rho)$ and $\rho(\alpha_1) = \tau_1$.*

*$dom(\rho)$ is the domain of $\rho$, defined as follows:*

$$dom(\{\ \}) = \varnothing \qquad dom(\{\ \overline{\alpha \mapsto \tau}, \alpha' \mapsto \tau'\ \}) = dom(\{\ \overline{\alpha \mapsto \tau}\ \}) \cup \{\ \alpha'\ \}$$

**Definition B.4** (Type substitution on type)**.** *Application of a type substitution to a type $\rho(\tau)$ is defined as follows:*

$$\rho(\tau_1 \to \tau_2) = \rho(\tau_1) \to \rho(\tau_2) \qquad \rho(\alpha) = \begin{cases} \tau & \text{if } (\alpha \mapsto \tau) \in \rho \\ \alpha & \text{if } \alpha \notin dom(\rho) \end{cases}$$

$$\rho(\{\ x : \tau\ \}) = \{\ x : \rho(\tau)\ \} \qquad \rho(\top^{\diamond}) = \top^{\diamond}$$

$$\rho(N[\overline{\tau}]) = N[\overline{\rho(\tau)}] \qquad \rho(\tau_1 \vee^{\diamond} \tau_2) = \rho(\tau_1) \vee^{\diamond} \rho(\tau_2)$$

$$\rho(\#C) = \#C \qquad \rho(\neg\tau) = \neg\rho(\tau)$$

**Definition B.5** (Type substitution on term)**.** *Application of a type substitution to a term $\rho(t)$ is defined as follows:*

$$\begin{aligned} \rho(x) &= x & \rho(t.x) &= \rho(t).x \\ \rho(t : \tau) &= \rho(t) : \rho(\tau) & \rho(C\ \{\ \overline{x = t}\ \}) &= C\ \{\ \overline{x = \rho(t)}\ \} \\ \rho(\lambda x.\ t) &= \lambda x.\ \rho(t) & \rho(\textbf{\textit{case}}\ x = t\ \textbf{\textit{of}}\ M) &= \textbf{\textit{case}}\ x = \rho(t)\ \textbf{\textit{of}}\ \rho(M) \\ \rho(t_0\ t_1) &= \rho(t_0)\ \rho(t_1) \end{aligned}$$

*Where type substitution $\rho(M)$ on case branches is defined as:*

$$\rho(\epsilon) = \epsilon \qquad \rho(\_ \to t) = \_ \to \rho(t) \qquad \rho(C \to t,\ M) = C \to \rho(t),\ \rho(M)$$

**Definition B.6** (Type substitution on typing context)**.** *Application of a type substitution to a typing context $\rho(\Gamma)$ is defined as follows:*

$$\rho(\epsilon) = \epsilon \qquad \rho(\Gamma{\cdot}(x : \tau)) = \rho(\Gamma){\cdot}(x : \rho(\tau)) \qquad \rho(\Gamma{\cdot}(x : \sigma)) = \rho(\Gamma){\cdot}(x : \sigma)$$

**Definition B.7** (Type substitution on subtyping context)**.** *Application of a type substitution to a subtyping context $\rho(\Sigma)$ is defined as follows:*

$$\rho(\epsilon) = \epsilon \qquad \rho(\Sigma \cdot (\tau_1 \leqslant \tau_2)) = \rho(\Sigma) \cdot (\rho(\tau_1) \leqslant \rho(\tau_2))$$

$$\rho(\Sigma \cdot \triangleright(\tau_1 \leqslant \tau_2)) = \rho(\Sigma) \cdot \triangleright(\rho(\tau_1) \leqslant \rho(\tau_2))$$

### *B.2 Well-Formedness*

The well-formedness rules are presented in Figure 26. They ensure that the declarations of a program lead to a decidable type inference algorithm by restricting the shapes of recursive types to regular trees. This is done by making sure that all recursive occurrences of class and type declarations are given the *same type arguments* $\overline{\alpha}$ as the declaration's head $N[\overline{\alpha}]$ itself. Note that well-formed type declaration may refer to each other freely, possibly forming mutually-recursive definitions.

**Definition B.8** (Occurrences)**.** *We define the occurrences of a type $\tau$, written $occs(\tau)$, as all the types transitively reachable by progressively traversing the subterms of $\tau$ and expanding the alias and class types as we encounter them. This is always a finite set, thanks to the regularity check (Section 2.3.1).*

The *type variables* of a piece of syntax $s$, written $TV(s)$, is defined in Section B.3. Function $guard_N(\tau)$ refers to the guardedness check described in Section 2.1.6.

**Theorem B.9** (Regularity)**.** *If $\mathcal{D}$ **wf**, then for all $\tau$, the set $occs(\tau)$ is finite.*

This notably means that given well-formed declarations $\mathcal{D}$, we can easily compute $\mathcal{S}(\tau)$.

**Proof** [Proof B.9 (Regularity)] Since each type constructor declared as $N[\overline{\alpha}]$ can only appear in its body (and transitively in the bodies of other declarations) with the same type variables $\overline{\alpha}$ as type arguments, the expansion $\tau$ of a type $N[\overline{\pi}]$ may only lead to $N$ occurrences of the form $N[\overline{\pi}]$, which itself has the same occurrences as $\tau$; thus the number of distinct type occurrences transitively reachable from a given declaration is finite. ∎

$\boxed{\mathcal{D}\ \mathit{wf}}$

$$\text{W-DECLS}\ \frac{\overline{\mathcal{D} \vdash d\ \mathit{wf}}^{\,d \in \mathcal{D}} \quad TV(\mathcal{D}) = \varnothing}{\mathcal{D}\ \mathit{wf}}$$

$\boxed{\mathcal{D} \vdash d\ \mathit{wf}}$

$$\text{W-ALS}\ \frac{\overline{\overline{\tau} = \overline{\alpha}}^{\,A[\overline{\tau}] \in \mathit{occs}(\pi)} \quad \mathit{guard}_A(\pi) \quad \pi\ \mathit{wf}}{\mathcal{D} \vdash \textbf{type}\ A[\overline{\alpha}] = \pi\ \mathit{wf}}$$

$$\text{W-CLS1}\ \frac{\overline{\overline{\tau} = \overline{\alpha}}^{\,C[\overline{\tau}] \in \mathit{occs}(\{\,\overline{x:\tau}\,\})} \quad \overline{\tau\ \mathit{wf}}}{\mathcal{D} \vdash \textbf{class}\ C[\overline{\alpha}] : \{\,\overline{x:\tau}\,\}\ \mathit{wf}}$$

$$\text{W-CLS2}\ \frac{C \notin \mathcal{S}(C[\overline{\alpha}]) \quad \mathit{guard}_C(D[\overline{\tau_i}] \wedge \{\,\overline{x:\tau}\,\}) \quad \overline{\overline{\tau} = \overline{\alpha}}^{\,C[\overline{\tau}] \in \mathit{occs}(D[\overline{\tau_i}] \wedge \{\,\overline{x:\tau}\,\})} \quad \overline{\tau\ \mathit{wf}}}{\mathcal{D} \vdash \textbf{class}\ C[\overline{\alpha}] : D[\overline{\tau_i}] \wedge \{\,\overline{x:\tau}\,\}\ \mathit{wf}}$$

$\boxed{\tau\ \mathit{wf}}$

$$\frac{}{\top^\diamond\ \mathit{wf}} \qquad \frac{}{\alpha\ \mathit{wf}} \qquad \frac{\tau\ \mathit{wf}}{\neg\tau\ \mathit{wf}} \qquad \frac{\tau_1\ \mathit{wf} \quad \tau_2\ \mathit{wf}}{\tau_1 \vee^\diamond \tau_2\ \mathit{wf}} \qquad \frac{\tau_1\ \mathit{wf} \quad \tau_2\ \mathit{wf}}{\tau_1 \to \tau_2\ \mathit{wf}}$$

$$\frac{\tau\ \mathit{wf}}{\{\,x:\tau\,\}\ \mathit{wf}} \qquad \frac{}{\#C\ \mathit{wf}} \qquad \frac{\overline{\tau\ \mathit{wf}} \quad N[\overline{\tau}]\ \textbf{exp.}\ \pi}{N[\overline{\tau}]\ \mathit{wf}}$$

$\boxed{(\Sigma \mid \Xi)\ \mathit{wf}}$

$$\frac{}{\epsilon\ \mathit{wf}} \qquad \frac{\Sigma\ \mathit{wf} \quad \tau_1\ \mathit{wf} \quad \tau_2\ \mathit{wf}}{\Sigma{\cdot}(\triangleright)(\tau_1 \leqslant \tau_2)}$$

$\boxed{\mathcal{D} \vdash C\ \textbf{final}}$

$$\frac{}{\epsilon \vdash C\ \textbf{final}} \qquad \frac{\mathcal{D} \vdash C\ \textbf{final} \quad D \neq C}{\mathcal{D}{\cdot}(\textbf{class}\ C'[\overline{\alpha}] : D[\overline{\tau}]) \vdash C\ \textbf{final}}$$

$$\frac{\mathcal{D} \vdash C\ \textbf{final} \quad D \neq C}{\mathcal{D}{\cdot}(\textbf{class}\ C'[\overline{\alpha}] : D[\overline{\tau}] \wedge \{\,\overline{x:\pi}\,\}) \vdash C\ \textbf{final}}$$

Fig. 26. Well-formedness and finality rules.

### *B.3 Free type variables*

**Definition B.10** (Free type variables)**.** *The set of free type variables of a type $\tau$, written $TV(\tau)$, is defined as:*

$$\begin{aligned} TV(\tau_1 \to \tau_2) &= TV(\tau_1) \cup TV(\tau_2) & TV(\alpha) &= \{\alpha\} \\ TV(\{\,x:\tau\,\}) &= TV(\tau) & TV(\top^\diamond) &= \varnothing \\ TV(\#C) &= \varnothing & TV(\tau_1 \vee^\diamond \tau_2) &= TV(\tau_1) \cup TV(\tau_2) \\ TV(N[\overline{\tau}]) &= \textstyle\bigcup_{\overline{\tau}} TV(\tau) & TV(\neg\tau) &= TV(\tau) \end{aligned}$$

**Definition B.11** (Free type variables of declaration context)**.** *The free type variables of a declaration context $TV(\mathcal{D})$ is defined as:*

$$TV(\epsilon) = \varnothing \qquad TV(\mathcal{D}\cdot(\textbf{\textit{class}}\ C[\overline{\alpha}] : \tau)) = TV(\mathcal{D}) \cup (TV(\tau)\backslash\{\,\overline{\alpha}\,\})$$

$$TV(\mathcal{D}\cdot(\textbf{\textit{type}}\ A[\overline{\alpha}] = \tau)) = TV(\mathcal{D}) \cup (TV(\tau)\backslash\{\,\overline{\alpha}\,\})$$

**Definition B.12** (Free type variables of typing context)**.** *The free type variables of a typing context $TV(\Gamma)$ is defined as:*

$$TV(\epsilon) = \varnothing \qquad TV(\Gamma\cdot(x : \tau)) = TV(\Gamma) \cup TV(\tau) \qquad TV(\Gamma\cdot(X : \sigma)) = TV(\Gamma)$$

**Definition B.13** (Free type variables of constraining context)**.** *The free type variables of a constraining context $TV(\Xi)$ is defined as:*

$$TV(\epsilon) = \varnothing \qquad TV(\Xi\cdot(\alpha \leqslant^{\diamond} \tau)) = TV(\Xi) \cup \{\alpha\} \cup TV(\tau)$$

**Definition B.14** (Top-level free type variables)**.** *The set of top-level free type variables of a type $\tau$, written $TTV(\tau)$, is defined as:*

$$\begin{aligned}
TTV(\tau_1 \to \tau_2) &= \varnothing & TTV(\alpha) &= \{\alpha\}\\
TTV(\{\,x : \tau\,\}) &= \varnothing & TTV(\top^{\diamond}) &= \varnothing\\
TTV(\#C) &= \varnothing & TTV(\tau_1 \vee^{\diamond} \tau_2) &= TTV(\tau_1) \cup TTV(\tau_2)\\
TTV(N[\overline{\tau}]) &= TTV(\tau') \quad \textit{when } N[\overline{\tau}]\ \textbf{\textit{exp.}}\ \tau' & TTV(\neg\tau) &= TTV(\tau)
\end{aligned}$$

*The list of top-level free type variables of a type $\tau$ (i.e., with duplicates), written $TTV'(\tau)$, is defined similarly, except for the cases $TTV'(\alpha) = \alpha$ and $TTV'(\tau_1 \vee^{\diamond} \tau_2) = TTV'(\tau_1) \cdot TTV'(\tau_2)$.*

## C MLstruct Correctness Proofs

Finally, we now develop the correctness proofs of MLstruct in full detail.

### *C.1 Progress Proofs*

**Lemma C.1** (Progress — general)**.** *If $\epsilon, \epsilon \vdash P : \tau$ and $body(P)$ is not a value then $P \rightsquigarrow P'$ for some $P'$.*

**Proof** By induction on program typing derivations.

**Case T-Body.** By progress for terms (Lemma C.2).
**Case T-Def.** By E-Def.

■

**Lemma C.2** (Term progress)**.** *If $\epsilon, \epsilon \vdash t : \tau$ and $t$ is not a value then $t \rightsquigarrow t'$ for some $t'$.*

**Proof** By induction on typing derivations.

**Case T-Subs.** Immediate from the induction hypothesis.

**Case T-Obj.** $t = C\ \{\ \overline{x = t'}\ \}$ If all $t'$ are values, then $t$ is a value; otherwise $t$ reduces by E-Ctx and IH.

**Case T-Proj.** $t = t'.x$

If $t'$ is not a value, by IH we have $t' \rightsquigarrow t''$, and thus $t \rightsquigarrow t''.x$ by E-Ctx. Otherwise, by canonical form for record types (Lemma C.3), we have $t' = C\ R$ and $\{\ x = v'\ \} \in R$, and therefore $t \rightsquigarrow v'$ by E-Proj.

**Cases T-Var1, T-Var2.** $t = x$

Impossible since there is no rule that would type $x$ in an empty typing context.

**Case T-Abs.** $t = \lambda x.\ t'$ Immediate since $t$ is a value.

**Case T-App.** $t = t_0\ t_1$

We can apply the induction hypothesis on $t_0$ and $t_1$, which are given types in the premises of this typing rule. If either $t_0$ or $t_1$ is not a value, then $t$ can progress by E-Ctx, so we only have to consider the case where $t_0 = v_0$ and $t_1 = v_1$. By canonical form for function types (Lemma C.4), we have $v_0 = \lambda x.\ t'$. Then $t \rightsquigarrow [x \mapsto v_2]t'$ by E-App.

**Case T-Asc.** $t = t_1 : \tau$ Immediate since $t_1 : \tau \rightsquigarrow t_1$ by E-Asc.

**Case T-Case1.** $t = \textsf{case}\ x = t_1\ \textsf{of}\ \epsilon$

By IH, if $t_1$ is not a value, then $t$ progresses by E-Ctx. Moreover, by canonical form for bottom types (Lemma C.6), $t_1$ cannot be a value.

**Case T-Case2.** $t = \textsf{case}\ x = t_1\ \textsf{of}\ \_ \to t_2$

By IH, if $t_1$ is not a value, then $t$ progresses by E-Ctx. On the other hand, if $t_1 = v_1$, then $t \rightsquigarrow t_2$ by E-CaseWld.

**Case T-Case3.** $t = \textsf{case}\ x = t_1\ \textsf{of}\ C \to t_2,\ M$

By IH, if $t_1$ is not a value, then $t$ progresses by E-Ctx. On the other hand, if $t_1 = v_1$, either $v_1 = C_1\ R$ with $C_2 \in \mathcal{S}(C_1)$, in which case E-CaseCls1 applies, or E-CaseCls2 applies since scrutinees can only be classes by Lemma C.7 and canonical form for class types (Lemma C.5); in either case, $t$ progresses.

■

**Lemma C.3** (Canonical form for record types)**.** *If* $\epsilon, \Gamma \vdash v : \{\ x : \tau\ \}$ *then we have* $v = C\ R$ *for some* $C$ *and* $R$*, and* $\{\ x = v'\ \} \in R$.

**Proof** By induction on typing derivations for the statement: if $\epsilon, \Gamma \vdash v : \tau$ and $\epsilon \vdash \tau \leqslant \{\ x : \tau'\ \}$ then $\{\ x = v'\ \} \in v$. The only cases to consider are those rules that can type values:

**Case T-Subs.** Then the premises of the rule are $v : \tau''$ and $\tau'' \leqslant \tau$ for some $\tau''$. By S-Trans on $\tau'' \leqslant \tau$ and $\tau \leqslant \{\ x : \tau'\ \}$, we have $\tau'' \leqslant \{\ x : \tau'\ \}$. This allows us to apply the IH on the premise $v : \tau''$, by which we have $\{\ x = v'\ \} \in v$.

**Case T-Abs.** Then $\tau = \tau_1 \to \tau_2$. By consistency of subtyping (Theorem A.63), $\tau_1 \to \tau_2 \leqslant \{\ x : \tau'\ \}$ cannot be true, therefore this case is impossible.

**Case T-Obj.** Then $\tau = \#C \wedge \{\ \overline{x_i : \tau_i}^{\,i}\ \}$ and $v = C\ \{\ \overline{x_i = v_i}^{\,i}\ \}$. Then by consistency of subtyping (Theorem A.63) we know that there is an $i$ such that $x_i = x$. Given the

conclusion of T-Obj and the definition of field projection (Section 6.2), this implies that there is a $v' = v_i$ such that $\{ x = v' \} \in v$.

■

**Lemma C.4** (Canonical form for function types)**.** *If* $\epsilon, \Gamma \vdash v : \tau_1 \to \tau_2$ *then we have* $v = \lambda x.\, t$ *for some* $x$ *and* $t$.

**Proof** By induction on typing derivations for the statement: if $\epsilon, \Gamma \vdash v : \tau$, and $\epsilon \vdash \tau \leqslant \tau_1 \to \tau_2$ then $v = \lambda x.\, t$ for some $x$ and $t$. The only cases to consider are those rules that can type values:

**Case T-Subs.** Then the premises of the rule are $v : \tau'$ and $\tau' \leqslant \tau$ for some $\tau'$. By S-Trans on $\tau' \leqslant \tau$ and $\tau \leqslant \tau_1 \to \tau_2$, we have $\tau' \leqslant \tau_1 \to \tau_2$. Then the result follows from IH on $v : \tau'$.

**Case T-Abs.** Immediate.

**Case T-Obj.** Then $\tau = \{ \overline{x_i : \tau_i}^{\,i} \}$ for some $\overline{x_i}^{\,i}$ and $\overline{v_i}^{\,i}$. By consistency of subtyping (Theorem A.63), $\tau \leqslant \tau_1 \to \tau_2$ cannot be true, therefore this case is impossible.

■

**Lemma C.5** (Canonical form for class types)**.** *If* $\epsilon, \Gamma \vdash v : \#C$ *then we have* $v = C\ R$ *for some* $R$.

**Proof** By induction on typing derivations for the statement: if $\epsilon, \Gamma \vdash v : \tau$, and $\epsilon \vdash \tau \leqslant \#C$ then $v = C\ R$ for some $R$. The only cases to consider are those rules that can type values:

**Case T-Subs.** Then the premises of the rule are $v : \tau'$ and $\tau' \leqslant \tau$ for some $\tau'$. By S-Trans on $\tau' \leqslant \tau$ and $\tau \leqslant \#C$, we have $\tau' \leqslant \#C$. Then the result follows from IH on $v : \tau'$.

**Case T-Abs.** Then $\tau = \tau_1 \to \tau_2$ for some $\tau_1$ and $\tau_2$. By consistency of subtyping (Theorem A.63), $\tau \leqslant \#C$ cannot be true, therefore this case is impossible.

**Case T-Obj.** Immediate.

■

**Lemma C.6** (Canonical form for bottom type)**.** *For all* $v$, $\epsilon, \Gamma \vdash v : \bot$ *cannot be derived.*

**Proof** By case analysis on the last typing rule used in the typing derivation, assuming without loss of generality that this typing derivation is in subsumption-normalized form (Lemma 3.12). The only cases to consider are those rules that can type values:

**Cases T-Abs, T-Obj.** Immediate.

**Case T-Subs.** The premises are $\epsilon, \Gamma \vdash v : \tau$ and $\epsilon \vdash \tau \leqslant \tau'$ and the goal is to show that we cannot have $\tau' = \bot$, i.e., that $\tau \leqslant \bot$ cannot be derived. The typing derivation being subsumption-normalized, the first premise is not an application of T-Subs, so it must be an application of either T-Abs or T-Obj, meaning that

$\tau \in \{\, \tau_1 \to \tau_2,\ \#C \wedge \{\, \overline{x_i : \tau_i}^{\,i} \,\} \,\}$. We conclude that $\tau \leqslant \bot$ cannot be derived by consistency of subtyping (Theorem A.63).

■

**Lemma C.7** (Scrutinee types)**.** *If* $\epsilon, \Gamma \vdash$ ***case*** $x = t$ ***of*** $M : \tau$ *then we have* $\epsilon, \Gamma \vdash t : \#C$ *for some* $C$.

**Proof** By induction of typing derivations.

**Case T-Subs.** Then the former premise of the rule is $\epsilon, \Gamma \vdash \mathbf{case}\ x = v\ \mathbf{of}\ M : \tau'$ for some $\tau'$. The result follows from IH.

**Case T-Case1.** Then the premise of the rule is $\epsilon, \Gamma \vdash v : \bot$, which is impossible by canonical form for bottom type (Lemma C.6).

**Case T-Case2.** Then the former premise of the rule is $\epsilon, \Gamma \vdash v : \tau_1 \wedge \#C$ for some $\tau_1$ and $C$. Then by T-Subs with $\tau_1 \wedge \#C \leqslant \#C$ (S-AndOr12↷), we have $\epsilon, \Gamma \vdash v : \#C$.

**Case T-Case3.** Then the first premise of the rule is $\epsilon, \Gamma \vdash t : \#C \wedge \tau_1 \vee \neg\#C \wedge \tau_2$ for some $\tau_1$ and $\tau_2$ We have either $\epsilon, \Gamma \vdash t : \#C'$ or $\epsilon, \Gamma \vdash t : \neg\#C'$ for some $C'$. For the former, the result is immediate. For the latter, we have $\epsilon, \Gamma \vdash t : (\#C \wedge \tau_1 \vee \neg\#C \wedge \tau_2) \wedge \neg\#C$, which implies $\epsilon, \Gamma \vdash t : \tau_2$ by T-Subs since $(\#C \wedge \tau_1 \vee \neg\#C \wedge \tau_2) \wedge \neg\#C \equiv \neg\#C \wedge \tau_2 \leqslant \tau_2$. By IH on the last premise $\epsilon, \Gamma{\cdot}(x : \tau_2) \vdash \mathbf{case}\ x = x\ \mathbf{of}\ M : \tau$, we have $\epsilon, \Gamma{\cdot}(x : \tau_2) \vdash x : \#C''$ for some $C''$, i.e., $\tau_2 \leqslant \#C''$. Then we have $\epsilon, \Gamma \vdash t : \#C''$ by T-Subs.

■

### *C.2 Preservation Proofs*

**Lemma C.8** (Preservation — general)**.** *If* $\epsilon, \Gamma \vdash^\star P : \tau$ *and* $P \rightsquigarrow P'$*, then we have* $\epsilon, \Gamma \vdash^\star P' : \tau$.

**Proof** By induction on program typing derivations.

**Case T-Body.** By preservation for terms (Lemma C.12).

**Case T-Def.** $P = \mathbf{def}\ x = t\ ;\ P'$

The only applicable reduction rule is E-Def. The premises of the rule are $\Xi, \Gamma \vdash t : \tau$ and $\epsilon, \Gamma{\cdot}(x : \forall\Xi.\ \tau) \vdash^\star P' : \tau_P$ for some $\Xi$ and $\tau$. By substitution (Lemma C.9), we have $\epsilon, \Gamma \vdash [x \mapsto t]P' : \tau_P$.

■

**Lemma C.9** (Substitution)**.** *For all* $\mathcal{D}$ ***wf****,* $\Gamma$ *and* $\Xi$ *such that* $TV(\Gamma) \cap TV(\forall\Xi.\ \tau) = \varnothing$*:*

1. *If* $\epsilon, \Gamma{\cdot}(x : \forall\Xi.\ \tau) \vdash^\star P : \tau_P$ *and* $\Xi, \Gamma \vdash t : \tau$*, then* $\epsilon, \Gamma \vdash^\star [x \mapsto t]P : \tau_P$.
2. *If* $\Xi_0, \Gamma{\cdot}(x : \forall\Xi.\ \tau) \vdash t_P : \tau_P$ *and* $\Xi_0{\cdot}\Xi, \Gamma \vdash t : \tau$*, then* $\Xi_0, \Gamma \vdash [x \mapsto t]t_P : \tau_P$.

