Liquid Meta

Michael Borkowski
(summarizing joint work with Ranjit Jhala and Niki Vazou)

June 30, 2020

1 Our language λ_2

We work with a polymorphic, typed lambda calculus with call-by-value semantics which is augmented by refinement types, dependent function types, and existential types. Our language is based on the Sprite language λ in Jhala and Vazou's forthcoming manuscript [JV] and incorporates and extends aspects from the λ^H of Vazou et al [VSJ$^+$14]. The existential types were used in a metatheory by Knowles and Flanagan [KF09].

We start with the syntax of term-level expressions in our language:

$$\begin{array}{llll}
\textsf{Values} & v ::= & \texttt{true}, \texttt{false} & \textit{boolean constants} \\
 & \mid & 0, 1, 2, \ldots & \textit{integer constants} \\
 & \mid & x & \textit{variables} \\
 & \mid & \lambda x.e & \textit{abstractions} \\
 & \mid & \Lambda \alpha{:}k.e & \textit{type abstractions} \\
 & \mid & \wedge,\ \vee,\ \neg,\ \leftrightarrow,\ \leq,\ = & \textit{built-in primitives}
\end{array}$$

$$\begin{array}{llll}
\textsf{Expressions} & e ::= & v & \textit{values} \\
 & \mid & e_1\ e_2 & \textit{applications} \\
 & \mid & e\ [t] & \textit{type applications} \\
 & \mid & \texttt{let}\ x = e_1\ \texttt{in}\ e_2 & \textit{let expressions} \\
 & \mid & e_1 : t & \textit{annotations}
\end{array}$$

Next, we give the syntax of the types and binding environments used in our language:

$$\begin{array}{llll}
\textsf{Basic types} & b ::= & \textsf{Bool} & \textit{booleans} \\
 & \mid & \textsf{Int} & \textit{integers} \\
 & \mid & \alpha & \textit{type variables}
\end{array}$$

$$\begin{array}{llll}
\textsf{Types} & t ::= & b\{r\} & \textit{refinement} \\
 & \mid & x{:}t_x \rightarrow t & \textit{dependent function}
\end{array}$$

	$\mid$	$\exists\, x{:}t_x.\, t$	*existential*
	$\mid$	$\forall\, \alpha{:}k.\, t$	*polymorphic*
Kinds	$k :=$	B	*base kind*
	$\mid$	$*$	*star kind*
Environments	$\Gamma :=$	$\varnothing$	*empty*
	$\mid$	$\Gamma, x{:}t$	*bind variable*
	$\mid$	$\Gamma, \alpha{:}k$	*bind type variable*

Note that a basic type b is not a proper type; to express b as a type we must write $b\{x : \texttt{true}\}$. Next, we give the syntax of the Boolean predicates and constraints involved in refinements and subtyping judgments. The ternary judgment $\vdash_B \,:$ is the typing judgment in the underlying System F calculus.

Refinements $r := \{x : p\}$

Predicates $p := \{e \mid \exists\,\Gamma.\, \Gamma \vdash_B e : \mathsf{Bool}\}$ *expressions of type* Bool

In the metatheory here we require that all variables bound in the environment be distinct. In the mechanization we use the locally-named representation: free and bound variables become distinct objects in the syntax. All free variables have unique names and these names never conflict with bound variables, which eliminates the possibility of capture in substitution and the need to perform alpha-renamings during substitution. This came at the cost of needing formal lemmas which permit us to change the name of a variable bound in the environment to maintain the uniqueness of the free variables only.

Our definition of predicates above departs from the Sprite languages of [JV] by allowing predicates to be arbitrary expressions from the main language (which are Boolean typed under the appropriate binding environment). In [JV] however, predicates are quantifier-free first-order formulae over a vocabulary of integers and a limited number of relations. We initially took this approach, but were unable to fully define the denotational semantics for this type of language. In particular, when we define closing substitutions we need to define the substitution of a type $\theta(t)$ as the type resulting from t after performing substitutions for all variables bound to values in $\theta = (x_1 \mapsto v_1, \ldots, x_n \mapsto v_n)$. Substituting arbitrary expressions into t requires substituting arbitrary expressions into predicates, and it isn't clear how to do this for functions like $(\lambda x.x)$ without taking predicates to be all Boolean-typed program expressions.

Returning to our λ_2, we next define the operational semantics of the language. We treat the reduction rules (small step semantics) of the various built-in primitives as external to our language, and we denote by $\delta(c, v)$ a function specifying them. The reductions are defined in a curried manner, so for instance we have that $c\; v_1\; v_2 \hookrightarrow^* \delta(\delta(c, v_1), v_2)$. Currying gives us unary relations like $m{\leq}$ which is a partially evaluated version of the $\leq$ relation.

$$\delta(\wedge, \texttt{true}) := \lambda x.\, x \qquad\qquad \delta(\leftrightarrow, \texttt{true}) := \lambda x.\, x$$

$$\begin{aligned}
\delta(\wedge, \texttt{false}) &:= \lambda x.\, \texttt{false} & \delta(\leftrightarrow, \texttt{false}) &:= \lambda x.\, \neg x \\
\delta(\vee, \texttt{true}) &:= \lambda x.\, \texttt{true} & \delta(\leq, m) &:= m\leq \\
\delta(\vee, \texttt{false}) &:= \lambda x.\, x & \delta(m\leq, n) &:= \texttt{bval}(m \leq n) \\
\delta(\neg, \texttt{true}) &:= \texttt{false} & \delta(=, m) &:= m= \\
\delta(\neg, \texttt{false}) &:= \texttt{true} & \delta(m=, n) &:= \texttt{bval}(m = n)
\end{aligned}$$

Now we give the reduction rules for the small-step semantics. In what follows, e and its variants refer to an arbitrary expression, v refers to a value, x to a variable, and c refers to a built-in primitive.

$$\frac{}{c\ v \hookrightarrow \delta(c, v)}\ \textsc{E-Prim} \qquad \frac{e \hookrightarrow e'}{e\ e_1 \hookrightarrow e'\ e_1}\ \textsc{E-App1}$$

$$\frac{e \hookrightarrow e'}{v\ e \hookrightarrow v\ e'}\ \textsc{E-App2} \qquad \frac{}{(\lambda x.\, e)\ v \hookrightarrow e[v/x]}\ \textsc{E-AppAbs}$$

$$\frac{e \hookrightarrow e'}{e\ [t] \hookrightarrow e'\ [t]}\ \textsc{E-AppT} \qquad \frac{}{(\Lambda \alpha{:}k.\, e)\ [t] \hookrightarrow e[t/\alpha]}\ \textsc{E-AppTAbs}$$

$$\frac{e_x \hookrightarrow e_x'}{\texttt{let}\, x\!=\!e_x\, \texttt{in}\, e \hookrightarrow \texttt{let}\, x\!=\!e_x'\, \texttt{in}\, e}\ \textsc{E-Let} \qquad \frac{}{\texttt{let}\, x\!=\!v\, \texttt{in}\, e \hookrightarrow e[v/x]}\ \textsc{E-LetV}$$

$$\frac{e \hookrightarrow e'}{e : t \hookrightarrow e' : t}\ \textsc{E-Ann} \qquad \frac{}{v : t \hookrightarrow v}\ \textsc{E-AnnV}$$

We give the details of the type substitution operation used above in E-AppTAbs: *(note: decide on whether the type variable is a basic type or whether it cannot be refined)*

$$\begin{aligned}
\alpha\{x : p\}[t_\alpha/\alpha] &: = \text{strengthen}(t_\alpha, p[t_\alpha/\alpha]) & & \\
b\{x : p\}[t_\alpha/\alpha] &: = b\{x : p[t_\alpha/\alpha]\} & & \alpha \neq b \\
(x{:}t_x \to t)[t_\alpha/\alpha] &: = x{:}(t_x[t_\alpha/\alpha]) \to t[t_\alpha/\alpha] & & \\
(\exists\, x{:}t_x.\, t)[t_\alpha/\alpha] &: = \exists\, x{:}(t_x[t_\alpha/\alpha]).\, t[t_\alpha/\alpha] & & \\
(\forall\, \alpha'{:}k.\, t)[t_\alpha/\alpha] &: = \forall\, \alpha'{:}k.\, t[t_\alpha/\alpha] & & \alpha \neq \alpha'
\end{aligned}$$

The strengthen(t_α, p) function defined here takes p and conjoins it to the refinements inside t_α. For this operation to be well-defined, t_α must have base kind.

Next, we define the typing rules of our λ_2. The type judgments in the language λ_2 will be denoted $\vdash$ with a colon between term and type. For clarity, we distinguish between this and other judgments by using $\vdash$ with a subscript in most other settings. For instance, the judgement $\Gamma \vdash_w t : k$ says that type t is well-formed in environment Γ and has kind k:

$$\frac{b \in \{\mathsf{Bool}, \mathsf{Int}\}}{\Gamma \vdash_w b\{x : \mathtt{true}\} : B}\ \text{WF-Base} \qquad \frac{\alpha : k \in \Gamma}{\Gamma \vdash_w \alpha\{x : \mathtt{true}\} : k}\ \text{WF-Var}$$

$$\frac{\Gamma \vdash_w b\{x : \mathtt{true}\} : B \quad y{:}b, \lfloor\Gamma\rfloor \vdash_B p[y/x] : \mathsf{Bool} \quad y \notin \mathrm{dom}(\Gamma)}{\Gamma \vdash_w b\{x : p\} : B}\ \text{WF-Refn}$$

$$\frac{\Gamma \vdash_w t_x : k_x \quad y{:}t_x, \Gamma \vdash_w t[y/x] : k \quad y \notin \mathrm{dom}(\Gamma)}{\Gamma \vdash_w x{:}t_x \to t : *}\ \text{WF-Func}$$

$$\frac{\Gamma \vdash_w t_x : k_x \quad y{:}t_x, \Gamma \vdash_w t[y/x] : k \quad y \notin \mathrm{dom}(\Gamma)}{\Gamma \vdash_w \exists\, x{:}t_x.\, t : k}\ \text{WF-Exis}$$

$$\frac{\alpha'{:}k, \Gamma \vdash_w t[\alpha'/\alpha] : k_t \quad \alpha' \notin \mathrm{dom}(\Gamma)}{\Gamma \vdash_w \forall\, \alpha{:}k.t : *}\ \text{WF-Poly} \qquad \frac{\Gamma \vdash_w t : B}{\Gamma \vdash_w t : *}\ \text{WF-Kind}$$

The judgment $\vdash_w \Gamma$ says that the environment Γ is well formed, meaning that variables are only bound to well-formed types. We adopt the convention that our environments grow from right to left.

$$\frac{}{\vdash_w \varnothing}\ \text{WFE-Empty} \quad \frac{\Gamma \vdash_w t_x : k_x \quad \vdash_w \Gamma \quad x \notin \mathrm{dom}(\Gamma)}{\vdash_w x{:}t_x, \Gamma}\ \text{WFE-Bind}$$

$$\frac{\vdash_w \Gamma \quad \alpha \notin \mathrm{dom}(\Gamma)}{\vdash_w \alpha{:}k, \Gamma}\ \text{WFE-BindT}$$

Now we give the rules for the typing judgements. As with the reduction rules, we take the type of our built-in primitives to be external to our language. We denote by $ty(c)$ the function that specifies the most specific type possible for c. More details on $ty(c)$ are given in the next section. In order to express the exact type of variables, we introduce a "selfification" function that strengthens a refinement we the condition that a value is equal to itself; this is key to derive the fine grained type of $\lambda x.x$ being $x{:}\mathsf{Bool}\{z : \mathtt{true}\} \to \mathsf{Bool}\{z : z = x\}$. Our metatheory only requires that we can selfify base types and existential quantifications of them; this will simplify the development below. *The = in the $z = x$ definition below is overloaded, but in our mechanization we would use either $z \leftrightarrow x$ or $z = x$ depending on the base type. But if we can refine type variables, then = should be polymorphic.*

$$\begin{aligned}
\mathrm{self}(b\{z : p\}, x) &:= b\{z : p \wedge z = x\} \\
\mathrm{self}(z{:}t_z \to t, x) &:= z{:}t_z \to t \\
\mathrm{self}(\exists\, z{:}t_z.\, t, x) &:= \exists\, z{:}t_z.\, \mathrm{self}(t, x) \\
\mathrm{self}(\forall\, \alpha{:}k.\, t, x) &:= \forall\, \alpha{:}k.\, t
\end{aligned}$$

$$\frac{ty(c) = t}{\Gamma \vdash c : t}\ \text{T-Prim} \qquad \frac{x{:}t \in \Gamma}{\Gamma \vdash x : \text{self}(t, x)}\ \text{T-Var} \qquad \frac{\Gamma \vdash e \ : \ x{:}t_x \rightarrow t \qquad \Gamma \vdash e' : t_x}{\Gamma \vdash e\ e' : \exists\, x{:}t_x.\, t}\ \text{T-App}$$

$$\frac{y{:}t_x, \Gamma \vdash e[y/x] : t[y/x] \qquad \Gamma \vdash_w t_x : k_x \qquad y \notin \text{dom}(\Gamma)}{\Gamma \vdash \lambda x.\, e \ : \ x{:}t_x \rightarrow t}\ \text{T-Abs}$$

$$\frac{\Gamma \vdash e \ : \ \forall\, \alpha{:}k.\, s \qquad \Gamma \vdash_w t : k}{\Gamma \vdash e[t] : s[t/\alpha]}\ \text{T-AppT}$$

$$\frac{\alpha'{:}k, \Gamma \vdash e[\alpha'/\alpha] : t[\alpha'/\alpha] \qquad \alpha'{:}k, \Gamma \vdash_w t[\alpha'/\alpha] : k' \qquad \alpha' \notin \text{dom}(\Gamma)}{\Gamma \vdash \Lambda\alpha{:}k.e : \forall\, \alpha{:}k.\, t}\ \text{T-AbsT}$$

$$\frac{\Gamma \vdash e_x : t_x \qquad y{:}t_x, \Gamma \vdash e[y/x] : t[y/x] \qquad \Gamma \vdash_w t : k \qquad y \notin \text{dom}(\Gamma)}{\Gamma \vdash \texttt{let}\, x{=}e_x\, \texttt{in}\, e : t}\ \text{T-Let}$$

$$\frac{\Gamma \vdash e : t \qquad \Gamma \vdash_w t : k}{\Gamma \vdash e : t \ : \ t}\ \text{T-Ann} \qquad \frac{\Gamma \vdash e : s \qquad \Gamma \vdash s <: t \qquad \Gamma \vdash_w t : k}{\Gamma \vdash e : t}\ \text{T-Sub}$$

The last rule, T-Sub, uses the subtyping judgement $\Gamma \vdash s <: t$. The subtyping rules are as follows:

$$\frac{y{:}b\{x_1 : p_1\}, \Gamma \vdash_e p_2[y/x_2] \qquad y \notin \text{dom}(\Gamma)}{\Gamma \vdash b\{x_1 : p_1\} <: b\{x_2 : p_2\}}\ \text{S-Base}$$

$$\frac{\Gamma \vdash s_2 <: s_1 \qquad y{:}s_2, \Gamma \vdash t_1[y/x_1] <: t_2[y/x_2] \qquad y \notin \text{dom}(\Gamma)}{\Gamma \vdash x_1{:}s_1 \rightarrow t_1 <: x_2{:}s_2 \rightarrow t_2}\ \text{S-Func}$$

$$\frac{\Gamma \vdash v_x : t_x \qquad \Gamma \vdash t <: t'[v_x/x]}{\Gamma \vdash t <: \exists\, x{:}t_x.\, t'}\ \text{S-Witn} \qquad \frac{y{:}t_x, \Gamma \vdash t[y/x] <: t' \qquad y \notin \mathit{free}(t')}{\Gamma \vdash \exists\, x{:}t_x.\, t <: t'}\ \text{S-Bind}$$

$$\frac{\alpha{:}k, \Gamma \vdash t_1[\alpha/\alpha_1] <: t_2[\alpha/\alpha_2] \qquad \alpha \notin \text{dom}(\Gamma)}{\Gamma \vdash \forall\, \alpha_1{:}k.\, t_1 <: \forall\, \alpha_2{:}k.\, t_2}\ \text{S-Poly}$$

The first rule above, S-Base, uses the entailment judgement $\Gamma \vdash_e p$ which (roughly) states that predicate p is valid (in the sense of a logical formula) when universally quantified over all variables bound in environment Γ. We give the inference rule for the entailment judgement:

$$\frac{\forall\, \theta.\, \theta \in [\![\Gamma]\!] \Rightarrow \theta(p) \hookrightarrow^* \texttt{true}}{\Gamma \vdash_e p}\ \text{Ent-Pred}$$

Note that if we combine rules T-AbsT and S-Poly then we can derive the following judgment, which shows that polymorphic types are equivalent up to alpha-renaming bound type variables:

$$\frac{\alpha{:}k,\Gamma \vdash e[\alpha/\alpha_1] : t[\alpha/\alpha_2] \qquad \alpha{:}k,\Gamma \vdash_w t : k' \qquad \alpha \notin \mathrm{dom}(\Gamma)}{\Gamma \vdash \Lambda\alpha_1{:}k.e : \forall\,\alpha_2{:}k.\,t}\ \text{T-AbsT'}$$

