Liquid Meta

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

January 9, 2020

1 Our language λ_1

We work with a simply 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 language λ in Jhala's forthcoming manuscript [Jhala] and incorporates some aspects from the λ^H of Vazou et al [VSJ+14]. The existential types were used in a similar setting by Knowles and Flanagan [KF09].

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

Values	$v :=$	$\texttt{true}, \texttt{false}$	*boolean constants*
	$\mid$	$0, 1, 2, \ldots$	*integer constants*
	$\mid$	x	*variables*
	$\mid$	$\lambda x.e$	*abstractions*
	$\mid$	$e_1 \wedge e_2,\ e_1 \vee e_2,\ \neg e_1$	*built−in primitives*
	$\mid$	$e_1 \leq e_2,\ e_1 = e_2$	*built−in primitives*

Expressions	$e :=$	v	*values*
	$\mid$	$e_1\ e_2$	*applications*
	$\mid$	$\texttt{let}\ x = e_1\ \texttt{in}\ e_2$	*let expressions*
	$\mid$	$e_1 : t$	*annotations*

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

Base types	$b :=$	Bool	*booleans*
	$\mid$	Int	*integers*

Types	$t :=$	b	*base*
	$\mid$	$b\{r\}$	*refinement*
	$\mid$	$x{:}t_x \to t$	*dependent function*
	$\mid$	$\exists\, x{:}t_x.\, t$	*existential*

$$\begin{array}{rrll} \textsf{Refinements} & r ::= & \{x : p\} & \\ \\ \textsf{Environments} & \Gamma ::= & \varnothing & \textit{empty} \\ & \mid & \Gamma, x{:}t & \textit{bind variable} \end{array}$$

Next, we give the syntax of the Boolean predicates and constraints involved in refinements and subtyping judgments:

$$\begin{array}{rrll} \textsf{Predicates} & p ::= & \{e \mid \exists\, \Gamma.\, \Gamma \vdash e : \textsf{Bool}\} & \textit{expressions of type } \textsf{Bool} \\ \\ \textsf{Constraints} & c ::= & p & \textit{predicates} \\ & \mid & c_1 \wedge c_2 & \textit{conjunction} \\ & \mid & \forall\, x{:}b.\, p \Rightarrow c & \textit{implication} \end{array}$$

Our definition of predicates above departs from the languages of [Jhala] by allowing predicates to be arbitrary expressions from the main language (which are Boolean typed under the appropriate binding environment). In [Jhala] 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 expresssions in $\theta = (x_1 \mapsto e_1, \ldots, x_n \mapsto e_n)$. Substituting arbitrary expressions into t requires substituting arbitrary expressions into predicates, and it isn't clear how to do this for non-values like $((\lambda x.x)\,3)$ without taking predicates to be all Boolean-typed program expressions.

Returning to our λ_1, 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.

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

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.

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

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

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

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

Next, we define the typing rules of our λ_1. The type judgments in the language λ_1 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 $\Gamma \vdash_w t$ says that type t is well-formed in environment Γ:

$$\frac{}{\Gamma \vdash_w b}\ \text{WF-Base} \qquad \frac{\Gamma, x{:}b \vdash e : \mathsf{Bool}}{\Gamma \vdash_w b\{x : e\}}\ \text{WF-Refn}$$

$$\frac{\Gamma \vdash_w t_x \qquad \Gamma, x{:}t_x \vdash_w t}{\Gamma \vdash_w x{:}t_x \to t}\ \text{WF-Func} \qquad \frac{\Gamma \vdash_w t_x \qquad \Gamma, x{:}t_x \vdash_w t}{\Gamma \vdash_w \exists\, x{:}t_x.\, t}\ \text{WF-Exis}$$

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.

$$\frac{ty(c) = t}{\Gamma \vdash c : t}\ \text{T-Prim} \qquad \frac{x{:}t \in \Gamma}{\Gamma \vdash x : t}\ \text{T-Var} \qquad \frac{\Gamma, x{:}t_x \vdash e : t \qquad \Gamma \vdash_w t_x}{\Gamma \vdash \lambda x.\, e\ :\ x{:}t_x \to t}\ \text{T-Abs}$$

$$\frac{\Gamma \vdash e\ :\ x{:}t_x \to t \qquad \Gamma \vdash e' : t_x}{\Gamma \vdash e\ e' : \exists\, x{:}t_x.\, t}\ \text{T-App} \qquad \frac{\Gamma \vdash e_x : t_x \quad \Gamma, x{:}t_x \vdash e_2 : t \quad \Gamma \vdash_w t}{\Gamma \vdash \texttt{let}\, x = e_x\, \texttt{in}\, e : t}\ \text{T-Let}$$