**Proof** By induction on program typing derivations of $\epsilon, \Gamma{\cdot}(x : \forall\Xi.\ \tau) \vdash^\star P : \tau_P$ and typing derivations of $\Xi_0, \Gamma{\cdot}(x : \forall\Xi.\ \tau) \vdash t_P : \tau_P$. Note that the $TV(\Gamma) \cap TV(\forall\Xi.\ \tau) = \varnothing$ condition can always be obtained by renaming variables quantified in definitions, when necessary. The only difficult cases are for T-Body and T-Var2:

**Case T-Body.** $P = t_P$

The premises of the rule are $\epsilon$ ***cons.*** and $\epsilon, \Gamma{\cdot}(x : \forall\Xi.\tau) \vdash t_P : \tau_P$. By assumption, we have $\Xi, \Gamma \vdash t : \tau$ By IH, we have $\epsilon, \Gamma \vdash [x \mapsto t]t_P : \tau_P$. The result $\epsilon, \Gamma \vdash^\star [x \mapsto t]t_P : \tau_P$ then follows by T-Body, as $P = t_P$.

**Case T-Def.** $P = \mathbf{def}\ x' = t'\ ;\ P'$

If $x' = x$, then $[x \mapsto t]P = P$ and the result is immediate.

Otherwise, $[x \mapsto t]P = \mathbf{def}\ x' = [x \mapsto t]t'\ ;\ [x \mapsto t]P$. We can apply the IH on the second premise of T-Def, $\Xi', \Gamma{\cdot}(x : \forall\Xi.\ \tau) \vdash t' : \tau'$, to get $\Xi', \Gamma \vdash [x \mapsto t]t' : \tau'$. Then, the third premise of T-Def, $\epsilon, \Gamma{\cdot}(x : \forall\Xi.\ \tau){\cdot}(x' : \forall\Xi'.\ \tau') \vdash^\star P : \tau_P$, can be commuted (Lemma C.11) to $\epsilon, \Gamma{\cdot}(x' : \forall\Xi'.\ \tau'){\cdot}(x : \forall\Xi.\ \tau) \vdash^\star P : \tau_P$, on which we can apply the IH to get $\epsilon, \Gamma{\cdot}(x' : \forall\Xi'.\ \tau') \vdash^\star [x \mapsto t]P : \tau_P$. We then conclude by T-Def, for which we have just derived the last two premises (the first premise is unchanged).

**Case T-Subs.** The premises of the rule are $\Xi_0, \Gamma{\cdot}(x : \forall\Xi.\ \tau) \vdash t_P : \tau_1$ and $\Xi_0 \vdash \tau_1 \leqslant \tau_P$. By IH on the first premise, we have $\Xi_0, \Gamma \vdash [x \mapsto t]t_P : \tau_1$. Then $\Xi_0, \Gamma \vdash [x \mapsto t]t_P : \tau_P$ by T-Subs with the second premise.

**Case T-Obj.** $t_P = C\ \{\ \overline{x' = t'}\ \}$ $\tau_P = \#C \wedge \{\ \overline{x' : \tau'}\ \}$

The premises of the rule are $\overline{\Xi_0, \Gamma{\cdot}(x : \forall\Xi.\ \tau) \vdash t' : \tau'}$. By IH, we have $\overline{\Xi_0, \Gamma \vdash [x \mapsto t]t' : \tau'}$. Then $\Xi_0, \Gamma \vdash C\ \{\ \overline{x' = [x \mapsto t]t'}\ \} : \#C \wedge \{\ \overline{x' : \tau'}\ \}$ by T-Obj, i.e., $\Xi_0, \Gamma \vdash [x \mapsto t](C\ \{\ \overline{x' = t'}\ \}) : \#C \wedge \{\ \overline{x' : \tau'}\ \}$ by the definition of substitution.

**Case T-Proj.** $t_P = t'.x'$

The premise of the rule is $\Xi_0, \Gamma{\cdot}(x : \forall\Xi.\ \tau) \vdash t' : \{\ x' : \tau_P\ \}$. By IH, we have $\Xi_0, \Gamma \vdash [x \mapsto t]t' : \{\ x' : \tau_P\ \}$. Then $\Xi_0, \Gamma \vdash ([x \mapsto t]t').x' : \tau_P$ by T-Proj, i.e., $\Xi_0, \Gamma \vdash [x \mapsto t]t'.x' : \tau_P$ by the definition of substitution.

**Case T-Var1.** $t_P = x'$ $(\Gamma{\cdot}(x : \forall\Xi.\ \tau))(x') = \tau_P$

Since $x'$ is mapped to a simple type in the context $\Gamma{\cdot}(x : \forall\Xi.\ \tau)$, $x \neq x'$, then $\Gamma(x') = \tau_P$. Then $\Xi_0, \Gamma \vdash x' : \tau_P$, i.e., $\Xi_0, \Gamma \vdash [x \mapsto t]x' : \tau_P$ by the definition of substitution.

**Case T-Var2.** $t_P = x'$ $\rho(\tau'_P) \leqslant \tau_P$ $(\Gamma{\cdot}(x : \forall\Xi.\ \tau))(x') = \forall\Xi'.\ \tau'_P$ $\Xi_0 \vDash \rho(\Xi')$

There are two cases to consider:

**Case** $x' \neq x$**.** Then $[x \mapsto t]t_P = t_P$ and the result is immediate.

**Case** $x' = x$**.** Then $[x \mapsto t]t_P = t$ and moreover $(\Gamma{\cdot}(x : \forall\Xi.\ \tau))(x) = \forall\Xi'.\ \tau'_P$, thus $\forall\Xi.\ \tau = \forall\Xi'.\ \tau'_P$, and thus $\Xi = \Xi'$ and $\tau = \tau'_P$.

By assumption, $\Xi, \Gamma \vdash t : \tau$ so $\Xi', \Gamma \vdash t : \tau'_P$. By preservation of typing under substitution (Lemma A.28), $\rho(\Xi'), \rho(\Gamma) \vdash t : \rho(\tau'_P)$, i.e., $\rho(\Xi'), \Gamma \vdash t : \tau_P$ by T-Subs and since $TV(\Gamma) \cap dom(\rho) = \varnothing$ by assumption.

Moreover, since we have $\Xi_0 \vDash \rho(\Xi')$, this implies that $\Xi_0, \Gamma \vdash t : \tau_P$ (Lemma A.27), which is what we wanted to prove (remember $t = [x \mapsto t]t_P$).

**Case T-Abs.** $t_P = \lambda x'.\ t'$ $\tau_P = \tau_1 \to \tau_2$

There are two cases to consider:

**Case** $x' = x$**.** The premise of the rule is $\Xi_0, \Gamma\cdot(x : \forall\Xi.\ \tau)\cdot(x : \tau_1) \vdash t' : \tau_2$. Since the binding $(x : \forall\Xi.\ \tau)$ is shadowed, we can remove it from the typing context (Lemma C.10), i.e., $\Xi_0, \Gamma\cdot(x : \tau_1) \vdash t' : \tau_2$. Then $\Xi_0, \Gamma \vdash \lambda x.\ t' : \tau_1 \to \tau_2$ by T-Abs, which is the desired result since $[x \mapsto t]t_P = t_P$ and $x' = x$.

**Case** $x' \neq x$**.** The premise of the rule is $\Xi_0, \Gamma\cdot(x : \forall\Xi.\ \tau)\cdot(x' : \tau_1) \vdash t' : \tau_2$, which can be commuted (Lemma C.11) to $\Xi_0, \Gamma\cdot(x' : \tau_1)\cdot(x : \forall\Xi.\ \tau) \vdash t' : \tau_2$. By IH, we have $\Xi_0, \Gamma\cdot(x' : \tau_1) \vdash [x \mapsto t]t' : \tau_2$. Then $\Xi_0, \Gamma \vdash \lambda x'.\ [x \mapsto t]t' : \tau_1 \to \tau_2$, i.e., $\Xi_0, \Gamma \vdash [x \mapsto t]\lambda x'.\ t' : \tau_1 \to \tau_2$ by the definition of substitution.

**Case T-App.** $t_P = t_0 t_1$

The premises of the rule are $\Xi_0, \Gamma\cdot(x : \forall\Xi.\ \tau) \vdash t_0 : \tau_1 \to \tau_P$ and $\Xi_0, \Gamma\cdot(x : \forall\Xi.\ \tau) \vdash t_1 : \tau_1$ for some $\tau_1$. By IH, we have $\Xi_0, \Gamma \vdash [x \mapsto t]t_0 : \tau_1 \to \tau_P$ and $\Xi_0, \Gamma \vdash [x \mapsto t]t_1 : \tau_1$. Then $\Xi_0, \Gamma \vdash [x \mapsto t]t_0\ [x \mapsto t]t_1 : \tau_P$ by T-App, i.e., $\Xi_0, \Gamma \vdash [x \mapsto t](t_0\ t_1) : \tau_P$ by the definition of substitution.

**Case T-Asc.** $t_P = t' : \tau_P$

The premise of the rule is $\Xi_0, \Gamma\cdot(x : \forall\Xi.\ \tau) \vdash t' : \tau_P$. By IH, we have $\Xi_0, \Gamma \vdash [x \mapsto t]t' : \tau_P$. Then $\Xi_0, \Gamma \vdash ([x \mapsto t]t' : \tau_P) : \tau_P$ by T-Asc, i.e., $\Xi_0, \Gamma \vdash [x \mapsto t](t' : \tau_P) : \tau_P$ by the definition of substitution.

**Case T-Case1.** $t_P = \mathsf{case}\ x' = t_1\ \mathsf{of}\ \epsilon \quad \tau_P = \bot$

The premise of the rule is $\Xi_0, \Gamma\cdot(x : \forall\Xi.\ \tau) \vdash t_1 : \bot$. By IH, we have $\Xi_0, \Gamma \vdash [x \mapsto t]t_1 : \bot$. Then $\Xi_0, \Gamma \vdash \mathsf{case}\ x' = [x \mapsto t]t_1\ \mathsf{of}\ \epsilon : \bot$ by T-Case1, i.e., $\Xi_0, \Gamma \vdash [x \mapsto t]\mathsf{case}\ x' = t_1\ \mathsf{of}\ \epsilon : \bot$ by the definition of substitution.

**Case T-Case2.** $t_P = \mathsf{case}\ x' = t_1\ \mathsf{of}\ \_ \to t_2$

There are two cases to consider:

**Case** $x' = x$**.** The premises of the rule are $\Xi_0, \Gamma\cdot(x : \forall\Xi.\ \tau) \vdash t_1 : \tau_1$ and $\Xi_0, \Gamma\cdot(x : \forall\Xi.\ \tau)\cdot(x : \tau_1) \vdash t_2 : \tau_P$. Since the binding $(x : \forall\Xi.\ \tau)$ in the second premise is shadowed, we can remove it from the typing context (Lemma C.10), i.e., $\Xi_0, \Gamma\cdot(x : \tau_1) \vdash t_2 : \tau_P$. By IH on the first premise, we have $\Xi_0, \Gamma \vdash [x \mapsto t]t_1 : \tau_1$. Then $\Xi_0, \Gamma \vdash \mathsf{case}\ x = [x \mapsto t]t_1\ \mathsf{of}\ \_ \to t_2 : \tau_P$, i.e., $\Xi_0, \Gamma \vdash [x \mapsto t]\mathsf{case}\ x = t_1\ \mathsf{of}\ \_ \to t_2 : \tau_P$ by the definition of substitution.

**Case** $x' \neq x$**.** The premises of the rule are $\Xi_0, \Gamma\cdot(x : \forall\Xi.\ \tau) \vdash t_1 : \tau_1$ and $\Xi_0, \Gamma\cdot(x : \forall\Xi.\ \tau)\cdot(x' : \tau_1) \vdash t_2 : \tau_P$. The latter can be commuted (Lemma C.11) to $\Xi_0, \Gamma\cdot(x' : \tau_1)\cdot(x : \forall\Xi.\ \tau) \vdash t_2 : \tau_P$. By IH, we have $\Xi_0, \Gamma \vdash [x \mapsto t]t_1 : \tau_1$ and $\Xi_0, \Gamma\cdot(x' : \tau_1) \vdash [x \mapsto t]t_2 : \tau_P$. Then $\Xi_0, \Gamma \vdash \mathsf{case}\ x' = [x \mapsto t]t_1\ \mathsf{of}\ \_ \to [x \mapsto t]t_2$ by T-Case2, i.e., $\Xi_0, \Gamma \vdash [x \mapsto t]\mathsf{case}\ x' = t_1\ \mathsf{of}\ \_ \to t_2$ by the definition of substitution.

**Case T-Case3.** $t_P = \mathsf{case}\ x' = t_1\ \mathsf{of}\ C \to t_2,\ M$

There are two cases to consider:

**Case** $x' = x$**.** The premises of the rule are:

$$\Xi_0, \Gamma\cdot(x : \forall\Xi.\ \tau) \vdash t_1 : \#C \wedge \tau_1 \vee \neg\#C \wedge \tau_2 \tag{1}$$

$$\Xi_0, \Gamma\cdot(x : \forall\Xi.\ \tau)\cdot(x : \tau_1) \vdash t_2 : \tau_P \tag{2}$$

$$\Xi_0, \Gamma\cdot(x : \forall\Xi.\ \tau)\cdot(x : \tau_2) \vdash \mathsf{case}\ x' = x'\ \mathsf{of}\ M : \tau_P \tag{3}$$

By IH on (1), we have:

$$\Xi_0, \Gamma \vdash [x \mapsto t]t_1 : \#C \wedge \tau_1 \vee \neg\#C \wedge \tau_2 \tag{4}$$

Since the binding $(x : \forall\Xi.\ \tau)$ in (2) and (3) are shadowed, we can remove them from the typing contexts (Lemma C.10):

$$\Xi_0, \Gamma\cdot(x : \tau_1) \vdash t_2 : \tau_P \tag{5}$$

$$\Xi_0, \Gamma\cdot(x : \tau_2) \vdash \textbf{case}\ x = x\ \textbf{of}\ M : \tau_P \tag{6}$$

Then by T-Case3 on (4) and (5) and (6), we have:

$$\Xi_0, \Gamma \vdash \textbf{case}\ x = [x \mapsto t]t_1\ \textbf{of}\ C \to t_2,\ M : \tau_P$$

$$\text{i.e.,}\quad \Xi_0, \Gamma \vdash [x \mapsto t]\textbf{case}\ x = t_1\ \textbf{of}\ C \to t_2,\ M : \tau_P \tag{7}$$

**Case $x' \neq x$.** The premises of the rule are:

$$\Xi_0, \Gamma\cdot(x : \forall\Xi.\ \tau) \vdash t_1 : \#C \wedge \tau_1 \vee \neg\#C \wedge \tau_2 \tag{8}$$

$$\Xi_0, \Gamma\cdot(x : \forall\Xi.\ \tau)\cdot(x' : \tau_1) \vdash t_2 : \tau_P \tag{9}$$

$$\Xi_0, \Gamma\cdot(x : \forall\Xi.\ \tau)\cdot(x' : \tau_2) \vdash \textbf{case}\ x' = x'\ \textbf{of}\ M : \tau_P \tag{10}$$

The typing contexts in (9) and (10) can be commuted (Lemma C.11) to:

$$\Xi_0, \Gamma\cdot(x' : \tau_1)\cdot(x : \forall\Xi.\ \tau) \vdash t_2 : \tau_P \tag{11}$$

$$\Xi_0, \Gamma\cdot(x' : \tau_2)\cdot(x : \forall\Xi.\ \tau) \vdash \textbf{case}\ x' = x'\ \textbf{of}\ M : \tau_P \tag{12}$$

By IH on (8) and (11) and (12) respectively, we have:

$$\Xi_0, \Gamma \vdash [x \mapsto t]t_1 : \#C \wedge \tau_1 \vee \neg\#C \wedge \tau_2 \tag{13}$$

$$\Xi_0, \Gamma\cdot(x' : \tau_1) \vdash [x \mapsto t]t_2 : \tau_P \tag{14}$$

$$\Xi_0, \Gamma\cdot(x' : \tau_2) \vdash \textbf{case}\ x' = x'\ \textbf{of}\ [x \mapsto t]M : \tau_P \tag{15}$$

Then by T-Case3 on (13) and (14) and (15), we have:

$$\Xi_0, \Gamma \vdash \textbf{case}\ x' = [x \mapsto t]t_1\ \textbf{of}\ C \to [x \mapsto t]t_2,\ [x \mapsto t]M : \tau_P$$

$$\text{i.e.,}\quad \Xi_0, \Gamma \vdash [x \mapsto t]\textbf{case}\ x' = t_1\ \textbf{of}\ C \to t_2,\ M : \tau_P \tag{16}$$

■

**Lemma C.10** (Shadowing of typing contexts)**.** *For all $\gamma = \tau$ or $\sigma$, and $\gamma' = \tau'$ or $\sigma'$:*

1. *If $\Xi, \Gamma\cdot(x : \gamma)\cdot\Gamma'\cdot(x : \gamma')\cdot\Gamma'' \vdash^\star P : \tau_P$, then $\Xi, \Gamma\cdot\Gamma'\cdot(x : \gamma)\cdot(x : \gamma')\cdot\Gamma'' \vdash^\star P : \tau_P$ and $\Xi, \Gamma\cdot\Gamma'\cdot(x : \gamma')\cdot\Gamma'' \vdash^\star P : \tau_P$.*
2. *If $\Xi, \Gamma\cdot(x : \gamma)\cdot\Gamma'\cdot(x : \gamma')\cdot\Gamma'' \vdash t_P : \tau_P$, then $\Xi, \Gamma\cdot\Gamma'\cdot(x : \gamma)\cdot(x : \gamma')\cdot\Gamma'' \vdash t_P : \tau_P$ and $\Xi, \Gamma\cdot\Gamma'\cdot(x : \gamma')\cdot\Gamma'' \vdash t_P : \tau_P$.*

**Proof** By straightforward induction on typing derivations. The only non-trivial cases are T-Var1 and T-Var2.

**Case T-VAR1.** By the definition of $\Gamma(\cdot)$, if $(\Gamma{\cdot}(y:\gamma){\cdot}\Gamma'{\cdot}(y:\gamma'){\cdot}\Gamma'')(x) = \tau''$ for some $\tau''$, then $(\Gamma{\cdot}\Gamma'{\cdot}(y:\gamma){\cdot}(y:\gamma'){\cdot}\Gamma'')(x) = \tau''$ and $(\Gamma{\cdot}\Gamma'{\cdot}(y:\gamma'){\cdot}\Gamma'')(x) = \tau''$. The result then follows from T-VAR1.

**Case T-VAR2.** Similarly.

■

**Lemma C.11** (Commutativity of typing contexts)**.** *For all* $\Gamma'$ *such that* $x \notin dom(\Gamma')$*, and* $\gamma = \tau$ *or* $\sigma$*:*

1. *If* $\epsilon, \Gamma{\cdot}(x:\gamma){\cdot}\Gamma' \vdash^\star P : \tau_P$*, then* $\epsilon, \Gamma{\cdot}\Gamma'{\cdot}(x:\gamma) \vdash^\star P : \tau_P$.
2. *If* $\Xi, \Gamma{\cdot}(x:\gamma){\cdot}\Gamma' \vdash t_P : \tau_P$*, then* $\Xi, \Gamma{\cdot}\Gamma'{\cdot}(x:\gamma) \vdash t_P : \tau_P$.

**Proof** By induction on typing derivations.

**Case T-BODY.** By IH, followed by T-BODY.

**Case T-DEF.** $P = \mathbf{def}\ x' = t'\ ;\ P'$

The premises are $\epsilon$ ***cons.***, $\Xi', \Gamma{\cdot}(x:\gamma){\cdot}\Gamma' \vdash t' : \tau'$, and $\epsilon, \Gamma{\cdot}(x:\gamma){\cdot}\Gamma'{\cdot}(x' : \forall\Xi'.\ \tau') \vdash^\star P' : \tau_P$. By IH on the second premise, we have $\Xi', \Gamma{\cdot}\Gamma'{\cdot}(x:\gamma) \vdash t' : \tau'$. If $x' = x$, we can rearrange the third premise (Lemma C.10) to $\epsilon, \Gamma{\cdot}\Gamma'{\cdot}(x:\gamma){\cdot}(x' : \forall\Xi'.\ \tau') \vdash^\star P' : \tau_P$. If $x' \neq x$, then $x \notin dom(\Gamma'{\cdot}(x' : \forall\Xi'.\ \tau'))$ and $x' \notin dom((x:\gamma))$, so we have $\epsilon, \Gamma{\cdot}\Gamma'{\cdot}(x:\gamma){\cdot}(x' : \forall\Xi'.\ \tau') \vdash^\star P' : \tau_P$ by IH. The result $\epsilon, \Gamma{\cdot}\Gamma'{\cdot}(x:\gamma) \vdash^\star \mathbf{def}\ x' = t'\ ;\ P' : \tau_P$ then follows from T-DEF.

**Cases T-SUBS, T-RCD, T-PROJ, T-APP, T-ASC, T-CASE1.** By IH on the premises, followed by the respective rules.

**Case T-VAR1.** By the definition of $\Gamma(\cdot)$, since $x \notin dom(\Gamma')$ by assumption, if $(\Gamma{\cdot}(x:\gamma){\cdot}\Gamma')(x') = \tau'$ for some $\tau'$, then $(\Gamma{\cdot}\Gamma'{\cdot}(x:\gamma))(x') = \tau'$. The result then follows from T-VAR1.

**Case T-VAR2.** Similar to the case above.

**Case T-ABS.** $t_P = \lambda x'.\ t' \quad \tau_P = \tau_1 \to \tau_2$

The premise is $\Xi, \Gamma{\cdot}(x:\gamma){\cdot}\Gamma'{\cdot}(x' : \tau_1) \vdash t' : \tau_2$. If $x' = x$, we can rearrange it (Lemma C.10) to $\Xi, \Gamma{\cdot}\Gamma'{\cdot}(x:\gamma){\cdot}(x' : \tau_1) \vdash t' : \tau_2$. If $x' \neq x$, then $x \notin dom(\Gamma'{\cdot}(x' : \tau_1))$ and $x' \notin dom((x:\gamma))$, so we have $\Xi, \Gamma{\cdot}\Gamma'{\cdot}(x:\gamma){\cdot}(x' : \tau_1) \vdash t' : \tau_2$ by IH. The result $\Xi, \Gamma{\cdot}\Gamma'{\cdot}(x:\gamma) \vdash \lambda x'.\ t' : \tau_1 \to \tau_2$ then follows from T-ABS.

**Cases T-CASE2, T-CASE3.** Similar to the case above.

■

**Lemma C.12** (Term preservation)**.** *If* $\epsilon, \Gamma \vdash t : \tau$ *and* $t \rightsquigarrow t'$*, then* $\epsilon, \Gamma \vdash t' : \tau$.

**Proof** By induction on typing derivations. In the following, we sometimes abbreviate $\epsilon, \Gamma \vdash t : \tau$ to $t : \tau$.

**Case T-SUBS.** Immediate from the induction hypothesis.

**Case T-Obj.** $t = C\,\{\,\overline{x = t}\,\} \quad \tau = \#C \wedge \{\,\overline{x : \tau}\,\}$

There is only one rule that reduces objects, E-Ctx. By straightforward application of the induction hypothesis with the respective premises of T-Obj and E-Obj and by reapplication of T-Obj on $t'$.

**Case T-Proj.** $t = t_0.x \quad t_0 : \{\,x : \tau\,\}$

If $t \rightsquigarrow t_0'.x$ by E-Ctx, we conclude by IH.

Otherwise, $t \rightsquigarrow v_2$ reduces by E-Proj, meaning that $t_0 = v_1$ and $\{\,x = v_2\,\} \in v_1$. We conclude by inversion of object types (Lemma C.16), which gives us $v_2 : \tau$.

**Cases T-Var1,T-Var2.** Immediate since $t$ cannot reduce.

**Case T-Abs.** $t = \lambda x.\, t_0$ Immediate since $t$ cannot reduce.

**Case T-App.** $t = t_0\, t_1 \quad t_0 : \tau_1 \to \tau \quad t_1 : \tau_1$

There are two rules by which $t \rightsquigarrow t'$ can hold:

**Case E-Ctx.** The result holds by IH and T-App.

**Case E-App** $t_0 = \lambda x.\, t_0' \quad t_1 = v_1 \quad t \rightsquigarrow [x \mapsto v_1]t_0'$

By inversion (Lemma C.13), $\epsilon, \Gamma{\cdot}(x : \tau_1) \vdash t_0' : \tau$. Together with substitution (Lemma C.9, applicable since $\epsilon, \Gamma \vdash v_1 : \tau_1$), this gives us $\epsilon, \Gamma \vdash [x \mapsto v_1]t_0' : \tau$, i.e., $\epsilon, \Gamma \vdash t' : \tau$.