2 Preliminaries

For clarity, we distinguish between different typing judgments with a subscript. The type judgments in the underlying polymorphic lambda calculus (System F) will be denoted by $\vdash_B$ and a colon before the type. In order to speak about the base type underlying some type, we define a function that erases refinements in types:

$$\lfloor b\{x:p\}\rfloor := b, \quad \lfloor x{:}t_x \to t\rfloor := \lfloor t_x\rfloor \to \lfloor t\rfloor, \quad \lfloor \exists\,x{:}t_x.\,t\rfloor := \lfloor t\rfloor, \quad \text{and} \quad \lfloor \forall\,\alpha{:}k.\,t\rfloor := \forall\,\alpha{:}k.\,\lfloor t\rfloor$$

We start our development of the meta-theory by giving a definition of *type denotations.* Roughly speaking, the denotation of a type t without type variables is the class of value terms v with the correct underlying base type such that this term satisfies the refinement predicates that appear within the structure of t. We can't just define v having the correct base type as meaning a judgment $\varnothing \vdash_B v : \lfloor t\rfloor$ because our System F (see, for instance, Figure 23-1 in [Pierce]) has no concept of subtyping to express the alpha-equivalence of types under the renaming of binders, and so it cannot express every typing judgment that our full language can. For example, by rule T-AbsT and T-Sub we have a judgment that $\varnothing \vdash \Lambda\alpha{:}k.\,\alpha : \forall\,\gamma{:}k.\,\gamma$, but when $\alpha \neq \gamma$ we cannot express this in System F. In keeping with our use of the locally named representation we define "correct underlying base type" to mean that v has a System F typing judgment for *some* member of the alpha-equivalence class of $\lfloor t\rfloor$. We introduce the notation

$$\Gamma \vdash_B v :^{\alpha} \lfloor t\rfloor \ := \ \exists s.\, s \overset{\alpha}{\equiv} \lfloor t\rfloor \ \wedge\ \Gamma \vdash_B v : s$$

We formalize this notion with a recursive definition:

$$\begin{aligned}
[\![b\{x:p\}]\!] &:= \{v \mid (\varnothing \vdash_B v :^{\alpha} b) \wedge (p[v/x] \hookrightarrow^* \texttt{true})\}\\
[\![x{:}t_x \to t]\!] &:= \{v \mid (\varnothing \vdash_B v :^{\alpha} \lfloor t_x\rfloor \to \lfloor t\rfloor) \wedge (\forall\, v_x \in [\![t_x]\!].\, v\ v_x \hookrightarrow^* v' \,\text{such that}\, v' \in [\![t[v_x/x]]\!])\}\\
[\![\exists\,x{:}t_x.\,t]\!] &:= \{v \mid (\varnothing \vdash_B v :^{\alpha} \lfloor t\rfloor) \wedge (\exists\, v_x \in [\![t_x]\!].\, v \in [\![t[v_x/x]]\!])\}\\
[\![\forall\,\alpha{:}k.\,t]\!] &:= \{v \mid (\varnothing \vdash_B v :^{\alpha} \forall\,\alpha{:}k.\,\lfloor t\rfloor) \wedge (\forall\, t_\alpha.\, (\varnothing \vdash_w t_\alpha : k) \Rightarrow v\,[t_\alpha] \hookrightarrow^* v' \,\text{such that}\, v' \in [\![t[t_\alpha/\alpha]]\!])\}
\end{aligned}$$

The denotation of a type variable α is undefined; indeed, it is impossible for any expression to have type α with respect to an empty environment.

We also have the concept of the denotation of an environment Γ; we intuitively define this to be the set of all sequences of value bindings for term variables and type bindings for type variables that appear in Γ such that the values respect the denotations of the types of the corresponding variables and such that the types in the sequence are well formed with respect to the corresponding kinds. A closing substitution is just a sequence of value bindings to variables:

$$\theta = (x_1 \mapsto v_1,\ \ldots,\ x_n \mapsto v_n, \alpha_1 \mapsto t_1,\ \ldots,\ \alpha_m \mapsto t_m) \quad \text{with all}\, x_i, \alpha_j \,\text{distinct}$$

We use the shorthand $\theta(x)$ to refer to v_i if $x = x_i$ and we use $\theta(\alpha)$ to refer to t_j if $\alpha = \alpha_j$. We define $\theta(t)$ to be the type derived from t by substituting for all variables in θ:

$$\theta(t) := t[v_1/x_1] \cdots [v_n/x_n][t_1/\alpha_1] \cdots [t_m/\alpha_m]$$

Then we can formally define the denotation of an environment:

$$\begin{aligned}\llbracket\Gamma\rrbracket := \{\theta =&(x_1 \mapsto v_1, \dots, x_n \mapsto v_n, \alpha_1 \mapsto t_1, \dots, \alpha_m \mapsto t_m)\\ &\mid \forall\,(x : t) \in \Gamma.\, \theta(x) \in \llbracket\theta(t)\rrbracket \;\wedge\; \forall\,(\alpha : k) \in \Gamma.\, \varnothing \vdash_w \theta(\alpha) : k\}.\end{aligned}$$

For each built-in primitive constant or function c we define $ty(c)$ to include the most specific possible refinement type for c.

$$\begin{aligned}
ty(\texttt{true}) &:= \mathsf{Bool}\{x : x = \texttt{true}\}\\
ty(\texttt{false}) &:= \mathsf{Bool}\{x : x = \texttt{false}\}\\
ty(3) &:= \mathsf{Int}\{x : x = 3\}\\
ty(n) &:= \mathsf{Int}\{x : x = n\}\\
ty(\wedge) &:= x{:}\mathsf{Bool} \to y{:}\mathsf{Bool} \to \mathsf{Bool}\{v : v = x \wedge y\}\\
ty(\neg) &:= x{:}\mathsf{Bool} \to \mathsf{Bool}\{y : y = \neg x\}\\
ty(\leq) &:= x{:}\mathsf{Int} \to y{:}\mathsf{Int} \to \mathsf{Bool}\{v : v = (x \leq y)\}\\
ty(m\!\leq) &:= n{:}\mathsf{Int} \to \mathsf{Bool}\{v : v = (m \leq n)\}\\
ty(=) &:= x{:}\alpha \to y{:}\alpha \to \mathsf{Bool}\{v : v = (x = y)\}
\end{aligned}$$

and similarly for the others Note that we use $m\leq$ to represent an arbitrary member of the infinite family of primitives $0\leq$, $1\leq$, $2\leq, \ldots$. Then by the definitions above we get our primitive typing lemma:

Lemma 1. *(Primitive Typing) For every primitive c,*

1. $\varnothing \vdash c : ty(c)$.
2. *If $ty(c) = b\{x : p\}$, then $\varnothing \vdash_w ty(c) : B$, $c \in \llbracket ty(c)\rrbracket$ and for all c' such that $c' \neq c$, $c' \notin \llbracket ty(c)\rrbracket$.*
3. *If $ty(c) = x{:}t_x \to t$, then $\varnothing \vdash_w ty(c) : *$ and for each $v \in \llbracket t_x\rrbracket$, $\delta(c, v)$ is defined and we have both $\varnothing \vdash \delta(c, v) : t[v/x]$ and $\delta(c, v) \in \llbracket t[v/x]\rrbracket$. Thus $c \in \llbracket ty(c)\rrbracket$.*

3 Meta-theory

In this section, we seek to prove the operational soundness of our language λ_1. We begin by stating several standard properties and proving some basic facts used later on.

Lemma 2. *Values are closed under substitution of variables for values. If v is a value and $x \in \text{free}(v)$ then for any value v_x, we have that $v[v_x/x]$ is also a value.*

Lemma 3. *The operational semantics of λ_2 are deterministic: For every expression e there exists at most one term e' such that $e \hookrightarrow e'$. (Moreover there exists at most one value term v such that $e \hookrightarrow^* v$.) Furthermore, a value v cannot be evaluated any further.*

Lemma 4. *(Weakenings of Judgments) For any environments Γ, Γ' and $x, \alpha \notin dom(\Gamma', \Gamma)$:*

1. *If $\Gamma', \Gamma \vdash e : t$ then $\Gamma', x{:}t_x, \Gamma \vdash e : t$ and $\Gamma', \alpha{:}k, \Gamma \vdash e : t$.*
2. *If $\Gamma', \Gamma \vdash s <: t$ then $\Gamma', x{:}t_x, \Gamma \vdash s <: t$ and $\Gamma', \alpha{:}k, \Gamma \vdash s <: t$.*

3. *If* $\Gamma', \Gamma \vdash_e p$ *then* $\Gamma', x{:}t_x, \Gamma \vdash_e p$ *and* $\Gamma', \alpha{:}k, \Gamma \vdash_e p$.

4. *If* $\Gamma', \Gamma \vdash_w t : k$ *then* $\Gamma', x{:}t_x, \Gamma \vdash_w t : k$ *and* $\Gamma', \alpha{:}k, \Gamma \vdash_w t : k$.

Proof. The proof is by straightforward mutual induction on the derivation trees of each type of judgment. In the base cases we rely on weakening typing judgments in the underlying System F (WF-Base) and on the fact that we can add extra variables to a closing substitution that don't appear in a predicate (Ent-Pred). □

Lemma 5. *(Reflexivity of* $<:$*) If* $\Gamma \vdash_w t : k$ *and* t *is not a type variable then* $\Gamma \vdash t <: t$.

Proof. We proceed by induction of the structure of the derivation of $\Gamma \vdash_w t : k$. *We could dispense with the hypothesis that* $t \not\equiv \alpha$ *by either making* α *a basic type or adding another subtyping rule that states* $(\alpha : k) \in \Gamma \Rightarrow \Gamma \vdash \alpha <: \alpha$

Case WF-Refn: In the base case, we have $t \equiv bx : p$ and $k \equiv B$. By inversion, we have for some $y \notin \text{dom}(\Gamma)$, the judgment $y{:}b, \lfloor\Gamma\rfloor \vdash_B p[y/x] : \mathsf{Bool}$. Let $\theta \in [\![y{:}b\{x : p\}, \Gamma]\!]$. Then we have that $\theta(y) \in [\![\theta(b\{x : p\})]\!] = [\![b\{x : \theta(p)\}]\!]$, and so $\theta(p)[\theta(y)/x] \hookrightarrow^* \texttt{true}$. But $\theta(p)[\theta(y)/x] = \theta(p[y/x])$, and so by rule Ent-Pred, $y{:}b\{x : p\}, \Gamma \vdash_e p[y/x]$. By S-Base, we conclude $\Gamma \vdash b\{x : p\} <: b\{x : p\}$.

Case WF-Kind: We have $\Gamma \vdash_w t : *$ and by inversion we have $\Gamma \vdash_w t : B$. By induction, we get $\Gamma \vdash t <: t$ as desired.

Case WF-Func: We have $t \equiv x{:}t_x \to t'$ and $k \equiv *$. By inversion, for some $y \notin \text{dom}(\Gamma)$ and some k_x, k we have

$$\Gamma \vdash_w t_x : k_x \ \text{ and } \ y{:}t_x, \Gamma \vdash_w t'[y/x] : k.$$

By the inductive hypothesis, we have $\Gamma \vdash t_x <: t_x$ and also $y{:}t_x, \Gamma \vdash_w t'[y/x] <: t'[y/x]$. Then by rule S-Func we have $\Gamma \vdash x{:}t_x \to t' <: x{:}t_x \to t'$.

Case WF-Exis: We have $t \equiv \exists\, x{:}t_x.\, t'$, and by inversion, for some $y \notin \text{dom}(\Gamma)$ and some k_x we have

$$\Gamma \vdash_w t_x : k_x \ \text{ and } \ y{:}t_x, \Gamma \vdash_w t'[y/x] : k.$$

By rule T-Var and the fact that $y{:}t_x, \Gamma \vdash \text{self}(t_x, y) <: t_x$ we have $y{:}t_x, \Gamma \vdash y : t_x$. By the inductive hypothesis we have $y{:}t_x, \Gamma \vdash t'[y/x] <: t'[y/x]$. Then by rule S-Witn (where $v_x \equiv y$) we have $y{:}t_x, \Gamma \vdash t'[y/x] <: \exists\, x{:}t_x.\, t'$. Then applying rule S-Bind we have $\Gamma \vdash \exists\, x{:}t_x.\, t' <: \exists\, x{:}t_x.\, t'$ as desired.

Case WF-Poly: We have $t \equiv \Lambda\alpha{:}k.t'$ and $\Gamma \vdash_w \Lambda\alpha{:}k.t' : *$. By inversion we have for some $\alpha' \notin \text{dom}(\Gamma)$, $\alpha'{:}k, \Gamma \vdash_w t'[\alpha'/\alpha] : k_t$. By induction, we have $\alpha'{:}k, \Gamma \vdash t'[\alpha'/\alpha] <: t'[\alpha'/\alpha]$. By rule S-Poly we conclude that $\Gamma \vdash \forall\, \alpha{:}k.\, t' <: \forall\, \alpha{:}k.\, t'$. □

Lemma 6. *If* $\Gamma \vdash t_1 <: t_2$ *then* $\lfloor t_1 \rfloor \overset{\alpha}{=} \lfloor t_2 \rfloor$.

Proof. We proceed by structural induction on the derivation tree of $\Gamma \vdash t_1 <: t_2$. In the base case Sub-Base, $t_1 \equiv b\{x_1 : p_1\}$ and $t_2 \equiv b\{x_2 : p_2\}$. Then $\lfloor t_1 \rfloor = b = \lfloor t_2 \rfloor$

Case Sub-Func: We have $t_1 \equiv x_1{:}s_1 \to t_1'$ and $t_2 \equiv x_2{:}s_2 \to t_2'$. By inversion we have for some $y \notin \text{dom}(\Gamma)$ that $\Gamma \vdash s_2 <: s_1$ and $y{:}s_2, \Gamma \vdash t_1'[y/x_1] <: t_2'[y/x_2]$. By the inductive hypothesis, $\lfloor s_2 \rfloor \overset{\alpha}{=} \lfloor s_1 \rfloor$ and $\lfloor t_1' \rfloor = \lfloor t_1'[y/x_1] \rfloor \overset{\alpha}{=} \lfloor t_2'[y/x_2] \rfloor = \lfloor t_2' \rfloor$. Combining these we obtain $\lfloor t_1 \rfloor = \lfloor s_1 \rfloor \to \lfloor t_1' \rfloor \overset{\alpha}{=} \lfloor s_2 \rfloor \to \lfloor t_2' \rfloor = \lfloor t_2 \rfloor$ as desired.

Case Sub-Witn: We have $\Gamma \vdash t_1 <: t_2$ where $t_2 \equiv \exists\, x{:}t_x.\, t'$. By inversion $\Gamma \vdash t_1 <: t'[v_x/x]$ and by the inductive hypothesis $\lfloor t_1 \rfloor \overset{\alpha}{=} \lfloor t'[v_x/x] \rfloor = \lfloor t' \rfloor = \lfloor \exists\, x{:}t_x.\, t' \rfloor$.

Case Sub-Bind: We have $\Gamma \vdash t_1 <: t_2$ where $t_1 \equiv \exists\, x{:}t_x.\, t'$. By inversion for some $y \notin \mathrm{dom}(\Gamma)$ such that $y \notin \mathrm{free}(t_2)$ we have $y{:}t_x, \Gamma \vdash t'[y/x] <: t_2$. Then by the inductive hypothesis $\lfloor t_2 \rfloor \overset{\alpha}{=} \lfloor t'[y/x] \rfloor = \lfloor t' \rfloor = \lfloor \exists\, x{:}t_x.\, t' \rfloor$.

Case Sub-Poly: We have $t_1 \equiv \forall\, \alpha_1{:}k.\, t_1'$ and $t_2 \equiv \forall\, \alpha_2{:}k.\, t_2'$. By inversion, for some $\alpha \notin \mathrm{dom}(\Gamma)$, $\alpha{:}k, \Gamma \vdash t_1'[\alpha/\alpha_1] <: t_2'[\alpha/\alpha_2]$. By the inductive hypothesis we have $\lfloor t_1' \rfloor[\alpha/\alpha_1] = \lfloor t_1'[\alpha/\alpha_1] \rfloor \overset{\alpha}{=} \lfloor t_2'[\alpha/\alpha_2] \rfloor = \lfloor t_2' \rfloor[\alpha/\alpha_2]$. Thus under an additional alpha-equivalence, $\lfloor \forall\, \alpha_1{:}k.\, t_1' \rfloor \overset{\alpha}{=} \lfloor \forall\, \alpha_2{:}k.\, t_2' \rfloor$. □

Our proof of the soundness theorems begin with several helping lemmas.

Lemma 7. *(Selfification and Denotations) If $\varnothing \vdash_w t : k$ and if $e \hookrightarrow^* v$ such that $v \in [\![t]\!]$ then $v \in [\![\mathrm{self}(t,e)]\!]$.*

Proof. We observe that $\lfloor \mathrm{self}(t,e) \rfloor = \lfloor t \rfloor$ so $\varnothing \vdash_B v : \lfloor \mathrm{self}(t,e) \rfloor$ if and only if $\varnothing \vdash_B v : \lfloor t \rfloor$. Also $\varnothing \vdash_B e : \lfloor t \rfloor$ by the preservation theorem for System F. We now proceed by structural induction on t.