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

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

$$\frac{\Gamma, x_1{:}b\{x_1 : p_1\} \vdash_e p_2[x_1/x_2]}{\Gamma \vdash b\{x_1 : p_1\} <: b\{x_2 : p_2\}}\ \text{S-Base} \qquad \frac{\Gamma \vdash s_2 <: s_1 \quad \Gamma, x_2{:}s_2 \vdash t_1[x_2/x_1] <: t_2}{\Gamma \vdash x_1{:}s_1 \to t_1 <: x_2{:}s_2 \to t_2}\ \text{S-Func}$$

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

The first rule above, S-BASE, uses the entailment judgement $\Gamma \vdash_e c$ which states that constraint c is valid (in the sense of a first-order sentence) when universally quantified over all variables bound in environment Γ. We give the inference rules for the entailment judgement:

$$\frac{p \hookrightarrow^* \texttt{true}}{\varnothing \vdash_e p}\ \textsc{Ent-EmpP} \qquad \frac{\forall\, \theta.\, \theta \in [\![b\{x : p\}]\!] \Rightarrow \varnothing \vdash_e \theta(c)}{\varnothing \vdash_e \forall\, x{:}b.\, p \Rightarrow c}\ \textsc{Ent-EmpI}$$

$$\frac{\Gamma \vdash_e c_1 \quad \Gamma \vdash_e c_2}{\Gamma \vdash_e c_1 \wedge c_2}\ \textsc{Ent-EmpC} \qquad \frac{\Gamma \vdash_e \forall\, x{:}b.\, p \Rightarrow c}{\Gamma, x{:}b\{x : p\} \vdash_e c}\ \textsc{Ent-Ext}$$

2 Preliminaries

For clarity, we distinguish between different typing judgments with a subscript. The type judgments in the underlying typed lambda calculus 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:

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

To simplify the meta-theory, we follow [VSJ+14] in extending our language with a non-value expression $\bot$, which does not evaluate but which has any refinement type:

$$\frac{\Gamma \vdash_w b\{x : p\}}{\Gamma \vdash \bot : b\{x : p\}}\ \textsc{T-Bot}$$

We define a predicate $\mathsf{BotLess}(e)$ which holds if and only if the expression e does not contain $\bot$.

We start our development of the meta-theory by giving a definition of *type denotations*. Roughly speaking, the denotation of a type t is the class of expressions e with the correct underlying base type such that if e evaluates to a value, then this value satisfies the refinement predicates that appear within the structure of t. We formalize this notion with a recursive definition:

$$\begin{aligned}
[\![b]\!] &:= \{e \mid \varnothing \vdash_B e : b\} \\
[\![b\{x : p\}]\!] &:= \{e \mid (\varnothing \vdash_B e : b) \wedge (\text{if } e \hookrightarrow^* v \text{ then } p[v/x] \hookrightarrow^* \texttt{true})\} \\
[\![x{:}t_x \to t]\!] &:= \{e \mid (\varnothing \vdash_B e : \lfloor t_x \rfloor \to \lfloor t \rfloor) \wedge (\forall\, e_x \in [\![t_x]\!].\, e\ e_x \in [\![t[e_x/x]]\!])\} \\
[\![\exists\, x{:}t_x.\, t]\!] &:= \{e \mid (\varnothing \vdash_B e : \lfloor t \rfloor) \wedge (\exists\, e_x \in [\![t_x]\!].\, e \in [\![t[e_x/x]]\!])\}
\end{aligned}$$

The addition of $\bot$ guarantees that all denotations are non-empty.We can formalize the above intuition that type denotations capture terms that diverge with the following lemma:

Lemma 1. *Let t be a type such that $\varnothing \vdash_w t$ and e be an expression such that $\varnothing \vdash_B e : \lfloor t \rfloor$ but there does not exist any value v such that $e \hookrightarrow^* v$. Then $e \in [\![t]\!]$; moreover, such an e always exists so in particular $[\![t]\!] \neq \varnothing$.*