**Case T-Asc.** $t = t_0 : \tau \quad t' = t_0$

Immediate by the premise of the rule.

**Case T-Case1.** $t = \textsf{case}\ x = t_1\ \textsf{of}\ \epsilon$

Immediate since the only rule that can apply is E-Ctx, and it yields a term $t'$ that can still be typed at $\bot$ by T-Case1.

**Case T-Case2.** $t = \textsf{case}\ x = t_1\ \textsf{of}\ \_ \to t_2$

If the rule that applies is E-Ctx, by IH. Otherwise, the rule that applies is E-CaseWld, and we conclude by substitution.

**Case T-Case3.** $t = \textsf{case}\ x = t_1\ \textsf{of}\ C \to t_2,\ M \quad t_1 : \#C \wedge \tau_1 \vee \neg\#C \wedge \tau_2$

If the rule that applies is E-Ctx, by IH.

Otherwise, if E-CaseCls1 is the rule that applies, it means $t_1$ is an instance of a subclass of $C_2$, so by Lemma C.18 we know that $\epsilon, \Gamma \vdash t_1 : \tau_1$, and we can conclude by substitution (Lemma C.9).

Otherwise, E-CaseCls2 must be the rule that applies, so by Lemma C.18 we know that $\epsilon, \Gamma \vdash t_1 : \tau_2$, and we can conclude by substitution (Lemma C.9) and IH.

■

**Lemma C.13** (Inversion of function types)**.** *If* $\epsilon, \Gamma \vdash \lambda x.\, t : \tau_0$ *and* $\epsilon \vdash \tau_0 \leqslant \tau_1 \to \tau_2$*, then* $\epsilon, \Gamma{\cdot}(x : \tau_1) \vdash t : \tau_2$.

**Proof** Straightforward induction on typing derivations. The only rules that can be used to type such a lambda expression are:

**Case T-Subs.** Then the premises of the rule are $\epsilon, \Gamma \vdash \lambda x.\, t : \tau_0'$ and $\epsilon \vdash \tau_0' \leqslant \tau_0$ for some $\tau_0'$, on which we can apply the IH by S-Trans ($\tau_0' \leqslant \tau_0 \leqslant \tau_1 \to \tau_2$).

**Case T-Abs.** Then $\tau_0 = \tau_1' \to \tau_2'$ for some $\tau_1'$ and $\tau_2'$. The premise is $\epsilon, \Gamma \cdot (x : \tau_1') \vdash t : \tau_2'$. By Lemma C.14 we have $\epsilon \vdash \tau_1 \leqslant \tau_1'$ and $\epsilon \vdash \tau_2' \leqslant \tau_2$. Combined with strengthening (Lemma C.15) and T-Subs, this gives us the desired result.

■

**Lemma C.14** (Inversion of function subtyping). *If* $\epsilon \vdash \tau_0 \to \tau_1 \leqslant \tau_2 \to \tau_3$, *then* $\epsilon \vdash \tau_2 \leqslant \tau_0$ *and* $\epsilon \vdash \tau_1 \leqslant \tau_3$.

**Proof** By consistency of subtyping (Theorem A.63). ■

**Lemma C.15** (Strengthening). *If* $\epsilon, \Gamma \cdot (x : \tau_1) \vdash t : \tau$ *and* $\epsilon \vdash \tau_2 \leqslant \tau_1$, *then we have* $\epsilon, \Gamma \cdot (x : \tau_2) \vdash t : \tau$.

**Proof** By straightforward induction on typing derivations, using T-Subs for the T-Var1 case. ■

**Lemma C.16** (Inversion of object types). *If* $\epsilon, \Gamma \vdash C\ R : \tau_0$ *and* $\{ x = v \} \in C\ R$ *and* $\epsilon \vdash \tau_0 \leqslant \{ x : \tau \}$, *then* $\epsilon \vdash v : \tau$.

**Proof**

Straightforward induction on typing derivations. The only rules that can be used to type such a lambda expression are:

**Case T-Subs.** Then the premises of the rule are $\epsilon, \Gamma \vdash \lambda x.\, t : \tau_0'$ and $\epsilon \vdash \tau_0' \leqslant \tau_0$ for some $\tau_0'$, on which we can apply the IH by S-Trans ($\tau_0' \leqslant \tau_0 \leqslant \{ x : \tau \}$).

**Case T-Obj.** Then $\tau_0 = \#C \wedge \{ \overline{x_i : \tau_i}^{\,i} \}$ for some $C$ and $\overline{\tau_i}^{\,i}$. One of the premises is $\epsilon, \Gamma \vdash v : \tau_k$, where $x_k = x$. By Lemma C.17 we have $\epsilon \vdash \tau_k \leqslant \tau$. Combined with T-Subs, this gives us the desired result.

■

**Lemma C.17** (Inversion of object subtyping). *If* $\epsilon \vdash \#C \wedge \{ \overline{x_i : \tau_i}^{\,i} \} \leqslant \{ x_k : \tau \}$, *then* $\epsilon \vdash \tau_k \leqslant \tau$.

**Proof** Let $U_0^{C_0} = \#C$ and $\overline{U_i^{C_i} = \{ x_i : \tau_i \}}^{\,i}$. Since $\#C \wedge \{ \overline{x_i : \tau_i}^{\,i} \} \cong \bigwedge_{i' \in \{ 0, \bar{i} \}} (\bot \vee U_i^{C_i})$, by Lemma 4.22, we have:

$$\{ x_k : \tau \} \cong \textstyle\bigwedge_j (\pi_j' \vee V_j^{D_j}) \tag{1}$$

$$\overline{U_{k_j}^{C_{k_j}} \preceq V_j^{D_j}}^{\,j} \tag{2}$$

for some $\overline{\pi_j'}^{\,j}$ and $\overline{V_j^{D_j}}^{\,j}$ and $\overline{k_j}^{\,j}$. By S-Trans with Lemma A.7ↄ on S-AndOr12·, (1) implies:

$$\textstyle\bigwedge_j V_j^{D_j} \subseteq \{ x_k : \tau \} \tag{3}$$

By Lemma A.57, (3) implies:

$$V_l^{D_l} \subseteq \{ x_k : \tau \} \tag{4}$$

for some $l$. By Lemma 4.10, (4) implies:

$$V_l^{D_l} = \bigvee_p \{ x_k : \tau \} \tag{5}$$

Then $D_l = x_k$. By Lemma 4.9, (2) for $j = l$ implies:

$$C_{k_l} = x_k \tag{6}$$

i.e., $k_l = k$. Then (2) for $j = l$ becomes:

$$\{ x_k : \tau_k \} \preceq \bigvee_p \{ x_k : \tau \} \tag{7}$$

By case analysis on the $\preceq$ rules, (7) implies:

$$\tau_k \leqslant \bigvee_p \tau$$
$$\text{i.e.,} \quad \tau_k \leqslant \tau \tag{8}$$

■

**Lemma C.18** (Inversion of discriminated class types)**.** *Assume* $\epsilon, \Gamma \vdash v : \tau$ *where* $v$ *is the scrutinee of a case expression and* $\epsilon \vdash \tau \leqslant \#C \wedge \tau_1 \vee \neg\#C \wedge \tau_2$*. Then we have:*

- *If* $v = C_0\ R$ *and* $C_0$ *is a subclass of* $C$ *(i.e.,* $C \in \mathcal{S}(C_0)$*), then* $\epsilon, \Gamma \vdash v : \tau_1$*.*
- *Otherwise,* $\epsilon, \Gamma \vdash v : \tau_2$*.*

**Proof** By induction on typing derivations.The only rules that can be used to type a value are:

**Case T-Subs.** Then the premises of the rule are $\epsilon, \Gamma \vdash v : \tau'$ and $\epsilon \vdash \tau' \leqslant \tau$ for some $\tau'$, on which we can apply the IH by S-Trans ($\tau' \leqslant \tau \leqslant \#C \wedge \tau_1 \vee \neg\#C \wedge \tau_2$).

**Case T-Abs.** $v = \lambda x.\, t$

Impossible since scrutinees can only be classes (Lemma C.7).

**Case T-Obj.** $v = C_0\ R$

We have $R = \{ \overline{x = t} \}$ and $\tau = \#C_0 \wedge \{ \overline{x : \tau} \}$ and $\overline{t : \tau}$ and $C_0$ is final.

So we have $\#C_0 \wedge \{ \overline{x : \tau} \} \leqslant \#C \wedge \tau_1 \vee \neg\#C \wedge \tau_2$

i.e., $\#C_0 \wedge \{ \overline{x : \tau} \} \wedge (\#C \vee \neg\tau_2) \leqslant \#C \wedge \tau_1$

i.e., (1) $\#C_0 \wedge \#C \wedge \{ \overline{x : \tau} \} \vee \#C_0 \wedge \{ \overline{x : \tau} \} \wedge \neg\tau_2 \leqslant \#C \wedge \tau_1$ Then from the assumption, we have:

$$\#C_0 \wedge \{ \overline{x : \tau} \} \leqslant \#C \wedge \tau_1 \vee \neg\#C \wedge \tau_2$$
$$\text{i.e.,} \quad \#C_0 \wedge \{ \overline{x : \tau} \} \wedge (\#C \vee \neg\tau_2) \leqslant \#C \wedge \tau_1$$
$$\text{i.e.,} \quad \#C_0 \wedge \#C \wedge \{ \overline{x : \tau} \} \vee \#C_0 \wedge \{ \overline{x : \tau} \} \wedge \neg\tau_2 \leqslant \#C \wedge \tau_1 \tag{1}$$

**Case** $C \in \mathcal{S}(C_0)$**.** Then by S-ClsSub, we have:

$$\#C_0 \leqslant \#C$$
$$\text{i.e.,} \quad \#C_0 \wedge \#C \equiv \#C_0 \tag{2}$$

Then (1) and (2) imply:

$$\#C_0 \wedge \{\overline{x:\tau}\} \vee \#C_0 \wedge \{\overline{x:\tau}\} \wedge \neg\tau_2 \leqslant \#C \wedge \tau_1$$
$$\text{i.e.,} \quad \#C_0 \wedge \{\overline{x:\tau}\} \leqslant \#C \wedge \tau_1$$
$$\text{i.e.,} \quad \tau \leqslant \#C \wedge \tau_1 \tag{3}$$

By S-Trans on (3) and S-AndOr12$\supset$, we have:

$$\tau \leqslant \tau_1 \tag{4}$$

Then by T-Subs, the assumption $\epsilon, \Gamma \vdash v : \tau$ and (4) imply:

$$\epsilon, \Gamma \vdash v : \tau_1 \tag{5}$$

**Case** $C \notin \mathcal{S}(C_0)$**.** By S-Trans on S-AndOr12$\cdot$ and (1), we have:

$$\#C_0 \wedge \{\overline{x:\tau}\} \wedge \neg\tau_2 \leqslant \#C$$
$$\text{i.e.,} \quad \#C_0 \wedge \{\overline{x:\tau}\} \leqslant \#C \vee \tau_2 \tag{6}$$

**Case** $C_0 \in \mathcal{S}(C)$**.** This case is impossible because $C_0$ is final and $C_0 \neq C$ (since $C \notin \mathcal{S}(C_0)$).

**Case** $C_0 \notin \mathcal{S}(C)$**.** Then by S-ClsBot and Theorem A.9, we have:

$$\#C_0 \leqslant \neg\#C \tag{7}$$

Then (6) and (7) imply:

$$\#C_0 \wedge \{\overline{x:\tau}\} \wedge \neg\#C \leqslant \tau_2$$
$$\text{i.e.,} \quad \#C_0 \wedge \{\overline{x:\tau}\} \leqslant \tau_2$$
$$\text{i.e.,} \quad \tau \leqslant \tau_2 \tag{8}$$

Then by T-Subs, the assumption $\epsilon, \Gamma \vdash v : \tau$ and (8) imply:

$$\epsilon, \Gamma \vdash v : \tau_2 \tag{9}$$

■

### *C.3 Type Inference Soundness Proofs*

We first define a few judgements to be used in the remainder of this chapter.

The consistency of subtyping contexts is lifted to typing contexts through the bounds in the polymorphic bindings.

**Definition C.19** (Consistency of typing contexts)**.** *The consistency of typing contexts is defined as follows:*

$\boxed{\Gamma \ \textbf{\textit{cons.}}}$

$$\frac{}{\epsilon \ \textbf{\textit{cons.}}} \qquad \frac{\Gamma \ \textbf{\textit{cons.}}}{\Gamma\cdot(x:\tau) \ \textbf{\textit{cons.}}} \qquad \frac{\Gamma \ \textbf{\textit{cons.}} \qquad \Xi \ \textbf{\textit{cons.}}}{\Gamma\cdot(x:\forall\Xi.\,\tau) \ \textbf{\textit{cons.}}}$$

A constraining context is said to be guarded if none of the type variables appear on the top level of its bounds. Guardedness is also similarly raised to typing contexts.

**Definition C.20** (Guardedness of constraining contexts)**.** *The guardedness of constraining contexts is defined as follows:*

$$\boxed{\Xi \ \textbf{\textit{guard.}}} \qquad \frac{}{\epsilon \ \textbf{\textit{guard.}}} \qquad \frac{\alpha \notin TTV(\tau) \qquad \Xi \ \textbf{\textit{guard.}}}{\Xi{\cdot}(\alpha \leqslant^{\diamond} \tau) \ \textbf{\textit{guard.}}}$$

**Definition C.21** (Guardedness of typing contexts)**.** *The guardedness of typing contexts is defined as follows:*

$$\boxed{\Gamma \ \textbf{\textit{guard.}}} \qquad \frac{}{\epsilon \ \textbf{\textit{guard.}}} \qquad \frac{\Gamma \ \textbf{\textit{guard.}}}{\Gamma{\cdot}(x:\tau) \ \textbf{\textit{guard.}}} \qquad \frac{\Gamma \ \textbf{\textit{guard.}} \qquad \Xi \ \textbf{\textit{guard.}}}{\Gamma{\cdot}(x:\forall \Xi.\, \tau) \ \textbf{\textit{guard.}}}$$

**Lemma C.22** (Soundness of type inference — general)**.** *If* $\Gamma \Vdash^{\star} P : \pi \Rightarrow \Xi$ *and* $\Gamma$ ***cons.*** *and* ***err*** $\notin \Xi$*, then* $\Xi, \Gamma \vdash^{\star} P : \pi$*.*

**Proof** By induction on type inference derivations.

**Case I-Body.** By soundness of term inference (Lemma C.23).

**Case I-Def.** By soundness of term inference (Lemma C.23), we get the subtyping relationship necessary to apply the IH on $P$.

■

**Lemma C.23** (Soundness of term type inference)**.** *If* $\Xi^0, \Gamma \Vdash s : \pi \Rightarrow \Xi^1$ *and* $\Xi^0, \Gamma$ ***cons.*** *and* $\Xi^0, \Gamma$ ***guard.*** *and* ***err*** $\notin \Xi^1$*, then* $\Xi^0{\cdot}\Xi^1, \Gamma \vdash s : \pi$ *and* $\Xi^0{\cdot}\Xi^1$ ***cons.*** *and* $\Xi^0{\cdot}\Xi^1$ ***guard.***.

**Proof** By induction on term type inference derivations.

**Case I-Proj.** $s = t.x$

By IH, we have $\Xi_0{\cdot}\Xi_1 \vdash t : \tau$ and $\Xi_0{\cdot}\Xi_1$ ***cons.*** and $\Xi_0{\cdot}\Xi_1$ ***guard.***. And by sound constraining (Lemma 7.5), we have $\Xi_0{\cdot}\Xi_1{\cdot}\Xi_2 \vdash \tau \leqslant \{\, x : \alpha \,\}$ and $\Xi_0{\cdot}\Xi_1{\cdot}\Xi_2$ ***cons.*** and $\Xi_0{\cdot}\Xi_1{\cdot}\Xi_2$ ***guard.***. Therefore, by weakening (Lemma A.27) and T-Subs we have $\Xi_0{\cdot}\Xi_1{\cdot}\Xi_2 \vdash t : \{\, x : \alpha \,\}$ and by T-Proj we have $\Xi_0{\cdot}\Xi_1{\cdot}\Xi_2 \vdash t.x : \alpha$.

**Case I-Obj.** By straightforward applications of the IH and weakening.

**Case I-Var1.** By T-Var1.

**Case I-Var2.** $t = x \quad \Gamma(x) = \forall \Xi_1.\, \tau_1$

Let $\rho = [\overline{\alpha \mapsto \gamma_\alpha}^{\alpha \in S}]$. We have $\Xi^0{\cdot}\rho\Xi_1 \vDash \rho\Xi_1$ by S-Cons and S-Hyp. We also have $\Xi^0{\cdot}\rho\Xi_1 \vdash \rho\tau_1 \leqslant \rho\tau_1$ by S-Refl. Then we have $\Xi^0{\cdot}\rho\Xi_1 \vdash \forall \Xi_1.\, \tau_1 \leqslant^{\forall} \rho\tau_1$ by S-All, and by S-Var2, we have $\Xi^0{\cdot}\rho\Xi_1, \Gamma \vdash x : \rho\tau_1$ Since $\Gamma$ ***cons.***, we have $[\overline{\alpha \mapsto \tau_\alpha}^{\alpha \in S}]\Xi_1$ ***cons.*** for some $\overline{\tau_\alpha}^{\alpha \in S}$. Since $\overline{\gamma_\alpha \ \textbf{\textit{fresh}}}^{\alpha \in S}$, we have $[\overline{\alpha \mapsto \tau_\alpha}^{\alpha \in S}]\Xi_1 = [\overline{\gamma_\alpha \mapsto \tau_\alpha}^{\alpha \in S}]\rho\Xi_1$. Then $[\overline{\alpha \mapsto \tau_\alpha}^{\alpha \in S}]\Xi_1$ ***cons.*** implies $\rho\Xi_1$ ***cons.***. Similarly, we also have $\rho\Xi_1$ ***guard.***.

**Case I-Abs.** By straightforward applications of the IH.

**Cases I-App I-Asc, I-Case1.** By analogous reasoning to the I-Proj case, applying the IH and sound constraining (Lemma 7.5) successively on the premises, threading the inferred constraints through and weakening accordingly.

**Case I-Case2.** $t = \textbf{case}\ x = t_1\ \textbf{of}\ \_ \to t_2$

By IH, we have $\Xi_0{\cdot}\Xi_1$ ***cons.*** and $\Xi_0{\cdot}\Xi_1$ ***guard.*** and $\Xi_0{\cdot}\Xi_1, \Gamma \vdash t_1 : \tau_1$, which implies $\Xi_0{\cdot}\Xi_1{\cdot}\Xi_2{\cdot}\Xi_3, \Gamma \vdash t_1 : \tau_1$ by weakening. By sound constraining (Lemma 7.5), we have $\Xi_0{\cdot}\Xi_1{\cdot}\Xi_2$ ***cons.*** and $\Xi_0{\cdot}\Xi_1{\cdot}\Xi_2$ ***guard.*** and $\Xi_0{\cdot}\Xi_1{\cdot}\Xi_2 \vdash \tau_1 \leqslant \#C$, which implies $\Xi_0{\cdot}\Xi_1{\cdot}\Xi_2{\cdot}\Xi_3 \vdash \tau_1 \leqslant \tau_1 \wedge \#C$ by weakening S-AndOr2$\supset$ with S-Refl. Then by T-Subs, we have $\Xi_0{\cdot}\Xi_1{\cdot}\Xi_2{\cdot}\Xi_3, \Gamma \vdash t_1 : \tau_1 \wedge \#C$. By IH, we have $\Xi_0{\cdot}\Xi_1{\cdot}\Xi_2{\cdot}\Xi_3$ ***cons.*** and $\Xi_0{\cdot}\Xi_1{\cdot}\Xi_2{\cdot}\Xi_3$ ***guard.*** and $\Xi_0{\cdot}\Xi_1{\cdot}\Xi_2{\cdot}\Xi_3, \Gamma{\cdot}(x : \tau_1) \vdash t_2 : \tau$. Therefore, by T-Case2, we have $\Xi_0{\cdot}\Xi_1{\cdot}\Xi_2{\cdot}\Xi_3, \Gamma \vdash \textbf{case}\ x = t_1\ \textbf{of}\ \_ \to t_2 : \tau$.

**Case I-Case3.** $t = \textbf{case}\ x = t_1\ \textbf{of}\ C \to t_2,\ M$

By IH, we have $\Xi_0{\cdot}\Xi_1$ ***cons.*** and $\Xi_0{\cdot}\Xi_1$ ***guard.*** and $\Xi_0{\cdot}\Xi_1, \Gamma \vdash t_1 : \tau_1$, which implies $\Xi_0{\cdot}\Xi_1{\cdot}\Xi_2{\cdot}\Xi_3{\cdot}\Xi_4, \Gamma \vdash t_1 : \tau_1$ by weakening. Then by IH, we have $\Xi_0{\cdot}\Xi_1{\cdot}\Xi_2$ ***cons.*** and $\Xi_0{\cdot}\Xi_1{\cdot}\Xi_2$ ***guard.*** and $\Xi_0{\cdot}\Xi_1{\cdot}\Xi_2, \Gamma{\cdot}(x : \alpha) \vdash t_2 : \tau_2$, which implies $\Xi_0{\cdot}\Xi_1{\cdot}\Xi_2{\cdot}\Xi_3{\cdot}\Xi_4, \Gamma{\cdot}(x : \alpha) \vdash t_2 : \tau_2 \vee \tau_3$ by weakening and S-Trans with S-AndOr11$\cdot$. Then by IH again, we have $\Xi_0{\cdot}\Xi_1{\cdot}\Xi_2{\cdot}\Xi_3$ ***cons.*** and $\Xi_0{\cdot}\Xi_1{\cdot}\Xi_2{\cdot}\Xi_3$ ***guard.*** and $\Xi_0{\cdot}\Xi_1{\cdot}\Xi_2{\cdot}\Xi_3, \Gamma{\cdot}(x : \beta) \vdash \textbf{case}\ x = x\ \textbf{of}\ M : \tau_3$, which implies $\Xi_0{\cdot}\Xi_1{\cdot}\Xi_2{\cdot}\Xi_3{\cdot}\Xi_4, \Gamma{\cdot}(x : \beta) \vdash \textbf{case}\ x = x\ \textbf{of}\ M : \tau_2 \vee \tau_3$ by weakening and S-Trans with S-AndOr12$\cdot$. By sound constraining (Lemma 7.5), we have $\Xi_4{\cdot}\Xi_0{\cdot}\Xi_1{\cdot}\Xi_2{\cdot}\Xi_3$ ***cons.*** and $\Xi_4{\cdot}\Xi_0{\cdot}\Xi_1{\cdot}\Xi_2{\cdot}\Xi_3$ ***guard.*** and $\Xi_4{\cdot}\Xi_0{\cdot}\Xi_1{\cdot}\Xi_2{\cdot}\Xi_3 \vdash \tau_1 \leqslant \#C \wedge \alpha \vee \neg\#C \wedge \beta$, which imply $\Xi_0{\cdot}\Xi_1{\cdot}\Xi_2{\cdot}\Xi_3{\cdot}\Xi_4$ ***cons.*** and $\Xi_0{\cdot}\Xi_1{\cdot}\Xi_2{\cdot}\Xi_3{\cdot}\Xi_4$ ***guard.*** and $\Xi_0{\cdot}\Xi_1{\cdot}\Xi_2{\cdot}\Xi_3{\cdot}\Xi_4 \vdash \tau_1 \leqslant \#C \wedge \alpha \vee \neg\#C \wedge \beta$ by commutation. Then by T-Subs, we have $\Xi_0{\cdot}\Xi_1{\cdot}\Xi_2{\cdot}\Xi_3{\cdot}\Xi_4, \Gamma \vdash t_1 : \#C \wedge \alpha \vee \neg\#C \wedge \beta$. Therefore, by T-Case3, we have $\Xi_0{\cdot}\Xi_1{\cdot}\Xi_2{\cdot}\Xi_3{\cdot}\Xi_4, \Gamma \vdash \textbf{case}\ x = t_1\ \textbf{of}\ C \to t_2,\ M : \tau_2 \vee \tau_3$.

■

**Proof** [Proof 7.5 (Soundness of Constraining)] By Lemma C.24 and Theorem C.25. ■

**Lemma C.24** (Sufficiency of Constraining)**.**

1. *If* $\Sigma \vdash \tau_1 \ll \tau_2 \Rightarrow \Xi$ *and* $\tau_1, \tau_2$ ***wf*** *and* ***err*** $\notin \Xi$*, then* $\Xi{\cdot}\Sigma \vdash \tau_1 \leqslant \tau_2$.
2. *If* $\Sigma \vdash \mathrm{D}^0 \Rightarrow \Xi$ *and* $\mathrm{D}^0$ ***wf*** *and* ***err*** $\notin \Xi$*, then* $\Xi{\cdot}\Sigma \vdash \mathrm{D}^0 \leqslant \bot$.

**Proof**

By induction on constraining derivations.