In the first case, $t \equiv b\{z : p\}$ and $\mathrm{self}(t,e) = b\{z : p \wedge z = e\}$. By hypothesis, $v \in [\![t]\!]$ so $\varnothing \vdash_B v : b$ and $p[v/z] \hookrightarrow^* \texttt{true}$. Then we also have that $(p \wedge z = e)[v/z] = (p[v/z] \wedge v = e) \hookrightarrow^* (\texttt{true} \wedge v = e) \hookrightarrow^* (\texttt{true} \wedge v = v) \hookrightarrow^* \texttt{true}$. Therefore $v \in [\![\mathrm{self}(t,e)]\!]$, as desired.

Second, if $t \equiv \exists\, y{:}s_y.\, s$ then

$$[\![\mathrm{self}(t,e)]\!] = \{\hat{v} \mid \varnothing \vdash_B \hat{v} : \lfloor \mathrm{self}(s,e) \rfloor \;\wedge\; (\exists\, v_y \in [\![s_y]\!].\, \hat{v} \in [\![\mathrm{self}(s,e)[v_y/y]]\!])\}$$

We do have $v \in [\![\exists\, y{:}s_y.\, s]\!]$, so there exists $v_y \in [\![s_y]\!]$ such that $v \in [\![s[v_y/y]]\!]$. Then by the inductive hypothesis $v \in [\![\mathrm{self}(s[v_y/y], e)]\!] = [\![\mathrm{self}(s[v_y/y], e[v_y/y])]\!] = [\![\mathrm{self}(s,e)[v_y/y]$ where we again use the fact that $e = e[v_y/y]$ because e is typed in the empty environment and so cannot have free variables. Thus $v \in [\![\mathrm{self}(t,e)]\!]$.

In the remaining cases, $\mathrm{self}(t,e) = t$ and the lemma holds trivially. □

Lemma 8. *(Type Denotations) Our typing and subtyping relations are sound with respect to the denotational semantics of our types:*
1. If $\Gamma \vdash t_1 <: t_2$ then $\forall \theta. \theta \in [\![\Gamma]\!] \Rightarrow [\![\theta(t_1)]\!] \subseteq [\![\theta(t_2)]\!]$.
2. If $\Gamma \vdash e : t$, then $\forall \theta. \theta \in [\![\Gamma]\!] \Rightarrow \theta(e) \hookrightarrow^* v'$ such that $v' \in [\![\theta(t)]\!]$.

The proof is by mutual induction on the derivation trees of the respective subtyping and typing judgements. The need for mutual induction contrasts with Lemma 4 of [VSJ+14] and comes from the appearance of the typing judgement $\Gamma \vdash v_x : t_x$ in the antecedent of rule S-Witn.

Proof. (1) Suppose $\Gamma \vdash t_1 <: t_2$. We proceed by induction on the derivation tree of the subtyping relation.

Case Sub-Base: We have that $\Gamma \vdash b\{x_1 : p_1\} <: b\{x_2 : p_2\}$ where $t_1 \equiv b\{x_1 : p_1\}$ and $t_2 \equiv b\{x_2 : p_2\}$. By inversion, for some $y \notin \mathrm{dom}(\Gamma)$ we have

$$y{:}b\{x_1 : p_1\}, \Gamma \vdash_e p_2[y/x_2].$$

By inversion of Ent-Pred we have

$$\forall\, \theta'.\, \theta' \in [\![y{:}b\{x_1 : p_1\}, \Gamma]\!] \Rightarrow \theta'(p_2[y/x_2]) \hookrightarrow^* \texttt{true}. \tag{1}$$

We need to show $\forall\theta.\ \theta \in [\![\Gamma]\!] \Rightarrow [\![\theta(b\{x_1 : p_1\})]\!] \subseteq [\![\theta(b\{x_2 : p_2\})]\!]$. Equivalently,

$$\forall\theta.\, \theta \in [\![\Gamma]\!] \Rightarrow \{v \mid \varnothing \vdash_B v : b \ \wedge\ (\theta(p_1[v/x_1]) \hookrightarrow^* \texttt{true})\} \tag{2}$$
$$\subseteq \{v \mid \varnothing \vdash_B v : b \ \wedge\ (\theta(p_2[v/x_2]) \hookrightarrow^* \texttt{true})\} \tag{3}$$

Let $\theta \in [\![\Gamma]\!]$ be a closing substitution and let v a term in $[\![\theta(t_1)]\!]$. Then $\theta(t_1) = b\{x_1 : \theta(p_1)\}$ and $\theta(p_1[v/x_1]) \hookrightarrow^* \texttt{true}$. Let $\theta' = (y \mapsto v, \theta) \in [\![y{:}b\{x_1 : p_1\}, \Gamma]\!]$. By (1) we have $\theta'(p_2[y/x_2]) \hookrightarrow^* \texttt{true}$ and $\theta'(p_2[y/x_2]) = \theta(p_2[y/x_2][v/y]) = \theta(p_2[v/x_2])$, which proves $v \in [\![\theta(t_2)]\!]$.

Case Sub-Func: We have that $\Gamma \vdash x_1{:}s_1 \to t_1' <: x_2{:}s_2 \to t_2'$ where $t_1 \equiv x_1{:}s_1 \to t_1'$ and $t_2 \equiv x_2{:}s_2 \to t_2'$. By inversion of this rule, for some $y \notin \text{dom}(\Gamma)$,

$$\Gamma \vdash s_2 <: s_1 \quad \text{and} \quad y{:}s_2, \Gamma \vdash t_1'[y/x_1] <: t_2'[y/x_2]$$

By the inductive hypothesis,

$$\forall\theta.\, \theta \in [\![\Gamma]\!] \Rightarrow [\![\theta(s_2)]\!] \subseteq [\![\theta(s_1)]\!]$$

and

$$\forall\theta'.\, \theta' \in [\![y{:}s_2, \Gamma]\!] \Rightarrow [\![\theta'(t_1'[y/x_1])]\!] \subseteq [\![\theta'(t_2'[y/x_2])]\!] \tag{4}$$

We need to show $\forall\theta.\ \theta \in [\![\Gamma]\!] \Rightarrow [\![\theta(x_1{:}s_1 \to t_1')]\!] \subseteq [\![\theta(x_2{:}s_2 \to t_2')]\!]$. Equivalently,

$$\forall\theta.\, \theta \in [\![\Gamma]\!] \Rightarrow \{v \mid \varnothing \vdash_B v : \lfloor\theta(s_1)\rfloor \to \lfloor\theta(t_1')\rfloor \ \wedge\ (\forall\, v' \in [\![\theta(s_1)]\!].\, v\, v' \hookrightarrow^* v^* \in [\![\theta(t_1')[v'/x_1]]\!])\} \tag{5}$$
$$\subseteq \{v \mid \varnothing \vdash_B v : \lfloor\theta(s_2)\rfloor \to \lfloor\theta(t_2')\rfloor \ \wedge\ (\forall\, v' \in [\![\theta(s_2)]\!].\, v\, v' \hookrightarrow^* v^* \in [\![\theta(t_2')[v'/x_2]]\!])\} \tag{6}$$

Fix $\theta \in [\![\Gamma]\!]$ and let v be a term in set (5) and let $v' \in [\![\theta(s_2)]\!]$. Then by induction, $v' \in [\![\theta(s_1)]\!]$. So there exists a value v^* such that $(v\, v') \hookrightarrow^* v^*$ and $v^* \in [\![(\theta(t_1')[v'/x_1]]\!]$. Let $\theta' = (y \mapsto v', \theta)$. From (4) we also have that $[\![\theta'(t_1'[y/x_1])]\!] \subseteq [\![\theta'(t_2'[y/x_2])]\!]$. But $\theta'(t_1'[y/x_1]) = \theta(t_1'[y/x_1])[v'/y] = \theta(t_1')[v'/x_1]$ and $\theta'(t_2'[y/x_2]) = \theta(t_2')[v'/x_2]$. Therefore $v^* \in [\![\theta(t_1')[v'/x_2]]\!] \subseteq [\![\theta(t_2')[v'/x_2]]\!]$ and so v is in set (6) as desired. Finally, given $\varnothing \vdash_B v : \lfloor\theta(s_1)\rfloor \to \lfloor\theta(t_1')\rfloor$ we have $\varnothing \vdash_B v : \lfloor\theta(s_2)\rfloor \to \lfloor\theta(t_2')\rfloor$ by Lemma 6.

Case Sub-Witn: We have that $\Gamma \vdash t_1 <: \exists\, x{:}t_x.\, t_2'$ where $t_2 \equiv \exists\, x{:}t_x.\, t_2'$. By inversion, there exists some value term v_x such that

$$\Gamma \vdash v_x : t_x \quad \text{and} \quad \Gamma \vdash t_1 <: t_2'[v_x/x].$$

By the inductive hypothesis, we have

$$\forall\, \theta.\, \theta \in [\![\Gamma]\!] \Rightarrow [\![\theta(t_1)]\!] \subseteq [\![\theta(t_2'[v_x/x])]\!] \tag{7}$$

and by mutual induction we also have that there exists a value v' such that

$$\forall\, \theta.\, \theta \in [\![\Gamma]\!] \Rightarrow \theta(v_x) \hookrightarrow^* v' \in [\![\theta(t_x)]\!];$$

values are closed under substitution and cannot be reduced further, so we must have that $v' = \theta(v_x)$. We need to show that $\forall\, \theta$, if $\theta \in [\![\Gamma]\!]$, then $[\![\theta(t_1)]\!] \subseteq [\![\theta(\exists\, x{:}t_x.\, t_2')]\!]$. Fix some $\theta \in [\![\Gamma]\!]$. Then

$$[\![\theta(\exists\, x{:}t_x.\, t_2')]\!] = \{v \mid \varnothing \vdash_B v : \lfloor\theta(t_2')\rfloor \ \wedge\ (\exists\, v' \in [\![\theta(t_x)]\!].\, v \in [\![\theta(t_2')[v'/x]]\!])\} \tag{8}$$

because $\theta(\exists\, x{:}t_x.\, t'_2) = \exists\, x{:}\theta(t_x).\, \theta(t'_2)$. Let $v \in [\![\theta(t_1)]\!]$ and let $v' = \theta(v_x) \in [\![\theta(t_x)]\!]$ be as above. Then by (7), $v \in [\![\theta(t'_2[v_x/x])]\!] = [\![\theta(t'_2)[\theta(v_x)/x]]\!] = [\![\theta(t'_2)[v'/x]]\!]$. By 7 and by definition of the denotation of a type, $\varnothing \vdash_B v : \lfloor\theta(t'_2[v_x/x])\rfloor = \lfloor\theta(t'_2)\rfloor$. Therefore v is in the right hand side of (8).

Case Sub-Bind: We have that $\Gamma \vdash \exists\, x{:}t_x.\, t'_1 <: t_2$ where $t_1 \equiv \exists\, x{:}t_x.\, t'_1$. By inversion we have for some $y \notin \mathrm{dom}(\Gamma)$

$$y{:}t_x, \Gamma \vdash t'_1[y/x] <: t_2 \quad \text{and} \quad y \notin \mathit{free}(t_2).$$

By the inductive hypothesis, we have

$$\forall\, \theta'.\, \theta' \in [\![y{:}t_x, \Gamma]\!] \Rightarrow [\![\theta'(t'_1[y/x])]\!] \subseteq [\![\theta'(t_2)]\!]. \tag{9}$$

We need to show that for every $\theta \in [\![\Gamma]\!]$ that it holds that $[\![\theta(\exists\, x{:}t_x.\, t'_1)]\!] \subseteq [\![\theta(t_2)]\!]$. Fix some $\theta \in [\![\Gamma]\!]$ and let $v \in [\![\theta(\exists\, x{:}t_x.\, t'_1)]\!]$. By definition, $\theta(\exists\, x{:}t_x.\, t'_1) = \exists\, x{:}\theta(t_x).\, \theta(t'_1)$ so

$$[\![\theta(\exists\, x{:}t_x.\, t'_1)]\!] = \{v \mid \varnothing \vdash_B v : \lfloor\theta(t'_1)\rfloor \;\wedge\; (\exists\, v' \in [\![\theta(t_x)]\!].\, v \in [\![\theta(t'_1)[v'/x]]\!])\}. \tag{10}$$

Take v' as in (10) and let $\theta' = (y \mapsto v', \theta)$. We note that $\theta' \in [\![y{:}t_x, \Gamma]\!]$ because $\theta'(y) = v' \in [\![\theta(t_x)]\!] = [\![\theta'(t_x)]\!]$ where the last equality follows from the fact that x cannot appear free in t_x. Then $v \in [\![\theta(t'_1)[v'/x]]\!] = [\![\theta(t'_1[y/x])[v'/y]]\!] = [\![\theta'(t'_1[y/x])]\!]$, so from (9) we can conclude $v \in [\![\theta'(t_2)]\!] = [\![\theta(t_2)]\!]$ because y does not appear free in t_2 so $\theta'(t_2) = \theta(t_2)$.

Case Sub-Poly: We have that $\Gamma \vdash \forall\, \alpha_1{:}k.\, t'_1 <: \forall\, \alpha_2{:}k.\, t'_2$, where $t_1 \equiv \forall\, \alpha_1{:}k.\, t'_1$ and $t_2 \equiv \forall\, \alpha_2{:}k.\, t'_2$. By inversion, for some $\alpha \notin \mathrm{dom}(\Gamma)$, $\alpha{:}k, \Gamma \vdash t_1[\alpha/\alpha_1] <: t_2[\alpha/\alpha_2]$. By the inductive hypothesis, we have that for all $\theta' \in [\![\alpha{:}k, \Gamma]\!]$ we have $[\![\theta'(t_1[\alpha/\alpha_1])]\!] \subseteq [\![\theta'(t_2[\alpha/\alpha_2])]\!]$. We need to show

$$\forall \theta.\; \theta \in [\![\Gamma]\!] \Rightarrow [\![\theta(\forall\, \alpha_1{:}k.\, t_1)]\!] \subseteq [\![\theta(\forall\, \alpha_2{:}k.\, t_2)]\!].$$

Let $\theta \in [\![\Gamma]\!]$ arbitrary and let $v \in [\![\theta(\forall\, \alpha_1{:}k.\, t_1)]\!] = [\![\forall\, \alpha_1{:}k.\, \theta(t_1)]\!]$. Then $\varnothing \vdash_B v : \forall\, \alpha_1{:}k.\, \lfloor\theta(t_1)\rfloor$. By Lemma **??**, we also have $\varnothing \vdash_B v : \forall\, \alpha_2{:}k.\, \lfloor\theta(t_2)\rfloor$ because $\lfloor\theta(t_1[\alpha/\alpha_1]\rfloor \overset{\alpha}{=} \lfloor\theta(t_2[\alpha/\alpha_2])\rfloor$ and $\lfloor\forall\, \alpha_1{:}k.\, \theta(t_1)\rfloor = \lfloor\forall\, \alpha_2{:}k.\, \theta(t_2)\rfloor$. Now let t_α be a type such that $\varnothing \vdash_w t_\alpha : k$. Then we have that there exists a value v' such that $v\,[t_\alpha] \hookrightarrow^* v'$ and $v' \in [\![\theta(t_1)[t_\alpha/\alpha_1]]\!] = [\![\theta(t_1[\alpha/\alpha_1])[t_\alpha/\alpha]]\!]$. Let $\theta' = (\alpha \mapsto t_\alpha, \theta)$. Then $v' \in [\![\theta'(t_1[\alpha/\alpha_1])]\!]$ and by induction we have

$$v' \in [\![\theta'(t_2[\alpha/\alpha_2])]\!] = [\![\theta(t_2[\alpha/\alpha_2])[t_\alpha/\alpha]]\!] = [\![\theta(t_2)[t_\alpha/\alpha_2]]\!].$$

This proves that $v \in [\![\forall\, \alpha_2{:}k.\, \theta(t_2)]\!] = [\![\theta(\forall\, \alpha_2{:}k.\, t_2)]\!]$ as desired.

(2) Suppose $\Gamma \vdash e : t$. We proceed by induction on the derivation tree of the typing relation.

Case T-Prim: We have $\Gamma \vdash e : t$ where $e \equiv c$, a built-in primitive function or constant. By inversion, $ty(c) = t$. Let $\theta \in [\![\Gamma]\!]$. In one case $t \equiv b\{x : p\}$; then by Lemma 1 on constants, $\theta(c) = c \in [\![ty(c)]\!] = [\![\theta(ty(c))]\!]$. In the other case, $ty(c) \equiv x{:}t_x \to t'$; by Lemma 1, $c\; v_x \hookrightarrow \delta(c, v_x) \in [\![t'[v_x/x]]\!]$ for any $v_x \in [\![t_x]\!]$. There are no free variables in c or t so $\theta(c)$ is a value and $\theta(c) = c \in [\![ty(c)]\!] = [\![\theta(ty(c))]\!]$.