Proof. We prove this by induction on the structure of type t. The cases that $t \equiv b$, a base type, or $t \equiv b\{x : p\}$ follow trivially from the definition of $[\![t]\!]$ (we can take $e = \bot$). In the

next case, suppose $t \equiv x{:}t_x \to t'$. Let $e_x \in [\![t_x]\!]$. We note that $e\ e_x$ cannot reduce to a value: the only rule that we can ever apply is E-App1 because e is not a value and for any e' such that $e \hookrightarrow^* e'$, e' is also not a value. Thus we can never apply E-Prim, E-App2, or E-AppAbs and no expression of the form $e'\ e_x$ is ever a value. Then by induction we have that $e\ e_x \in [\![t[e_x/x]]\!]$ and so $e \in [\![x{:}t_x \to t']\!]$. To show that $[\![x{:}t_x \to t']\!] \neq \varnothing$, we use the inductive hypothesis to get an $e_x \in [\![t_x]\!]$ and use it again to get a term $e' \in [\![t'[e_x/x]]\!]$ that doesn't evaluate to a value. Set $e = \lambda x.\, e'$. Then $\varnothing \vdash_B e : \lfloor t_x \rfloor \to \lfloor t' \rfloor$ and for any $\hat{e} \in [\![t_x]\!]$, either $\hat{e}$ doesn't evaluate to a value (in which case $e\ \hat{e}$ doesn't evaluate to a value either and so $e\ \hat{e} \in [\![t'[e_x/x]]\!]$ by induction) or $\hat{e} \hookrightarrow^* \hat{v}$ for some value $\hat{v}$. Then $e\ \hat{e} \hookrightarrow^* e\ \hat{v} \hookrightarrow e'[v/x] = e' \in [\![t'[e_x/x]]\!]$ and so by Lemma 3 $e\ \hat{e} \in [\![t'[e_x/x]]\!]$. Therefore $e \in [\![t]\!]$.

In the final case, suppose $t \equiv \exists\, x{:}t_x.\, t'$. By the inductive hypothesis, there exists some $e_x \in [\![t_x]\!]$. Suppose that $\varnothing \vdash_B e : t'$ and that e doesn't evaluate to a value. Then by induction we have that $e \in [\![t'[e_x/x]]\!]$ so $e \in [\![\exists\, x{:}t_x.\, t']\!]$. Finally, to show that such an e must exist, we have by the inductive hypothesis that there must exists some $e \in [\![t'[e_x/x]]\!]$ that does not evaluate to a value, and so $e \in [\![\exists\, x{:}t_x.\, t']\!]$ as well. □

We also have the concept of the denotation of an environment Γ; we intuitively define this to be the set of all sequences of expression bindings for the variables in Γ such that the expressions respect the denotations of the types of the corresponding variables. A closing substitution is just a sequence of expression bindings to variables:

$$\theta = (x_1 \mapsto e_1,\ \ldots,\ x_n \mapsto e_n) \quad \text{with all } x_i \text{ distinct}$$

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

$$\theta(t) := t[e_1/x_1] \cdots [e_n/x_n]$$

Then we can formally define the denotation of an environment:

$$[\![\Gamma]\!] := \{\theta = (x_1 \mapsto e_1, \ldots, x_n \mapsto e_n) \mid \forall\, (x : t) \in \Gamma.\, \theta(x) \in [\![\theta(t)]\!]\}.$$

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

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

and similarly for $ty(\vee)$, $ty(=)$, and $ty(m=)$. 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 2. *(Primitive Typing) For every primitive c, $\varnothing \vdash c : ty(c)$ and*

1. *If $ty(c) = b\{x : p\}$, then $c \in [\![ty(c)]\!]$ and for all c' such that $c' \neq c$, $c' \notin [\![ty(c)]\!]$.*
2. *If $ty(c) = x{:}t_x \to t$, then for each $v \in [\![t_x]\!]$, $\delta(c, v)$ is defined and $\delta(c, v) \in [\![t[v/x]]\!]$. Thus $c \in [\![ty(c)]\!]$.*

3 Meta-theory

In this section, we seek to prove the operational soundness of our language λ_1. Our proof of the soundness theorem begins with several helping lemmas.

Lemma 3. *(Preservation of Denotations) If $e \hookrightarrow^* e'$ then $e \in [\![t]\!]$ iff $e' \in [\![t]\!]$.*

Proof. We proceed by a case split on the definition of the denotation of a type; in other words, we use induction on the size of type t (the number of arrows or existential quantifiers appearing in t).