**Case C-Hyp.** Immediate by S-Hyp.

**Case C-Assum.** By IH on the latter premise, we have $\Xi{\cdot}\Sigma{\cdot}\triangleright(\tau_1 \leqslant \tau_2) \vdash \mathrm{dnf}^0(\tau_1 \wedge \neg\tau_2) \leqslant \bot$. By Lemma 7.3, we have $\mathrm{dnf}^0(\tau_1 \wedge \neg\tau_2) \equiv \tau_1 \wedge \neg\tau_2$. Then we have $\Xi{\cdot}\Sigma{\cdot}\triangleright(\tau_1 \leqslant \tau_2) \vdash \tau_1 \wedge \neg\tau_2 \leqslant \bot$, which implies $\Xi{\cdot}\Sigma{\cdot}\triangleright(\tau_1 \leqslant \tau_2) \vdash \tau_1 \leqslant \tau_2$ by Theorem A.9, which implies $\Xi{\cdot}\Sigma \vdash \tau_1 \leqslant \tau_2$ by S-Assum.

**Case C-Or.** Then $\mathrm{D}^0 = \mathrm{D}^0_1 \vee \mathrm{C}^0_1$ for some $\mathrm{D}^0_1$ and $\mathrm{C}^0_1$, and $\Xi = \Xi_1{\cdot}\Xi_2$ for some $\Xi_1$ and $\Xi_2$. By IH on the former premise, we have $\Xi_1{\cdot}\Sigma \vdash \mathrm{D}^0_1 \leqslant \bot$. By IH on the latter premise, we have $\Xi_2{\cdot}\Xi_1{\cdot}\Sigma \vdash \mathrm{C}^0_1 \leqslant \bot$, which implies $\Xi{\cdot}\Sigma \vdash \mathrm{C}^0_1 \leqslant \bot$ by commutation. $\Xi_1{\cdot}\Sigma \vdash \mathrm{D}^0_1 \leqslant \bot$ implies $\Xi{\cdot}\Sigma \vdash \mathrm{D}^0_1 \leqslant \bot$ by Lemma A.23. Then by S-AndOr2$\cdot$, we have $\Xi{\cdot}\Sigma \vdash \mathrm{D}^0_1 \vee \mathrm{C}^0_1 \leqslant \bot$.

**Case C-Bot.** Immediate by S-Refl.

**Case C-Cls1.** Then $\mathrm{D}^0 = \mathcal{I}[\#C_1] \wedge \neg(\mathrm{U} \vee \#C_2)$ for some $C_1$ and $C_2$ and U. From the premise, we have $\Sigma \vdash \#C_1 \leqslant \#C_2$ by S-ClsSub, which implies $\Sigma \vdash \#C_1 \wedge \mathcal{F} \wedge \mathcal{R} \leqslant \mathrm{U} \vee \#C_2$ by S-Trans with S-AndOr11$\supset$ and S-AndOr12$\cdot$, which implies $\Sigma \vdash \#C_1 \wedge \mathcal{F} \wedge \mathcal{R} \wedge \neg(\mathrm{U} \vee \#C_2) \leqslant \bot$ by Theorem A.9, i.e., $\Sigma \vdash \mathcal{I}[\#C_1] \wedge \neg(\mathrm{U} \vee \#C_2) \leqslant \bot$.

**Case C-Cls2.** Then $\mathrm{D}^0 = \mathcal{I}[\#C_1] \wedge \neg(\mathrm{U} \vee \#C_2)$ for some $C_1$ and $C_2$ and U. By IH on the latter premise, we have $\Xi\cdot\Sigma \vdash \mathcal{I}[\#C_1] \wedge \neg\mathrm{U} \leqslant \bot$. Since $\neg(\mathrm{U} \vee \#C_2) \leqslant \neg\mathrm{U}$ by S-AndOr11$\cdot$ and S-NegInv, we have $\Xi\cdot\Sigma \vdash \mathcal{I}[\#C_1] \wedge \neg(\mathrm{U} \vee \#C_2) \leqslant \mathcal{I}[\#C_1] \wedge \neg\mathrm{U}$ by Lemma A.7$\supset$ with S-Refl. Then we have $\Xi\cdot\Sigma \vdash \mathcal{I}[\#C_1] \wedge \neg(\mathrm{U} \vee \#C_2) \leqslant \bot$ by S-Trans.

**Case C-Cls3.** Then $\mathrm{D}^0 = \mathcal{I}^{\mathcal{N}}[\top] \wedge \neg(\mathrm{U} \vee \#C)$ for some $C$ and U. By IH on the premise, we have $\Xi\cdot\Sigma \vdash \mathcal{I}^{\mathcal{N}}[\top] \wedge \neg\mathrm{U} \leqslant \bot$. Since $\neg(\mathrm{U} \vee \#C) \leqslant \neg\mathrm{U}$ by S-AndOr11$\cdot$ and S-NegInv, we have $\Xi\cdot\Sigma \vdash \mathcal{I}^{\mathcal{N}}[\top] \wedge \neg(\mathrm{U} \vee \#C) \leqslant \mathcal{I}^{\mathcal{N}}[\top] \wedge \neg\mathrm{U}$ by Lemma A.7$\supset$ with S-Refl. Then we have $\Xi\cdot\Sigma \vdash \mathcal{I}^{\mathcal{N}}[\top] \wedge \neg(\mathrm{U} \vee \#C) \leqslant \bot$ by S-Trans.

**Case C-Fun1.** Then $\mathrm{D}^0 = \mathcal{I}[\mathrm{D}_1 \to \mathrm{D}_2] \wedge \neg(\mathrm{D}_3 \to \mathrm{D}_4)$ for some $\mathrm{D}_1$ and $\mathrm{D}_2$ and $\mathrm{D}_3$ and $\mathrm{D}_4$, and $\Xi = \Xi_1\cdot\Xi_2$ for some $\Xi_1$ and $\Xi_2$. By IH on the former premise, we have $\Xi_1\cdot\triangleleft\Sigma \vdash \mathrm{D}_3 \leqslant \mathrm{D}_1$, which implies $\triangleleft(\Xi\cdot\Sigma) \vdash \mathrm{D}_3 \leqslant \mathrm{D}_1$ by Lemma A.23. By IH on the latter premise, we have $\Xi_2\cdot\Xi_1\cdot\triangleleft\Sigma \vdash \mathrm{D}_2 \leqslant \mathrm{D}_4$, which implies $\triangleleft(\Xi\cdot\Sigma) \vdash \mathrm{D}_2 \leqslant \mathrm{D}_4$ by Lemma A.23. Then by S-FunDepth, we have $\Xi\cdot\Sigma \vdash \mathrm{D}_1 \to \mathrm{D}_2 \leqslant \mathrm{D}_3 \to \mathrm{D}_4$, which implies $\Xi\cdot\Sigma \vdash \mathcal{N} \wedge \mathrm{D}_1 \to \mathrm{D}_2 \wedge \mathcal{R} \leqslant \mathrm{D}_3 \to \mathrm{D}_4$ by S-Trans with S-AndOr11$\supset$ and S-AndOr12$\supset$, i.e., $\Xi\cdot\Sigma \vdash \mathcal{I}[\mathrm{D}_1 \to \mathrm{D}_2] \leqslant \mathrm{D}_3 \to \mathrm{D}_4$, which implies $\Xi\cdot\Sigma \vdash \mathcal{I}[\mathrm{D}_1 \to \mathrm{D}_2] \wedge \neg(\mathrm{D}_3 \to \mathrm{D}_4) \leqslant \bot$ by Theorem A.9.

**Case C-Rcd1.** Then $\mathrm{D}^0 = \mathcal{I}[\{\, \overline{x : \mathrm{D}_x}^{x \in S} \,\}] \wedge \neg\{\, y : \mathrm{D} \,\}$ for some $\overline{\mathrm{D}_x}^{x \in S}$ and D. By IH on the premise, we have $\Xi\cdot\triangleleft\Sigma \vdash \mathrm{D}_y \leqslant \mathrm{D}$, which implies $\triangleleft(\Xi\cdot\Sigma) \vdash \mathrm{D}_y \leqslant \mathrm{D}$ by Lemma A.23. Then by S-RcdDepth, we have $\Xi\cdot\Sigma \vdash \{\, y : \mathrm{D}_y \,\} \leqslant \{\, y : \mathrm{D} \,\}$, which implies $\Xi\cdot\Sigma \vdash \mathcal{N} \wedge \mathcal{F} \wedge \{\, \overline{x : \mathrm{D}_x}^{x \in S} \,\} \leqslant \{\, y : \mathrm{D} \,\}$ by S-Trans with S-AndOr11$\supset$ and S-AndOr12$\supset$, i.e., $\Xi\cdot\Sigma \vdash \mathcal{I}[\{\, \overline{x : \mathrm{D}_x}^{x \in S} \,\}] \leqslant \{\, y : \mathrm{D} \,\}$, which implies $\Xi\cdot\Sigma \vdash \mathcal{I}[\{\, \overline{x : \mathrm{D}_x}^{x \in S} \,\}] \wedge \neg\{\, y : \mathrm{D} \,\} \leqslant \bot$ by Theorem A.9.

**Cases C-NotBot, C-Fun2, C-Rcd2, C-Rcd3.** Then $\mathit{err} \in \Xi$.

**Case C-Var1.** By S-Hyp, we have $\Xi\cdot(\alpha \leqslant \neg\mathrm{C})\cdot\Sigma \vdash \alpha \leqslant \neg\mathrm{C}$, which implies $\Xi\cdot(\alpha \leqslant \neg\mathrm{C})\cdot\Sigma \vdash \mathrm{C} \wedge \alpha \leqslant \bot$ by Theorem A.9.

**Case C-Var2.** By S-Hyp, we have $\Xi\cdot(\alpha \leqslant \mathrm{C})\cdot\Sigma \vdash \alpha \leqslant \mathrm{C}$, which implies $\Xi\cdot(\alpha \leqslant \mathrm{C})\cdot\Sigma \vdash \alpha \wedge \neg\mathrm{C} \leqslant \bot$ by Theorem A.9.

■

**Theorem C.25** (Consistency of constraining)**.** *If* $\Xi$ ***cons.*** *and* $\Xi$ ***guard.*** *and* $\Xi \vdash \tau \ll \pi \Rightarrow \Xi'$ *and* $\mathit{err} \notin \Xi'$*, then* $\Xi\cdot\Xi'$ ***cons.*** *and* $\Xi\cdot\Xi'$ ***guard.***.

**Proof** By Lemma C.26. ■

In the remainder of this section, we consider the reformulated type constraining rules in Figure 27. In these rules, we assume that we always start derivations with an empty $\Sigma$, so that we start only with bounds, and all these bounds are in $\Xi$. It is easy to see that they are equivalent to the ones presented in Figure 20.

$\boxed{\Xi, \Sigma \vdash \tau \ll \tau \Rightarrow \Xi}$

C-Hyp
$$\frac{(\tau_1 \leqslant \tau_2) \in \Xi{\cdot}\Sigma}{\Xi, \Sigma \vdash \tau_1 \ll \tau_2 \Rightarrow \epsilon}$$

C-Assum
$$\frac{(\tau_1 \leqslant \tau_2) \notin \Xi{\cdot}\Sigma \qquad \Xi, \Sigma{\cdot}\triangleright(\tau_1 \leqslant \tau_2) \vdash \mathrm{dnf}^0(\tau_1 \wedge \neg\tau_2) \Rightarrow \Xi'}{\Xi, \Sigma \vdash \tau_1 \ll \tau_2 \Rightarrow \Xi'}$$

$\boxed{\Xi, \Sigma \vdash \mathrm{D}^0 \Rightarrow \Xi}$

C-Or
$$\frac{\Xi, \Sigma \vdash \mathrm{D}^0 \Rightarrow \Xi' \qquad \Xi{\cdot}\Xi', \Sigma \vdash \mathrm{C}^0 \Rightarrow \Xi''}{\Xi, \Sigma \vdash \mathrm{D}^0 \vee \mathrm{C}^0 \Rightarrow \Xi'{\cdot}\Xi''}$$

C-Bot
$$\frac{}{\Xi, \Sigma \vdash \bot \Rightarrow \epsilon}$$

C-NotBot
$$\frac{}{\Xi, \Sigma \vdash \mathrm{I}^0 \wedge \neg\bot \Rightarrow \textbf{\textit{err}}}$$

C-Cls1
$$\frac{C_2 \in \mathcal{S}(\#C_1)}{\Xi, \Sigma \vdash \mathcal{I}[\#C_1] \wedge \neg(\mathrm{U} \vee \#C_2) \Rightarrow \epsilon}$$

C-Cls2
$$\frac{C_2 \notin \mathcal{S}(\#C_1) \qquad \Xi, \Sigma \vdash \mathcal{I}[\#C_1] \wedge \neg\mathrm{U} \Rightarrow \Xi'}{\Xi, \Sigma \vdash \mathcal{I}[\#C_1] \wedge \neg(\mathrm{U} \vee \#C_2) \Rightarrow \Xi'}$$

C-Cls3
$$\frac{\Xi, \Sigma \vdash \mathcal{I}^{\mathcal{N}}[\top] \wedge \neg\mathrm{U} \Rightarrow \Xi'}{\Xi, \Sigma \vdash \mathcal{I}^{\mathcal{N}}[\top] \wedge \neg(\mathrm{U} \vee \#C) \Rightarrow \Xi'}$$

C-Fun1
$$\frac{\Xi, {\triangleleft}\Sigma \vdash \mathrm{D}_3 \ll \mathrm{D}_1 \Rightarrow \Xi' \qquad \Xi{\cdot}\Xi', {\triangleleft}\Sigma \vdash \mathrm{D}_2 \ll \mathrm{D}_4 \Rightarrow \Xi''}{\Xi, \Sigma \vdash \mathcal{I}[\mathrm{D}_1 \to \mathrm{D}_2] \wedge \neg(\mathrm{D}_3 \to \mathrm{D}_4) \Rightarrow \Xi'{\cdot}\Xi''}$$

C-Fun2
$$\frac{}{\Xi, \Sigma \vdash \mathcal{I}^{\to}[\top] \wedge \neg(\mathrm{D}_1 \to \mathrm{D}_2) \Rightarrow \textbf{\textit{err}}}$$

C-Rcd1
$$\frac{y \in S \qquad \Xi, {\triangleleft}\Sigma \vdash \mathrm{D}_y \ll \mathrm{D} \Rightarrow \Xi'}{\Xi, \Sigma \vdash \mathcal{I}[\{\, \overline{x : \mathrm{D}_x}^{\,x \in S} \,\}] \wedge \neg\{\, y : \mathrm{D} \,\} \Rightarrow \Xi'}$$

C-Rcd2
$$\frac{y \notin S}{\Xi, \Sigma \vdash \mathcal{I}[\{\, \overline{x : \mathrm{D}_x}^{\,x \in S} \,\}] \wedge \neg\{\, y : \mathrm{D} \,\} \Rightarrow \textbf{\textit{err}}}$$

C-Rcd3
$$\frac{}{\Xi, \Sigma \vdash \mathcal{I}^{\{\}}[\top] \wedge \neg\{\, x : \mathrm{D} \,\} \Rightarrow \textbf{\textit{err}}}$$

C-Var1
$$\frac{\Xi{\cdot}(\alpha \leqslant \neg\mathrm{C}), \Sigma \vdash \mathit{lb}_{\Xi}(\alpha) \ll \neg\mathrm{C} \Rightarrow \Xi'}{\Xi, \Sigma \vdash \mathrm{C} \wedge \alpha \Rightarrow \Xi'{\cdot}(\alpha \leqslant \neg\mathrm{C})}$$

C-Var2
$$\frac{\Xi{\cdot}(\mathrm{C} \leqslant \alpha), \Sigma \vdash \mathrm{C} \ll \mathit{ub}_{\Xi}(\alpha) \Rightarrow \Xi'}{\Xi, \Sigma \vdash \mathrm{C} \wedge \neg\alpha \Rightarrow \Xi'{\cdot}(\mathrm{C} \leqslant \alpha)}$$

Fig. 27. Reformulated normal form constraining rules. The only difference with the rules of Figure 20 is that we now explicitly split the subtyping context into a constraining part $\Xi$ and a plain subtyping part $\Sigma$.

**Lemma C.26** (Consistency of constraining)**.**

1. *If* ${\triangleleft}\Sigma{\cdot}\Delta \vdash \Xi$ ; $\rho$ ***cons.*** *and* $\Xi$ ***guard.*** *and* $\Xi, \Sigma \vdash \tau \ll \pi \Rightarrow \Xi'$ *and* $\textbf{\textit{err}} \notin \Xi'$*, then* ${\triangleleft}\Sigma{\cdot}\Delta \vdash \Xi{\cdot}\Xi'$ ; $\rho'$ ***cons.*** *and* $\Xi{\cdot}\Xi'$ ***guard.*** *for some* $\rho'$*.*
2. *If* ${\triangleleft}\Sigma{\cdot}\Delta \vdash \Xi$ ; $\rho$ ***cons.*** *and* $\Xi$ ***guard.*** *and* $\Xi, \Sigma \vdash \bigvee_{i \in 1..n} \mathrm{C}_i^0 \Rightarrow \Xi'$ *and* $\overline{\mathit{TTV}'(\mathrm{C}_i^0) \textit{ are distinct}}^{\,i \in 1..n}$ *and* $\textbf{\textit{err}} \notin \Xi'$*, then* ${\triangleleft}\Sigma{\cdot}\Delta \vdash \Xi{\cdot}\Xi'$ ; $\rho'$ ***cons.*** *and* $\Xi{\cdot}\Xi'$ ***guard.*** *for some* $\rho'$*.*

**Proof** By induction on constraining derivations.

**Cases C-Hyp, C-Bot, C-Cls1.** Immediate since $\Xi' = \epsilon$.
**Case C-Assum.** Then the premise of the rule is:

$$\Xi, \Sigma{\cdot}\triangleright(\tau \leqslant \pi) \vdash \mathrm{dnf}^0(\tau \wedge \neg\pi) \Rightarrow \Xi' \tag{1}$$

From the assumptions, we have:

$$\triangleleft\Sigma{\cdot}\Delta \vdash \Xi\,;\ \rho\ \textbf{\textit{cons.}} \tag{2}$$

By Lemma A.26 with Lemma A.18, (2) implies:

$$\triangleleft(\Sigma{\cdot}\triangleright(\tau \leqslant \pi)){\cdot}\Delta \vdash \Xi\,;\ \rho\ \textbf{\textit{cons.}} \tag{3}$$

Then by IH on (3) and (1), we have:

$$\triangleleft(\Sigma{\cdot}\triangleright(\tau \leqslant \pi)){\cdot}\Delta \vdash \Xi{\cdot}\Xi'\,;\ \rho'\ \textbf{\textit{cons.}}$$
$$\text{i.e.,}\quad \triangleleft\Sigma{\cdot}(\tau \leqslant \pi){\cdot}\Delta \vdash \Xi{\cdot}\Xi'\,;\ \rho'\ \textbf{\textit{cons.}} \tag{4}$$

for some $\rho'$. By Lemma C.24, $\Xi, \Sigma \vdash \tau \ll \pi \Rightarrow \Xi'$ implies:

$$\Xi{\cdot}\Xi'{\cdot}\Sigma \vdash \tau \leqslant \pi \tag{5}$$

By Lemma A.23 with Lemma A.18, (5) implies:

$$\Xi{\cdot}\Xi'{\cdot}\triangleleft\Sigma{\cdot}\Delta \vdash \tau \leqslant \pi \tag{6}$$

Then by Lemma A.26 with (6), (4) implies:

$$\triangleleft\Sigma{\cdot}\Delta \vdash \Xi{\cdot}\Xi'\,;\ \rho'\ \textbf{\textit{cons.}} \tag{7}$$

**Case C-Or.** Then the premises of the rule are:

$$\Xi, \Sigma \vdash \bigvee_{i \in 1..n-1} \mathrm{C}_i^0 \Rightarrow \Xi_1' \tag{8}$$
$$\Xi{\cdot}\Xi_1', \Sigma \vdash \mathrm{C}_n^0 \Rightarrow \Xi_2' \tag{9}$$

where $\Xi' = \Xi_1'{\cdot}\Xi_2'$. Then by IH on (8), we have:

$$\triangleleft\Sigma{\cdot}\Delta \vdash \Xi{\cdot}\Xi_1'\,;\ \rho''\ \textbf{\textit{cons.}} \tag{10}$$

for some $\rho''$. Then by IH on (10) and (9), we have:

$$\triangleleft\Sigma{\cdot}\Delta \vdash \Xi{\cdot}\Xi_1'{\cdot}\Xi_2'\,;\ \rho'\ \textbf{\textit{cons.}}$$
$$\text{i.e.,}\quad \triangleleft\Sigma{\cdot}\Delta \vdash \Xi{\cdot}\Xi'\,;\ \rho'\ \textbf{\textit{cons.}} \tag{11}$$

for some $\rho'$.
**Cases C-Cls2, C-Cls3, C-Rcd1.** Immediate by IH on the premise.
**Case C-Fun1.** Similar to case C-Or.
**Case C-Var1.** Then the premise of the rule is:

$$\Xi{\cdot}(\alpha \leqslant \neg\mathrm{C}), \Sigma \vdash lb_\Xi(\alpha) \ll \neg\mathrm{C} \Rightarrow \Xi_1' \tag{12}$$

where $\bigvee_{i \in 1..n} \mathrm{C}_\mathrm{i}^0 = \mathrm{C} \wedge \alpha$ and $\Xi' = \Xi_1'{\cdot}(\alpha \leqslant \neg\mathrm{C})$ for some $\alpha$ and C and $\Xi_1'$. From the assumption, we have:

$$\triangleleft\Sigma{\cdot}\Delta \vdash \Xi\,;\ \rho\ \textbf{\textit{cons.}} \tag{13}$$

By Lemma A.26 with Lemma A.18, (13) implies:

$$(\alpha \leqslant \neg \mathrm{C}) \cdot \Xi_1' \cdot \lhd \Sigma \cdot \Delta \vdash \Xi \,;\, \rho \ \textbf{\textit{cons.}} \tag{14}$$

Since $TTV'(\mathrm{C} \wedge \alpha)$ are distinct, by the syntax of RDNF, we have $\alpha \notin TTV(\mathrm{C})$. Then we have:

$$\Xi \cdot (\alpha \leqslant \neg \mathrm{C}) \ \textbf{\textit{guard.}} \tag{15}$$

Since (15) implies $\alpha \notin TTV(lb_{\Xi}(\alpha)) \cup TTV(\neg \mathrm{C})$, by Lemma C.28 on (12) followed by Lemma A.23, we have:

$$\rhd \Xi_\alpha \cdot \rhd (\alpha \leqslant \neg \mathrm{C}) \cdot \rho'_\alpha (\Xi_{\not\alpha} \cdot \Xi_1' \cdot \Sigma) \vdash lb_{\Xi}(\alpha) \leqslant \neg \mathrm{C} \tag{16}$$

where $split_\alpha(\Xi,\ dom(\rho) \backslash \{\, \alpha \,\}) = (\Xi_\alpha,\ \Xi_{\not\alpha})$ and $\rho'_\alpha = [\alpha \mapsto \alpha \wedge ub_{\Xi \cdot (\alpha \leqslant \tau)}(\alpha) \vee lb_{\Xi \cdot (\alpha \leqslant \tau)}(\alpha)]$. By Lemma A.23 with Lemma A.18, (16) implies:

$$\rhd \Xi_\alpha \cdot \rhd (\alpha \leqslant \neg \mathrm{C}) \cdot \rho'_\alpha (\Xi_{\not\alpha} \cdot \Xi_1' \cdot \lhd \Sigma) \vdash lb_{\Xi}(\alpha) \leqslant \neg \mathrm{C} \tag{17}$$

Then by Lemma C.27 on (14), (15), and (17), we have:

$$\begin{aligned} &\Xi_1' \cdot \lhd \Sigma \cdot \Delta \vdash \Xi \cdot (\alpha \leqslant \neg \mathrm{C}) \,;\, \rho' \ \textbf{\textit{cons.}} \\ \text{i.e.,}\quad &\lhd \Sigma \cdot (\Xi_1' \cdot \Delta) \vdash \Xi \cdot (\alpha \leqslant \neg \mathrm{C}) \,;\, \rho' \ \textbf{\textit{cons.}} \end{aligned} \tag{18}$$

for some $\rho'$. Then by IH on (18) and (12), we have:

$$\lhd \Sigma \cdot (\Xi_1' \cdot \Delta) \vdash \Xi \cdot (\alpha \leqslant \neg \mathrm{C}) \cdot \Xi_1' \,;\, \rho' \ \textbf{\textit{cons.}} \tag{19}$$

By Lemma A.18, we have:

$$\Xi \cdot (\alpha \leqslant \neg \mathrm{C}) \cdot \Xi_1' \cdot \lhd \Sigma \cdot \Delta \models \lhd \Sigma \cdot (\Xi_1' \cdot \Delta) \tag{20}$$

Then by Lemma A.26 with (20), (19) implies:

$$\begin{aligned} &\lhd \Sigma \cdot \Delta \vdash \Xi \cdot (\alpha \leqslant \neg \mathrm{C}) \cdot \Xi_1' \,;\, \rho' \ \textbf{\textit{cons.}} \\ \text{i.e.,}\quad &\lhd \Sigma \cdot \Delta \vdash \Xi \cdot \Xi' \,;\, \rho' \ \textbf{\textit{cons.}} \end{aligned} \tag{21}$$

**Case C-Var2.** Similar to case C-Var1.

■

**Lemma C.27.** *If* $(\alpha \leqslant^\diamond \tau) \cdot \Sigma \vdash \rhd \Xi_\rhd \cdot \Xi \,;\, \rho$ ***cons.*** *and* $\Xi \cdot (\alpha \leqslant^\diamond \tau)$ ***guard.*** *and* $\rhd \Xi_\rhd \cdot \rhd \Xi_\alpha \cdot \rhd (\alpha \leqslant^\diamond \tau) \cdot \rho'_\alpha (\Xi_{\not\alpha} \cdot \Sigma) \vdash lb^\diamond_\Xi(\alpha) \leqslant^\diamond \tau$, *where* $split_\alpha(\Xi,\ dom(\rho) \backslash \{\, \alpha \,\}) = (\Xi_\alpha,\ \Xi_{\not\alpha})$ *and* $\rho'_\alpha = [\alpha \mapsto \alpha \wedge ub_{\Xi \cdot (\alpha \leqslant^\diamond \tau)}(\alpha) \vee lb_{\Xi \cdot (\alpha \leqslant^\diamond \tau)}(\alpha)]$, *then* $\Sigma \vdash \rhd \Xi_\rhd \cdot \Xi \cdot (\alpha \leqslant^\diamond \tau) \,;\, \rho'$ ***cons.*** *for some* $\rho'$.