Case T-Var: We have $\Gamma \vdash e : t$ where $e \equiv x$ and $t \equiv \mathrm{self}(t', x)$. By inversion, $(x{:}t') \in \Gamma$. Then for any $\theta \in [\![\Gamma]\!]$, we have by definition $\theta(x) \in [\![\theta(t')]\!]$. By Lemma 7, we have $\theta(x) \in [\![\mathrm{self}(\theta(t'), \theta(x))]\!] = [\![\theta(\mathrm{self}(t', x))]\!] = [\![\theta(t)]\!]$.

Case T-App: We have $\Gamma \vdash e : t$ where $e \equiv e'\; e_x$ and $t \equiv \exists\, x{:}t_x.\, t'$. By inversion, $\Gamma \vdash e' : x{:}t_x \to t'$ and $\Gamma \vdash e_x : t_x$. By the inductive hypothesis we have both

$$\forall \theta.\, \theta \in [\![\Gamma]\!] \Rightarrow \exists\, v'.\, \theta(e') \hookrightarrow^* v' \text{ and } v' \in [\![\theta(x{:}t_x \to t')]\!] \tag{11}$$

and

$$\forall \theta.\, \theta \in [\![\Gamma]\!] \Rightarrow \exists\, v_x.\, \theta(e_x) \hookrightarrow^* v_x \text{ and } v_x \in [\![\theta(t_x)]\!]. \tag{12}$$

Fix some $\theta \in [\![\Gamma]\!]$ and let v' and v_x be as above; we must show that there exists some value v such that $\theta(e) = \theta(e')\ \theta(e_x) \hookrightarrow^* v$ and $v \in [\![\theta(t)]\!] = [\![\exists\, x{:}\theta(t_x).\, \theta(t')]\!]$. Because $v' \in [\![\theta(x{:}t_x \to t')]\!] = [\![x{:}\theta(t_x) \to \theta(t')]\!]$, we have that $\varnothing \vdash_B v' : \lfloor t_x \rfloor \to \lfloor t' \rfloor$ and that there exists a v such that $v'\ v_x \hookrightarrow^* v$ and $v \in [\![\theta(t')[v_x/x]]\!]$. Note that $\theta(e) = \theta(e')\ \theta(e_x) \hookrightarrow^* v'\ v_x \hookrightarrow^* v$ and we have $\varnothing \vdash_B v'\ v_x : \lfloor t' \rfloor$ because $\varnothing \vdash_B v_x : \lfloor t_x \rfloor$. Then by the soundness of the underlying System F, we have $\varnothing \vdash_B v : \lfloor t' \rfloor$. Thus we conclude that $v \in [\![\exists\, x{:}\theta(t_x).\, \theta(t')]\!]$ as required.

Case T-Abs: We have $\Gamma \vdash e : t$ where $e \equiv \lambda x.e'$ and $t \equiv x{:}t_x \to t'$. By inversion, there exists some $y \notin \mathrm{dom}(\Gamma)$ such that $y{:}t_x, \Gamma \vdash e'[y/x] : t'[y/x]$. By the inductive hypothesis,

$$\forall \theta'.\, \theta' \in [\![y{:}t_x, \Gamma]\!] \Rightarrow \exists\, v'.\, \theta'(e'[y/x]) \hookrightarrow^* v' \text{ and } v' \in [\![\theta'(t'[y/x])]\!]. \tag{13}$$

We need to show that for every $\theta \in [\![\Gamma]\!]$, there exists a value v such that $\theta(e) \hookrightarrow^* v$ and

$$\begin{aligned} v \in [\![\theta(x{:}t_x \to t')]\!] &= [\![x{:}\theta(t_x) \to \theta(t')]\!] \\ &= \{\hat{v} \mid (\varnothing \vdash_B \hat{v} : \lfloor \theta(t_x) \rfloor \to \lfloor \theta(t') \rfloor) \wedge (\forall\, v_x \in [\![\theta(t_x)]\!].\ \exists v'.\, \hat{v}\ v_x \hookrightarrow^* v' \in [\![\theta(t')[v_x/x]]\!])\} \end{aligned}$$

Let $\theta \in [\![\Gamma]\!]$ and let $v = \lambda x.\theta(e') = \theta(e)$. By repeated application of the substitution lemma for System F, $\varnothing \vdash_B v : \lfloor \theta(t) \rfloor$. Let $v_x \in [\![\theta(t_x)]\!]$ a value. Then let

$$\theta' := (y \mapsto v_x, \theta) \in [\![y{:}t_x, \Gamma]\!].$$

By rule E-AppAbs $v\ v_x \hookrightarrow \theta(e')[v_x/x] = \theta(e'[y/x])[v_x/y] = \theta'(e'[y/x]) \hookrightarrow^* v'$ and $v' \in [\![\theta'(t'[y/x])]\!] = [\![\theta(t')[v_x/x]]\!]$ by (13), which proves that $v \in [\![\theta(t)]\!]$.

Case T-AppT We have $\Gamma \vdash e : t$ where $e \equiv e'\ [t']$ and $t \equiv s[t'/\alpha]$. By inversion, $\Gamma \vdash e' : \forall\, \alpha{:}k.\, s$ and $\Gamma \vdash_w t' : k$. By the inductive hypothesis

$$\forall\, \theta.\, \theta \in [\![\Gamma]\!] \Rightarrow \exists v'.\, \theta(e') \hookrightarrow^* v' \text{ and } v' \in [\![\forall\, \alpha{:}k.\, \theta(s)]\!]. \tag{14}$$

Let $\theta \in [\![\Gamma]\!]$. Then $\theta(e) = \theta(e')\ [\theta(t')] \hookrightarrow^* v'\ [\theta(t')]$. By definition of a denotation, we have that there exists v'' such that $v'\ [\theta(t')] \hookrightarrow^* v''$ and $v'' \in [\![\theta(s)[\theta(t')/\alpha]]\!] = [\![\theta(t)]\!]$.

Case T-AbsT. We have $\Gamma \vdash e : t$ where $e \equiv \Lambda\alpha{:}k.e'$ and $t \equiv \forall\, \alpha{:}k.\, t'$. By inversion we have that there exists $\alpha' \notin \mathrm{dom}(\Gamma)$ such that $\alpha'{:}k, \Gamma \vdash e'[\alpha'/\alpha] : t'[\alpha'/\alpha]$, and by the inductive hypothesis,

$$\forall\, \theta'.\, \theta' \in [\![\alpha'{:}k, \Gamma]\!] \Rightarrow \exists v'.\, \theta'(e'[\alpha'/\alpha]) \hookrightarrow^* v' \text{ and } v' \in [\![\theta'(t'[\alpha'/\alpha])]\!]. \tag{15}$$

Let $\theta \in [\![\Gamma]\!]$ arbitrary and let t_α be a type such that $\varnothing \vdash_w t_\alpha : k$. Then we have $\theta' := (\alpha' \mapsto t_\alpha, \theta)$. Let $v = \Lambda\alpha{:}k.\theta(e')$. Then from (15), we have that v' exists such that

$$v\ [t_\alpha] \hookrightarrow \theta(e')[t_\alpha/\alpha] = \theta(e'[\alpha'/\alpha])[t_\alpha/\alpha'] = \theta'(e'[\alpha'/\alpha]) \hookrightarrow^* v'$$

and $v' \in [\![\theta'(t'[\alpha'/\alpha])]\!] = [\![\theta(t')[t_\alpha/\alpha]]\!]$. Finally, by repeated application of the System F substitution lemma, $\varnothing \vdash_B c : \lfloor \theta(t) \rfloor$. This proves that $v \in [\![\theta(t)]\!]$.

Case T-Let: We have $\Gamma \vdash e : t$ where $e \equiv \mathtt{let}\, x = e_x \,\mathtt{in}\, e'$. By inversion, we have for some $y \notin \mathrm{dom}(\Gamma)$, that $\Gamma \vdash e_x : t_x$, $(y{:}t_x, \Gamma) \vdash e'[y/x] : t[y/x]$, and $\Gamma \vdash_w t$ for some t_x. Then by the inductive hypothesis we have

$$\forall \theta.\, \theta \in [\![\Gamma]\!] \Rightarrow \exists v_x.\, \theta(e_x) \hookrightarrow^* v_x \text{ and } v_x \in [\![\theta(t_x)]\!]$$

and

$$\forall \theta'.\, \theta' \in [\![y{:}t_x, \Gamma]\!] \Rightarrow \exists v'.\, \theta'(e'[y/x]) \hookrightarrow^* v' \text{ and } v' \in [\![\theta'(t[y/x])]\!]. \tag{16}$$

Let $\theta \in [\![\Gamma]\!]$. Let v_x be as above and let $\theta' = (y \mapsto v_x, \theta) \in [\![y{:}t_x, \Gamma]\!]$ because we chose $\theta'(x) = v_x \in [\![\theta(t_x)]\!]$. From the operational semantics $\theta(\mathtt{let}\, x = e_x \,\mathtt{in}\, e') = (\mathtt{let}\, x = \theta(e_x) \,\mathtt{in}\, \theta(e')) \hookrightarrow^* (\mathtt{let}\, x = v_x \,\mathtt{in}\, \theta(e')) \hookrightarrow \theta(e')[v_x/x] = \theta(e'[y/x])[v_x/y] = \theta'(e'[y/x]) \hookrightarrow^* v'$ for some value v'. Then from (16),

$$v' \in [\![\theta'(t[y/x])]\!] = [\![\theta(t[y/x])[v_x/y]]\!] = [\![\theta(t[y/x])]\!],$$

where the last equality follows from the fact that the judgment $\Gamma \vdash_w t$ implies that y cannot be free in $\theta(t)$. In conclusion we have $\theta(e) \hookrightarrow^* v'$ and $v' \in [\![\theta(t[y/x])]\!]$ as required.

Case T-Ann: We have $\Gamma \vdash e : t$ where $e \equiv (e' : t)$. By inversion, $\Gamma \vdash e' : t$ and by the inductive hypothesis, there exists some value v such that $\theta(e') \hookrightarrow^* v$ and $v \in [\![\theta(t)]\!]$. By the operational semantics of type annotations, $\theta(e) = (\theta(e') : \theta(t)) \hookrightarrow^* (v : \theta(t)) \hookrightarrow v \in [\![\theta(t)]\!]$, as required.

Case T-Sub: We have $\Gamma \vdash e : t$ and by inversion, we have $\Gamma \vdash e : s$ and $\Gamma \vdash s <: t$ for some type s. By the inductive hypothesis, $\forall \theta.\, \theta \in [\![\Gamma]\!] \Rightarrow \exists v.\, \theta(e) \hookrightarrow^* v$ and $v \in [\![\theta(s)]\!]$. By mutual induction, part 1 of the Lemma gives us that $\forall \theta.\, \theta \in [\![\Gamma]\!] \Rightarrow [\![\theta(s)]\!] \subseteq [\![\theta(t)]\!]$. Then we conclude that $\forall \theta.\, \theta \in [\![\Gamma]\!] \Rightarrow \exists v.\, \theta(e) \hookrightarrow^* v$ and $\in [\![\theta(t)]\!]$ as desired. $\square$

Lemma 9. *(Exactness) The self() function satisfies the following properties:*
1. We have both $\Gamma \vdash \mathrm{self}(t, e) <: \mathrm{self}(\mathrm{self}(t, e), e)$ *and* $\Gamma \vdash \mathrm{self}(\mathrm{self}(t, e), e) <: \mathrm{self}(t, e)$.
2. If $\Gamma \vdash s <: t$ *and* $\Gamma \vdash e : t_e$ *then* $\Gamma \vdash \mathrm{self}(s, e) <: \mathrm{self}(t, e)$.
3. If $\Gamma \vdash v : t$ *then* $\Gamma \vdash v : \mathrm{self}(t, v)$.

Proof. (1) The first subtyping judgment follows from the fact that $\mathrm{self}(t', e) <: t'$ holds for any type t'; in particular, set $t' \equiv \mathrm{self}(t, e)$. The second judgment follows (by induction on t) from the fact for any predicate p we have $\Gamma \vdash b\{x_1 : p \wedge x_1 = e \wedge x_1 = e\} <: b\{x_1 : p \wedge x_1 = e\}$.

(2) We proceed by induction on the judgment $\Gamma \vdash s <: t$.

Case S-Base: We have $s \equiv b\{x_1 : p_1\}$ and $t \equiv b\{x_2 : p_2\}$. By inversion $y{:}b\{x_1 : p_1\}, \Gamma \vdash_e p_2[y/x_2]$ for some $y \notin \mathrm{dom}(\Gamma)$. By inversion of Ent-Pred we have $\forall \theta.\, \theta \in [\![y{:}b\{x_1 : p_1\}, \Gamma]\!] \Rightarrow \theta(p_2[y/x_2]) \hookrightarrow^* \mathtt{true}$. Let $\theta' \in [\![y{:}b\{x_1 : p_1 \wedge x_1 = e\}, \Gamma]\!]$. Then in particular $\theta'(p_2[y/x_2]) \hookrightarrow^* \mathtt{true}$. We also have $\theta'(y) = e \hookrightarrow^* \mathtt{true}$, so $\theta'((p_2 \wedge x_2 = e)[y/x_2]) \hookrightarrow^* \mathtt{true}$ and thus $y{:}b\{x_1 : p_1 \wedge x_1 = e\}, \Gamma \vdash_e (p_2 \wedge x_2 = e)[y/x_2]$ by rule Ent-Pred. Then by rule S-Base we conclude that $\Gamma \vdash b\{x_1 : p_1 \wedge p_1 = e\} <: b\{x_2 : p_2 \wedge p_2 = e\}$.

Case S-Func and S-Poly: The cases are trivial because in the former both s and t must be function types and in the latter both s and t must be polymorphic types. In either case $\mathrm{self}(s, e) = e$ and $\mathrm{self}(t, e) = t$.

Case S-Witn: We have $\Gamma \vdash s <: t$ where $t \equiv \exists\, x{:}t_x.\, t'$. By inversion we have $\Gamma \vdash v_x : t_x$ and $\Gamma \vdash s <: t'[v_x/x]$ for some value v_x. By the inductive hypothesis, $\Gamma \vdash \mathrm{self}(s, e) <: \mathrm{self}(t'[v_x/x], e)$. We have $\mathrm{self}(t'[v_x/x], e) = \mathrm{self}(t', e)[v_x/x]$ because x cannot appear free in e. Then by rule S-Witn we can conclude that $\Gamma \vdash \mathrm{self}(s, e) <: \exists\, x{:}t_x.\, \mathrm{self}(t', e)$, where the latter type is equal to $\mathrm{self}(t, e)$ by definition.

Case S-Bind: We have $\Gamma \vdash s <: t$ where $s \equiv \exists\, x{:}s_x.\, s'$. By inversion we have $y{:}s_x, \Gamma \vdash s'[y/x] <: t$ for some $y \notin \mathrm{dom}(\Gamma)$ such that $y \notin \mathit{free}(t)$. By the inductive hypothesis, $y{:}s_x, \Gamma \vdash \mathrm{self}(s'[y/x], e) <: \mathrm{self}(t, e)$. We have $\mathrm{self}(s'[y/x], e) = \mathrm{self}(s', e)[y/x]$ because x cannot appear free in e. Then by rule S-Bind (because $y \notin \Gamma$ so y cannot appear free in e) we have $\Gamma \vdash \exists\, s{:}s_x.\, \mathrm{self}(s', e) <: t$.

(3) We proceed by induction on the structure of the judgment $\Gamma \vdash v : t$. Several cases don't apply because v must be a value (T-App, T-AppT, T-Let, and T-Ann).

Case T-Prim: We have $v \equiv c$. If c is a constant then it already has an exact type, that is, $t \equiv \mathrm{self}(b\{x : \mathtt{true}\}, c)$ and this follows from part (1). Similarly, for each primitive function the type is already exact.

Case T-Var: We have $v \equiv x$ and $t \equiv \mathrm{self}(t', x)$. Then by T-Sub and part (1) $\Gamma \vdash v : \mathrm{self}(t, x)$.

Case T-Abs: We have $v \equiv \lambda x.\, e'$ and $t \equiv x{:}t_x \to t'$. We have $\mathrm{self}(x{:}t_x \to t', v) = x{:}t_x \to t'$ and so this case follows trivially.

Case T-AbsT: We have $v \equiv \Lambda\alpha{:}k.\, e$ and $t \equiv \forall\, \alpha{:}k.\, t'$. We have that $\mathrm{self}(\forall\, \alpha{:}k.\, t', v) = \forall\, \alpha{:}k.\, t'$ and so this case also follow trivially.

