

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	$v :=$	<code>true, false</code>	<i>boolean constants</i>
		<code>0, 1, 2, ...</code>	<i>integer constants</i>
		<code>x</code>	<i>variables</i>
		<code>$\lambda x. e$</code>	<i>abstractions</i>
		<code>$e_1 \wedge e_2, e_1 \vee e_2, \neg e_1$</code>	<i>built-in primitives</i>
		<code>$e_1 \leq e_2, e_1 = e_2$</code>	<i>built-in primitives</i>
Expressions	$e :=$	<code>v</code>	<i>values</i>
		<code>$e_1 e_2$</code>	<i>applications</i>
		<code>let $x = e_1$ in e_2</code>	<i>let expressions</i>
		<code>$e_1 : t$</code>	<i>annotations</i>

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

Base types	$b :=$	<code>Bool</code>	<i>booleans</i>
		<code>Int</code>	<i>integers</i>
Types	$t :=$	<code>b</code>	<i>base</i>
		<code>$b\{r\}$</code>	<i>refinement</i>
		<code>$x:t_x \rightarrow t$</code>	<i>dependent function</i>
		<code>$\exists x:t_x. t$</code>	<i>existential</i>

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

Environments $\Gamma := \emptyset$ *empty*
 $\quad \mid \quad \Gamma, x:t$ *bind variable*

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

Predicates $p := \{e \mid \exists \Gamma. \Gamma \vdash e : \text{Bool}\}$ *expressions of type Bool*

Constraints $c := p$ *predicates*
 $\quad \mid \quad c_1 \wedge c_2$ *conjunction*
 $\quad \mid \quad \forall x:b. p \Rightarrow c$ *implication*

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 expressions in $\theta = (x_1 \mapsto e_1, \dots, x_n \mapsto e_n)$. 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.

$\delta(\wedge, \text{true}) := \lambda x. x$	$\delta(\leq, m) := m \leq$
$\delta(\wedge, \text{false}) := \lambda x. \text{false}$	$\delta(m \leq, n) := \text{bval}(m \leq n)$
$\delta(\vee, \text{true}) := \lambda x. \text{true}$	$\delta(=, m) := m =$
$\delta(\vee, \text{false}) := \lambda x. x$	$\delta(m =, n) := \text{bval}(m = n)$
$\delta(\neg, \text{true}) := \text{false}$	
$\delta(\neg, \text{false}) := \text{true}$	

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.

$$\begin{array}{c}
\frac{}{c \ v \hookrightarrow \delta(c, v)} \text{E-PRIM} \qquad \frac{e \hookrightarrow e'}{e \ e_1 \hookrightarrow e' \ e_1} \text{E-APP1} \\
\\
\frac{e \hookrightarrow e'}{v \ e \hookrightarrow v \ e'} \text{E-APP2} \qquad \frac{}{(\lambda x. e) \ v \hookrightarrow e[v/x]} \text{E-APPABS} \\
\\
\frac{e_x \hookrightarrow e'_x}{\text{let } x = e_x \text{ in } e \hookrightarrow \text{let } x = e'_x \text{ in } e} \text{E-LET} \qquad \frac{}{\text{let } x = v \text{ in } e \hookrightarrow e[v/x]} \text{E-LETV} \\
\\
\frac{e \hookrightarrow e'}{e : t \hookrightarrow e' : t} \text{E-ANN} \qquad \frac{}{v : t \hookrightarrow v} \text{E-ANNV}
\end{array}$$

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 :

$$\begin{array}{c}
\frac{}{\Gamma \vdash_w b} \text{WF-BASE} \qquad \frac{\Gamma, x:b \vdash e : \text{Bool}}{\Gamma \vdash_w b\{x:e\}} \text{WF-REFN} \\
\\
\frac{\Gamma \vdash_w t_x \quad \Gamma, x:t_x \vdash_w t}{\Gamma \vdash_w x:t_x \rightarrow t} \text{WF-FUNC} \qquad \frac{\Gamma \vdash_w t_x \quad \Gamma, x:t_x \vdash_w t}{\Gamma \vdash_w \exists x:t_x. t} \text{WF-EXIS}
\end{array}$$