The proof for the $\cdot$ direction is shown below. The $\wr$ direction is symmetric.

**Proof**

By Lemma A.37, $(\alpha \leqslant \tau) \cdot \Sigma \vdash \rhd \Xi_\rhd \cdot \Xi \,;\, \rho$ ***cons.*** implies:

$$\rhd \Xi_\rhd \cdot \rhd \Xi_\alpha \cdot \rho_\alpha \Xi_{\not\alpha} \cdot \rho_\alpha ((\alpha \leqslant \tau) \cdot \Sigma) \models \rho_\alpha \Xi_\alpha \tag{1}$$

$$\rho_\alpha ((\alpha \leqslant \tau) \cdot \Sigma) \vdash \rhd \Xi_\rhd \cdot \rhd \Xi_\alpha \cdot \rho_\alpha \Xi_{\not\alpha} \,;\, \rho'_1 \ \textbf{\textit{cons.}} \tag{2}$$

for some $\rho'_1$, where $\rho_\alpha = [\alpha \mapsto \alpha \wedge ub_\Xi(\alpha) \vee lb_\Xi(\alpha)]$ and $dom(\rho'_1) = dom(\rho) \backslash \{\, \alpha \,\}$.

Let $\rho_\tau = [\alpha \mapsto \alpha \wedge \tau]$. By Lemma A.29 on (1), we have:

$$\rho_\tau(\triangleright\Xi_\triangleright \cdot \triangleright\Xi_\alpha \cdot \rho_\alpha \Xi_{\not\alpha} \cdot \rho_\alpha((\alpha \leqslant \tau) \cdot \Sigma)) \vDash \rho_\tau \rho_\alpha \Xi_\alpha \tag{3}$$

By Corollary A.33 and Corollary A.34, we have:

$$(\alpha \leqslant \tau) \vdash \pi \equiv \rho_\tau \pi \quad \textit{for all } \pi \tag{4}$$

$$\triangleright(\alpha \leqslant \tau) \vdash \pi \equiv \rho_\tau \pi \quad \textit{for all } \pi \textit{ where } \alpha \notin \mathit{TTV}(\pi) \tag{5}$$

By S-Trans on Lemma A.18 and (4), we have:

$$\triangleright\Xi_\triangleright \cdot \triangleright\Xi_\alpha \cdot \triangleright(\alpha \leqslant \tau) \vDash \rho_\tau(\triangleright\Xi_\triangleright \cdot \triangleright\Xi_\alpha) \tag{6}$$

Then by Lemma A.23 on (3) with (6), we have:

$$\triangleright\Xi_\triangleright \cdot \triangleright\Xi_\alpha \cdot \triangleright(\alpha \leqslant \tau) \cdot \rho_\tau \rho_\alpha(\Xi_{\not\alpha} \cdot (\alpha \leqslant \tau) \cdot \Sigma) \vDash \rho_\tau \rho_\alpha \Xi_\alpha \tag{7}$$

Expanding the composition, we have:

$$\rho_\tau \circ \rho_\alpha = [\alpha \mapsto \alpha \wedge \tau \wedge \rho_\tau \mathit{ub}_\Xi(\alpha) \vee \rho_\tau \mathit{lb}_\Xi(\alpha)] \tag{8}$$

By Lemma A.7 on S-Refl and (5), we have:

$$\triangleright(\alpha \leqslant \tau) \vdash \alpha \wedge \tau \wedge \mathit{ub}_\Xi(\alpha) \vee \mathit{lb}_\Xi(\alpha) \equiv \alpha \wedge \tau \wedge \rho_\tau \mathit{ub}_\Xi(\alpha) \vee \rho_\tau \mathit{lb}_\Xi(\alpha)$$

$$\text{i.e.,} \quad \triangleright(\alpha \leqslant \tau) \vdash \alpha \wedge \mathit{ub}_{\Xi \cdot (\alpha \leqslant \tau)}(\alpha) \vee \mathit{lb}_{\Xi \cdot (\alpha \leqslant \tau)}(\alpha) \equiv \alpha \wedge \tau \wedge \rho_\tau \mathit{ub}_\Xi(\alpha) \vee \rho_\tau \mathit{lb}_\Xi(\alpha) \tag{9}$$

Then by Lemma A.31 on (9), we have:

$$\triangleright(\alpha \leqslant \tau) \vdash \rho'_\alpha \pi \equiv \rho_\tau \rho_\alpha \pi \quad \textit{for all } \pi \tag{10}$$

By S-Trans on Lemma A.18 and (10), we have:

$$\triangleright(\alpha \leqslant \tau) \cdot \rho'_\alpha(\Xi_{\not\alpha} \cdot (\alpha \leqslant \tau) \cdot \Sigma) \vDash \rho_\tau \rho_\alpha(\Xi_{\not\alpha} \cdot (\alpha \leqslant \tau) \cdot \Sigma) \tag{11}$$

$$\rho_\tau \rho_\alpha \Xi_\alpha \vDash \rho'_\alpha \Xi_\alpha \tag{12}$$

Then by Lemma A.23 on (7) with (11), followed by Lemma A.19 with (12), we have:

$$\triangleright\Xi_\triangleright \cdot \triangleright\Xi_\alpha \cdot \triangleright(\alpha \leqslant \tau) \cdot \rho'_\alpha(\Xi_{\not\alpha} \cdot (\alpha \leqslant \tau) \cdot \Sigma) \vDash \rho'_\alpha \Xi_\alpha \tag{13}$$

From the assumption, we have:

$$\triangleright\Xi_\triangleright \cdot \triangleright\Xi_\alpha \cdot \triangleright(\alpha \leqslant \tau) \cdot \rho'_\alpha(\Xi_{\not\alpha} \cdot \Sigma) \vdash \mathit{lb}_\Xi(\alpha) \leqslant \tau \tag{14}$$

By S-AndOr2· on S-AndOr11↷/S-AndOr12↷ and (14), we have:

$$\triangleright\Xi_\triangleright \cdot \triangleright\Xi_\alpha \cdot \triangleright(\alpha \leqslant \tau) \cdot \rho'_\alpha(\Xi_{\not\alpha} \cdot \Sigma) \vdash \alpha \wedge \tau \wedge \mathit{ub}_\Xi(\alpha) \vee \mathit{lb}_\Xi(\alpha) \leqslant \tau \tag{15}$$

By Corollary A.34, we have:

$$\triangleright\Xi_\alpha \cdot \triangleright(\alpha \leqslant \tau) \vdash \tau \equiv \rho'_\alpha \tau \tag{16}$$

Then by S-Trans on (15) and (16), we have:

$$\triangleright\Xi_\triangleright \cdot \triangleright\Xi_\alpha \cdot \triangleright(\alpha \leqslant \tau) \cdot \rho'_\alpha(\Xi_{\not\alpha} \cdot \Sigma) \vdash \rho'_\alpha \alpha \leqslant \rho'_\alpha \tau \tag{17}$$

Then by Lemma A.23 on (13) with (17), we have:

$$\triangleright\Xi_\triangleright \cdot \triangleright\Xi_\alpha \cdot \triangleright(\alpha \leqslant \tau) \cdot \rho'_\alpha(\Xi_{\not\alpha} \cdot \Sigma) \vDash \rho'_\alpha \Xi_\alpha \tag{18}$$

By S-Cons on (18) with (17), we have:

$$\triangleright\Xi_{\triangleright}\cdot\triangleright\Xi_{\alpha}\cdot\triangleright(\alpha \leqslant \tau)\cdot\rho'_{\alpha}(\Xi_{\not\alpha}\cdot\Sigma) \vDash \rho'_{\alpha}\Xi_{\alpha}\cdot\rho'_{\alpha}(\alpha \leqslant \tau) \tag{19}$$

By S-Trans on S-AndOr11↷, we have:

$$(\alpha \leqslant \tau) \vdash \alpha \wedge ub_{\Xi}(\alpha) \leqslant \tau \tag{20}$$

Then by S-AndOr2· on (20) and (14), we have:

$$\triangleright\Xi_{\triangleright}\cdot\triangleright\Xi_{\alpha}\cdot(\alpha \leqslant \tau)\cdot\rho'_{\alpha}(\Xi_{\not\alpha}\cdot\Sigma) \vdash \alpha \wedge ub_{\Xi}(\alpha) \vee lb_{\Xi}(\alpha) \leqslant \tau \tag{21}$$

By Corollary A.34, we have:

$$\triangleright\Xi_{\alpha} \vdash \tau \equiv \rho_{\alpha}\tau \tag{22}$$

Then by S-Trans on (21) and (22), we have:

$$\begin{aligned}&\triangleright\Xi_{\triangleright}\cdot\triangleright\Xi_{\alpha}\cdot(\alpha \leqslant \tau)\cdot\rho'_{\alpha}(\Xi_{\not\alpha}\cdot\Sigma) \vdash \alpha \wedge ub_{\Xi}(\alpha) \vee lb_{\Xi}(\alpha) \leqslant \rho_{\alpha}\tau\\ \text{i.e.,}\quad &\triangleright\Xi_{\triangleright}\cdot\triangleright\Xi_{\alpha}\cdot(\alpha \leqslant \tau)\cdot\rho'_{\alpha}(\Xi_{\not\alpha}\cdot\Sigma) \vdash \rho_{\alpha}\alpha \leqslant \rho_{\alpha}\tau\end{aligned} \tag{23}$$

By S-AndOr2↷ and Lemma A.7 on S-Hyp and S-Refl, we have:

$$\begin{aligned}&(\alpha \leqslant \tau) \vdash \alpha \wedge \tau \wedge ub_{\Xi}(\alpha) \vee lb_{\Xi}(\alpha) \equiv \alpha \wedge ub_{\Xi}(\alpha) \vee lb_{\Xi}(\alpha)\\ \text{i.e.,}\quad &(\alpha \leqslant \tau) \vdash \alpha \wedge ub_{\Xi\cdot(\alpha\leqslant\tau)}(\alpha) \vee lb_{\Xi\cdot(\alpha\leqslant\tau)}(\alpha) \equiv \alpha \wedge ub_{\Xi}(\alpha) \vee lb_{\Xi}(\alpha)\end{aligned} \tag{24}$$

By Lemma A.31 on (24), we have:

$$(\alpha \leqslant \tau) \vdash \rho'_{\alpha}\pi \equiv \rho_{\alpha}\pi \quad \textit{for all } \pi \tag{25}$$

By S-Trans on Lemma A.18 and (25), we have:

$$\rho_{\alpha}(\Xi_{\not\alpha}\cdot\Sigma) \vDash \rho'_{\alpha}(\Xi_{\not\alpha}\cdot\Sigma) \tag{26}$$

Then by Lemma A.23 on (23) with (26), we have:

$$\triangleright\Xi_{\triangleright}\cdot\triangleright\Xi_{\alpha}\cdot(\alpha \leqslant \tau)\cdot\rho_{\alpha}(\Xi_{\not\alpha}\cdot\Sigma) \vdash \rho_{\alpha}\alpha \leqslant \rho_{\alpha}\tau \tag{27}$$

Then by Lemma A.29 and Lemma A.23 with (27), (2) implies:

$$(\alpha \leqslant \tau)\cdot\rho_{\alpha}\Sigma \vdash \triangleright\Xi_{\triangleright}\cdot\triangleright\Xi_{\alpha}\cdot\rho_{\alpha}\Xi_{\not\alpha}\ ;\ \rho'_1 \textbf{ cons.} \tag{28}$$

Then by Lemma A.43 on (28), we have:

$$\rho_{\tau}\rho_{\alpha}\Sigma \vdash \triangleright\Xi_{\triangleright}\cdot\triangleright\Xi_{\alpha}\cdot\triangleright(\alpha \leqslant \tau)\cdot\rho_{\tau}\rho_{\alpha}\Xi_{\not\alpha}\ ;\ \rho'_2 \textbf{ cons.} \tag{29}$$

for some $\rho'_2$. By Lemma A.36 on (29) with (9), we have:

$$\rho'_{\alpha}\Sigma \vdash \triangleright\Xi_{\triangleright}\cdot\triangleright\Xi_{\alpha}\cdot\triangleright(\alpha \leqslant \tau)\cdot\rho'_{\alpha}\Xi_{\not\alpha}\ ;\ \rho'_3 \textbf{ cons.} \tag{30}$$

for some $\rho'_3$. Then by the definition of consistency on (19) and (30), we have:

$$\Sigma \vdash \Xi\cdot(\alpha \leqslant \tau)\ ;\ \rho'_3 \circ \rho'_{\alpha} \textbf{ cons.} \tag{31}$$

■

**Lemma C.28.**

1. *If* $\Xi, \Sigma \vdash \tau_1 \ll \tau_2 \Rightarrow \Xi'$ *and* $\alpha \notin TTV(\tau_1) \cup TTV(\tau_2)$ *and* $\mathbf{err} \notin \Xi'$, *then* $\triangleright\Xi_\alpha \cdot \rho(\Xi_{\not\alpha} \cdot \Xi' \cdot \Sigma) \vdash \tau_1 \leqslant \tau_2$, *where* $split_\alpha(\Xi, \varnothing) = (\Xi_\alpha, \Xi_{\not\alpha})$ *and* $\rho = [\alpha \mapsto \alpha \wedge ub_\Xi(\alpha) \vee lb_\Xi(\alpha)]$.
2. *If* $\Xi, \Sigma \vdash \mathrm{D}^0 \Rightarrow \Xi'$ *and* $\alpha \notin TTV(\mathrm{D}^0)$ *and* $\mathbf{err} \notin \Xi'$, *where* $\mathrm{D}^0 = \bigvee_{i \in 1..n} \mathrm{C}_i^0$, *then* $\triangleright\Xi_\alpha \cdot \rho(\Xi_{\not\alpha} \cdot \Xi' \cdot \Sigma) \vdash \mathrm{D}^0 \leqslant \bot$, *where* $split_\alpha(\Xi, \varnothing) = (\Xi_\alpha, \Xi_{\not\alpha})$ *and* $\rho = [\alpha \mapsto \alpha \wedge ub_\Xi(\alpha) \vee lb_\Xi(\alpha)]$.

**Proof** By induction on constraining derivations.

**Case C-Hyp.** Since $\alpha \notin TTV(\tau_1) \cup TTV(\tau_2)$, we have from the premise:

$$(\tau_1 \leqslant \tau_2) \in \Xi_{\not\alpha} \cdot \Sigma$$
$$\text{i.e.,} \quad (\rho\tau_1 \leqslant \rho\tau_2) \in \rho(\Xi_{\not\alpha} \cdot \Sigma) \tag{1}$$

Then by S-Hyp on (1), we have:

$$\rho(\Xi_{\not\alpha} \cdot \Sigma) \vdash \rho\tau_1 \leqslant \rho\tau_2 \tag{2}$$

By Corollary A.34, we have:

$$\triangleright\Sigma_\alpha \vdash \tau_1 \equiv \rho\tau_1 \tag{3}$$
$$\triangleright\Sigma_\alpha \vdash \tau_2 \equiv \rho\tau_2 \tag{4}$$

Then by S-Trans on (2), (3), and (4), we have:

$$\triangleright\Sigma_\alpha \cdot \rho(\Xi_{\not\alpha} \cdot \Sigma) \vdash \tau_1 \leqslant \tau_2 \tag{5}$$

**Case C-Assum.** Then the premise of the rule is:

$$\Xi, \Sigma \cdot \triangleright(\tau_1 \leqslant \tau_2) \vdash \mathrm{dnf}^0(\tau_1 \wedge \neg\tau_2) \Rightarrow \Xi' \tag{6}$$

By IH on (6), we have:

$$\triangleright\Xi_\alpha \cdot \rho(\Xi_{\not\alpha} \cdot \Xi' \cdot \Sigma \cdot \triangleright(\tau_1 \leqslant \tau_2)) \vdash \mathrm{dnf}^0(\tau_1 \wedge \neg\tau_2) \leqslant \bot \tag{7}$$

By Corollary A.33, we have:

$$\Xi_\alpha \vdash \tau_1 \equiv \rho\tau_1 \tag{8}$$
$$\Xi_\alpha \vdash \tau_2 \equiv \rho\tau_2 \tag{9}$$

Then by S-Trans on Lemma A.18, (8), and (9), we have:

$$\triangleright\Xi_\alpha \cdot \triangleright(\tau_1 \leqslant \tau_2) \models \rho \triangleright (\tau_1 \leqslant \tau_2) \tag{10}$$

Then by Lemma A.23 with (10), (7) implies:

$$\triangleright\Xi_\alpha \cdot \rho(\Xi_{\not\alpha} \cdot \Xi' \cdot \Sigma) \cdot \triangleright(\tau_1 \leqslant \tau_2) \vdash \mathrm{dnf}^0(\tau_1 \wedge \neg\tau_2) \leqslant \bot \tag{11}$$

By S-Trans on Lemma 7.3 and (11), we have:

$$\triangleright\Xi_\alpha \cdot \rho(\Xi_{\not\alpha} \cdot \Xi' \cdot \Sigma) \cdot \triangleright(\tau_1 \leqslant \tau_2) \vdash \tau_1 \wedge \neg\tau_2 \leqslant \bot \tag{12}$$

By Theorem A.9 on (12), we have:

$$\triangleright\Xi_\alpha \cdot \rho(\Xi_{\not\alpha} \cdot \Xi' \cdot \Sigma) \cdot \triangleright(\tau_1 \leqslant \tau_2) \vdash \tau_1 \leqslant \tau_2 \tag{13}$$

By S-Assum on (13), we have:

$$\triangleright\Xi_\alpha{\cdot}\rho(\Xi_{\not\alpha}{\cdot}\Xi'{\cdot}\Sigma) \vdash \tau_1 \leqslant \tau_2 \tag{14}$$

**Case C-Or.** It is easy to see that if $TTV'(\mathrm{C}^0_k)$ are not distinct for some $k$, we can deduplicate them before preceeding, and duplicate them again in the conclusion. Therefore we can assume that $\overline{TTV'(\mathrm{C}^0_i)}^{\,i\,\in\,1..n}$ are distinct.
The premises of the rule are:

$$\Xi, \Sigma \vdash \bigvee_{i\,\in\,1..n-1} \mathrm{C}^0_i \Rightarrow \Xi'_1 \tag{15}$$

$$\Xi{\cdot}\Xi', \Sigma \vdash \mathrm{C}^0_n \Rightarrow \Xi'_2 \tag{16}$$

where $\Xi' = \Xi'_1{\cdot}\Xi'_2$ for some $\Xi'_1$ and $\Xi'_2$. By IH on (15), we have:

$$\triangleright\Xi_\alpha{\cdot}\rho(\Xi_{\not\alpha}{\cdot}\Xi'_1{\cdot}\Sigma) \vdash \bigvee_{i\,\in\,1..n-1} \mathrm{C}^0_i \leqslant \bot \tag{17}$$

By IH on (16), we have:

$$\triangleright\Xi_\alpha{\cdot}\triangleright\Xi'_{1\,\alpha}{\cdot}\rho'(\Xi_{\not\alpha}{\cdot}\Xi'_{1\,\not\alpha}{\cdot}\Xi'_2{\cdot}\Sigma) \vdash \mathrm{C}^0_n \leqslant \bot \tag{18}$$

where $split_\alpha(\Xi'_1,\ \varnothing) = (\Xi'_{1\,\alpha},\ \Xi'_{1\,\not\alpha})$ and $\rho' = [\alpha \mapsto \alpha \wedge ub_{\Xi{\cdot}\Xi'_1}(\alpha) \vee lb_{\Xi{\cdot}\Xi'_1}(\alpha)]$.
By Lemma C.29 on (15), we have:

$$\Xi'_1 \ \textbf{\textit{guard.}} \tag{19}$$

By Lemma A.18, we have:

$$\rho\Xi'_1 \vDash \rho\Xi'_{1\,\alpha} \tag{20}$$

By Corollary A.33, we have:

$$\Xi_\alpha \vdash \pi \equiv [\alpha \mapsto \alpha \wedge ub_{\Xi_\alpha}(\alpha) \vee lb_{\Xi_\alpha}(\alpha)]\pi \quad \textit{for all } \pi$$

$$\text{i.e.,} \quad \Xi_\alpha \vdash \pi \equiv \rho\pi \quad \textit{for all } \pi \tag{21}$$

Then by S-Trans on (20) and (21), we have:

$$\Xi_\alpha{\cdot}\rho\Xi'_1 \vDash \Xi'_{1\,\alpha} \tag{22}$$

By Lemma A.21 on (22), we have:

$$\triangleright\Xi_\alpha{\cdot}\triangleright\rho\Xi'_1 \vDash \triangleright\Xi'_{1\,\alpha} \tag{23}$$

By Lemma A.19 on (23) and Lemma A.18, we have:

$$\triangleright\Xi_\alpha{\cdot}\rho\Xi'_1 \vDash \triangleright\Xi'_{1\,\alpha} \tag{24}$$

By Corollary A.34, we have:

$$\triangleright\Xi_\alpha \vdash ub_{\Xi'_1}(\alpha) \equiv [\alpha \mapsto \alpha \wedge ub_{\Xi_\alpha}(\alpha) \vee lb_{\Xi_\alpha}(\alpha)]ub_{\Xi'_1}(\alpha)$$

$$\text{i.e.,} \quad \triangleright\Xi_\alpha \vdash ub_{\Xi'_1}(\alpha) \equiv \rho ub_{\Xi'_1}(\alpha) \tag{25}$$

By S-AndOr2$\circlearrowright$ on S-Hyp, we have:

$$\rho\Xi'_1 \vdash \rho\alpha \leqslant \rho ub_{\Xi'_1}(\alpha)$$

$$\text{i.e.,} \quad \rho\Xi'_1 \vdash \alpha \wedge ub_\Xi(\alpha) \vee lb_\Xi(\alpha) \leqslant \rho ub_{\Xi'_a}(\alpha) \tag{26}$$

Then by S-TRANS on S-ANDOR12·, (26) and (25), we have:

$$\triangleright \Xi_\alpha \cdot \rho \Xi'_1 \vdash lb_\Xi(\alpha) \leqslant ub_{\Xi'_1}(\alpha) \tag{27}$$

By S-ANDOR2⤸ on S-REFL and (27), we have:

$$\triangleright \Xi_\alpha \cdot \rho \Xi'_1 \vdash lb_\Xi(\alpha) \leqslant lb_\Xi(\alpha) \wedge ub_{\Xi'_1}(\alpha) \tag{28}$$

Then by S-ANDOR11⤸ and (28), we have:

$$\triangleright \Xi_\alpha \cdot \rho \Xi'_1 \vdash lb_\Xi(\alpha) \equiv lb_\Xi(\alpha) \wedge ub_{\Xi'_1}(\alpha) \tag{29}$$