Case T-Sub: Inverting we have $\Gamma \vdash e : s$ and $\Gamma s <: t$. By the inductive hypothesis, $\Gamma \vdash e : \mathrm{self}(s, e)$. ☐

Lemma 10. *(The Substitution Lemma) If* $\Gamma \vdash v_x : t_x$ *and if* $\Gamma \vdash_w t_\alpha : k$ *then*
1. If $\Gamma', x{:}t_x, \Gamma \vdash t_1 <: t_2$ *and* $\vdash_w \Gamma', x{:}t_x, \Gamma$ *then*

$$\Gamma'[v_x/x], \Gamma \vdash t_1[v_x/x] <: t_2[v_x/x],$$

and if $\Gamma', \alpha{:}k, \Gamma \vdash t_1 <: t_2$ *and* $\vdash_w \Gamma', \alpha{:}k, \Gamma$ *then*

$$\Gamma'[t_\alpha/\alpha], \Gamma \vdash t_1[t_\alpha/\alpha] <: t_2[t_\alpha/\alpha].$$

2. If $\Gamma', x{:}t_x, \Gamma \vdash e : t$ *and* $\vdash_w \Gamma', x{:}t_x, \Gamma$*then*

$$\Gamma'[v_x/x], \Gamma \vdash e[v_x/x] : t[v_x/x],$$

and if $\Gamma', x{:}t_x, \Gamma \vdash e : t$ *and* $\vdash_w \Gamma', \alpha{:}k, \Gamma$ *then*

$$\Gamma'[t_\alpha/\alpha], \Gamma \vdash e[t_\alpha/\alpha] : t[t_\alpha/\alpha].$$

3. If $\Gamma', x{:}t_x, \Gamma \vdash_w t : k$ *and* $\vdash_w \Gamma', x{:}t_x, \Gamma$*then*

$$\Gamma'[v_x/x], \Gamma \vdash_w t[v_x/x] : k,$$

and if $\Gamma', x{:}t_x, \Gamma \vdash_w t : k$ *and* $\vdash_w \Gamma', \alpha{:}k, \Gamma$ *then*

$$\Gamma'[t_\alpha/\alpha], \Gamma \vdash_w t[t_\alpha/\alpha] : k.$$

Proof. (1) Suppose $\Gamma \vdash v_x : t_x$ and $\Gamma', x{:}t_x, \Gamma \vdash t_1 <: t_2$. We proceed by mutual induction (for parts 1 and 2) on the derivation tree of the subtyping relation. The proofs for substitution for a type variable are entirely similar, except where specifically noted.

Case Sub-Base: First, we have that $\Gamma', x{:}t_x, \Gamma \vdash b\{x_1 : p_1\} <: b\{x_2 : p_2\}$ where $t_1 \equiv b\{x_1 : p_1\}$ and $t_2 \equiv b\{x_2 : p_2\}$. By inversion, for some $y \notin \mathrm{dom}(\Gamma', x{:}t_x, \Gamma)$ we have

$$y{:}b\{x_1 : p_1\}, \Gamma', x{:}t_x, \Gamma \vdash_e p_2[y/x_2].$$

By inversion of Ent-Pred we have

$$\forall \theta^*.\, \theta^* \in [\![y{:}b\{x_1 : p_1\}, \Gamma', x{:}t_x, \Gamma]\!] \Rightarrow \theta^*(p_2[y/x_2]) \hookrightarrow^* \texttt{true}. \tag{17}$$

Let $\hat\theta = (y \mapsto v_y, \theta', \theta) \in [\![y{:}\{x_1 : p_1[v_x/x]\}, \Gamma'[v_x/x], \Gamma]\!]$ arbitrary and let $\theta^* = (y \mapsto v_y, \theta', x \mapsto v_x, \theta) \in [\![y{:}\{x_1 : p_1\}, \Gamma', x{:}t_x, \Gamma]\!]$. Then $\hat\theta(p_2[v_x/x][y/x_2]) = \hat\theta(p_2[y/x_2])[v_x/x] = \theta^*(p_2[y/x_2]) \hookrightarrow^* \texttt{true}$. Then we have that $y{:}\{x_1 : p_1[v_x/x]\}, \Gamma'[v_x/x], \Gamma \vdash_e p_2[v_x/x][y/x_2]$ by rule Ent-Pred, and $\Gamma'[v_x/x], \Gamma \vdash b\{x_1 : p_1[v_x/x]\} <: b\{x_2 : p_2[v_x/x]\}$ by Sub-Base.

Case Sub-Func: We have that $\Gamma', x : t_x, \Gamma \vdash x_1 : s_1 \to t_1' <: x_2 : s_2 \to t_2'$ where $t_1 \equiv x_1 : s_1 \to t_1'$ and $t_2 \equiv x_2 : s_2 \to t_2'$. By inversion, there exists some $y \notin \mathrm{dom}(\Gamma', x{:}t_x, \Gamma)$ such that

$$\Gamma', x : t_x, \Gamma \vdash s_2 <: s_1 \quad \text{and} \quad y{:}s_2, \Gamma', x : t_x, \Gamma \vdash t_1'[y/x_1] <: t_2'[y/x_2].$$

Applying the inductive hypothesis to the above, we get

$$\Gamma'[v_x/x], \Gamma \vdash s_2[v_x/x] <: s_1[v_x/x] \tag{18}$$

and

$$y{:}s_2[v_x/x], \Gamma'[v_x/x], \Gamma \vdash t_1'[y/x_1][v_x/x] <: t_2'[y/x_2][v_x/x] \tag{19}$$

We necessarily have that $x \neq y$ and v_x contains only free variables from Γ, so $t_1'[y/x_1][v_x/x] = t_1'[v_x/x][y/x_1]$ and $t_2'[y/x_2][v_x/x] = t_2'[v_x/x][y/x_2]$. By rule Sub-Fun applied to (18) and (19),

$$\Gamma'[v_x/x], \Gamma \vdash x_1 : s_1[v_x/x] \to t_1'[v_x/x] <: x_2 : s_2[v_x/x] \to t_2'[v_x/x]$$

This is the same as $\Gamma'[v_x/x], \Gamma \vdash t_1[v_x/x] <: t_2[v_x/x]$.

Case Sub-Witn: We have that $t_2 \equiv \exists\, y{:}t_y.\, t'$ and $\Gamma', x : t_x, \Gamma \vdash t_1 <: \exists\, y{:}t_y.\, t'$. By inversion, there exists some value v_y such that

$$\Gamma', x : t_x, \Gamma \vdash v_y : t_y \quad \text{and} \quad \Gamma', x : t_x, \Gamma \vdash t_1 <: t'[v_y/y]. \tag{20}$$

By the inductive hypothesis we have that $\Gamma'[v_x/x], \Gamma \vdash t_1[v_x/x] <: t'[v_y/y][v_x/x]$. By our convention that free and bound variables are distinct (and because v_x contains only free variables from Γ and v_y contains only free variables from $\Gamma', x{:}t_x, \Gamma$) we have $t'[v_y/y][v_x/x] = t'[v_x/x][v_y[v_x/x]/y]$. By the inductive hypothesis, we also have $\Gamma'[v_x/x], \Gamma \vdash v_y[v_x/x] : t_y[v_x/x]$. Applying rule Sub-Witn we have $\Gamma'[v_x/x], \Gamma \vdash t_1[v_x/x] <: \exists\, y{:}t_y[v_x/x].\, t'[v_x/x]$ as desired.

Case Sub-Bind: We have that $t_1 \equiv \exists\, y{:}t_y.\, t$ and $\Gamma', x : t_x, \Gamma \vdash \exists\, y{:}t_y.\, t <: t_2$. By inversion, we have for some $z \notin \mathrm{dom}(\Gamma', x : t_x, \Gamma)$ such that $z \notin \mathrm{free}(t_2)$,

$$z{:}t_y, \Gamma', x : t_x, \Gamma \vdash t[z/y] <: t_2.$$

By the inductive hypothesis,

$$z{:}t_y[v_x/x], \Gamma'[v_x/x], \Gamma \vdash t[z/y][v_x/x] <: t_2[v_x/x]. \tag{21}$$

Because $x \neq z$ and v_x contains only free variables from Γ we have $t[z/y][v_x/x] = t[v_x/x][z/y]$. Thus we can apply rule Sub-Bind to conclude that $\Gamma'[v_x/x], \Gamma \vdash \exists\, y{:}t_y[v_x/x].\, t[v_x/x] <: t_2[v_x/x]$.

Case Sub-Poly: We have that $\Gamma', x{:}t_x, \Gamma \vdash \forall\, \alpha_1{:}k.\, t_1' <: \forall\, \alpha_2{:}k.\, t_2'$ where $t_1 \equiv \forall\, \alpha_1{:}k.\, t_1'$ and $t_2 \equiv \forall\, \alpha_2{:}k.\, t_2'$. By inversion, there exists some $\alpha \notin \mathrm{dom}(\Gamma', x{:}t_x, \Gamma)$ such that

$$\alpha{:}k, \Gamma', x : t_x, \Gamma \vdash t_1'[\alpha/\alpha_1] <: t_2'[\alpha/\alpha_2].$$

Applying the inductive hypothesis to the above, we get

$$\alpha{:}k, \Gamma'[v_x/x], \Gamma \vdash t_1'[\alpha/\alpha_1][v_x/x] <: t_2'[\alpha/\alpha_2][v_x/x] \tag{22}$$

We necessarily have that $x \neq \alpha$ and v_x contains only free variables from Γ, so $t_1'[\alpha/\alpha_1][v_x/x] = t_1'[v_x/x][\alpha/\alpha_1]$ and $t_2'[\alpha/\alpha_1][v_x/x] = t_2'[v_x/x][\alpha/\alpha_1]$. By rule Sub-Fun applied to (22),

$$\Gamma'[v_x/x], \Gamma \vdash \forall\, \alpha_1{:}k.\, t_1'[v_x/x] <: \forall\, \alpha_2{:}k.\, t_2'[v_x/x].$$

This is the same as $\Gamma'[v_x/x], \Gamma \vdash t_1[v_x/x] <: t_2[v_x/x]$.

(2) Suppose $\Gamma \vdash v_x : t_x$ and $\Gamma', x : t_x, \Gamma \vdash e : t$. We proceed by induction on the derivation tree of the typing judgment $e : t$. The proofs for substitution of a type variable are entirely similar, except where specifically noted.

Case T-Prim: We have $\Gamma', x : t_x, \Gamma \vdash e : t$ where $e \equiv c$. By inversion, $t = ty(c)$. By Lemma 1, neither c nor $ty(c)$ contain any free variables so $c[v_x/x] = c$ and $ty(c)[v_x/x] = ty(c)$. By rule T-Prim, $\Gamma'[v_x/x], \Gamma \vdash c : ty(c)$ because the environment can be arbitrary, and so $\Gamma'[v_x/x], \Gamma \vdash c[v_x/x] : ty(c)[v_x/x]$.

Case T-Var: We have $\Gamma', x : t_x, \Gamma \vdash e : t$ where $e \equiv y$. and $t \equiv \mathrm{self}(t', y)$. By inversion we have $y{:}t' \in \Gamma', x : t_x, \Gamma$. There are three possible cases for where y is bound in the environment.

First suppose that $y{:}t' \in \Gamma$. Then, necessarily, $y \neq x$ and $y[v_x/x] = y$. Now $x{:}t_x$ is bound to the left of Γ, so x cannot appear free in t' and $t' = t'[v_x/x]$. By rule T-Var we have $\Gamma'[v_x/x], \Gamma \vdash y : \mathrm{self}(t', y)$ and so $\Gamma'[v_x/x], \Gamma \vdash y[v_x/x] : \mathrm{self}(t', y)[v_x/x]$.

Next suppose $y = x$. Then $t' = t_x$. Also, x cannot appear in t_x (i.e. x cannot be free in its own type). So $t_x = t_x[v_x/x] = t'[v_x/x]$. We also have $v_x = x[v_x/x] = y[v_x/x]$. By hypothesis, $\Gamma \vdash v_x : t_x$ and by Lemma 4 this judgment remains true with respect to more bindings on variables that don't appear in Γ; thus $\Gamma'[v_x/x], \Gamma \vdash v_x : t_x$. By Lemma 9 we have $\Gamma'[v_x/x], \Gamma \vdash v_x : \mathrm{self}(t_x, v_x)$. Then we conclude $\Gamma'[v_x/x], \Gamma \vdash y[v_x/x] : \mathrm{self}(t', y)[v_x/x]$.

Finally, suppose $y{:}t \in \Gamma'$. Then $y{:}t[v_x/x] \in \Gamma'[v_x/x]$, and by rule T-Var we have $\Gamma'[v_x/x], \Gamma \vdash y : t[v_x/x]$. We must have that $y \neq x$ and thus $y[v_x/x] = y$. Thus we conclude that $\Gamma'[v_x/x], \Gamma \vdash y[v_x/x] : t[v_x/x]$.

For the type variable substitution proof, we have $\Gamma', \alpha : k, \Gamma \vdash y : t$ and $\vdash_w \Gamma', \alpha : k, \Gamma$. By inversion we have $y{:}t \in \Gamma', \alpha; k, \Gamma$. We proceed as above, except that there are only two cases: $y{:}t \in \Gamma$ or $y{:}t \in \Gamma'$.

Case T-App:We have $\Gamma', x{:}t_x, \Gamma \vdash e : t$ where $e \equiv e_1\ e_2$ and $t \equiv \exists\, y{:}t_y.\, t'$. By inversion, $\Gamma', x{:}t_x, \Gamma \vdash e_1 : y{:}t_y \to t'$ and $\Gamma', x{:}t_x, \Gamma \vdash e_2 : t_y$. By the inductive hypothesis,

$$\Gamma'[v_x/x], \Gamma \vdash e_1[v_x/x] : y{:}t_y[v_x/x] \to t'[v_x/x] \tag{23}$$

and

$$\Gamma'[v_x/x], \Gamma \vdash e_2[v_x/x] : t_y[v_x/x]. \tag{24}$$

By applying rule T-App

$$\Gamma'[v_x/x], \Gamma \vdash e_1[v_x/x]\ e_2[v_x/x] : y{:}t_y[v_x/x] \to t'[v_x/x]. \tag{25}$$

Now by the definition of substitution we have $e_1[v_x/x]\ e_2[v_x/x] = (e_1\ e_2)[v_x/x] \equiv e[v_x/x]$ and $y{:}t_y[v_x/x] \to t'[v_x/x] = (y{:}t_y \to t')[v_x/x] \equiv t[v_x/x]$. Therefore, we conclude that $\Gamma'[v_x/x], \Gamma \vdash e[v_x/x] : t[v_x/x]$.

Case T-ABS:We have $\Gamma', x : t_x, \Gamma \vdash e : t$ where $e \equiv \lambda y.e'$ and $t \equiv y{:}t_y \to t'$. By inversion, $z{:}t_y, \Gamma', x{:}t_x, \Gamma \vdash e'[z/y] : t'[z/y]$ and $\Gamma', x{:}t_x, \Gamma \vdash_w t_y : k_y$ for some $z \notin \mathrm{dom}(\Gamma)$. By the inductive hypothesis

$$z{:}t_y[v_x/x], \Gamma'[v_x/x], \Gamma \vdash e'[z/y][v_x/x] : t'[z/y][v_x/x] \text{ and } \Gamma'[v_x/x], \Gamma \vdash_w t_y[v_x/x] : k_y. \tag{26}$$

We must have $x \neq z$ and we have $x \neq y$ because bound and free variables are taken to be distinct. Moreover, v_x contains only free variables from Γ, so $e'[z/y][v_x/x] = e'[v_x/x][z/y]$ and $t'[z/y][v_x/x] = t'[v_x/x][z/y]$. Then by rule T-ABS

$$\Gamma'[v_x/x], \Gamma' \vdash \lambda y.(e'[v_x/x]) : y{:}t_y[v_x/x] \to t'[v_x/x]. \tag{27}$$

By definition of substitution, we can rewrite the above as

$$\Gamma'[v_x/x], \Gamma \vdash (\lambda y.e')[v_x/x] : y{:}t_y \to t'[v_x/x].$$

Case T-APPT:We have $\Gamma', x{:}t_x, \Gamma \vdash e : t$ where $e \equiv e'\ [t']$ and $t \equiv s[t'/\alpha']$. By inversion, $\Gamma', x{:}t_x, \Gamma \vdash e' : \forall\, \alpha'{:}k'.\, s$ and $\Gamma', x{:}t_x, \Gamma \vdash_w t' : k'$. By the inductive hypothesis,

$$\Gamma'[v_x/x], \Gamma \vdash e'[v_x/x] : \forall\, \alpha'{:}k'.\, s[v_x/x] \tag{28}$$

and

$$\Gamma'[v_x/x], \Gamma \vdash_w t'[v_x/x] : k'. \tag{29}$$

By applying rule T-APPT

$$\Gamma'[v_x/x], \Gamma \vdash e'[v_x/x]\ [t'[v_x/x]] : s[v_x/x][t'[v_x/x]/\alpha']. \tag{30}$$

Now by the definition of substitution we have $e'[v_x/x]\ [t'[v_x/x]] = (e'\ [t'])[v_x/x] \equiv e[v_x/x]$ and $s[v_x/x][t'[v_x/x]/\alpha'] = s[t'/\alpha'][v_x/x] \equiv t[v_x/x]$. Therefore, we conclude that $\Gamma'[v_x/x], \Gamma \vdash e[v_x/x] : t[v_x/x]$.