First, suppose that $t \equiv b$, a base type. We appeal to the soundness of the underlying bare type system. In particular, if $e \in [\![b]\!]$ then $\varnothing \vdash_B e : b$ and so $\varnothing \vdash_B e' : b$ and $e' \in [\![b]\!]$. Conversely if $e' \in [\![b]\!]$ then by determinism of the operational semantics and the typing relation, $e \in [\![b]\!]$. So we see that $e \in [\![b]\!]$. We use this argument implicitly in each of the other cases because we can replace b with any bare type.

Second, suppose $t \equiv b\{x : p\}$. If $e \in [\![t]\!]$ then it holds that if $e \hookrightarrow^* v$ for some value v then $p[v/x] \hookrightarrow^* \texttt{true}$. Suppose $e' \hookrightarrow^* v$ for some value v; then by transitive closure $e \hookrightarrow^* v$, so $p[v/x] \hookrightarrow^* \texttt{true}$ and we conclude $e' \in [\![t]\!]$.

If $e' \in [\![t]\!]$ then it holds that if $e' \hookrightarrow^* v$ for some value v then $p[v/x] \hookrightarrow^* \texttt{true}$. Suppose $e \hookrightarrow^* v$. We appeal to the determinism of the operational semantics: by hypothesis, $e \hookrightarrow^* e'$, so it must be the case that $e' \hookrightarrow^* v$. Then $p[v/x] \hookrightarrow^* \texttt{true}$ and therefore $e \in [\![t]\!]$.

Next, suppose $t \equiv x{:}t_x \to t'$. If $e \in [\![t]\!]$ then it holds that for every $v \in [\![t_x]\!]$, we have $e\,v \in [\![t'[v/x]]\!]$. Because $e \hookrightarrow^* e'$, we have $(e\,v) \hookrightarrow^* (e'\,v)$ by the operational semantics and so by induction (on the structure of t) we have $e'\,v \in [\![t'[v/x]]\!]$ and thus $e' \in [\![t]\!]$. If $e' \in [\![t]\!]$ then it holds that $\forall\, v \in [\![t_x]\!].\, e'\,v \in [\![t'[v/x]]\!]$. We appeal to the determinism of the operations semantics: by hypothesis, $e \hookrightarrow^* e'$, so it must be the case that $e\,v \hookrightarrow^* e'\,v$. Then by induction we have $e\,v \in [\![t'[v/x]]\!]$, and thus $e \in [\![t]\!]$.

Finally, suppose $t \equiv \exists\, x{:}t_x.\, t'$. We have $e \in [\![t]\!]$ if and only if there exists some $v \in [\![t_x]\!]$ such that $e \in [\![t'[v/x]]\!]$. Then by induction $e \in [\![t'[v/x]]\!]$ if and only if $e' \in [\![t'[v/x]]\!]$. By definition, $e' \in [\![t]\!]$ if and only if there exists $v \in [\![t_x]\!]$ such that $e' \in [\![t'[v/x]]\!]$, which completes the proof. □

Lemma 4. *(Declarative Entailments) Our entailment judgement is sound with respect to the denotations of the environment: If $\Gamma \vdash_e c$, then $\forall\, \theta.\, \theta \in [\![\Gamma]\!] \Rightarrow \varnothing \vdash_e \theta(c)$.*

Proof. We proceed by induction on (the length of) Γ. In the base case $\Gamma = \varnothing$, so $\theta(c) = c$ and the result is vacuous. In the inductive case, suppose we have $\Gamma', x{:}t \vdash_e c$. By inversion of Ent-Ext, we must have that $t \equiv b\{x : p\}$ and $\Gamma' \vdash_e \forall\, x{:}b.\, p \Rightarrow c$. By the inductive hypothesis, for any $\theta' \in [\![\Gamma']\!]$, $\varnothing \vdash_e \theta'(\forall\, x{:}b.\, p \Rightarrow c)$, or equivalently, $\varnothing \vdash_e \forall\, x{:}b.\, \theta'(p) \Rightarrow \theta'(c)$. By inversion of Ent-EmpI, we have for all $\theta'' = (x \mapsto e) \in [\![b\{x : p\}]\!]$, $\varnothing \vdash_e \theta'(c)[e/x]$. Therefore, for any $\theta \in [\![\Gamma', x{:}t]\!]$, if we write $\theta = (\theta', x \mapsto e)$ we have $\varnothing \vdash_e \theta(c)$.