Then by (29) and S-DISTR, we have:

$$\begin{aligned} \triangleright \Xi_\alpha \cdot \rho \Xi'_1 \vdash \quad & \alpha \wedge ub_{\Xi \cdot \Xi'_1}(\alpha) \vee lb_{\Xi \cdot \Xi'_1}(\alpha) \\ & = \alpha \wedge ub_\Xi(\alpha) \wedge ub_{\Xi'_1}(\alpha) \vee lb_\Xi(\alpha) \vee lb_{\Xi'_1}(\alpha) \\ & \equiv \alpha \wedge ub_\Xi(\alpha) \wedge ub_{\Xi'_1}(\alpha) \vee lb_\Xi(\alpha) \wedge ub_{\Xi'_1}(\alpha) \vee lb_{\Xi'_1}(\alpha) \\ & \equiv (\alpha \wedge ub_\Xi(\alpha) \vee lb_\Xi(\alpha)) \wedge ub_{\Xi'_1}(\alpha) \vee lb_{\Xi'_1}(\alpha) \end{aligned} \tag{30}$$

By S-ANDOR2⤸ on S-REFL and S-HYP, followed by S-TRANS with S-ANDOR11·, we have:

$$\rho \Xi'_1 \vdash \alpha \wedge ub_\Xi(\alpha) \vee lb_\Xi(\alpha) \leqslant (\alpha \wedge ub_\Xi(\alpha) \vee lb_\Xi(\alpha)) \wedge \rho ub_{\Xi'_1}(\alpha) \vee \rho lb_{\Xi'_1}(\alpha) \tag{31}$$

Similarly, by S-ANDOR2· on S-REFL and S-HYP, followed by S-TRANS with S-ANDOR11⤸, we have:

$$\rho \Xi'_1 \vdash (\alpha \wedge ub_\Xi(\alpha) \vee lb_\Xi(\alpha)) \wedge \rho ub_{\Xi'_1}(\alpha) \vee \rho lb_{\Xi'_1}(\alpha) \leqslant \alpha \wedge ub_\Xi(\alpha) \vee lb_\Xi(\alpha) \tag{32}$$

By Corollary A.34, we have:

$$\triangleright \Xi_\alpha \vdash ub_{\Xi'_1}(\alpha) \equiv \rho ub_{\Xi'_1}(\alpha) \tag{33}$$

$$\triangleright \Xi_\alpha \vdash lb_{\Xi'_1}(\alpha) \equiv \rho lb_{\Xi'_1}(\alpha) \tag{34}$$

Then by S-TRANS on (31)/(32), (33), and (34), we have:

$$\triangleright \Xi_\alpha \cdot \rho \Xi'_1 \vdash \alpha \wedge ub_\Xi(\alpha) \vee lb_\Xi(\alpha) \equiv (\alpha \wedge ub_\Xi(\alpha) \vee lb_\Xi(\alpha)) \wedge ub_{\Xi'_1}(\alpha) \vee lb_{\Xi'_1}(\alpha) \tag{35}$$

Then by S-TRANS on (35) and (30), we have:

$$\triangleright \Xi_\alpha \cdot \rho \Xi'_1 \vdash \alpha \wedge ub_\Xi(\alpha) \vee lb_\Xi(\alpha) \equiv \alpha \wedge ub_{\Xi \cdot \Xi'_1}(\alpha) \vee lb_{\Xi \cdot \Xi'_1}(\alpha) \tag{36}$$

By Lemma A.31 on (36), we have:

$$\triangleright \Xi_\alpha \cdot \rho \Xi'_1 \vdash \rho \pi \equiv \rho' \pi \quad \textit{for all } \pi \tag{37}$$

Then by S-TRANS on Lemma A.18 and (37), we have:

$$\triangleright \Xi_\alpha \cdot \rho (\Xi_{\not\alpha} \cdot \Xi'_1 \cdot \Xi'_2 \cdot \Sigma) \vDash \rho' (\Xi_{\not\alpha} \cdot \Xi'_{1\not\alpha} \cdot \Xi'_2 \cdot \Sigma) \tag{38}$$

Then by Lemma A.23 with (24) and (38), (18) implies:

$$\triangleright \Xi_\alpha \cdot \rho (\Xi_{\not\alpha} \cdot \Xi'_1 \cdot \Xi'_2 \cdot \Sigma) \vDash \mathrm{C}^0_n \leqslant \bot \tag{39}$$

Then by S-AndOr2· on (17) and (39), we have:

$$\triangleright\Xi_\alpha{\cdot}\rho(\Xi_{\mathscr{A}}{\cdot}\Xi_1'{\cdot}\Xi_2'{\cdot}\Sigma) \vdash \bigvee_{i\in 1..n} \mathrm{C}_i^0 \leqslant \bot$$
$$\text{i.e.,}\quad \triangleright\Xi_\alpha{\cdot}\rho(\Xi_{\mathscr{A}}{\cdot}\Xi'{\cdot}\Sigma) \vdash \mathrm{D}^0 \leqslant \bot \tag{40}$$

**Case C-Bot.** Immediate by S-ToB⊃.

**Case C-Cls1.** Then $\mathrm{D}^0 = \mathcal{I}[\#C_1] \wedge \neg(\mathrm{U} \vee \#C_2)$ for some $C_1$, $C_2$, $\mathcal{I}[\#C_1]$, and U. By S-ClsSub on the premise $C_2 \in \mathcal{S}(\#C_1)$, we have:

$$\#C_1 \leqslant \#C_2 \tag{41}$$

By S-Trans on S-AndOr11⊃, (41), and S-AndOr12·, we have:

$$\mathcal{I}[\#C_1] \leqslant \mathrm{U} \vee \#C_2 \tag{42}$$

Then by Theorem A.9, (42) implies:

$$\mathcal{I}[\#C_1] \wedge \neg(\mathrm{U} \vee \#C_2) \leqslant \bot \tag{43}$$

**Cases C-Cls2, C-Cls3.** Then $\mathrm{D}^0 = \mathcal{I}^{\mathcal{N}}[\mathcal{N}] \wedge \neg(\mathrm{U} \vee \#C)$ for some $\mathcal{N}$, $C$, $\mathcal{I}^{\mathcal{N}}[\mathcal{N}]$, and U. The premise of the rule is:

$$\Xi, \Sigma \vdash \mathcal{I}^{\mathcal{N}}[\mathcal{N}] \wedge \neg\mathrm{U} \Rightarrow \Xi' \tag{44}$$

By IH on (44), we have:

$$\triangleright\Xi_\alpha{\cdot}\rho(\Xi_{\mathscr{A}}{\cdot}\Xi'{\cdot}\Sigma) \vdash \mathcal{I}^{\mathcal{N}}[\mathcal{N}] \wedge \neg\mathrm{U} \leqslant \bot \tag{45}$$

By S-AndOr11· followed by S-NegInv, we have:

$$\neg(\mathrm{U} \vee \#C) \leqslant \neg\mathrm{U} \tag{46}$$

Then by S-Trans on (46) and (45), we have:

$$\triangleright\Xi_\alpha{\cdot}\rho(\Xi_{\mathscr{A}}{\cdot}\Xi'{\cdot}\Sigma) \vdash \mathcal{I}^{\mathcal{N}}[\mathcal{N}] \wedge \neg(\mathrm{U} \vee \#C) \leqslant \bot \tag{47}$$

**Case C-Fun1.** Then $\mathrm{D}^0 = \mathcal{I}[\mathrm{D}_1 \to \mathrm{D}_2] \wedge \neg(\mathrm{D}_3 \to \mathrm{D}_4)$ for some $\overline{\mathrm{D}_j}^{\,j\in 1..4}$ and $\mathcal{I}[\mathrm{D}_1 \to \mathrm{D}_2]$. The premises of the rule are:

$$\Xi, \lhd\Sigma \vdash \mathrm{D}_3 < \mathrm{D}_1 \Rightarrow \Xi_1' \tag{48}$$
$$\Xi{\cdot}\Xi_1', \lhd\Sigma \vdash \mathrm{D}_2 < \mathrm{D}_4 \Rightarrow \Xi_2' \tag{49}$$

for some $\Xi_1'$ and $\Xi_2'$, where $\Xi' = \Xi_1'{\cdot}\Xi_2'$. By Lemma C.24 on (48) and (49), we have:

$$\Xi{\cdot}\lhd\Sigma{\cdot}\Xi_1' \vdash \mathrm{D}_3 \leqslant \mathrm{D}_1 \tag{50}$$
$$\Xi{\cdot}\Xi_1'{\cdot}\lhd\Sigma{\cdot}\Xi_2' \vdash \mathrm{D}_2 \leqslant \mathrm{D}_4 \tag{51}$$

By Lemma A.23 with Lemma A.18, (50) and (51) imply:

$$\Xi{\cdot}\Xi'{\cdot}\lhd\Sigma \vdash \mathrm{D}_3 \leqslant \mathrm{D}_1 \tag{52}$$
$$\Xi{\cdot}\Xi'{\cdot}\lhd\Sigma \vdash \mathrm{D}_2 \leqslant \mathrm{D}_4 \tag{53}$$

By Corollary A.33, we have:

$$\Xi_\alpha \vdash \pi \equiv [\alpha \mapsto \alpha \wedge ub_{\Xi_\alpha}(\alpha) \vee lb_{\Xi_\alpha}(\alpha)]\pi \quad \textit{for all } \pi$$
$$\text{i.e.,}\quad \Xi_\alpha \vdash \pi \equiv \rho\pi \quad \textit{for all } \pi \tag{54}$$

By S-TRANS on Lemma A.18 and (54), we have:

$$\Xi_\alpha{\cdot}\rho(\Xi_{\not\alpha}{\cdot}\Xi'{\cdot}\lhd\Sigma) \vDash \Xi_{\not\alpha}{\cdot}\Xi'{\cdot}\lhd\Sigma \tag{55}$$

Then by Lemma A.23 with (55), (52) and (53) imply:

$$\Xi_\alpha{\cdot}\rho(\Xi_{\not\alpha}{\cdot}\Xi'{\cdot}\lhd\Sigma) \vdash \mathrm{D}_3 \leqslant \mathrm{D}_1 \tag{56}$$

$$\Xi_\alpha{\cdot}\rho(\Xi_{\not\alpha}{\cdot}\Xi'{\cdot}\lhd\Sigma) \vdash \mathrm{D}_2 \leqslant \mathrm{D}_4 \tag{57}$$

Then by S-FUNDEPTH on (56) and (57), we have:

$$\triangleright\Xi_\alpha{\cdot}\rho(\Xi_{\not\alpha}{\cdot}\Xi'{\cdot}\Sigma) \vdash \mathrm{D}_1 \to \mathrm{D}_2 \leqslant \mathrm{D}_3 \to \mathrm{D}_4 \tag{58}$$

By S-TRANS on S-ANDOR11$\wr$, S-ANDOR12$\wr$, and (58), we have:

$$\triangleright\Xi_\alpha{\cdot}\rho(\Xi_{\not\alpha}{\cdot}\Xi'{\cdot}\Sigma) \vdash \mathcal{I}[\mathrm{D}_1 \to \mathrm{D}_2] \leqslant \mathrm{D}_3 \to \mathrm{D}_4 \tag{59}$$

By Theorem A.9, (59) implies:

$$\triangleright\Xi_\alpha{\cdot}\rho(\Xi_{\not\alpha}{\cdot}\Xi'{\cdot}\Sigma) \vdash \mathcal{I}[\mathrm{D}_1 \to \mathrm{D}_2] \wedge \neg(\mathrm{D}_3 \to \mathrm{D}_4) \leqslant \bot \tag{60}$$

**Case C-RCD1.** Similar to case C-FUN1.

**Case C-VAR1.** Then $\mathrm{D}^0 = \mathrm{C} \wedge \beta$ and $\Xi' = \Xi'_1{\cdot}(\beta \leqslant \neg\mathrm{C})$ for some $\beta$, C, and $\Xi'_1$. By S-HYP, we have:

$$\rho(\beta \leqslant \neg\mathrm{C}) \vDash \rho\beta \leqslant \rho\neg\mathrm{C}$$

$$\text{i.e.,}\quad \rho(\beta \leqslant \neg\mathrm{C}) \vDash \rho\beta \leqslant \neg\rho\mathrm{C} \tag{61}$$

By Theorem A.9, (61) implies:

$$\rho(\beta \leqslant \neg\mathrm{C}) \vDash \rho\mathrm{C} \wedge \rho\beta \leqslant \bot$$

$$\text{i.e.,}\quad \rho(\beta \leqslant \neg\mathrm{C}) \vDash \rho(\mathrm{C} \wedge \beta) \leqslant \bot \tag{62}$$

By Corollary A.34, we have:

$$\triangleright\Xi_\alpha \vdash \mathrm{C} \wedge \beta \equiv [\alpha \mapsto \alpha \wedge ub_{\Xi_\alpha}(\alpha) \vee lb_{\Xi_\alpha}(\alpha)](\mathrm{C} \wedge \beta)$$

$$\text{i.e.,}\quad \triangleright\Xi_\alpha \vdash \mathrm{C} \wedge \beta \equiv \rho(\mathrm{C} \wedge \beta) \tag{63}$$

Then by S-TRANS on (63) and (62), we have:

$$\triangleright\Xi_\alpha{\cdot}\rho(\Xi_{\not\alpha}{\cdot}\Xi'_1{\cdot}(\beta \leqslant \neg\mathrm{C}){\cdot}\Sigma) \vdash \mathrm{C} \wedge \beta \leqslant \bot \tag{64}$$

**Case C-VAR2.** Similar to case C-VAR1.

■

**Lemma C.29** (Guardedness of constraining)**.**

1. *If* $\Xi, \Sigma \vdash \tau_1 \ll \tau_2 \Rightarrow \Xi'$ *and* $\boldsymbol{err} \notin \Xi'$*, then* $\Xi'$ ***guard.***.
2. *If* $\Xi, \Sigma \vdash \bigvee_i \mathrm{C}_i^0 \Rightarrow \Xi'$ *and* $\overline{TTV'(\mathrm{C}_i^0)}^i$ *are distinct and* $\boldsymbol{err} \notin \Xi'$*, then* $\Xi'$ ***guard.***.

**Proof** By straightforward induction on constraining derivations. ■

### C.4 Type Inference Termination Proof

The basic intuition is that by Theorem B.9, we know that in well-formed declarations contexts, there is only a finite number of types that can be reached by expanding all the user-defined type constructors in a given type. Therefore, the number of types that may be reached while applying constraining rules is finite, and since each traversed type is saved as part of the current subtyping hypotheses, all executions of constraining will eventually halt.

**Proof** [Proof 7.6 (Termination of Constraining)]

Let $T_i$ be the set of type pairs that are constrained at any recursive depth $i$ of the type constraining algorithm.

We can see from the constraining rules of Figure 20 that if we start from the constraint $\Xi \vdash \tau_0 \leqslant \pi_0$, then $T_0 = \{\, \tau_0 \leqslant \pi_0 \,\}$ and $\overline{T_i \subseteq T'_i}^{\,i}$ where:

$$
\begin{aligned}
T'_0 &= \{\, \tau_0 \leqslant \pi_0 \,\} \cup \Xi \\
T'_{i+1} = \quad & \{\, \mathrm{D}_3 \leqslant \mathrm{D}_1 \mid \mathcal{I}[\mathrm{D}_1 \to \mathrm{D}_2] \wedge \neg S^{\neg}[\mathrm{D}_3 \to \mathrm{D}_4] \in S_i \,\} \\
& \cup \{\, \mathrm{D}_2 \leqslant \mathrm{D}_4 \mid \mathcal{I}[\mathrm{D}_1 \to \mathrm{D}_2] \wedge \neg S^{\neg}[\mathrm{D}_3 \to \mathrm{D}_4] \in S_i \,\} \\
& \cup \{\, \mathrm{D}_y \leqslant \mathrm{D} \mid \mathcal{I}[\{\, \overline{x : \mathrm{D}_x}^{\,x} \,\}] \wedge \neg S^{\neg}[\{\, y : \mathrm{D} \,\}] \in S_i,\ y \in \{\, \overline{x} \,\} \,\} \\
& \cup \{\, \textstyle\bigvee_{\tau \in S} \tau \leqslant \neg \mathrm{C} \mid \mathrm{C} \wedge \alpha \in S_i,\ S \in \mathcal{P}(\{\, \pi \mid \pi \leqslant \alpha \in \bigcup_{j \leqslant i} T'_j \,\}) \,\} \\
& \cup \{\, \alpha \leqslant \neg \mathrm{C} \mid \mathrm{C} \wedge \alpha \in S_i \,\} \\
& \cup \{\, \mathrm{C} \leqslant \textstyle\bigwedge_{\tau \in S} \tau \mid \mathrm{C} \wedge \neg\alpha \in S_i,\ S \in \mathcal{P}(\{\, \pi \mid \alpha \leqslant \pi \in \bigcup_{j \leqslant i} T'_j \,\}) \,\} \\
& \cup \{\, \mathrm{C} \leqslant \alpha \mid \mathrm{C} \wedge \neg\alpha \in S_i \,\} \\
S_i = \{\, & \mathrm{C} \mid (\tau \leqslant \pi) \in T'_i,\ \mathrm{dnf}^0(\tau \wedge \neg\pi) = \textstyle\bigvee_i \mathrm{C}_i,\ \mathrm{C} \in \{\, \overline{\mathrm{C}_i}^{\,i} \,\} \,\}
\end{aligned}
$$

$S_i$ puts each constraint in $T'_i$ into RDNF, as is done by C-Assum. The first two components in the inductive definition of $T'_i$ correspond to the premises of C-Fun1, and the third component to the premise of C-Rcd1. In addition to the pairs of types constrained (i.e., the hypotheses assumed), $T'_i$ also contains the bounds assumed in the premises of C-Var1 and C-Var2, as seen in the fifth and seventh components. Therefore we can simply look up the bounds from the union of $T'_j$ for $j \leqslant i$ in the fourth and sixth components, which correspond to the premise of C-Var1 and C-Var2 respectively. To exclude hypotheses assumed by C-Assum, which may not end up being assumed as a bound by C-Var1 and C-Var2, we overapproximate by considering all subsets of such pairs of types.

Next, we show that the size of $\bigcup_i T'_i$ is bounded.

The functions $\overline{collect_c}^{\,c}$ traverse a type and collect the type variables, class and alias types, nominal tags, and record labels, which we abbreviate as $\overline{c}$, reachable from the type.

$$
\begin{aligned}
c \ ::= \ & \mathrm{TV} \quad \text{(type variables)} \\
\mid \ & \mathrm{CA} \quad \text{(class and alias types)} \\
\mid \ & \mathrm{NT} \quad \text{(nominal tags)} \\
\mid \ & \mathrm{RL} \quad \text{(record labels)} \\
N[\overline{\tau}]^* \ ::= \ & \epsilon \mid N[\overline{\tau}]^* \cdot N[\overline{\tau}]
\end{aligned}
$$

$$collect_c^{N[\overline{\tau}]^*}(\tau_1 \to \tau_2) = collect_c^{N[\overline{\tau}]^*}(\tau_1) \cup collect_c^{N[\overline{\tau}]^*}(\tau_2)$$

$$collect_c^{N[\overline{\tau}]^*}(\{\, x : \tau \,\}) = \begin{cases} collect_c^{N[\overline{\tau}]^*}(\tau) \cup \{\, x \,\} & \text{if } c = \mathrm{RL} \\ collect_c^{N[\overline{\tau}]^*}(\tau) & \text{otherwise} \end{cases}$$

$$collect_c^{N[\overline{\tau}]^*}(N[\overline{\tau}]) = \begin{cases} collect_c^{N[\overline{\tau}]^* \cdot N[\overline{\tau}]}(\tau') \cup \{\, N[\overline{\tau}] \,\} & \text{if } N[\overline{\tau}] \notin N[\overline{\tau}]^* \text{ and } c = \mathrm{CA} \\ collect_c^{N[\overline{\tau}]^* \cdot N[\overline{\tau}]}(\tau') & \text{if } N[\overline{\tau}] \notin N[\overline{\tau}]^* \text{ and } c \neq \mathrm{CA} \\ \varnothing & \text{if } N[\overline{\tau}] \in N[\overline{\tau}]^* \end{cases}$$

$$\text{where } N[\overline{\tau}] \textbf{ exp. } \tau'$$

$$collect_c^{N[\overline{\tau}]^*}(\#C) = \begin{cases} \{\, \#C \,\} & \text{if } c = \mathrm{NT} \\ \varnothing & \text{otherwise} \end{cases}$$

$$collect_c^{N[\overline{\tau}]^*}(\alpha) = \begin{cases} \{\, \alpha \,\} & \text{if } c = \mathrm{TV} \\ \varnothing & \text{otherwise} \end{cases}$$

$$collect_c^{N[\overline{\tau}]^*}(\top^\diamond) = \varnothing$$

$$collect_c^{N[\overline{\tau}]^*}(\tau_1 \vee^\diamond \tau_2) = collect_c^{N[\overline{\tau}]^*}(\tau_1) \cup collect_c^{N[\overline{\tau}]^*}(\tau_2)$$

$$collect_c^{N[\overline{\tau}]^*}(\neg\tau) = collect_c^{N[\overline{\tau}]^*}(\tau)$$

Similarly, the function *depth* traverses a type and measures the nesting depth of type constructors up to the first recursive occurrence of a class or alias type.

$$depth^{N[\overline{\tau}]}(\tau_1 \to \tau_2) = max(depth^{N[\overline{\tau}]}(\tau_1), depth^{N[\overline{\tau}]}(\tau_2)) + 1$$

$$depth^{N[\overline{\tau}]}(\{\, x : \tau \,\}) = depth^{N[\overline{\tau}]}(\tau) + 1$$

$$depth^{N[\overline{\tau}]}(N[\overline{\tau}]) = \begin{cases} depth^{N[\overline{\tau}]^* \cdot N[\overline{\tau}]}(\tau') & \text{if } N[\overline{\tau}] \notin N[\overline{\tau}]^*, \text{where } N[\overline{\tau}] \textbf{ exp. } \tau' \\ 0 & \text{if } N[\overline{\tau}] \in N[\overline{\tau}]^* \end{cases}$$

$$depth^{N[\overline{\tau}]}(\#C) = depth^{N[\overline{\tau}]}(\alpha) = depth^{N[\overline{\tau}]}(\top^\diamond) = 0$$

$$depth^{N[\overline{\tau}]}(\tau_1 \vee^\diamond \tau_2) = max(depth^{N[\overline{\tau}]}(\tau_1), depth^{N[\overline{\tau}]}(\tau_2))$$

$$depth^{N[\overline{\tau}]}(\neg\tau) = depth^{N[\overline{\tau}]}(\tau)$$

By the Theorem B.9, if $\mathcal{D}$ ***wf***, then for all $\tau$, the sets $\overline{collect_c(\tau)}^c$ are finite, and $depth(\tau)$ is finite.

Given a set of types $S$, we can collect the $\overline{c}$ reachable from it as $\overline{collect_c(S) = \bigcup_{\tau \in S} collect_c(\tau)}^c$ and the type constructor nesting depth as $depth(S) = \max_{\tau \in S} depth(\tau)$. Then we can inductively construct the universes $U_i$ of C's up to depth $i$ that only contain $\overline{collect_c(S)}^c$ without duplicates, as do the results of $\mathrm{dnf}^0$. Notice that all of $U_i$ are finite.

For any $S'$ where $\overline{collect_c(S') \subseteq collect_c(S)}^c$, $depth(S')$ is the type constructor nesting depth after expanding class and alias types up to the first recursive occurrences, while $\mathrm{dnf}^0$ expands class and alias types on the top level, which by the guardedness check does not include their first recursive occurrences. Since the RDNF subexpression unnesting in the first three components of the inductive definition of $T_i'$, the Boolean algebraic connectives in the remaining four components, and $\mathrm{dnf}^0$ in $S_i$ all preserve the depth and do not introduce

new $\overline{c}$, we have:

$$\begin{aligned}
&S_i \subseteq U_{depth(T_0')} \\
&T_i' \subseteq T_0' \cup \quad (\{ \textstyle\bigvee_{\tau \in S} \tau \mid S \in \mathcal{P}(U_{depth(T_0')}) \} \cup collect_{\mathrm{TV}}(T_0')) \\
&\qquad\qquad \times (\{ \textstyle\bigvee_{\tau \in S} \tau \mid S \in \mathcal{P}(U_{depth(T_0')}) \} \cup \{ \textstyle\bigwedge_{\tau \in S} \tau \mid S \in \mathcal{P}(U_{depth(T_0')}) \} \\
&\qquad\qquad\quad \cup \{ \tau \mid \tau \in U_{depth(T_0')} \} \cup collect_{\mathrm{TV}}(T_0'))
\end{aligned}$$

Therefore the set $T = \bigcup_i T_i$ of all pairs of types ever constrained by the algorithm is bounded by:

$$\begin{aligned}
T \subseteq \textstyle\bigcup_i T_i' \subseteq T_0' \cup \quad &(\{ \textstyle\bigvee_{\tau \in S} \tau \mid S \in \mathcal{P}(U_{depth(T_0')}) \} \cup collect_{\mathrm{TV}}(T_0')) \\
&\times (\{ \textstyle\bigvee_{\tau \in S} \tau \mid S \in \mathcal{P}(U_{depth(T_0')}) \} \cup \{ \textstyle\bigwedge_{\tau \in S} \tau \mid S \in \mathcal{P}(U_{depth(T_0')}) \} \\
&\quad \cup \{ \tau \mid \tau \in U_{depth(T_0')} \} \cup collect_{\mathrm{TV}}(T_0'))
\end{aligned}$$

and is thus finite.