The judgement $\vdash_w \Gamma$ says that the environment Γ is well formed, meaning that variables are only bound to well-formed types:

$$\frac{}{\vdash_w \emptyset} \text{WFE-EMPTY} \qquad \frac{\Gamma \vdash_w t_x \quad \vdash_w \Gamma}{\vdash_w \Gamma, x:t_x} \text{WFE-BIND}$$

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.

$$\begin{array}{c}
\frac{ty(c) = t}{\Gamma \vdash c : t} \text{T-PRIM} \qquad \frac{x:t \in \Gamma}{\Gamma \vdash x : t} \text{T-VAR} \qquad \frac{\Gamma, x:t_x \vdash e : t \quad \Gamma \vdash_w t_x}{\Gamma \vdash \lambda x. e : x:t_x \rightarrow t} \text{T-ABS} \\
\\
\frac{\Gamma \vdash e : x:t_x \rightarrow t \quad \Gamma \vdash e' : t_x}{\Gamma \vdash e \ e' : \exists x:t_x. t} \text{T-APP} \qquad \frac{\Gamma \vdash e_x : t_x \quad \Gamma, x:t_x \vdash e_2 : t \quad \Gamma \vdash_w t}{\Gamma \vdash \text{let } x = e_x \text{ in } e : t} \text{T-LET} \\
\\
\frac{\Gamma \vdash e : t}{\Gamma \vdash e : t : t} \text{T-ANN} \qquad \frac{\Gamma \vdash e : s \quad \Gamma \vdash s <: t \quad \Gamma \vdash_w t}{\Gamma \vdash e : t} \text{T-SUB}
\end{array}$$

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

$$\begin{array}{c}
\frac{\Gamma, x_1 : b\{x_1 : p_1\} \vdash_e p_2[x_1/x_2]}{\Gamma \vdash b\{x_1 : p_1\} <: b\{x_2 : p_2\}} \text{S-BASE} \quad \frac{\Gamma \vdash s_2 <: s_1 \quad \Gamma, x_2 : s_2 \vdash t_1[x_2/x_1] <: t_2}{\Gamma \vdash x_1 : s_1 \rightarrow t_1 <: x_2 : s_2 \rightarrow t_2} \text{S-FUNC} \\
\\
\frac{\Gamma \vdash e_x : t_x \quad \Gamma \vdash t <: t'[e_x/x]}{\Gamma \vdash t <: \exists x : t_x. t'} \text{S-WITN} \quad \frac{\Gamma, x : t_x \vdash t <: t' \quad x \notin \text{free}(t')}{\Gamma \vdash \exists x : t_x. t <: t'} \text{S-BIND}
\end{array}$$

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:

$$\frac{\forall \theta. \theta \in \llbracket \Gamma \rrbracket \Rightarrow \theta(p) \hookrightarrow^* \mathbf{true}}{\Gamma \vdash_e p} \text{ENT-PRED}$$

The constraints in the grammar above are only used in the algorithmic typing rules. The constraints represent SMT instances that capture a predicate and its binding environment.

$$\begin{array}{c}
\frac{p \hookrightarrow^* \mathbf{true}}{\text{SMT}(p)} \text{ENT-EMP} \quad \frac{\text{SMT}(c_1) \quad \text{SMT}(c_2)}{\text{SMT}(c_1) \wedge \text{SMT}(c_2)} \text{ENT-CONJ} \\
\\
\frac{\text{SMT}(\forall x : b. p \Rightarrow c)}{\text{SMT}(p[v/x] \Rightarrow c[v/x])} \text{ENT-WITN}
\end{array}$$

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:

$$\llbracket b\{x : p\} \rrbracket := b, \quad \llbracket x : t_x \rightarrow t \rrbracket := \llbracket t_x \rrbracket \rightarrow \llbracket t \rrbracket, \quad \text{and} \quad \llbracket \exists x : t_x. t \rrbracket := \llbracket t \rrbracket$$