Case T-ABST:We have $\Gamma', x : t_x, \Gamma \vdash e : t$ where $e \equiv \Lambda\alpha : k.e'$ and $t \equiv \forall\, \alpha{:}k.\, t'$. By inversion, $\alpha'{:}k, \Gamma', x{:}t_x, \Gamma \vdash e'[\alpha'/\alpha] : t'[\alpha'/\alpha]$ and $\alpha'{:}k, \Gamma', x{:}t_x, \Gamma \vdash_w t'[\alpha'/\alpha] : k'$ for some $\alpha' \notin \mathrm{dom}(\Gamma)$. By the inductive hypothesis

$$\alpha'{:}k, \Gamma'[v_x/x], \Gamma \vdash e'[\alpha'/\alpha][v_x/x] : t'[\alpha'/\alpha][v_x/x] \text{ and } \alpha'{:}k, \Gamma'[v_x/x], \Gamma \vdash_w t'[\alpha'/\alpha][v_x/x] : k'. \tag{31}$$

We must have $x \neq \alpha'$ and we have $x \neq \alpha$ because bound and free variables are taken to be distinct. Moreover, v_x contains only free variables from Γ, so $e'[\alpha'/\alpha][v_x/x] = e'[v_x/x][\alpha'/\alpha]$ and $t'[\alpha'/\alpha][v_x/x] = t'[v_x/x][\alpha'/\alpha]$. Then by rule T-ABST

$$\Gamma'[v_x/x], \Gamma' \vdash \Lambda\alpha : k.(e'[v_x/x]) : \forall\, \alpha{:}k.\, t'[v_x/x]. \tag{32}$$

By definition of substitution, we can rewrite the above as

$$\Gamma'[v_x/x], \Gamma \vdash (\Lambda\alpha : k.e')[v_x/x] : \forall\, \alpha{:}k.\, t'[v_x/x].$$

In the type variable substitution lemma, where we have an environment $\Gamma', \beta{:}k_\beta, \Gamma$ and $\Gamma \vdash_w t_\beta : k_\beta$ this proof is similar except that we argue that $e'[\alpha'/\alpha][t_\beta/\beta] = e'[t_\beta/\beta][\alpha'/\alpha]$

and $t'[\alpha'/\alpha][t_\beta/\beta] = t'[t_\beta/\beta][\alpha'/\alpha]$ because $\alpha' \neq \beta$ and only free variables from Γ may appear in t_β.

Case T-LET: We have $\Gamma', x{:}t_x, \Gamma \vdash e : t$ where $e \equiv (\texttt{let}\, y{=}e_1 \,\texttt{in}\, e_2)$ and $t \equiv t_2$. By inversion, $\Gamma', x{:}t_x, \Gamma \vdash e_1 : t_1$ and $z{:}t_1, \Gamma, x{:}t_x, \Gamma \vdash e_2[z/y] : t_2[z/y]$ for some type t_1 and some $y \notin \mathrm{dom}(\Gamma)$. By the inductive hypothesis we have

$$\Gamma'[v_x/x], \Gamma \vdash e_1[v_x/x] : t_1[v_x/x] \tag{33}$$

and

$$y{:}t_1[v_x/x], \Gamma'[v_x/x], \Gamma \vdash e_2[z/y][v_x/x] : t_2[z/y][e_x/x] \tag{34}$$

We must have $x \neq z$ and we have $x \neq y$ because bound and free variables are taken to be distinct. Moreover, v_x contains only free variables from Γ, so $e_2[z/y][v_x/x] = e_2[v_x/x][z/y]$ and $t_2[z/y][v_x/x] = t_2[v_x/x][z/y]$. Then by rule T-LET,

$$\Gamma'[v_x/x], \Gamma \vdash \texttt{let}\, y{=}e_1[v_x/x] \,\texttt{in}\, e_2[v_x/x] : t_2[v_x/x] \tag{35}$$

which we can write as

$$\Gamma'[v_x/x], \Gamma \vdash (\texttt{let}\, y{=}e_1 \,\texttt{in}\, e_2)[v_x/x] : t_2[v_x/x].$$

Case T-ANN: We have $\Gamma', x{:}t_x, \Gamma \vdash e : t$ where $e \equiv (e' : t)$. By inversion, we have $\Gamma', x{:}t_x, \Gamma \vdash e' : t$ and by the inductive hypothesis, $\Gamma'[v_x/x], \Gamma \vdash e'[v_x/x] : t[v_x/x]$. By rule T-ANN, we get

$$\Gamma'[v_x/x], \Gamma \vdash (e'[v_x/x] : t[v_x/x]) : t[v_x/x] \tag{36}$$

By definition of substitution, $(e'[v_x/x] : t[v_x/x]) = (e' : t)[v_x/x] = e[v_x/x]$, so from (36) we immediately get $\Gamma'[v_x/x], \Gamma \vdash e[v_x/x] : t[v_x/x]$.

Case T-SUB: We have $\Gamma', x{:}t_x, \Gamma \vdash e : t$. By inversion, we have $\Gamma', x{:}t_x, \Gamma \vdash e : s$, $\Gamma', x{:}t_x, \Gamma \vdash s <: t$, and $\Gamma', x{:}t_x, \Gamma \vdash_w t : k$ for some type s and kind k. By the inductive hypothesis we have

$$\Gamma'[v_x/x], \Gamma \vdash e[v_x/x] : s[v_x/x] \tag{37}$$

and by part (1) of the Lemma we have

$$\Gamma'[v_x/x], \Gamma \vdash s[v_x/x] <: t[v_x/x]. \tag{38}$$

By part (3) of the Lemma we also have $\Gamma'[v_x/x], \Gamma \vdash_w t[v_x/x] : k$. Then by rule T-SUB we have $\Gamma'[v_x/x], \Gamma \vdash e[v_x/x] : t[v_x/x]$.

(3) Suppose $\Gamma \vdash v_x : t_x$ and $\Gamma', x : t_x, \Gamma \vdash_w t : k$. We proceed by induction on the derivation tree of the typing judgment $e : t$. The proofs for substitution of a type variable are entirely similar, except where specifically noted.

Case WF-REFN: We have $\Gamma', x{:}t_x, \Gamma \vdash t : k$ where $t \equiv b\{y : p\}$ and $k \equiv B$. By inversion we have $z{:}b, \lfloor \Gamma', x{:}t_x, \Gamma \rfloor \vdash_B p[z/y] : \mathsf{Bool}$ for some $z \notin \mathrm{dom}(\Gamma', x{:}t_x, \Gamma)$. By the Substitution Lemma for System F we have

$$z{:}b, \lfloor \Gamma', \Gamma \rfloor \vdash_B p[z/y][v_x/x] : \mathsf{Bool}.$$

We must have $z \neq x$ and $x \neq x$ and v_x contains only free variables from Γ, so $p[z/y][v_x/x] = p[v_x/x][z/y]$. We also have that $\lfloor \Gamma', \Gamma \rfloor = \lfloor \Gamma'[v_x/x], \Gamma \rfloor$ because all refinements are erased . By rule WF-REFN we conclude that $\Gamma'[v_x/x], \Gamma \vdash_w b\{y : p[v_x/x]\} : B$.

Case WF-Kind: We have $\Gamma', x{:}t_x, \Gamma \vdash t : *$. By inversion, we have We have $\Gamma', x{:}t_x, \Gamma \vdash t : B$. By the inductive hypothesis, $\Gamma'[v_x/x], \Gamma \vdash t[v_x/x] : B$, and by rule WF-Kind we conclude $\Gamma'[v_x/x], \Gamma \vdash t[v_x/x] : *$.

Case WF-Var: We have $\Gamma', x{:}t_x, \Gamma \vdash \alpha' : k'$. By inversion we have that $\alpha'{:}k' \in \Gamma', x{:}t_x, \Gamma$. There are two possibilities for where α' appears in the environment: either in Γ or Γ' (we cannot have $\alpha' = x$ because one is a term variable and one is a type variable). Then $\alpha' = \alpha'[v_x/x]$. By rule WF-Var we thus have $\Gamma'[v_x/x], \Gamma \vdash_w \alpha'[v_x/x] : k'$.

Case WF-Func: We have $\Gamma', x{:}t_x, \Gamma \vdash t : k$ where $t \equiv y{:}t_y \to t'$ and $k \equiv *$. By inversion we have

$$\Gamma', x{:}t_x, \Gamma \vdash_w t_y : k_y \quad \text{and} \quad z{:}t_y, \Gamma', x{:}t_x, \Gamma \vdash_w t'[z/y] : k' \tag{39}$$

for some $z \notin \mathrm{dom}(\Gamma', x{:}t_x, \Gamma)$. By the inductive hypothesis on the above,

$$\Gamma'[v_x/x], \Gamma \vdash_w t_y[v_x/x] : k_y \quad \text{and} \quad z{:}t_y[v_x/x], \Gamma'[v_x/x], \Gamma \vdash_w t'[z/y][v_x/x] : k'. \tag{40}$$

We must have $z \neq x$ and v_x contains only free variables from Γ so $t'[z/y][v_x/x] = t'[v_x/x][z/y]$. Then by rule WF-Func we conclude $\Gamma'[v_x/x], \Gamma \vdash_w y{:}t_y[v_x/x] \to t'[v_x/x] : *$ and $y{:}t_y[v_x/x] \to t'[v_x/x] = (y{:}t_y \to t')[v_x/x]$.

Case WF-Exis: We have $\Gamma', x{:}t_x, \Gamma \vdash t : k$ where $t \equiv \exists\, y{:}t_y.\, t'$. By inversion we have

$$\Gamma', x{:}t_x, \Gamma \vdash_w t_y : k_y \quad \text{and} \quad z{:}t_y, \Gamma', x{:}t_x, \Gamma \vdash_w t'[z/y] : k' \tag{41}$$

for some $z \notin \mathrm{dom}(\Gamma', x{:}t_x, \Gamma)$. By the inductive hypothesis on the above,

$$\Gamma'[v_x/x], \Gamma \vdash_w t_y[v_x/x] : k_y \quad \text{and} \quad z{:}t_y[v_x/x], \Gamma'[v_x/x], \Gamma \vdash_w t'[z/y][v_x/x] : k'. \tag{42}$$

We must have $z \neq x$ and v_x contains only free variables from Γ so $t'[z/y][v_x/x] = t'[v_x/x][z/y]$. Then by rule WF-Exis we conclude $\Gamma'[v_x/x], \Gamma \vdash_w \exists\, y{:}t_y[v_x/x].\, t'[v_x/x] : k$ and $\exists\, y{:}t_y[v_x/x].\, t'[v_x/x] = (\exists\, y{:}t_y.\, t')[v_x/x]$.

Case WF-Poly: We have $\Gamma', x{:}t_x, \Gamma \vdash t : k$ where $t \equiv \forall\, \alpha{:}k'.\, t'$ and $k \equiv *$. By inversion we have $\alpha'{:}k', \Gamma', x{:}t_x, \Gamma \vdash_w t'[\alpha'/\alpha] : k_{t'}$for some $\alpha' \notin \mathrm{dom}(\Gamma', x{:}t_x, \Gamma)$. By the inductive hypothesis, $al'{:}k', Gamma'[v_x/x], \Gamma \vdash_w t'[\alpha'/\alpha][v_x/x] : k_{t'}$. We must have $x \neq \alpha'$ and v_x contains only free variables from Γ so $t'[\alpha'/\alpha][v_x/x] = t'[v_x/x][\alpha'/\alpha]$. Then by rule WF-Poly we conclude $\Gamma'[v_x/x], \Gamma \vdash_w \forall\, \alpha{:}k'.\, t'[v_x/x] : *$ and $\forall\, \alpha{:}k'.\, t'[v_x/x] = (\forall\, \alpha{:}k'.\, t')[v_x/x]$.

Finally, suppose $\Gamma \vdash_w t_\alpha : k$ and $\Gamma', \alpha{:}k, \Gamma \vdash e : t$. We give the proof of WF-Refn for type variable substitution because it is slightly different as System F types do contain type variables. In this case we have $t \equiv b\{y : p\}$ and $k \equiv B$. By inversion we have $z{:}b, \lfloor \Gamma', \alpha{:}k, \Gamma \rfloor \vdash_B p[y/x] : \mathsf{Bool}$ for some $z \notin \mathrm{dom}(\Gamma', x{:}t_x, \Gamma)$. By the Substitution Lemma for System F we have

$$z{:}b, \lfloor \Gamma'[t_\alpha/\alpha], \Gamma \rfloor \vdash_B p[z/y][t_\alpha/\alpha] : \mathsf{Bool}.$$

We must have $z \neq \alpha$ and $y \neq \alpha$ and t_α contains only free variables from Γ, so $p[z/y][t_\alpha/\alpha] = p[t_\alpha/\alpha][z/y]$. By rule WF-Refn we conclude that $\Gamma'[t_\alpha/\alpha], \Gamma \vdash_w b\{y : p[t_\alpha/\alpha]\} : B$.

We also consider the proof of WF-Var. Here we have $\Gamma', \alpha{:}k, \Gamma \vdash \alpha' : k'$. By inversion we have that $\alpha'{:}k' \in \Gamma', \alpha{:}k, \Gamma$. There are three possibilities for where α appears in the environment. First, consider either $\alpha'{:}k' \in \Gamma'$ or $\alpha'{:}k' \in \Gamma$. Then $\alpha \neq \alpha'$ so $\alpha'[t_\alpha/\alpha] = \alpha'$. So we still have $\alpha'{:}k' \in \Gamma'[t_\alpha/\alpha], \Gamma$ and by rule WF-Var, $\Gamma'[t_\alpha/\alpha], \Gamma \vdash_w \alpha' : k$. The other possibility is that $\alpha = \alpha'$ and $k = k'$ in which case $\alpha'[t_\alpha/\alpha] = t_\alpha$. By hypothesis $\Gamma \vdash_w t_\alpha : k$ and by repeated application of Lemma 4 we have $\Gamma'[t_\alpha/\alpha], \Gamma \vdash_w t_\alpha : k$. In other words, $\Gamma'[t_\alpha/\alpha], \Gamma \vdash_w \alpha'[t_\alpha/\alpha] : k'$. □

Lemma 11. *(Well-formedness of types in judgments) If* $\Gamma \vdash e : t$ *and* $\vdash_w \Gamma$ *then* $\Gamma \vdash_w t : k$ *for some kind* k*.*

Proof. We proceed by induction on the derivation tree of the judgment $\Gamma \vdash e : t$.

Case T-Prim: We have $e \equiv c$. By inversion, $t = ty(c)$ and by Lemma 1 we have $\varnothing \vdash_w ty(c) : k$, where k is either B or $*$ depending on whether c is a Boolean/integer constant or function. By repeated application of Lemma 4, we have $\Gamma \vdash_w ty(c) : k$.

Case T-Var: We have $\Gamma \vdash e : t$ where $e \equiv x$ and $t \equiv \mathrm{self}(t', x)$. By inversion, $x{:}t' \in \Gamma$, so we can write $\Gamma \equiv \Gamma'', x{:}t', \Gamma'$ and by repeated inversion of WFE-Bind, $\vdash_w x{:}t', \Gamma'$. Inverting once more we get $\Gamma' \vdash_w t' : k$. Inductively applying Lemma 4 gives us $\Gamma \vdash_w t' : k$. By a simple inductive argument, selfification of well-formed types are themselves well-formed. Thus we conclude that $\Gamma \vdash_w \mathrm{self}(t', x) : k$.

Case T-App: We have $\Gamma \vdash e : t$ where $e \equiv e_1\ e_2$ and $t \equiv \exists\, x{:}t_x.\, t'$. By inversion, $\Gamma \vdash e_1 : x{:}t_x \to t'$ and $\Gamma \vdash e_2 : t_x$. By the inductive hypothesis we have $\Gamma \vdash_w x{:}t_x \to t' : k$ where $k \equiv *$ because WF-Func is the only rule that could have resulted in a well-formedness judgment for a function type. By inverting rule WF-Func (on the aforementioned judgment), we have $\Gamma \vdash_w t_x : k_x$ and $y{:}t_x, \Gamma, \vdash_w t'[y/x] : k'$ for some $y \notin \mathrm{dom}(\Gamma)$. By rule WF-Exis, $\Gamma \vdash_w \exists\, x{:}t_x.\, t' : k'$.

Case T-Abs: We have $\Gamma \vdash e : t$ where $e \equiv \lambda x.\, e'$ and $t \equiv x{:}t_x \to t'$. By inversion, we have $y{:}t_x, \Gamma \vdash e[y/x] : t'[y/x]$ and $\Gamma \vdash_w t_x : k_x$ for some $y \notin \mathrm{dom}(\Gamma)$. By the inductive hypothesis, we have $y{:}t_x, \Gamma \vdash_w t'[y/x] : k'$ for some kind k'. By rule WF-Func we have $\Gamma \vdash_w x{:}t_x \to t' : *$.

Case T-AppT: We have $\Gamma \vdash e : t$ where $e \equiv e'\ [t']$ and $t \equiv s[t'/\alpha]$. By inversion, $\Gamma \vdash e' : \forall\, \alpha{:}k.\, s$ and $\Gamma \vdash_w t' : k$. By the inductive hypothesis we have $\Gamma \vdash_w \forall\, \alpha{:}k.\, s : k'$ where $k' \equiv *$ because WF-Poly is the only rule that could have resulted in a well-formedness judgment for a polymorphic type. By inverting rule WF-Poly (on the aforementioned judgment), we have $\alpha'{:}k, \Gamma \vdash_w s[\alpha'/\alpha] : k_s$ for some $\alpha' \notin \mathrm{dom}(\Gamma)$. By the Substitution Lemma, we have $\Gamma \vdash_w s[\alpha'/\alpha][t'/\alpha'] : k_s$. We conclude by noting that $s[\alpha'/\alpha][t'/\alpha'] = s[t'/\alpha]$.