□

Lemma 5. *(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* $\forall \theta.\theta \in [\![\Gamma]\!] \Rightarrow \theta(e) \in [\![\theta(t)]\!]$.
3. TODO: Do I need to prove that well-formedness is sound too?

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 $\Gamma \vdash e_x : t_x$ 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,

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

By inversion of Ent-Ext we have

$$\Gamma \vdash_e \forall\, x_1{:}b.\, p_1 \Rightarrow p_2[x_1/x_2]. \tag{1}$$

By Lemma 4 we have

$$\forall\, \theta.\, \theta \in [\![\Gamma]\!] \Rightarrow \varnothing \vdash_e \theta(\forall\, x_1{:}b.\, p_1 \Rightarrow p_2[x_1/x_2]),$$

or equivalently

$$\forall\, \theta.\, \theta \in [\![\Gamma]\!] \Rightarrow \varnothing \vdash_e \forall\, x_1{:}b.\, \theta(p_1) \Rightarrow \theta(p_2)[x_1/x_2] \tag{2}$$

By inversion of rule Ent-EmpI, we have

$$\forall\, \theta.\, \theta \in [\![\Gamma]\!] \Rightarrow \forall\, (x_1 \mapsto e) \in [\![b\{x_1 : \theta(p_1)\}]\!] \Rightarrow \varnothing \vdash_e \theta(p_2)[x_1/x_2][e/x_1]. \tag{3}$$

Then by inversion of rule Ent-EmpP, we obtain

$$\forall\, \theta.\, \theta \in [\![\Gamma]\!] \Rightarrow \forall\, (x_1 \mapsto e) \in [\![b\{x_1 : \theta(p_1)\}]\!] \Rightarrow \theta(p_2)[e/x_2] \hookrightarrow^* \texttt{true}. \tag{4}$$

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

$$\forall \theta.\, \theta \in [\![\Gamma]\!] \Rightarrow \{e \mid \varnothing \vdash_B e : b \ \wedge\ (\text{if } e \hookrightarrow^* v \text{ then } \theta(p_1[v/x_1]) \hookrightarrow^* \texttt{true})\} \tag{5}$$

$$\subseteq \{e \mid \varnothing \vdash_B e : b \ \wedge\ (\text{if } e \hookrightarrow^* v \text{ then } \theta(p_2[v/x_2]) \hookrightarrow^* \texttt{true})\} \tag{6}$$

Let $\theta \in [\![\Gamma]\!]$ be a closing substitution and let e a term in set (5), and suppose $e \hookrightarrow^* v$. Then $\theta(p_1[v/x_1]) \hookrightarrow^* \texttt{true}$, and so $\theta' = (\theta, x_1 \mapsto v) \in [\![\Gamma, b\{x_1 : \theta(p_1)\}]\!]$. From (4) we have $\theta(p_2[v/x_2]) = \theta(p_2)[v/x_2] \hookrightarrow^* \texttt{true}$, and so e lies in set (6), thus proving the desired containment.

Case Sub-Fun: 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,

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

By the inductive hypothesis,

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

and

$$\forall \theta.\, \theta \in [\![\Gamma, x_2{:}s_2]\!] \Rightarrow [\![\theta(t_1'[x_2/x_1])]\!] \subseteq [\![\theta(t_2')]\!] \tag{7}$$

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,

$$\forall \theta.\, \theta \in [\![\Gamma]\!] \Rightarrow \{e \mid \varnothing \vdash_B e : \lfloor s_1 \rfloor \to \lfloor t_1' \rfloor \ \wedge\ (\forall\, e' \in [\![\theta(s_1)]\!].\, e\, e' \in [\![\theta(t_1'[e'/x_1])]\!])\} \tag{8}$$

$$\subseteq \{e \mid \varnothing \vdash_B e : \lfloor s_2 \rfloor \to \lfloor t_2' \rfloor \ \wedge\ (\forall\, e' \in [\![\theta(s_2)]\!].\, e\, e' \in [\![\theta(t_2'[e'/x_2])]\!])\} \tag{9}$$

Fix $\theta \in [\![\Gamma]\!]$ and let e be a term in set (8) and let $e' \in [\![\theta(s_2)]\!]$. Then by induction, $e' \in [\![\theta(s_1)]\!]$. So $(e\, e') \in [\![(\theta(t_1'[e'/x_1])]\!]$. Let $\theta' = (\theta, x_2 \mapsto e')$. From (7) we also have that $[\![\theta'(t_1'[x_2/x_1])]\!] \subseteq [\![\theta'(t_2')]\!]$. But $\theta'(t_1'[x_2/x_1]) = \theta(t_1'[x_2/x_1])[e'/x_2] = \theta(t_1'[e'/x_1])$ and $\theta'(t_2') = \theta(t_2')[e'/x_2]$. Therefore $(e\ e') \in [\![\theta(t_1'[e'/x_2])]\!] \subseteq [\![\theta(t_2'[e'/x_2])]\!]$ and so e is in set (9) as desired.