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:

$$\begin{aligned}
\llbracket b \rrbracket &:= \{e \mid \emptyset \vdash_B e : b\} \\
\llbracket b\{x : p\} \rrbracket &:= \{e \mid (\emptyset \vdash_B e : b) \wedge (\text{if } e \hookrightarrow^* v \text{ then } p[v/x] \hookrightarrow^* \mathbf{true})\} \\
\llbracket x : t_x \rightarrow t \rrbracket &:= \{e \mid (\emptyset \vdash_B e : \llbracket t_x \rrbracket \rightarrow \llbracket t \rrbracket) \wedge (\forall v_x \in \llbracket t_x \rrbracket. e v_x \in \llbracket t[v_x/x] \rrbracket)\} \\
\llbracket \exists x : t_x. t \rrbracket &:= \{e \mid (\emptyset \vdash_B e : \llbracket t \rrbracket) \wedge (\exists v_x \in \llbracket t_x \rrbracket. e \in \llbracket t[v_x/x] \rrbracket)\}
\end{aligned}$$

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:

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

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:

$$\llbracket \Gamma \rrbracket := \{ \theta = (x_1 \mapsto e_1, \dots, x_n \mapsto e_n) \mid \forall (x : t) \in \Gamma. \theta(x) \in \llbracket \theta(t) \rrbracket \}.$$

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

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

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 $0\leq, 1\leq, 2\leq, \dots$. Then by the definitions above we get our primitive typing lemma:

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

1. $\emptyset \vdash c : ty(c)$ and $\emptyset \vdash_w ty(c)$.
2. If $ty(c) = b\{x : p\}$, then $c \in \llbracket ty(c) \rrbracket$ and for all c' such that $c' \neq c$, $c' \notin \llbracket ty(c) \rrbracket$.
3. If $ty(c) = x:t_x \rightarrow t$, then for each $v \in \llbracket t_x \rrbracket$, $\delta(c, v)$ is defined and we have both $\emptyset \vdash \delta(c, v) : t[v/x]$ 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 proving several standard properties and basic facts used later on.

Lemma 2. *The operational semantics of λ_1 are deterministic: For every expression e there exists at most one term e' such that $e \hookrightarrow e'$.*

Lemma 3. (*Weakenings of Judgments*) For any environment Γ and $x \notin \text{dom}(\Gamma)$:

1. If $\Gamma \vdash e : t$ then $\Gamma, x:t_x \vdash e : t$.

2. If $\Gamma \vdash s <: t$ then $\Gamma, x:t_x \vdash s <: t$.

3. If $\Gamma \vdash_e p$ then $\Gamma, x:t_x \vdash_e p$.

4. If $\Gamma \vdash_w t$ then $\Gamma, x:t_x \vdash_w t$.

Proof. The proof is by mutual induction on the derivation tree of each type of judgement.

(1) **TODO**

(2) **TODO**

(3) **TODO**

(4) **TODO** □

Lemma 4. (*Reflexivity of $<:$*) If $\Gamma \vdash_w t$ and t is not a base type, then $\Gamma \vdash t <: t$.

TODO: Write up the proof from my notes

Proof. We proceed by induction of the structure of the derivation of $\Gamma \vdash_w t$.

Case WF-REFN: In the base case, we have.... □

Our proof of the soundness theorem begins with several helping lemmas.

Lemma 5. (*Preservation of Denotations*) If $e \hookrightarrow^* e'$ then $e \in \llbracket t \rrbracket$ iff $e' \in \llbracket t \rrbracket$.

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 $e \in \llbracket b \rrbracket$ then $\emptyset \vdash_B e : b$ and so $\emptyset \vdash_B e' : b$ and $e' \in \llbracket b \rrbracket$. Conversely if $e' \in \llbracket b \rrbracket$ then by determinism of the operational semantics and the typing relation, $e \in \llbracket b \rrbracket$. So we see that $e \in \llbracket b \rrbracket$. 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 $e \in \llbracket t \rrbracket$ then it holds that if $e \hookrightarrow^* v$ for some value v then $p[v/x] \hookrightarrow^* \mathbf{true}$. Suppose $e' \hookrightarrow^* v$ for some value v ; then by transitive closure $e \hookrightarrow^* v$, so $p[v/x] \hookrightarrow^* \mathbf{true}$ and we conclude $e' \in \llbracket t \rrbracket$.