Since C-Hyp ensures that the subtyping context $\Sigma$ reachable by the subtyping algorithm cannot contain duplicates, we have $\Sigma \subseteq T \cup \{ \textbf{\textit{err}} \}$. Since $T$ is finite, $\Sigma$ is also finite. Since recursive calls to the constraining algorithm always increases the size of $\Sigma$, this implies that constraining always terminates. ■

### *C.5 Type Inference Completeness Proofs*

**Lemma C.30** (Completeness of type inference — general)**.** *If* $\Xi, \Gamma \vdash^\star P : \tau$*, then* $\Gamma \Vdash^\star P : \tau' \Rightarrow \Xi'$ *for some* $\Xi'$ *and* $\tau'$ *so that* $\forall \Xi'.\ \tau' \leqslant^\forall \forall \Xi.\ \tau$*.*

**Proof** By induction on program typing derivations.

**Case T-Body.** Then $P = t$ for some $t$. The premises of the rule are:

$$\Xi \ \textbf{\textit{cons.}} \tag{1}$$

$$\Xi, \Gamma \vdash t : \tau \tag{2}$$

By Lemma C.34 on (2) and (1), we have:

$$\Gamma \Vdash t : \tau' \Rightarrow \Xi' \tag{3}$$

$$\Xi \vdash \rho\tau' \leqslant \tau \tag{4}$$

$$\Xi \models \rho\Xi' \tag{5}$$

for some $\tau'$ and $\Xi'$ and $\rho$, where $dom(\rho) = fresh((3))$. By I-Body on (3), we have:

$$\Gamma \Vdash^\star t : \tau' \Rightarrow \Xi' \tag{6}$$

By S-All on (4) and (5), we have:

$$\forall \Xi'.\ \tau' \leqslant^\forall \forall \Xi.\ \tau \tag{7}$$

**Case T-Def.** Then $P = \textbf{def}\ x = t\ ;\ P'$ for some $x$ and $t$ and $P'$. The premises of the rule are:

$$\Xi_1\ \textbf{\textit{cons.}} \tag{8}$$
$$\Xi_1, \Gamma \vdash t : \tau_1 \tag{9}$$
$$\Xi, \Gamma{\cdot}(x : \forall \Xi_1.\, \tau_1) \vdash^\star P' : \tau \tag{10}$$

By Lemma C.34 on (9) and (8), we have:

$$\Gamma \Vdash t : \tau_1' \Rightarrow \Xi_1' \tag{11}$$
$$\Xi_1 \vdash \rho_1 \tau_1' \leqslant \tau_1 \tag{12}$$
$$\Xi_1 \vDash \rho_1 \Xi_1' \tag{13}$$

for some $\tau_1'$ and $\Xi_1'$ and $\rho_1$, where $dom(\rho_1) = fresh((11))$. By S-All on (12) and (13), we have:

$$\forall \Xi_1'.\, \tau_1' \leqslant^\forall \forall \Xi_1.\, \tau_1 \tag{14}$$

By Lemma C.31 on (10) and (14), we have:

$$\Xi, \Gamma{\cdot}(x : \forall \Xi_1'.\, \tau_1') \vdash^\star P' : \tau \tag{15}$$

By IH on (15), we have:

$$\Gamma{\cdot}(x : \forall \Xi_1'.\, \tau_1') \Vdash^\star P' : \tau' \Rightarrow \Xi' \tag{16}$$
$$\forall \Xi'.\, \tau' \leqslant^\forall \forall \Xi.\, \tau \tag{17}$$

for some $\tau'$ and $\Xi'$. By I-Body on (11) and (16), we have:

$$\Gamma \Vdash^\star \textbf{def}\ x = t\ ;\ P' : \tau' \Rightarrow \Xi' \tag{18}$$

■

**Lemma C.31** (Strengthening)**.** *If* $\Xi, \Gamma{\cdot}(x : \sigma_1) \vdash t : \tau$ *and* $\epsilon \vdash \sigma_2 \leqslant^\forall \sigma_1$, *then* $\Xi, \Gamma{\cdot}(x : \sigma_2) \vdash t : \tau$.

**Proof** By straightforward induction on typing derivations. ■

**Definition C.32.** *We write* $fresh(A)$ *to denote all the type variables that are taken as fresh in the given derivation* $A$.

**Definition C.33.** *We say* $\rho$ *extends* $\rho'$ *if* $[\overline{\alpha \mapsto \tau}^{(\alpha \mapsto \tau) \in \rho, \alpha \in dom(\rho')}] = \rho'$.

**Lemma C.34** (Completeness of polymorphic type inference)**.** *If* $\Xi, \Gamma \vdash t : \tau$ *and* $\Xi$ ***cons.*** *and* $\Xi \vDash \rho_0 \Xi_0$, *then (A)* $\Xi_0, \Gamma \Vdash t : \tau' \Rightarrow \Xi'$ *and* $\Xi \vdash \rho\tau' \leqslant \tau$ *and* $\Xi \vDash \rho(\Xi_0{\cdot}\Xi')$ *for some* $\tau'$ *and* $\Xi'$ *and* $\rho$, *where* $\textbf{\textit{err}} \notin \Xi'$ *and* $\rho$ *extends* $\rho_0$ *and* $dom(\rho) \backslash dom(\rho_0) = fresh(A)$.

**Proof** By induction on term typing derivations.

**Case T-Subs.** Then the premises of the rule are:

$$\Xi, \Gamma \vdash t : \tau_1 \tag{1}$$
$$\Xi \vdash \tau_1 \leqslant \tau \tag{2}$$

for some $\tau_1$. By IH on (1), we have:

$$\Xi_0, \Gamma \Vdash t : \tau' \Rightarrow \Xi' \tag{3}$$
$$\Xi \vdash \rho\tau' \leqslant \tau_1 \tag{4}$$
$$\Xi \vDash \rho(\Xi_0 \cdot \Xi') \tag{5}$$

for some $\tau'$ and $\Xi'$ and $\rho$, where $\boldsymbol{err} \notin \Xi'$ and $\rho$ extends $\rho_0$ and $dom(\rho) \backslash dom(\rho_0) = fresh((3))$. By S-Trans on (4) and (2), we have:

$$\Xi \vdash \rho\tau' \leqslant \tau \tag{6}$$

**Case T-Obj.** Then $t = C\ \{\ \overline{x_i = t_i}^{\,i}\ \}$ and $\tau = \#C \wedge \{\ \overline{x_i : \tau_i}^{\,i}\ \}$ for some $C$ and $\overline{x_i}^{\,i}$ and $\overline{t_i}^{\,i}$ and $\overline{\tau_i}^{\,i}$. The premises of the rule are:

$$\overline{\Xi, \Gamma \vdash t_i : \tau_i}^{\,i} \tag{7}$$
$$C\ \boldsymbol{final} \tag{8}$$

Then for each $i$, repeat the following:

*Assume the following:*

$$\Xi \vDash \rho_{i-1}(\overline{\Xi_j}^{\,j \in 0..i-1}) \tag{9}$$
$$\overline{\Xi \vdash \rho_{i-1}\tau'_j \leqslant \tau_j}^{\,j \in 1..i-1} \tag{10}$$

*By IH on (7), we have:*

$$\overline{\Xi_j}^{\,j \in 0..i-1}, \Gamma \Vdash t_i : \tau'_i \Rightarrow \Xi_i \tag{11}$$
$$\Xi \vdash \rho_i\tau'_i \leqslant \tau_i \tag{12}$$
$$\Xi \vDash \rho_i(\overline{\Xi_j}^{\,j \in 0..i-1} \cdot \Xi_i) \tag{13}$$

*for some $\tau'_i$ and $\Xi_i$ and $\rho_i$, where $\boldsymbol{err} \notin \Xi_i$ and $\rho_i$ extends $\rho_{i-1}$ and $dom(\rho_i) \backslash dom(\rho_{i-1}) = fresh((11))$. Since $\rho_i$ extends $\rho_{i-1}$ and $dom(\rho_i) \backslash dom(\rho_{i-1})$ are picked to be fresh in (11), which means they could not have appeared in $\overline{\tau'_j}^{\,j \in 1..i-1}$, we have:*

$$\overline{\rho_i\tau'_j = \rho_{i-1}\tau'_j}^{\,j \in 1..i-1} \tag{14}$$

*Then (10) implies:*

$$\overline{\Xi \vdash \rho_i\tau'_j \leqslant \tau_j}^{\,j \in 1..i-1} \tag{15}$$

Then in the end we have:

$$\overline{\overline{\Xi_j}^{\,j\in 0..i-1}, \Gamma \Vdash t_i : \tau'_i \Rightarrow \Xi_i}^{\,i} \tag{16}$$

$$\Xi \vDash \rho(\Xi_0 \cdot \overline{\Xi_i}^{\,i}) \tag{17}$$

$$\overline{\Xi \vdash \rho\tau'_i \leqslant \tau_i}^{\,i} \tag{18}$$

for some $\overline{\tau'_i}^{\,i}$ and $\overline{\Xi_i}^{\,i}$ and $\rho$, where $\mathbf{err} \notin \overline{\Xi_i}^{\,i}$ and $\rho$ extends $\rho_0$ and $dom(\rho)\backslash dom(\rho_0) = \bigcup_i (dom(\rho_i)\backslash dom(\rho_{i-1})) = \bigcup_i fresh((16)_i)$. By I-Obj on (16) and (8), we have:

$$\Xi_0, \Gamma \Vdash C\ \{\ \overline{x_i = t_i}^{\,i}\ \} : \#C \wedge \{\ \overline{x_i : \tau'_i}^{\,i}\ \} \Rightarrow \overline{\Xi_i}^{\,i} \tag{19}$$

By S-RcdDepth on (18), we have:

$$\overline{\Xi \vdash \{\ x_i : \rho\tau'_i\ \} \leqslant \{\ x_i : \tau_i\ \}}^{\,i} \tag{20}$$

By Lemma A.7ↄ on S-Refl and (20), we have:

$$\Xi \vdash \#C \wedge \{\ \overline{x_i : \rho\tau'_i}^{\,i}\ \} \leqslant \#C \wedge \{\ \overline{x_i : \tau_i}\ \}$$

$$\text{i.e.,} \quad \Xi \vdash \rho(\#C \wedge \{\ \overline{x_i : \tau'_i}^{\,i}\ \}) \leqslant \#C \wedge \{\ \overline{x_i : \tau_i}\ \} \tag{21}$$

**Case T-Proj.** Then $t = t'.x$ for some $t'$ and $x$. The premise of the rule is:

$$\Xi, \Gamma \vdash t' : \{\ x : \tau\ \} \tag{22}$$

By IH on (22), we have:

$$\Xi_0, \Gamma \Vdash t' : \tau' \Rightarrow \Xi_1 \tag{23}$$

$$\Xi \vdash \rho_1\tau' \leqslant \{\ x : \tau\ \} \tag{24}$$

$$\Xi \vDash \rho_1(\Xi_0 \cdot \Xi_1) \tag{25}$$

for some $\tau'$ and $\Xi_1$ and $\rho_1$, where $\mathbf{err} \notin \Xi_1$ and $\rho_1$ extends $\rho_0$ and $dom(\rho_1)\backslash dom(\rho_0) = fresh((23))$. Introduce a fresh $\alpha$ and let $\rho = [\alpha \mapsto \tau, \overline{\beta \mapsto \pi}^{(\beta \mapsto \pi) \in \rho_1}]$. Then we have:

$$\rho\tau' = \rho_1\tau' \tag{26}$$

$$\rho(\{\ x : \alpha\ \}) = \{\ x : \tau\ \} \tag{27}$$

$$\rho(\Xi_0 \cdot \Xi_1) = \rho_1(\Xi_0 \cdot \Xi_1) \tag{28}$$

Then (24) and (25) imply:

$$\Xi \vdash \rho\tau' \leqslant \rho(\{\ x : \alpha\ \}) \tag{29}$$

$$\Xi \vDash \rho(\Xi_0 \cdot \Xi_1) \tag{30}$$

By Lemma 7.8 on (29) and (30), we have:

$$\Xi_0 \cdot \Xi_1, \epsilon \vdash \tau' \ll \{\ x : \alpha\ \} \Rightarrow \Xi_2 \tag{31}$$

for some $\Xi_2$, where $\mathbf{err} \notin \Xi_2$ and $\Xi \vDash \rho\Xi_2$. Then by I-Proj on (23) and (31), we have:

$$\Xi_0, \Gamma \Vdash t'.x : \alpha \Rightarrow \Xi_1 \cdot \Xi_2 \tag{32}$$

Since $\rho\alpha = \tau$, by S-REFL, we have:

$$\Xi \vdash \rho\alpha \leqslant \tau \tag{33}$$

(30) and $\Xi \models \rho\Xi_2$ implies:

$$\Xi \models \rho(\Xi_0 \cdot \Xi_1 \cdot \Xi_2) \tag{34}$$

**Case T-VAR1.** Immediate by I-VAR1.

**Case T-VAR2.** Then $t = x$ and $\Gamma(x) = \forall \Xi_1.\ \tau_1$ for some $x$ and $\Xi_1$ and $\tau_1$. By the definition of $\leqslant^{\forall}$, we have:

$$\Xi \models \rho_1 \Xi_1 \tag{35}$$

$$\Xi \vdash \rho_1 \tau_1 \leqslant \tau \tag{36}$$

for some $\rho_1$, where $dom(\rho_1) = TV(\Xi_1) \cup TV(\tau_1) =: S$. Introduce a fresh $\gamma_\alpha$ for each $\alpha \in S$. Then by I-VAR2, we have:

$$\Xi_0, \Gamma \Vdash x : [\overline{\alpha \mapsto \gamma_\alpha}^{\alpha \in S}]\tau_1 \Rightarrow [\overline{\alpha \mapsto \gamma_\alpha}^{\alpha \in S}]\Xi_1 \tag{37}$$

Let $\rho = [\overline{\gamma_\alpha \mapsto \rho_1 \alpha}^{\alpha \in S}]$. Then we have:

$$\begin{aligned} & \rho \circ [\overline{\alpha \mapsto \gamma_\alpha}^{\alpha \in S}] \\ = \ & \rho_1 \circ [\overline{\gamma_\alpha \mapsto \alpha}^{\alpha \in S}] \circ [\overline{\alpha \mapsto \gamma_\alpha}^{\alpha \in S}] \\ = \ & \rho_1 \circ [\overline{\gamma_\alpha \mapsto \alpha}^{\alpha \in S}] \end{aligned} \tag{38}$$

Since $\overline{\gamma_\alpha}^{\alpha \in S}$ are picked to be fresh, which means they could not have appeared in $\Xi_1$ and $\tau_1$, we have:

$$[\overline{\gamma_\alpha \mapsto \alpha}^{\alpha \in S}]\Xi_1 = \Xi_1 \tag{39}$$

$$[\overline{\gamma_\alpha \mapsto \alpha}^{\alpha \in S}]\tau_1 = \tau_1 \tag{40}$$

Then we have:

$$\begin{aligned} \rho_1 \Xi_1 &= \rho_1([\overline{\gamma_\alpha \mapsto \alpha}^{\alpha \in S}]\Xi_1) \\ &= \rho([\overline{\alpha \mapsto \gamma_\alpha}^{\alpha \in S}]\Xi_1) \end{aligned} \tag{41}$$

$$\begin{aligned} \rho_1 \tau_1 &= \rho_1([\overline{\gamma_\alpha \mapsto \alpha}^{\alpha \in S}]\tau_1) \\ &= \rho([\overline{\alpha \mapsto \gamma_\alpha}^{\alpha \in S}]\tau_1) \end{aligned} \tag{42}$$

Then (35) and (36) imply:

$$\Xi \models \rho([\overline{\alpha \mapsto \gamma_\alpha}^{\alpha \in S}]\Xi_1) \tag{43}$$

$$\Xi \vdash \rho([\overline{\alpha \mapsto \gamma_\alpha}^{\alpha \in S}]\tau_1) \leqslant \tau \tag{44}$$

**Case T-ABS.** Then $t = \lambda x.\ t'$ and $\tau = \tau_1 \to \tau_2$ for some $x$ and $t'$ and $\tau_1$ and $\tau_2$. The premise of the rule is:

$$\Xi, \Gamma \cdot (x : \tau_1) \vdash t' : \tau_2 \tag{45}$$

Introduce a fresh $\alpha$. By Lemma C.35 on (45), we have:

$$\Xi\cdot(\alpha \leqslant \tau_1), \Gamma\cdot(x:\alpha) \vdash t' : \tau' \tag{46}$$
$$\Xi \vdash [\alpha \mapsto \tau_1]\tau' \leqslant \tau_2 \tag{47}$$

By IH on (46), we have:

$$\Xi_0, \Gamma\cdot(x:\alpha) \Vdash t' : \tau'' \Rightarrow \Xi' \tag{48}$$
$$\Xi\cdot(\alpha \leqslant \tau_1) \vdash \rho_1\tau'' \leqslant \tau' \tag{49}$$
$$\Xi\cdot(\alpha \leqslant \tau_1) \vDash \rho_1(\Xi_0\cdot\Xi') \tag{50}$$

for some $\tau''$ and $\Xi'$ and $\rho_1$, where $\boldsymbol{err} \notin \Xi'$ and $\rho_1$ extends $\rho_0$ and $dom(\rho_1)\backslash dom(\rho_0) = fresh((48))$. By I-Abs on (48), we have:

$$\Xi_0, \Gamma \Vdash \lambda x.\, t' : \alpha \to \tau'' \Rightarrow \Xi' \tag{51}$$

By Lemma A.29, (49) and (50) imply:

$$[\alpha \mapsto \tau_1](\Xi\cdot(\alpha \leqslant \tau_1)) \vdash [\alpha \mapsto \tau_1] \circ \rho_1\tau'' \leqslant [\alpha \mapsto \tau_1]\tau'$$
$$\text{i.e.,}\quad [\alpha \mapsto \tau_1]\Xi\cdot(\tau_1 \leqslant \tau_1) \vdash [\alpha \mapsto \tau_1] \circ \rho_1\tau'' \leqslant [\alpha \mapsto \tau_1]\tau' \tag{52}$$
$$[\alpha \mapsto \tau_1](\Xi\cdot(\alpha \leqslant \tau_1)) \vDash [\alpha \mapsto \tau_1] \circ \rho_1(\Xi_0\cdot\Xi')$$
$$\text{i.e.,}\quad [\alpha \mapsto \tau_1]\Xi\cdot(\tau_1 \leqslant \tau_1) \vDash [\alpha \mapsto \tau_1] \circ \rho_1(\Xi_0\cdot\Xi') \tag{53}$$

By S-Cons on Lemma A.18 and S-Refl, we have:

$$[\alpha \mapsto \tau_1]\Xi \vDash [\alpha \mapsto \tau_1]\Xi\cdot(\tau_1 \leqslant \tau_1) \tag{54}$$

By Lemma A.23 with (53), (51) and (52) imply:

$$[\alpha \mapsto \tau_1]\Xi \vdash [\alpha \mapsto \tau_1] \circ \rho_1\tau'' \leqslant [\alpha \mapsto \tau_1]\tau' \tag{55}$$
$$[\alpha \mapsto \tau_1]\Xi \vDash [\alpha \mapsto \tau_1] \circ \rho_1(\Xi_0\cdot\Xi') \tag{56}$$

Since $\alpha$ is picked to be fresh, which means it could not have appeared in $\Xi$, we have $[\alpha \mapsto \tau_1]\Xi = \Xi$. Then (54) and (55) imply:

$$\Xi \vdash [\alpha \mapsto \tau_1] \circ \rho_1\tau'' \leqslant [\alpha \mapsto \tau_1]\tau' \tag{57}$$
$$\Xi \vDash [\alpha \mapsto \tau_1] \circ \rho_1(\Xi_0\cdot\Xi') \tag{58}$$

By S-Trans on (57) and (47), we have:

$$\Xi \vdash [\alpha \mapsto \tau_1] \circ \rho_1\tau'' \leqslant \tau_2 \tag{59}$$

By S-FunDepth on S-Refl and (60), we have:

$$\Xi \vdash \tau_1 \to [\alpha \mapsto \tau_1] \circ \rho_1\tau'' \leqslant \tau_1 \to \tau_2$$
$$\text{i.e.,}\quad \Xi \vdash [\alpha \mapsto \tau_1] \circ \rho_1(\alpha \to \tau'') \leqslant \tau_1 \to \tau_2 \tag{60}$$

**Cases T-App, T-Asc, T-Case1, T-Case2, T-Case3.** Similar to case T-Proj.

■

**Lemma C.35.** *If* $\Xi, \Gamma\cdot(x:\tau_1) \vdash t : \tau$, *then* $\Xi\cdot(\alpha \leqslant \tau_1), \Gamma\cdot(x:\alpha) \vdash t : \tau'$ *and* $\Xi \vdash [\alpha \mapsto \tau_1]\tau' \leqslant \tau$ *for any* $\alpha$ ***fresh*** *and some* $\tau'$.

**Proof** By straightforward induction on typing derivations. ∎

**Proof** [Proof 7.8 (Completeness of Constraining)] By Theorem 7.6, we have:

$$\Xi_0, \epsilon \vdash \tau_1 \ll \tau_2 \Rightarrow \Xi' \tag{1}$$

for some $\Xi'$. The result then follows from Lemma C.36. ∎

**Lemma C.36** (Necessity of Constraining)**.**

1. *If* $\Xi \vdash \rho\tau_1 \leqslant \rho\tau_2$ *and* $\Xi$ ***cons.*** *and* $\Xi \vDash \rho\Xi_0$ *and* $\Xi_0, \Sigma \vdash \tau_1 \ll \tau_2 \Rightarrow \Xi'$*, then* $\Xi \vDash \rho\Xi'$.
2. *If* $\Xi \vdash \rho\mathrm{D}^0 \leqslant \bot$ *and* $\Xi$ ***cons.*** *and* $\Xi \vDash \rho\Xi_0$ *and* $\Xi_0, \Sigma \vdash \mathrm{D}^0 \Rightarrow \Xi'$*, then* $\Xi \vDash \rho\Xi'$.