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 term e_x such that

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

By the inductive hypothesis, we have

$$\forall\, \theta.\, \theta \in [\![\Gamma]\!] \Rightarrow [\![\theta(t_1)]\!] \subseteq [\![\theta(t_2'[e_x/x])]\!] \tag{10}$$

and by mutual induction we also have

$$\forall\, \theta.\, \theta \in [\![\Gamma]\!] \Rightarrow \theta(e_x) \in [\![\theta(t_x)]\!].$$

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

$$[\![\theta(\exists\, x{:}t_x.\, t_2')]\!] = \{e \mid \varnothing \vdash_B e : \lfloor \theta(t_2') \rfloor \ \wedge\ (\exists\, e' \in [\![\theta(t_x)]\!].\, e \in [\![\theta(t_2')[e'/x]]\!])\} \tag{11}$$

because $\theta(\exists\, x{:}t_x.\, t_2') = \exists\, x{:}\theta(t_x).\, \theta(t_2')$. Let $e \in [\![\theta(t_1)]\!]$ and set $e' = \theta(e_x) \in [\![\theta(t_x)]\!]$. Then by (10), $e \in [\![\theta(t_2'[e_x/x])]\!]$ and by definition of the denotation of a type, in every case $\varnothing \vdash_B e : \lfloor \theta(t_2'[e_x/x]) \rfloor = \lfloor \theta(t_2') \rfloor$. We conclude by noting $\theta(t_2'[e_x/x]) = \theta(t_2')[e'/x]$ and so e is in the right hand side of (11).

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

$$\Gamma, x{:}t_x \vdash t_1' <: t_2 \quad \text{and} \quad x \notin \mathit{free}(t_2).$$

By the inductive hypothesis, we have

$$\forall\, \theta.\, \theta \in [\![\Gamma, x{:}t_x]\!] \Rightarrow [\![\theta(t_1')]\!] \subseteq [\![\theta(t_2)]\!]. \tag{12}$$

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 $e \in [\![\theta(\exists\, x{:}t_x.\, t_1')]\!]$. By definition, $\theta(\exists\, x{:}t_x.\, t_1') = \exists\, x{:}\theta(t_x).\, \theta(t_1')$ so

$$[\![\theta(\exists\, x{:}t_x.\, t_1')]\!] = \{e \mid \varnothing \vdash_B e : \lfloor \theta(t_1') \rfloor \ \wedge\ (\exists\, e' \in [\![\theta(t_x)]\!].\, e \in [\![\theta(t_1')[e'/x]]\!])\}. \tag{13}$$

Take e' as in (13) and let $\theta' = (\theta, x \mapsto e')$. We note that $\theta' \in [\![\Gamma, x{:}t_x]\!]$ because $\theta'(x) = e' \in [\![\theta(t_x)]\!] = [\![\theta'(t_x)]\!]$ where the last equality follows from the fact that x cannot appear free in

t_x. Then $e \in [\![\theta'(t_1')]\!]$, so from (12) we can conclude $e \in [\![\theta'(t_2)]\!] = [\![\theta(t_2)]\!]$ because x does not appear free in t_2 so $\theta'(t_2) = \theta(t_2)$.

(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 2 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 2, $\delta(c, v) \in [\![t'[v/x]]\!]$ for any $v \in [\![t_x]\!]$. There are no free variables in c or t so again $\theta(c) \in [\![\theta(ty(c))]\!]$.

Case T-Var: We have $\Gamma \vdash e : t$ where $e \equiv x$. By inversion, $(x{:}t) \in \Gamma$. Then for any $\theta \in [\![\Gamma]\!]$, we have by definition $\theta(x) \in [\![\theta(t)]\!]$ as desired.