Case T-AbsT: We have $\Gamma \vdash e : t$ where $e \equiv \Lambda\alpha : k.\, e'$ and $t \equiv \forall\, \alpha{:}k.\, t'$. By inversion, we have $\alpha'{:}k, \Gamma \vdash e'[\alpha'/\alpha] : t'[\alpha'/\alpha]$ and $\alpha'{:}k, \Gamma \vdash_w t'[\alpha'/\alpha] : k'$ for some $\alpha' \notin \mathrm{dom}(\Gamma)$. By rule WF-Func we have $\Gamma \vdash_w \forall\, \alpha{:}k.\, t' : *$.

Case T-Let: We have $\Gamma \vdash e : t$ where $e \equiv \mathtt{let}\, x = e_x\, \mathtt{in}\, e'$. By inversion we have, in particular, that $\Gamma \vdash_w t : k$ for some kind k.

Case T-Ann: We have $\Gamma \vdash e : t$ where $e \equiv e' : t$. By inversion we have, in particular, that $\Gamma \vdash_w t : k$ for some kind k.

Case T-Sub: We have $\Gamma \vdash e : t$. By inversion we have, in particular, $\Gamma \vdash_w t : k$ for some kind k. □

Lemma 12. *(Witnesses and subtyping) If* $\Gamma \vdash v_x : t_x$ *and* $y{:}t_x, \Gamma \vdash_w t : k$ *then* $\Gamma \vdash t[v_x/x] <: \exists\, x{:}t_x.\, t$*.*

Proof. By Lemma 5, we have that $y{:}t_x \Gamma \vdash t <: t$ and by the Substitution Lemma we have $\Gamma \vdash t[v_x/x] <: t[v_x/x]$. Applying rule S-Witn, we get $\Gamma \vdash t[v_x/x] <: \exists\, x{:}t_x.\, t$. □

Lemma 13. *(Narrowing) We can replace types by subtypes in the environment. More precisely, if* $\Gamma \vdash s_x <: t_x$ *then*
1. If $\Gamma', x{:}t_x, \Gamma \vdash_w t : k$ *then*

$$\Gamma', x{:}s_x, \Gamma \vdash_w t : k.$$

2. If $\Gamma', x{:}t_x, \Gamma \vdash_e p$ *then*

$$\Gamma', x{:}s_x, \Gamma \vdash_e p.$$

3. If $\Gamma', x{:}t_x, \Gamma \vdash t_1 <: t_2$ *then*

$$\Gamma', x{:}s_x, \Gamma \vdash t_1 <: t_2.$$

4. If $\Gamma', x{:}t_x, \Gamma \vdash e : t$ *then*

$$\Gamma', x{:}s_x, \Gamma \vdash e : t.$$

Proof. (1) We proceed by induction on the derivation tree of $\Gamma', x{:}t_x, \Gamma \vdash_w t : k$.

Case WF-Refn: We have $\Gamma', x{:}t_x, \Gamma \vdash_w b\{y : p\} : B$. By inversion, we have

$$z{:}b, \lfloor\Gamma'\rfloor, x{:}\lfloor t_x\rfloor, \lfloor\Gamma\rfloor \vdash_B p[z/y] : \mathsf{Bool} \tag{43}$$

for some $z \notin \text{dom}(\Gamma', x{:}t_x, \Gamma)$. By Lemma 6, $\lfloor s_x\rfloor \overset{\alpha}{=} \lfloor t_x\rfloor$. Judgments in our System F remain valid *mutatis mutandis* under alpha-renaming bound variables in types in the environment, so we obtain $z{:}b, \lfloor\Gamma'\rfloor, x{:}\lfloor s_x\rfloor, \lfloor\Gamma\rfloor \vdash_B p[z/y] : \mathsf{Bool}$. Applying rule WF-Refn, $\Gamma', x{:}s_x, \Gamma \vdash_w b\{y : p\} : B$.

Case WF-Kind: We have $\Gamma', x{:}t_x, \Gamma \vdash_w t : *$. By inversion we have $\Gamma', x{:}t_x, \Gamma \vdash_w t : B$. By the inductive hypothesis, $\Gamma', x{:}s_x, \Gamma \vdash_w t : B$. Then by rule WF-Kind, $\Gamma', x{:}s_x, \Gamma \vdash t : *$.

Case WF-Var: We have $\Gamma', x{:}t_x, \Gamma\alpha : k$. By inversion we have $\alpha{:}k \in \Gamma', x{:}t_x, \Gamma$ and thus $\alpha{:}k \in \Gamma', x{:}s_x, \Gamma$. By rule WF-Var we conclude $\Gamma', x{:}s_x, \Gamma \vdash_w \alpha : k$.

Case WF-Func: We have $\Gamma', x{:}t_x, \Gamma \vdash_w y{:}t_y \to t : *$. By inversion, $\Gamma', x{:}t_x, \Gamma \vdash_w t_y : k_y$ and $z{:}t_y, \Gamma', x{:}t_x, \Gamma \vdash_w t[z/y] : k$ for some $z \notin \text{dom}(\Gamma', x{:}t_x, \Gamma) = \text{dom}(\Gamma', x{:}s_x, \Gamma)$. By the inductive hypothesis, $\Gamma', x{:}s_x, \Gamma \vdash_w t_y : k_y$ and $z{:}t_y, \Gamma', x{:}s_x, \Gamma \vdash_w t[z/y] : k$. Then applying WF-Func we conclude $\Gamma', x{:}s_x, \Gamma \vdash_w y{:}t_y \to t : *$.

Case WF-Exis: We have $\Gamma', x{:}t_x, \Gamma \vdash_w \exists\, y{:}t_y.\, t : k$. By inversion, $\Gamma', x{:}t_x, \Gamma \vdash_w t_y : k_y$ and $z{:}t_y, \Gamma', x{:}t_x, \Gamma \vdash_w t[z/y] : k$ for some $z \notin \text{dom}(\Gamma', x{:}t_x, \Gamma) = \text{dom}(\Gamma', x{:}s_x, \Gamma)$. By the inductive hypothesis, $\Gamma', x{:}s_x, \Gamma \vdash_w t_y : k_y$ and $z{:}t_y, \Gamma', x{:}s_x, \Gamma \vdash_w t[z/y] : k$. Then applying WF-Exis we conclude $\Gamma', x{:}s_x, \Gamma \vdash_w \exists\, y{:}t_y.\, t : k$.

Case WF-Poly: We have $\Gamma', x{:}t_x, \Gamma \vdash_w \forall\, \alpha{:}k.\, t : *$. By inversion, we have $\alpha'{:}k, \Gamma', x{:}t_x, \Gamma \vdash_w t[\alpha'/\alpha] : k_t$ for some $\alpha \notin \text{dom}(\Gamma', x{:}t_x, \Gamma \vdash_w) = \text{dom}(\Gamma', x{:}s_x, \Gamma \vdash_w)$. By the inductive hypothesis we have, $\alpha'{:}k, \Gamma', x{:}s_x, \Gamma \vdash_w t[\alpha'/\alpha] : k_t$ and by rule WF-Poly we conclude $\Gamma', x{:}s_x, \Gamma \vdash_w \forall\, \alpha{:}k.\, t : *$.

(2) We have $\Gamma', x{:}t_x, \Gamma \vdash_e p$ and by inversion of the only rule we have that $\forall \theta.\, \theta \in [\![\Gamma', x{:}t_x, \Gamma]\!] \Rightarrow \theta(p) \hookrightarrow^* \texttt{true}$. We observe that if $\theta \in [\![\Gamma', x{:}s_x, \Gamma]\!]$ then $\theta(x) \in [\![\theta(s_x)]\!] \subseteq [\![\theta(t_x)]\!]$ by Lemma 8; therefore $\theta \in [\![\Gamma', x{:}t_x, \Gamma]\!]$. Then we have the statement $\forall \theta.\, \theta \in [\![\Gamma', x{:}s_x, \Gamma]\!] \Rightarrow \theta(p) \hookrightarrow^* \texttt{true}$, and by rule Ent-Pred we conclude $\Gamma', x{:}s_x, \Gamma \vdash_e p$.

(3) We proceed by mutual induction on the derivation of the subtyping and typing judgments (part 4).

Case S-Base: We have $\Gamma', x{:}t_x, \Gamma \vdash b\{y_1 : p_1\} <: b\{y_2 : p_2\}$. By inversion, we have $z{:}b\{y_1 : p_1\}, \Gamma', x{:}t_x, \Gamma \vdash_e p2[z/y_2]$ for some $z \notin \text{dom}(\Gamma', x{:}t_x, \Gamma)$. By part (2) we have $z{:}b\{y_1 : p_1\}, \Gamma', x{:}s_x, \Gamma \vdash_e p[z/y_2]$ and by rule S-Base we conclude $\Gamma', x{:}s_x, \Gamma \vdash b\{y_1 : p_1\} <: b\{y_2 : p_2\}$.

Case S-Func: We have $\Gamma', x{:}t_x, \Gamma \vdash y_1{:}s_1 \to t_1 <: y_2{:}s_2 \to t_2$. By inversion we have $\Gamma', x{:}t_x, \Gamma \vdash s_2 <: s_1$ and $z{:}s_2, \Gamma', x{:}t_x, \Gamma \vdash t_1[z/y_1] <: t_2[z/y_2]$ for some $z \notin \text{dom}(\Gamma', x{:}t_x, \Gamma)$. By the inductive hypothesis, we have $\Gamma', x{:}s_x, \Gamma \vdash s_2 <: s_1$ and $z{:}s_2, \Gamma', x{:}s_x, \Gamma \vdash t_1[z/y_1] <: t_2[z/y_2]$. By rule S-Func we conclude $\Gamma', x{:}s_x, \Gamma \vdash y_1{:}s_1 \to t_1 <: y_2{:}s_2 \to t_2$.

Case S-Witn: We have $\Gamma', x{:}t_x, \Gamma \vdash t <: \exists\, y{:}t_y.\, t'$. By inversion we have $\Gamma', x{:}t_x, \Gamma \vdash v_y : t_y$ and $\Gamma', x{:}t_x, \Gamma \vdash t <: t'[v_y/y]$ for some value v_y. By the inductive hypothesis we have $\Gamma', x{:}s_x, \Gamma \vdash v_y : t_y$ and $\Gamma', x{:}s_x, \Gamma t <: t'[v_y/y]$. We conclude by rule S-Witn that $\Gamma', x{:}s_x, \Gamma \vdash t <: \exists\, y{:}t_y.\, t'$.

Case S-Bind: We have $\Gamma', x{:}t_x, \Gamma \vdash \exists\, y{:}t_y.\, t <: t$. By inversion, $z{:}t_y, \Gamma', x{:}t_x, \Gamma \vdash t[z/y] <: t'$ for some $z \notin \mathrm{dom}(\Gamma', x{:}t_x, \Gamma)$ such that $z \notin \mathit{free}(t')$. By the inductive hypothesis, $z{:}t_y, \Gamma', x{:}s_x, \Gamma \vdash t[z/y] <: t'$. Then by rule S-Bind we conclude $\Gamma', x{:}s_x, \Gamma \vdash \exists\, y{:}t_y.\, t <: t'$.

Case S-Poly: We have $\Gamma', x{:}t_x, \Gamma \vdash \forall\, \alpha_1{:}k.\, t_1 <: \forall\, \alpha_2{:}k.\, t_2$. By inversion, $\alpha{:}k, \Gamma', x{:}t_x, \Gamma \vdash t_1[\alpha/\alpha_1] <: t_2[\alpha/\alpha_2]$ for some $\alpha \notin \mathrm{dom}(\Gamma', x{:}t_x, \Gamma)$. By the inductive hypothesis, we have $\alpha{:}k, \Gamma', x{:}s_x, \Gamma \vdash t_1[\alpha/\alpha_1] <: t_2[\alpha/\alpha_2]$ and by rule S-Poly we get $\Gamma', x{:}s_x, \Gamma \vdash \forall\, \alpha_1{:}k.\, t_1 <: \forall\, \alpha_2{:}k.\, t_2$.

(4) As in part (3), we proceed by mutual induction on the derivation of the subtyping and typing judgments.

Case T-Prim: We have $\Gamma', x{:}t_x, \Gamma \vdash c : t$. By inversion, $ty(c) = t$, so by T-Prim we have $\Gamma', x{:}s_x, \Gamma \vdash c : t$.

Case T-Var: We have $\Gamma', x{:}t_x, \Gamma \vdash y : \mathrm{self}(t, y)$. By inversion, $y{:}t \in \Gamma', x{:}t_x, \Gamma$. If $y \neq x$ then $y{:}t \in \Gamma', x{:}s_x, \Gamma$ and we conclude by rule T-Var. If $y = x$ and $t = t_x$ then we have that $\Gamma', x{:}s_x, \Gamma \vdash y : \mathrm{self}(s_x, y)$. By Lemma 9, $\Gamma', x{:}s_x, \Gamma \vdash \mathrm{self}(s_x, y) <: \mathrm{self}(t_x, y)$. By rule T-Sub we conclude that $\Gamma', x{:}s_x, \Gamma \vdash y : \mathrm{self}(t_x, y)$.

Case T-App: We have $\Gamma', x{:}t_x, \Gamma \vdash e\, e' : \exists\, y{:}t_y.\, t'$. By inversion, $\Gamma', x{:}t_x, \Gamma \vdash e : y{:}t_y \to t'$ and $\Gamma', x{:}t_x, \Gamma \vdash e' : t_y$. Applying the inductive hypothesis twice, $\Gamma', x{:}s_x, \Gamma \vdash e : y{:}t_y \to t'$ and $\Gamma', x{:}s_x, \Gamma \vdash e' : t_y$. By rule T-App we conclude that $\Gamma', x{:}s_x, \Gamma \vdash e\, e' : \exists\, y{:}t_y.\, t'$.

Case T-Abs: We have $\Gamma', x{:}t_x, \Gamma \vdash \lambda x.\, e : y{:}t_y \to t'$. By inversion, $z{:}t_y, \Gamma', x{:}t_x, \Gamma \vdash e[z/y] : t'[z/y]$ and $\Gamma', x{:}t_x, \Gamma t_y : k_y$ for some $z \notin \mathrm{dom}(\Gamma', x{:}t_x, \Gamma)$. Applying the inductive hypothesis twice, $z{:}t_y, \Gamma', x{:}s_x, \Gamma \vdash e[z/y] : t'[z/y]$ and $\Gamma', x{:}s_x, \Gamma t_y : k_y$. Then by rule T-Abs we conclude $\Gamma', x{:}s_x, \Gamma \vdash \lambda x.\, e : y{:}t_y \to t'$.

Case T-AppT: We have $\Gamma', x{:}t_x, \Gamma \vdash e\,[t'] : s[t'/\alpha]$. By inversion, $\Gamma', x{:}t_x, \Gamma \vdash e : \forall\, \alpha{:}k.\, s$ and $\Gamma', x{:}t_x, \Gamma t' : k'$. By part (1) and by applying the inductive hypothesis, $\Gamma', x{:}s_x, \Gamma \vdash e : \forall\, \alpha{:}k.\, s$ and $\Gamma', x{:}s_x, \Gamma t' : k'$. Applying rule T-AppT we conclude $\Gamma', x{:}s_x, \Gamma \vdash e\,[t'] : s[t'/\alpha]$

Case T-AbsT: We have $\Gamma', x{:}t_x, \Gamma \vdash \Lambda\alpha{:}k.\, e : \forall\, \alpha{:}k.\, t$. By inversion, $\alpha'{:}k, \Gamma', x{:}t_x, \Gamma \vdash e[\alpha'/\alpha] : t[\alpha'/\alpha]$ and $\alpha'{:}k, \Gamma', x{:}t_x, \Gamma \vdash_w t[\alpha'/\alpha] : k'$ for some $\alpha' \notin \mathrm{dom}(\Gamma', x{:}t_x, \Gamma)$. By the inductive hypothesis, $\alpha'{:}k, \Gamma', x{:}s_x, \Gamma \vdash e[\alpha'/\alpha] : t[\alpha'/\alpha]$ and $\alpha'{:}k, \Gamma', x{:}s_x, \Gamma \vdash_w t[\alpha'/\alpha] : k'$. We conclude by rule T-AbsT that $\Gamma', x{:}s_x, \Gamma \vdash \Lambda\alpha{:}k.\, e : \forall\, \alpha{:}k.\, t$.