**Proof** By induction on constraining derivations.

**Cases C-Hyp, C-Bot, C-Cls1.** Immediate by S-Empty since $\Xi' = \epsilon$.
**Case C-Assum.** From the assumptions, we have:

$$\Xi \vdash \rho\tau_1 \leqslant \rho\tau_2 \tag{1}$$

The premise of the rule is:

$$\Xi_0, \Sigma{\cdot}\triangleright(\tau_1 \leqslant \tau_2) \vdash \mathrm{dnf}^0(\tau_1 \wedge \neg\tau_2) \Rightarrow \Xi' \tag{2}$$

By Theorem A.9, (1) implies:

$$\begin{aligned} &\Xi \vdash \rho\tau_1 \wedge \neg\rho\tau_2 \leqslant \bot \\ \text{i.e.,}\quad &\Xi \vdash \rho(\tau_1 \wedge \neg\tau_2) \leqslant \bot \end{aligned} \tag{3}$$

By Lemma 7.3, we have:

$$\tau_1 \wedge \neg\tau_2 \equiv \mathrm{dnf}^0(\tau_1 \wedge \neg\tau_2) \tag{4}$$

By Lemma A.29, (4) implies:

$$\rho(\tau_1 \wedge \neg\tau_2) \equiv \rho\mathrm{dnf}^0(\tau_1 \wedge \neg\tau_2) \tag{5}$$

By S-Trans on (5) and (3), we have:

$$\Xi \vdash \rho\mathrm{dnf}^0(\tau_1 \wedge \neg\tau_2) \leqslant \bot \tag{6}$$

The result then follows from IH on (2) and (6).
**Case C-Or.** Then $\mathrm{D}^0 = \mathrm{D}^0_1 \vee \mathrm{C}^0$ for some $\mathrm{D}^0_1$ and $\mathrm{C}^0$. From the assumptions, we have:

$$\Xi - \rho(\mathrm{D}^0_1 \vee \mathrm{C}^0) \leqslant \bot \tag{7}$$

$$\Xi \vDash \rho\Xi_0 \tag{8}$$

The premises of the rule are:

$$\Xi_0, \Sigma \vdash \mathrm{D}^0_1 \Rightarrow \Xi'_1 \tag{9}$$

$$\Xi_0{\cdot}\Xi'_1, \Sigma \vdash \mathrm{C}^0 \Rightarrow \Xi'_2 \tag{10}$$

for some $\Xi'_1$ and $\Xi'_2$, where $\Xi' = \Xi'_1 \cdot \Xi'_2$. By S-AndOr11$\cdot$ and S-AndOr12$\cdot$ respectively, we have:

$$\begin{aligned} & \rho \mathrm{D}^0_1 \leqslant \rho \mathrm{D}^0_1 \vee \rho \mathrm{C}^0 \\ \text{i.e.,} \quad & \rho \mathrm{D}^0_1 \leqslant \rho(\mathrm{D}^0_1 \vee \mathrm{C}^0) \end{aligned} \tag{11}$$

$$\begin{aligned} & \rho \mathrm{C}^0 \leqslant \rho \mathrm{D}^0_1 \vee \rho \mathrm{C}^0 \\ \text{i.e.,} \quad & \rho \mathrm{C}^0 \leqslant \rho(\mathrm{D}^0_1 \vee \mathrm{C}^0) \end{aligned} \tag{12}$$

By S-Trans with (7), (11) and (12) respectively imply:

$$\Xi \vdash \rho \mathrm{D}^0_1 \leqslant \bot \tag{13}$$

$$\Xi \vdash \rho \mathrm{C}^0 \leqslant \bot \tag{14}$$

By IH on (13) and (8) and (9), we have:

$$\Xi \models \rho \Xi'_1 \tag{15}$$

(8) and (15) imply:

$$\begin{aligned} & \Xi \models \rho \Xi_0 \cdot \rho \Xi'_1 \\ \text{i.e.,} \quad & \Xi \models \rho(\Xi_0 \cdot \Xi'_1) \end{aligned} \tag{16}$$

By IH on (14) and (16) and (10), we have:

$$\Xi \models \rho \Xi'_2 \tag{17}$$

(15) and (17) imply:

$$\begin{aligned} & \Xi \models \rho \Xi'_1 \cdot \rho \Xi'_2 \\ \text{i.e.,} \quad & \Xi \models \rho \Xi' \end{aligned} \tag{18}$$

**Case C-NotBot.** Then $\mathrm{D}^0 = \mathcal{N} \wedge \mathcal{F} \wedge \mathcal{R} \wedge \neg\bot$ for some $\mathcal{N}$ and $\mathcal{F}$ and $\mathcal{R}$. From the assumptions, we have:

$$\begin{aligned} & \Xi \vdash \rho(\mathcal{N} \wedge \mathcal{F} \wedge \mathcal{R} \wedge \neg\bot) \leqslant \bot \\ \text{i.e.,} \quad & \Xi \vdash \mathcal{N} \wedge \rho\mathcal{F} \wedge \rho\mathcal{R} \wedge \neg\bot \leqslant \bot \end{aligned} \tag{19}$$

$$\Xi \ \textit{cons}. \tag{20}$$

By S-Trans on S-ToB$\cdot$ and Theorem A.6, we have:

$$\mathcal{N} \wedge \rho\mathcal{F} \wedge \rho\mathcal{R} \leqslant \neg\bot \tag{21}$$

By S-AndOr2$\curlyvee$ on S-Refl and (21), we have:

$$\mathcal{N} \wedge \rho\mathcal{F} \wedge \rho\mathcal{R} \leqslant \mathcal{N} \wedge \rho\mathcal{F} \wedge \rho\mathcal{R} \wedge \neg\bot \tag{22}$$

By S-Trans on (22) and (19), we have:

$$\Xi \vdash \mathcal{N} \wedge \rho\mathcal{F} \wedge \rho\mathcal{R} \leqslant \bot \tag{23}$$

Since $TTV(\mathcal{N} \wedge \rho\mathcal{F} \wedge \rho\mathcal{R}) \cup TTV(\bot) = \varnothing$, by Lemma 3.4 on (20) and (23), we have:

$$\triangleright\Xi \vdash \mathcal{N} \wedge \rho\mathcal{F} \wedge \rho\mathcal{R} \leqslant \bot \tag{24}$$

Notice that $\mathcal{N} \wedge \rho\mathcal{F} \wedge \rho\mathcal{R}$ is in CDN-normalized form. Since none of $\{ \mathcal{N}, \rho\mathcal{F}, \rho\mathcal{R} \}$ is a negation, $\mathcal{N} \wedge \rho\mathcal{F} \wedge \rho\mathcal{R}$ is complement-free. Then by Lemma 4.22 on (24), we have:

$$\bot \cong \bigwedge_j (\pi'_j \vee V_j^{D_j}) \tag{25}$$

for some $\overline{\pi'_j}^{\,j}$ and $\overline{D_j}^{\,j}$ and $\overline{V_j^{D_j}}^{\,j}$, where $\bigwedge_j V_j^{D_j}$ is complement-free. By S-AndOr12·, we have:

$$\overline{V_j^{D_j} \subseteq \pi'_j \vee V_j^{D_j}}^{\,j} \tag{26}$$

By Lemma A.7↷ on (26), we have:

$$\bigwedge_j V_j^{D_j} \subseteq \bigwedge_j (\pi'_j \vee V_j^{D_j}) \tag{27}$$

By S-Trans on (27) and (25), we have:

$$\bigwedge_j V_j^{D_j} \subseteq \bot \tag{28}$$

which is impossible since $\bigwedge_j V_j^{D_j}$ is complement-free. Therefore this case is impossible.

**Case C-Cls2.** Then $\mathrm{D}^0 = \mathcal{I}[\#C_1] \wedge \neg(\mathrm{U} \vee \#C_2)$ for some $C_1$ and $C_2$ and $\mathcal{I}[\#C_1]$ and U. From the assumptions, we have:

$$\Xi \vdash \rho(\mathcal{I}[\#C_1] \wedge \neg(\mathrm{U} \vee \#C_2)) \leqslant \bot \tag{29}$$

$$\Xi \ \textit{cons.} \tag{30}$$

The premises of the rule are:

$$C_2 \notin \mathcal{S}(\#C_1) \tag{31}$$

$$\Xi_0, \Sigma \vdash \mathcal{I}[\#C_1] \wedge \neg\mathrm{U} \Rightarrow \Xi' \tag{32}$$

By Theorem A.9 on (29), we have:

$$\Xi \vdash \rho\mathcal{I}[\#C_1] \leqslant \rho(\mathrm{U} \vee \#C_2)$$

$$\text{i.e.,} \quad \Xi \vdash \rho\mathcal{I}[\#C_1] \leqslant \rho\tau^0 \vee \bigvee_j \#C'_j \vee \#C_2 \tag{33}$$

for some $\tau^0 \in \{ \bot,\ \mathrm{D}_1 \to \mathrm{D}_2,\ \{ y : \mathrm{D}_1 \} \}$ and $\overline{C'_j}^{\,j}$, where $\mathrm{U} = \rho\tau^0 \vee \bigvee_j \#C'_j$. Since $TTV(\rho\mathcal{I}[\#C_1]) \cup TTV(\rho\tau^0 \vee \bigvee_j \#C'_j \vee \#C_2) = \varnothing$, by Lemma 3.4 on (30) and (33), we have:

$$\triangleright\Xi \vdash \rho\mathcal{I}[\#C_1] \leqslant \rho\tau^0 \vee \bigvee_j \#C'_j \vee \#C_2 \tag{34}$$

By Lemma 4.22 on (34), we have:

$$\rho\mathcal{I}[\#C_1] \cong \bigvee_i (\tau'_i \wedge X_i^{C_i}) \tag{35}$$

$$\overline{\triangleright\Xi \vdash X_i^{C_i} \preceq Y_i}^{\,i} \tag{36}$$

for some $\overline{\tau_i'}^{i}$ and $\overline{C_i}^{i}$ and $\overline{X_i^{C_i}}^{i}$ and $\overline{Y_i \in \{\, \rho\tau^0,\ \#C_2,\ \overline{\#C_j'}^{j} \,\}}^{i}$, where $\bigvee_i X_i^{C_i}$ is complement-free. By S-AndOr12$\curlywedge$, we have:

$$\overline{\tau_i' \wedge X_i^{C_i} \subseteq X_i^{C_i}}^{i} \tag{37}$$

By Lemma A.7$\cdot$ on (37), we have:

$$\bigvee_i (\tau_i' \wedge X_i^{C_i}) \subseteq \bigvee_i X_i^{C_i} \tag{38}$$

By S-Trans on (35) and (38), we have:

$$\rho\mathcal{I}[\#C_1] \cong \bigvee_i X_i^{C_i} \tag{39}$$

By Corollary A.61, (39) implies:

$$\rho\mathcal{I}[\#C_1] \cong X_k^{C_k} \tag{40}$$

for some $k$.

**Case** $C_k \in \{\, \bot, \not\top \,\}$**.** Then we have:

$$X_k^{C_k} \equiv \bot \tag{41}$$

By S-Trans on (40) and (41), we have:

$$\rho\mathcal{I}[\#C_1] \leqslant \bot \tag{42}$$

By S-AndOr11$\curlywedge$, we have:

$$\begin{aligned} &\rho\mathcal{I}[\#C_1] \wedge \rho(\neg \mathrm{U}) \leqslant \rho\mathcal{I}[\#C_1] \\ \text{i.e.,}\quad &\rho(\mathcal{I}[\#C_1] \wedge \neg \mathrm{U}) \leqslant \rho\mathcal{I}[\#C_1] \end{aligned} \tag{43}$$

By S-Trans on (43) and (42), we have:

$$\rho(\mathcal{I}[\#C_1] \wedge \neg \mathrm{U}) \leqslant \bot \tag{44}$$

The result then follows from IH on (32) and (44).

**Case** $C_k \notin \{\, \bot, \not\top \,\}$**.** Let $X_k^{C_k} = \bigwedge_l X_{kl}^{C_k}$ for some $\overline{X_{kl}^{C_k}}^{l}$ where $\overline{X_{kl}^{C_k}}^{l}$ are not intersections. By S-AndOr11$\curlywedge$ and S-AndOr12$\curlywedge$, we have:

$$\overline{X_k^{C_k} \subseteq X_{kl}^{C_k}}^{l} \tag{45}$$

By S-Trans on (40) and (45), we have:

$$\overline{\rho\mathcal{I}[\#C_1] \subseteq X_{kl}^{C_k}}^{l} \tag{46}$$

Notice that $\rho\mathcal{I}[\#C_1]$ is in CDN-normalized form. Since none of the conjuncts of $\rho\mathcal{I}[\#C_1]$ is a negation, $\rho\mathcal{I}[\#C_1]$ is complement-free. Then by Lemma A.57, (46) implies:

$$\overline{\rho\tau_l^0 \subseteq X_{kl}^{C_k}}^{l} \tag{47}$$

for some $\overline{\tau_l^0 \in \{\mathcal{N}, \mathcal{F}, \mathcal{R}\}}^l$, where $\mathcal{I}[\#C_1] = \mathcal{N} \wedge \mathcal{F} \wedge \mathcal{R}$. By Lemma A.62, (47) implies:

$$\overline{\tau_l^0 \neq \top}^l \tag{48}$$

Then by Lemma 4.10 on (47), we have:

$$\overline{X_{kl}^{C_k} = \rho\tau_l^0}^l \tag{49}$$

By the syntax of $X_k^{C_k}$ and (49), we have:

$$\overline{\rho\tau_l^0 = \rho\tau_1^0}^l \tag{50}$$

Then we have:

$$X_k^{C_k} = \bigwedge_l \rho\tau_1^0 \tag{51}$$

Then (36) implies:

$$\triangleright\Xi \vdash \bigwedge_l \rho\tau_1^0 \preceq Y_k \tag{52}$$

Since $\preceq$ implies $\leqslant$, (52) implies:

$$\triangleright\Xi \vdash \bigwedge_l \rho\tau_1^0 \preceq Y_k$$
$$\text{i.e.,} \quad \triangleright\Xi \vdash \rho\tau_1^0 \leqslant Y_k \tag{53}$$

By Theorem A.63 on (53), (31) implies $Y_k \neq \#C_2$. By S-AndOr11$\diamond$ and S-AndOr12$\diamond$, we have:

$$\rho\mathcal{I}[\#C_1] = \rho(\mathcal{N} \wedge \mathcal{F} \wedge \mathcal{R}) \leqslant \rho\tau_1^0 \tag{54}$$
$$Y_k \leqslant \tau^0 \vee \bigvee_j \#C'_j = \mathrm{U} \tag{55}$$

By S-Trans on (54) and (53) and (55), we have:

$$\triangleright\Xi \vdash \rho\mathcal{I}[\#C_1] \leqslant \mathrm{U} \tag{56}$$

By Theorem A.9, (56) implies:

$$\triangleright\Xi \vdash \rho\mathcal{I}[\#C_1] \wedge \neg\mathrm{U} \leqslant \bot \tag{57}$$

By Lemma A.23 with Lemma A.18, (57) implies:

$$\Xi \vdash \rho\mathcal{I}[\#C_1] \wedge \neg\mathrm{U} \leqslant \bot \tag{58}$$

The result then follows from IH on (32) and (58).

**Case C-Cls3.** Similar to case C-Cls2.

**Case C-Fun1.** Then $\mathrm{D}^0 = \mathcal{I}[\mathrm{D}_1 \to \mathrm{D}_2] \wedge \neg(\mathrm{D}_3 \to \mathrm{D}_4)$ for some $\overline{\mathrm{D}_i}^{i \in 1..4}$. From the assumptions, we have:

$$\Xi \vdash \rho(\mathcal{I}[\mathrm{D}_1 \to \mathrm{D}_2] \wedge \neg(\mathrm{D}_3 \to \mathrm{D}_4)) \leqslant \bot \tag{59}$$
$$\Xi \ \textit{cons.} \tag{60}$$
$$\Xi \models \rho\Xi_0 \tag{61}$$

The premises of the rule are:

$$\Xi_0, \lhd\Sigma \vdash \mathrm{D}_3 \ll \mathrm{D}_1 \Rightarrow \Xi_1' \tag{62}$$

$$\Xi_0 \cdot \Xi_1', \lhd\Sigma \vdash \mathrm{D}_2 \ll \mathrm{D}_4 \Rightarrow \Xi_2' \tag{63}$$

for some $\Xi_1'$ and $\Xi_2'$, where $\Xi' = \Xi_1' \cdot \Xi_2'$. By Theorem A.9 on (59), we have:

$$\Xi \vdash \rho\mathcal{I}[\mathrm{D}_1 \to \mathrm{D}_2] \leqslant \rho(\mathrm{D}_3 \to \mathrm{D}_4) \tag{64}$$

Since $TTV(\rho\mathcal{I}[\mathrm{D}_1 \to \mathrm{D}_2]) \cup TTV(\rho(\mathrm{D}_3 \to \mathrm{D}_4)) = \varnothing$, by Lemma 3.4 on (60) and (64), we have:

$$\triangleright\Xi \vdash \rho\mathcal{I}[\mathrm{D}_1 \to \mathrm{D}_2] \leqslant \rho(\mathrm{D}_3 \to \mathrm{D}_4) \tag{65}$$

By Lemma 4.22 on (65), we have:

$$\rho\mathcal{I}[\mathrm{D}_1 \to \mathrm{D}_2] \cong \bigvee_i (\tau_i' \wedge X_i^{C_i}) \tag{66}$$

$$\overline{\triangleright\Xi \vdash X_i^{C_i} \preceq \rho(\mathrm{D}_3 \to \mathrm{D}_4)}^{\,i} \tag{67}$$

for some $\overline{\tau_i'}^{\,i}$ and $\overline{C_i}^{\,i}$ and $\overline{X_i^{C_i}}^{\,i}$, where $\bigvee_i X_i^{C_i}$ is complement-free. By S-AndOr12$\curlyvee$, we have:

$$\overline{\tau_i' \wedge X_i^{C_i} \subseteq X_i^{C_i}}^{\,i} \tag{68}$$

By Lemma A.7$\cdot$ on (68), we have:

$$\bigvee_i (\tau_i' \wedge X_i^{C_i}) \subseteq \bigvee_i X_i^{C_i} \tag{69}$$

By S-Trans on (66) and (69), we have:

$$\rho\mathcal{I}[\mathrm{D}_1 \to \mathrm{D}_2] \subseteq \bigvee_i X_i^{C_i} \tag{70}$$

By Lemma 4.9, (67) implies that each of $\overline{C_i}^{\,i}$ is either bottom, arrow, or a negated record field. By Corollary A.61, (70) implies:

$$\rho\mathcal{I}[\mathrm{D}_1 \to \mathrm{D}_2] \subseteq X_k^{C_k} \tag{71}$$

for some $k$.

**Case** $C_k \in \{\, \bot, \not\curlywedge \,\}$**.** Then we have:

$$X_k^{C_k} \equiv \bot \tag{72}$$

By S-Trans on (71) and (72), we have:

$$\rho\mathcal{I}[\mathrm{D}_1 \to \mathrm{D}_2] \leqslant \bot \tag{73}$$

which is impossible by the same reasoning as case C-NotBot. Therefore this case is impossible.

**Case** $C_k = \to$**.** Let $X_k^{C_k} = \bigwedge_l X_{kl}^{C_k}$ for some $\overline{X_{kl}^{C_k}}^{\,l}$ where $\overline{X_{kl}^{C_k}}^{\,l}$ are not intersections. By S-AndOr11$\curlyvee$ and S-AndOr12$\curlyvee$, we have:

$$\overline{X_k^{C_k} \subseteq X_{kl}^{C_k}}^{\,l} \tag{74}$$

By S-Trans on (71) and (74), we have:

$$\overline{\rho\mathcal{I}[\mathrm{D}_1 \to \mathrm{D}_2] \subseteq X_{kl}^{C_k}}^{l} \tag{75}$$

Notice that $\rho\mathcal{I}[\mathrm{D}_1 \to \mathrm{D}_2]$ is in CDN-normalized form. Since none of the conjuncts of $\rho\mathcal{I}[\mathrm{D}_1 \to \mathrm{D}_2]$ is a negation, $\rho\mathcal{I}[\mathrm{D}_1 \to \mathrm{D}_2]$ is complement-free. Then by Lemma A.57, (75) implies:

$$\overline{\rho\tau_l^0 \subseteq X_{kl}^{C_k}}^{l} \tag{76}$$

for some $\overline{\tau_l^0 \in \{\mathcal{N}, \mathcal{F}, \mathcal{R}\}}^{l}$, where $\mathcal{I}[\mathrm{D}_1 \to \mathrm{D}_2] = \mathcal{N} \wedge \mathcal{F} \wedge \mathcal{R}$. By Lemma A.62, (76) implies:

$$\overline{\tau_l^0 \neq \top}^{l} \tag{77}$$

Then by Lemma 4.10 on (76), we have:

$$\overline{X_{kl}^{C_k} = \rho\tau_l^0}^{l} \tag{78}$$

By the syntax of $X_k^{C_k}$ and (78), we have:

$$\overline{\rho\tau_l^0 = \rho\tau_1^0}^{l} \tag{79}$$

Then we have:

$$X_k^{C_k} = \bigwedge_l \rho\tau_1^0 \tag{80}$$

Then (67) implies:

$$\triangleright\Xi \vdash \bigwedge_l \rho\tau_1^0 \preceq \rho(\mathrm{D}_3 \to \mathrm{D}_4) \tag{81}$$

Since $\preceq$ implies $\leqslant$, (81) implies:

$$\begin{aligned} &\triangleright\Xi \vdash \textstyle\bigwedge_l \rho\tau_1^0 \leqslant \rho(\mathrm{D}_3 \to \mathrm{D}_4) \\ \text{i.e.,}\quad &\triangleright\Xi \vdash \rho\tau_1^0 \leqslant \rho(\mathrm{D}_3 \to \mathrm{D}_4) \end{aligned} \tag{82}$$

By Theorem A.63 on (82), we have:

$$\tau_1^0 = \mathrm{D}_1 \to \mathrm{D}_2 \tag{83}$$

$$\Xi \vdash \rho\mathrm{D}_3 \leqslant \rho\mathrm{D}_1 \tag{84}$$

$$\Xi \vdash \rho\mathrm{D}_2 \leqslant \rho\mathrm{D}_4 \tag{85}$$

By IH on (84) and (61) and (62), we have:

$$\Xi \vDash \rho\Xi_1' \tag{86}$$

(61) and (86) imply:

$$\begin{aligned} &\Xi \vDash \rho\Xi_0 \cdot \rho\Xi_1' \\ \text{i.e.,}\quad &\Xi \vDash \rho(\Xi_0 \cdot \Xi_1') \end{aligned} \tag{87}$$

By IH on (85) and (87) and (63), we have:

$$\Xi \vDash \rho\Xi_2' \tag{88}$$

(86) and (88) imply:

$$\Xi \vDash \rho\Xi'_1 \cdot \rho\Xi'_2$$
$$\text{i.e.,} \quad \Xi \vDash \rho(\Xi'_1 \cdot \Xi'_2) \tag{89}$$

**Case** $C_k = \mathcal{X}$**.** Then $X_k^{C_k} = \neg \bigvee_j \{ x : \pi_j \}$ for some $\overline{\pi_j}^j$. Then (71) implies:

$$\rho\mathcal{I}[\mathrm{D}_1 \to \mathrm{D}_2] \cong \neg \bigvee_j \{ x : \pi_j \} \tag{90}$$

By S-AndOr11·, we have:

$$\{ x : \pi_1 \} \subseteq \bigvee_j \{ x : \pi_j \} \tag{91}$$

By S-NegInv on (91), we have:

$$\neg \bigvee_j \{ x : \pi_j \} \subseteq \neg\{ x : \pi_1 \} \tag{92}$$

By S-Trans on (90) and (92), we have:

$$\rho\mathcal{I}[\mathrm{D}_1 \to \mathrm{D}_2] \subseteq \neg\{ x : \pi_1 \} \tag{93}$$

By Theorem A.9 on (93), we have:

$$\rho\mathcal{I}[\mathrm{D}_1 \to \mathrm{D}_2] \wedge \{ x : \pi_1 \} \subseteq \bot \tag{94}$$

which is impossible by the same reasoning as case C-NotBot. Therefore this case is impossible.

**Case C-Rcd1.** Similar to case C-Fun1.

**Cases C-Fun2, C-Rcd2, C-Rcd3.** Similar to case C-NotBot.

**Case C-Var1.** Then $\mathrm{D}^0 = \mathrm{C} \wedge \alpha$ and $\Xi' = \Xi'_1 \cdot (\alpha \leqslant \neg \mathrm{C})$ for some C and $\alpha$ and $\Xi'_1$. From the assumptions, we have:

$$\Xi \vdash \rho(\mathrm{C} \wedge \alpha) \leqslant \bot \tag{95}$$
$$\Xi \ \textit{cons.} \tag{96}$$
$$\Xi \vDash \rho\Xi_0 \tag{97}$$

The premise of the rule is:

$$\Xi_0 \cdot (\alpha \leqslant \neg \mathrm{C}), \Sigma \vdash lb_{\Xi_0}(\alpha) \ll \neg \mathrm{C} \Rightarrow \Xi'_1 \tag{98}$$

By Theorem A.9, (95) implies:

$$\Xi \vdash \rho\alpha \leqslant \neg\rho\mathrm{C}$$
$$\text{i.e.,} \quad \Xi \vdash \rho\alpha \leqslant \rho(\neg \mathrm{C}) \tag{99}$$

By S-AndOr2 on S-Hyp, we have:

$$\Xi_0 \vdash lb_{\Xi_0}(\alpha) \leqslant \alpha \tag{100}$$

By S-Hyp, we have:

$$(\alpha \leqslant \neg \mathrm{C}) \vdash \alpha \leqslant \neg \mathrm{C} \tag{101}$$

By S-Trans on (100) and (101), we have:

$$\Xi_0 \cdot (\alpha \leqslant \neg \mathrm{C}) \vdash lb_{\Xi_0}(\alpha) \leqslant \neg \mathrm{C} \tag{102}$$

By Lemma A.29, (102) implies:

$$\rho(\Xi_0 \cdot (\alpha \leqslant \neg \mathrm{C})) \vdash \rho lb_{\Xi_0}(\alpha) \leqslant \rho(\neg \mathrm{C}) \tag{103}$$

By S-Cons on (97) and (99), we have:

$$\Xi \vDash \rho\Xi_0 \cdot (\rho\alpha \leqslant \rho(\neg \mathrm{C}))$$
$$\text{i.e.,} \quad \Xi \vDash \rho(\Xi_0 \cdot (\alpha \leqslant \mathrm{C})) \tag{104}$$

By Lemma A.23 with (104), (103) implies:

$$\Xi \vdash \rho lb_{\Xi_0}(\alpha) \leqslant \rho(\neg \mathrm{C}) \tag{105}$$

By IH on (105) and (104) and (98), we have:

$$\Xi \vDash \rho\Xi_1' \tag{106}$$

By S-Cons on (106) and (99), we have:

$$\Xi \vDash \rho\Xi_1' \cdot (\rho\alpha \leqslant \rho(\neg \mathrm{C}))$$
$$\text{i.e.,} \quad \Xi \vDash \rho\Xi' \tag{107}$$

**Case C-Var2.** Similar to case C-Var1.

■