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, $\Gamma, x{:}t_x \vdash e' : t'$ and by the inductive hypothesis,

$$\forall \theta'.\, \theta' \in [\![\Gamma, x{:}t_x]\!] \Rightarrow \theta'(e') \in [\![\theta'(t')]\!]. \tag{14}$$

Let $\theta \in [\![\Gamma]\!]$ and let $e_x \in [\![\theta(t_x)]\!]$. Then let

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

Then from (14),

$$\theta(e')[e_x/x] = \theta'(e') \in [\![\theta'(t')]\!] = [\![\theta(t')[e_x/x]]\!]. \tag{15}$$

We need to show that for every $\theta \in [\![\Gamma]\!]$, it holds that

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

We have $\varnothing \vdash_B \theta(e) : \lfloor t_x \rfloor \to \lfloor t' \rfloor$ because substitutions do not affect bare types, only the refinement predicates. We have $\theta(e)\ e_x = (\lambda x.\theta(e'))\ e_x$. There are two cases for e_x: first, if e_x does not evaluate to any value then the only reduction rules we can ever apply are E-App1 and E-App2 and so $\theta(e)\ e_x$ can never evaluate to a value. By Lemma 1 we have that $\theta(e)\ e_x \in [\![t'[e_x/x]]\!]$. In the other case suppose that there exists some value such that $e_x \hookrightarrow^* v_x$. Then $\theta(e)\ e_x = (\lambda x.\theta(e'))\ e_x \hookrightarrow^* (\lambda x.\theta(e'))\ v_x \hookrightarrow \theta(e')[v_x/x] \in [\![\theta(t')[v_x/x]]\!]$. By a forthcoming lemma (TODO: this), $[\![\theta(t')[v_x/x]]\!] \subseteq [\![\theta(t')[e_x/x]]\!]$. Then by Lemma 1 we conclude that $\theta(e)\ e_x \in [\![\theta(t')[e_x/x]]\!]$ and so $\theta(e) \in [\![\theta(t)]\!]$.

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

$$\forall \theta.\, \theta \in [\![\Gamma]\!] \Rightarrow \theta(e') \in [\![\theta(x{:}t_x \to t')]\!] \tag{16}$$

and

$$\forall \theta.\, \theta \in [\![\Gamma]\!] \Rightarrow \theta(e_x) \in [\![\theta(t_x)]\!]. \tag{17}$$

From (16), we have that for all $\theta \in [\![\Gamma]\!]$, $\theta(e')\ \theta(e_x) \in [\![\theta(t')[\theta(e_x)/x]]\!]$. Thus

$$\theta(e) = \theta(e')\ \theta(e_x) \in [\![\exists\, x{:}\theta(t_x).\, \theta(t')]\!] = [\![\theta(\exists\, x{:}t_x.\, t')]\!]. \tag{18}$$

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

$$\forall \theta.\, \theta \in [\![\Gamma]\!] \Rightarrow \theta(e_x) \in [\![\theta(t_x)]\!]$$

and

$$\forall \theta'.\, \theta' \in [\![\Gamma, x{:}t_x]\!] \Rightarrow \theta'(e') \in [\![\theta'(t)]\!]. \tag{19}$$

Let $\theta \in [\![\Gamma]\!]$. There are two cases for the semantics of e_x. In the case that there exists some value v_x such that $\theta(e_x) \hookrightarrow^* v_x$, let $\theta' = (\theta, x \mapsto v_x) \in [\![\Gamma, x{:}t_x]\!]$ because we chose $\theta'(x) = v_x \in [\![\theta(t_x)]\!]$. From the operational semantics $\theta(\texttt{let}\, x{=}e_x\, \texttt{in}\, e') = \texttt{let}\, x{=}\theta(e_x)\, \texttt{in}\, \theta(e') \hookrightarrow^* \texttt{let}\, x{=}v_x\, \texttt{in}\, \theta(e') \hookrightarrow \theta(e')[v_x/x]$. Then from (19),

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

where the last equality follows from the fact that the judgement $\Gamma \vdash_w t$ implies $\varnothing \vdash_w \theta(t)$ by part 3 of this lemma, which in turn implies that x cannot be free in $\theta(t)$. The above implies $\theta(e) \hookrightarrow^* \theta(e')[v_x/x]$, so by Lemma 3, $\theta(e) \in [\![\theta(t)]\!]$.