Case T-Let: We have $\Gamma', x{:}t_x, \Gamma \vdash \mathtt{let}\, y{=}e_y\, \mathtt{in}\, e : t$. By inversion, we get $\Gamma', x{:}t_x, \Gamma \vdash e_y : t_y$ and $z{:}t_y, \Gamma', x{:}t_x, \Gamma \vdash e[z/y] : t[z/y]$ and $\Gamma', x{:}t_x, \Gamma \vdash_w t : k$ for some $z \notin \mathrm{dom}(\Gamma', x{:}t_x, \Gamma)$. By the inductive hypothesis, $\Gamma', x{:}s_x, \Gamma \vdash e_y : t_y$ and $z{:}t_y, \Gamma', x{:}s_x, \Gamma \vdash e[z/y] : t[z/y]$ and $\Gamma', x{:}s_x, \Gamma \vdash_w t : k$. By rule T-Let we conclude $\Gamma', x{:}s_x, \Gamma \vdash \mathtt{let}\, y{=}e_y\, \mathtt{in}\, e : t$.

Case T-Ann: We have $\Gamma', x{:}t_x, \Gamma \vdash (e : t) : t$. By inversion, $\Gamma', x{:}t_x, \Gamma \vdash e : t$ and $\Gamma', x{:}t_x, \Gamma \vdash_w t : k$. By part (1) and by the inductive hypothesis, $\Gamma', x{:}s_x, \Gamma \vdash e : t$ and $\Gamma', x{:}s_x, \Gamma \vdash_w t : k$. By rule T-Ann we conclude $\Gamma', x{:}s_x, \Gamma \vdash (e : t) : t$.

Case T-Sub: We have $\Gamma', x{:}t_x, \Gamma \vdash e : t$. By inversion, $\Gamma', x{:}t_x, \Gamma \vdash e : s$, $\Gamma', x{:}t_x, \Gamma \vdash s <: t$ and $\Gamma', x{:}t_x, \Gamma \vdash_w t : k$. By the (mutually) inductive hypothesis, we have $\Gamma', x{:}s_x, \Gamma \vdash e : s$, $\Gamma', x{:}s_x, \Gamma \vdash s <: t$ and $\Gamma', x{:}s_x, \Gamma \vdash_w t : k$. By rule T-Sub we conclude $\Gamma', x{:}s_x, \Gamma \vdash e : t$.

□

Lemma 14. *If $\Gamma \vdash t <: t'$ and $\Gamma \vdash t' <: t''$ then $\Gamma \vdash t <: t''$.*

Proof. We proceed by induction on the combined size of the derivation trees of $\Gamma \vdash t' <: t'$

and $\Gamma \vdash t' <: t''$. We first consider the cases where none of t, t', or t'' are existential types. By examination of the subtyping rules, we see that t, t', and t'' must have the same form, and so there are three such cases:

Case refinement types: In this case $t \equiv b\{x_1 : p_1\}$, $t' \equiv b\{x_2 : p_2\}$, and $t'' \equiv b\{x_3 : p_3\}$. The last rule used in each of $\Gamma \vdash t' <: t'$ and $\Gamma \vdash t' <: t''$ must have been S-BASE. Inverting each of these we have

$$y{:}b\{x_1 : p_1\}, \Gamma \vdash_e p_2[y/x_2] \quad \text{and} \quad z{:}b\{x_2 : p_2\}, \Gamma \vdash_e p_3[z/x_3] \tag{44}$$

for some $y, z \notin \text{dom}(\Gamma)$. We can invert ENT-PRED to get

$$\forall \theta.\, \theta \in [\![y{:}b\{x_1 : p_1\}, \Gamma]\!] \Rightarrow \theta(p_2[y/x_2]) \hookrightarrow^* \texttt{true} \tag{45}$$

and

$$\forall \theta.\, \theta \in [\![z{:}b\{x_2 : p_2\}, \Gamma]\!] \Rightarrow \theta(p_3[z/x_3]) \hookrightarrow^* \texttt{true}. \tag{46}$$

Let $\theta \in [\![y{:}b\{x_1{:}p_1\}, \Gamma]\!]$. Let $v = \theta(y)$ Then $v \in [\![\theta(b\{x_1{:}p_1\})]\!]$, and so $\theta(p_1)[v/x_1] \hookrightarrow^* \texttt{true}$.

Case function types:

Case polymorphic types:

□

Lemma 15. *Let $<:^*$ denote the reflexive and transitive closure of the subtyping judgment $<:$. If $\Gamma \vdash t <:^* t'$ then $\Gamma \vdash t <: t'$.*

Lemma 16. *If $\Gamma \vdash \lambda x.\, e : x{:}t_x \to t$ then $y{:}t_x, \Gamma e : t[y/x]$ for some $y \notin \text{dom}(\Gamma)$.*

Theorem 17. *(The Progress Theorem) If $\varnothing \vdash e : t$ then either e is a value or there exists a term e' such that $e \hookrightarrow e$'.*

Proof. We proceed by induction on the derivation tree of the judgment $\varnothing \vdash e : t$.

Case T-PRIM: This case holds trivially because $e \equiv c$ is a value.

Case T-VAR: This case cannot occur because $\Gamma = \varnothing$.

Case T-APP: We have $\varnothing \vdash e : t$ where $e \equiv e_1\ e_2$ and $t \equiv \exists\, x{:}t_x.\, t'$. By inversion, $\varnothing \vdash e_1 : x{:}t_x \to t'$ and $\varnothing \vdash e_2 : t_x$. We split on five cases for the structure of e_1 and e_2.

First, consider $e_1 \equiv c$ and $e_2 \equiv v$; then by rule E-PRIM $e \equiv c\ v \hookrightarrow \delta(c, v)$, which is defined by Lemma 1. Second, consider $e_1 \equiv c$ and e_2 not a value. By the inductive hypothesis (applied to $\varnothing \vdash e_2 : t_x$), there exists a term e_2' such that $e_2 \hookrightarrow e_2'$. Thus $e_1\ e_2 \hookrightarrow e_1\ e_2'$ by rule E-APP1.

Third, consider $e_1 \equiv \lambda x.e_1'$ and $e_2 \equiv v$. Then by the operational semantics, $\lambda x.e_2\ v \hookrightarrow e_2[v/x]$. Fourth, consider $e_1 \equiv \lambda x.e_1'$ and e_2 not a value. By the inductive hypothesis, there exists a term e_2' such that $e_2 \hookrightarrow e_2'$. Thus $e_1\ e_2 \hookrightarrow e_1\ e_2'$ by E-APP1 again.

This exhausts all possible cases in which e_1 could be a value in the empty environment. So, finally, consider e_1 not a value. Then by the inductive hypothesis there exists e_1' such that $e_1 \hookrightarrow e_1'$. By the operational semantics, $e_1\ e_2 \hookrightarrow e_1'\ e_2$.

Case T-ABS and T-ABST: These cases holds trivially because $e \equiv \lambda x.e'$ is a value and so is $e \equiv \Lambda\alpha : k.e'$.

Case T-APPT: We have $\varnothing \vdash e : t$ where $e \equiv e'\ [t']$ and $t \equiv s[t'/\alpha]$. By inversion, $\varnothing \vdash e' : \forall\, \alpha{:}k.\, s$ and $\varnothing \vdash_w t' : k$. There are two possible cases for the structure of e'. *Note: We'll get another case if we introduce any polymorphic primitives.*

First, if e' is a value then it can't be a variable because it is typed in the empty environment. So the only possibility is that $e' \equiv \Lambda\alpha{:}k'.e''$. Then by the operational semantics

$e \equiv \Lambda\alpha{:}k'.e''\ [t'] \hookrightarrow e''[t'/\alpha]$. Second, e' is not a value. Then by the inductive hypothesis there is some term e'' such that $e' \hookrightarrow e''$. Then by rule E-AppT of the operation semantics, $e \equiv e'\ [t'] \hookrightarrow e''\ [t']$.

Case T-Let: We have $\varnothing \vdash e : t$ where $e \equiv (\texttt{let}\, x\!=\!e_1\, \texttt{in}\, e_2)$. By inversion, $\varnothing \vdash e_1 : t_x$ and $y{:}t_x \vdash e_2[y/x] : t[y/x]$ for some y. First, suppose that $e_1 \equiv v$. Then by rule T-LetV, $\texttt{let}\, x\!=\!v\, \texttt{in}\, e_2 \hookrightarrow e_2[v/x]$. Second, suppose that e_1 is not a value. Then by the inductive hypothesis (applied to judgement $\varnothing \vdash e_1 : t_x$), there exists a term e_1' such that $e_1 \hookrightarrow e_1'$. Then by rule E-Let we have $\texttt{let}\, x\!=\!e_1\, \texttt{in}\, e_2 \hookrightarrow \texttt{let}\, x\!=\!e_1'\, \texttt{in}\, e_2$.

Case T-Ann: We have $\varnothing \vdash e : t$ where $e \equiv (e_1 : t)$. By inversion, $\varnothing \vdash e_1 : t$. By the inductive hypothesis either $e_1 \equiv v$ a value or there exists e_1' such that $e_1 \hookrightarrow e_1'$. In the former case $(v : t) \hookrightarrow v$ and in the latter case $(e_1 : t) \hookrightarrow (e_1' : t)$.

Case T-Sub: We have $\varnothing \vdash e : t$. By inversion, $\varnothing \vdash e : s$, $\varnothing \vdash s <: t$, and $\varnothing \vdash_w t : k$ for some type s. By the inductive hypothesis, either e is a value or there exists e' such that $e \hookrightarrow e'$ and we are done. □

Theorem 18. *(The Preservation Theorem) If $\varnothing \vdash e : t$ and $e \hookrightarrow e'$, then $\varnothing \vdash e' : t$.*

Proof. We proceed by induction on the derivation tree of the judgment $\varnothing \vdash e : t$.

Case T-Prim: Holds trivially because if $e \equiv c$ then there does not exist e' such that $c \hookrightarrow e'$.

Case T-Var: Holds trivially because if $e \equiv x$ then there does not exist e' such that $x \hookrightarrow e'$.

Case T-App: We have $\varnothing \vdash e : t$ where $e \equiv e_1\ e_2$ and $t \equiv \exists\, x{:}t_x.\, t'$ for some variable x and type t_x. By inversion, $\varnothing \vdash e_1 : x{:}t_x \to t'$ and $\varnothing \vdash e_2 : t_x$. We split on five cases for the structure of e_1 and e_2.

First, consider $e_1 \equiv c$ and $e_2 \equiv v$; then by the determinism of the semantics $e' = \delta(c, v)$. By Lemma 1 we have $\varnothing \vdash c : ty(c)$ and write $ty(c) = z{:}t_z \to t''$. By Lemma **??** we have $\varnothing \vdash z{:}t_z \to t'' <: x{:}t_x \to t'$. Inverting the last rule used in that derivation, which must be S-Func, we have $y{:}t_x \vdash t''[y/z] <: t'[y/z]$. By the Substitution Lemma we have $\varnothing \vdash t''[y/z][v/y] <: t'[y/x][v/y]$, which is the same as $\varnothing \vdash t''[v/z] <: t'[v/x]$. By Lemma 1, we have $\varnothing \vdash \delta(c, v) : t''[v/z]$. We also have $\varnothing \vdash t'[v/x] : k$ for some kind k by Lemma 11 and the Substitution Lemma. Then we can apply rule T-Sub to obtain $\varnothing \vdash \delta(c, v) : t'[v/x]$. By Lemma 12, $\varnothing \vdash t'[v/x] <: \exists\, x{:}t_x.\, t'$, and so by rule T-Sub again (by Lemma 11, we have $\varnothing \vdash_w \exists\, x{:}t_x.\, t'$), $\varnothing \vdash \delta(c, v) : \exists\, x{:}t_x.\, t'$.

Second, consider $e_1 \equiv c$ and e_2 not a value. By Theorem 17, there exists a term e_2' such that $e_2 \hookrightarrow e_2'$. By rule E-App2, $c\ e_2 \hookrightarrow c\ e_2'$ and by the determinism of the operational semantics, $e' \equiv c\ e_2'$. By the inductive hypothesis, $\varnothing \vdash e_2' : t_x$. We conclude by T-App that $\varnothing \vdash e' : \exists\, x{:}t_x.\, t'$.

Third, consider $e_1 \equiv \lambda x.e_1'$ and $e_2 \equiv v$. Then $e_1\ e_2 \hookrightarrow e_1'[v/x]$ and by determinism of the operational semantics, $e' \equiv e_1'[v/x]$. Consider the proof tree deriving $\varnothing \vdash \lambda x.e_1' : x{:}t_x \to t'$...

There are two rules that could have been used last in $\varnothing \vdash \lambda x.e_1' : x{:}t_x \to t'$. If the last rule used were T-Abs, then by inversion we have If the last rule used were T-Sub, then inversion gives us $\varnothing \vdash \lambda x.e_1' : s$ and $\varnothing \vdash s <: x{:}t_x \to t'$ for some type s.

Third, consider $e_1 \equiv \lambda x.e_1'$ and $e_2 \equiv v$. Then $e_1\ e_2 \hookrightarrow e_1'[v/x]$ and by determinism of the operational semantics, $e' \equiv e_1'[v/x]$. By inversion of T-Abs, we have $x{:}t_x \vdash e_1' : t'$, and by the substitution lemma we have $\varnothing \vdash e_1'[v/x] : t'[v/x]$. By Lemma 11, we have $\varnothing \vdash_w \exists\, x{:}t_x.\, t'$ and by inverting WF-Exis we have $x{:}t_x \vdash_w t'$. By Lemma 12, $\varnothing \vdash t'[v/x] <: \exists\, x{:}t_x.\, t'$, and so by rule T-Sub, $\varnothing \vdash e' : \exists\, x{:}t_x.\, t'$.

Fourth, consider $e_1 \equiv \lambda x.e_1'$ and e_2 not a value. By Theorem 17, there exists a term e_2' such that $e_2 \hookrightarrow e_2'$. By rule E-App2, $(\lambda x.e_1')\ e_2 \hookrightarrow (\lambda x.e_1')\ e_2'$ and by the determinism of the operational semantics, $e' \equiv (\lambda x.e_1')\ e_2'$. By the inductive hypothesis, $\varnothing \vdash e_2' : t_x$. We conclude by T-App that $\varnothing \vdash e' : \exists\, x{:}t_x.\, t'$.

This exhausts all possible cases in which e_1 could be a value in the empty environment. So, finally, consider e_1 not a value. Then by Theorem 17, there exists an e_1' such that $e_1 \hookrightarrow e_1'$. By determinism of the operational semantics, $e' \equiv e_1'\ e_2$. By the inductive hypothesis, $\varnothing \vdash e_1' : \exists\, x{:}t_x.\, t'$. By rule Syn-App, $\varnothing \vdash e' : \exists\, x{:}t_x.\, t'$.

Case T-Abs: Holds trivially because if $e \equiv \lambda x.e_1$ then there does not exist any e' such that $\lambda x.e_1 \hookrightarrow e'$.

Case T-AbsT: Holds trivially because if $e \equiv \Lambda\alpha{:}k.e_1$ then there does not exist any e' such that $\Lambda\alpha{:}k.e_1 \hookrightarrow e'$.

Case T-Let: We have $\varnothing \vdash e : t$ where $e \equiv (\texttt{let}\, x = e_1\, \texttt{in}\, e_2)$ and $t \equiv t_2$. By inversion, $\varnothing \vdash e_1 : t_1$, $\quad x{:}t_1 \vdash e_2 : t_2$, and $\varnothing \vdash_w t_2$ for some type t_1. First suppose that e_1 is not a value. Then by Theorem 17, there exists some term e_1' such that $e_1 \hookrightarrow e_1'$. By Rule E-Let, $\texttt{let}\, x = e_1\, \texttt{in}\, e_2 \hookrightarrow \texttt{let}\, x = e_1'\, \texttt{in}\, e_2$, and by determinism of the operational semantics, $e' \equiv \texttt{let}\, x = e_1'\, \texttt{in}\, e_2$. By the inductive hypothesis, $\varnothing \vdash e_1 : t_1$. Then by T-Let, $\varnothing \vdash e' : t_2$.

Second, suppose that $e_1 \equiv v$, for some value v. Then by rule E-LetV, $\texttt{let}\, x = v\, \texttt{in}\, e_2 \hookrightarrow e_2[v/x]$. By determinism of the operational semantics, $e' \equiv e_2[v/x]$. By the substitution lemma, $\varnothing \vdash e_2[v/x] : t_2[v/x]$. But by $\varnothing vdash_w t_2$, we know that x does not appear free in t_2 so $t_2[v/x] = t_2$ and $\varnothing \vdash e' : t_2$.

Case T-Ann: We have $\varnothing \vdash e : t$ where $e \equiv (e_1 : t)$ and $e \hookrightarrow e'$. By inversion, $\varnothing \vdash e_1 : t$. By Theorem 17 there exists e_1' such that $e_1 \hookrightarrow e_1'$. By rule E-Ann $(e_1 : t) \hookrightarrow (e_1' : t)$ and by the determinism of the operational semantics we must have $e' \equiv (e_1' : t)$. Then by the inductive hypothesis, $\varnothing \vdash e_1' : t$. By rule Syn-Ann, $\varnothing \vdash (e_1' : t) : t$.

Case T-Sub: We have $\varnothing \vdash e : t$. By inversion $\varnothing \vdash e : s$ and $\varnothing \vdash s <: t$ for some type s, and also $\varnothing \vdash_w t$. By the inductive hypothesis $\varnothing \vdash e' : s$. By rule Chk-Syn, $\varnothing \vdash e' : t$.

$\square$