In the second case, $\theta(e_x)$ does not reduce to any value. In that case, the only rule we can ever apply to $\theta(e) = \texttt{let}\, x{=}\theta(e_x)\, \texttt{in}\, \theta(e')$ is E-Let so $\theta(e)$ never reduces to a value, and by Lemma 1 $\theta(e) \in [\![\theta(t)]\!]$.

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, $\theta(e') \in [\![\theta(t)]\!]$. By the operational semantics of type annotations, $\theta(e) = (\theta(e') : \theta(t)) \hookrightarrow \theta(e') \in [\![\theta(t)]\!]$, so we conclude that $\theta(e) \in [\![\theta(t)]\!]$ by Lemma 3.

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, $\forall \theta.\, \theta \in [\![\Gamma]\!] \Rightarrow \theta(e) \in [\![\theta(s)]\!]$ and 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 $\forall \theta.\, \theta \in [\![\Gamma]\!] \Rightarrow \theta(e) \in [\![\theta(t)]\!]$.

Case T-Bot: We have $\Gamma \vdash \bot : t$ where $t \equiv b\{x : p\}$. Then because $\bot$ cannot evaluate, we have that $\bot \in [\![b\{x : \theta(p)\}]\!] = [\![\theta(t)]\!]$. □

Lemma 6. *(The Substitution Lemma) If* $\Gamma \vdash e_x : t_x$ *then*
1. If $\Gamma, x{:}t_x, \Gamma' \vdash t_1 <: t_2$ *then*

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

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

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

Proof. (1) Suppose $\Gamma \vdash e_x : t_x$ and $\Gamma, x{:}t_x, \Gamma' \vdash t_1 <: t_2$. We proceed by mutual induction on the derivation tree of the subtyping relation.

Case Sub-Base: 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,

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

By inversion of Ent-Ext we have

$$\Gamma, x{:}t_x, \Gamma' \vdash \forall x_1{:}b.\, p_1 \Rightarrow p_2[x_1/x_2]. \tag{20}$$

By Lemma 4, we have

$$\forall\, \theta^*.\, \theta^* \in [\![\Gamma, x{:}t_x, \Gamma']\!] \Rightarrow \varnothing \;\vdash_e \theta^*(\forall x_1{:}b.\, p_1 \Rightarrow p_2[x_1/x_2]) \tag{21}$$

$$= \forall\, x_1{:}b.\, \theta^*(p_1) \Rightarrow \theta^*(p_2)[x_1/x_2]. \tag{22}$$

Let e_x be an expression such that $\Gamma \vdash e_x : t_x$. The above is equivalent to:

$$\forall\, \theta, \theta'.\, (\theta, x \mapsto \theta(e_x), \theta') \in [\![\Gamma, x{:}t_x, \Gamma']\!] \Rightarrow \big(\varnothing \vdash_e \forall\, x_1{:}b.\, \theta'(\theta(p_1)[\theta(e_x)/x]) \Rightarrow \theta'(\theta(p_2)[\theta(e_x)/x]))[x_1/x_2]\big)$$

or equivalently,

$$\forall\, \theta, \theta'.\, (\theta, \theta') \in [\![\Gamma, \Gamma'[e_x/x]]\!] \Rightarrow \big(\varnothing \vdash_e \forall\, x_1{:}b.\, (\theta, \theta')(p_1[e_x/x]) \Rightarrow (\theta, \theta')(p_2[e_x/x][x_1/x_2])\big) \tag{23}$$

TODO: can't finish this one yet. □

General Todo list:

- does having call-by-value (non-lazy) semantics affect wha the definitions of denotations of types ($[\![\ ]\!]$) ought to be? Should any expressions being substituted into types become values?
- In Lemma 5, case T-Abs requires some additional lemma like: if $e \hookrightarrow^* v$ then $[\![t[v/x]]\!] \subseteq [\![t[e/x]]\!]$
- Lemma 5 needs a part (3): If $\Gamma \vdash_w t$ then $\forall \theta.\, \theta \in [\![\Gamma]\!] \Rightarrow \varnothing \vdash_w \theta(t)$.
- Lemma 6 needs a part (3): If $\Gamma, x{:}t_x, \Gamma' \vdash_w t$ and $\Gamma \vdash e_x : t_x$ then $\Gamma, \Gamma'[e_x/x] \vdash_w t[e_x/x]$.
- The entailment rules don't quite seem to fit with the denotation definitions. Can't prove case S-Base in the Substitution Lemma (Lemma 6).