If $e' \in \llbracket t \rrbracket$ then it holds that if $e' \hookrightarrow^* v$ for some value v then $p[v/x] \hookrightarrow^* \mathbf{true}$. 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 $p[v/x] \hookrightarrow^* \mathbf{true}$ and therefore $e \in \llbracket t \rrbracket$.

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

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

Lemma 6. (*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 $\forall \theta. \theta \in \llbracket \Gamma \rrbracket \Rightarrow \llbracket \theta(t_1) \rrbracket \subseteq \llbracket \theta(t_2) \rrbracket$.*
2. *If $\Gamma \vdash e : t$ then $\forall \theta. \theta \in \llbracket \Gamma \rrbracket \Rightarrow \theta(e) \in \llbracket \theta(t) \rrbracket$.*
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

$$\Gamma \vdash_e \forall x_1 : b. p_1 \Rightarrow p_2[x_1/x_2]. \quad (1)$$

By Lemma ?? we have

$$\forall \theta. \theta \in \llbracket \Gamma \rrbracket \Rightarrow \emptyset \vdash_e \theta(\forall x_1 : b. p_1 \Rightarrow p_2[x_1/x_2]),$$

or equivalently

$$\forall \theta. \theta \in \llbracket \Gamma \rrbracket \Rightarrow \emptyset \vdash_e \forall x_1 : b. \theta(p_1) \Rightarrow \theta(p_2)[x_1/x_2] \quad (2)$$

By inversion of rule ENT-EMPI, we have

$$\forall \theta. \theta \in \llbracket \Gamma \rrbracket \Rightarrow \forall (x_1 \mapsto e) \in \llbracket b\{x_1 : \theta(p_1)\} \rrbracket \Rightarrow \emptyset \vdash_e \theta(p_2)[x_1/x_2][e/x_1]. \quad (3)$$

Then by inversion of rule ENT-EMPP, we obtain

$$\forall \theta. \theta \in \llbracket \Gamma \rrbracket \Rightarrow \forall (x_1 \mapsto e) \in \llbracket b\{x_1 : \theta(p_1)\} \rrbracket \Rightarrow \theta(p_2)[e/x_2] \hookrightarrow^* \mathbf{true}. \quad (4)$$

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

$$\forall \theta. \theta \in \llbracket \Gamma \rrbracket \Rightarrow \{e \mid \emptyset \vdash_B e : b \wedge (\text{if } e \hookrightarrow^* v \text{ then } \theta(p_1[v/x_1]) \hookrightarrow^* \mathbf{true})\} \quad (5)$$

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

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

Case SUB-FUN: We have that $\Gamma \vdash x_1 : s_1 \rightarrow t'_1 <: x_2 : s_2 \rightarrow t'_2$ where $t_1 \equiv x_1 : s_1 \rightarrow t'_1$ and $t_2 \equiv x_2 : s_2 \rightarrow t'_2$. 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,

$$\forall \theta. \theta \in \llbracket \Gamma \rrbracket \Rightarrow \llbracket \theta(s_2) \rrbracket \subseteq \llbracket \theta(s_1) \rrbracket$$

and

$$\forall \theta. \theta \in \llbracket \Gamma, x_2:s_2 \rrbracket \Rightarrow \llbracket \theta(t'_1[x_2/x_1]) \rrbracket \subseteq \llbracket \theta(t'_2) \rrbracket \quad (7)$$

We need to show $\forall \theta. \theta \in \llbracket \Gamma \rrbracket \Rightarrow \llbracket \theta(x_1:s_1 \rightarrow t'_1) \rrbracket \subseteq \llbracket \theta(x_2:s_2 \rightarrow t'_2) \rrbracket$. Equivalently,

$$\forall \theta. \theta \in \llbracket \Gamma \rrbracket \Rightarrow \{e \mid \emptyset \vdash_B e : [s_1] \rightarrow [t'_1] \wedge (\forall e' \in \llbracket \theta(s_1) \rrbracket. e e' \in \llbracket \theta(t'_1[e'/x_1]) \rrbracket)\} \quad (8)$$

$$\subseteq \{e \mid \emptyset \vdash_B e : [s_2] \rightarrow [t'_2] \wedge (\forall e' \in \llbracket \theta(s_2) \rrbracket. e e' \in \llbracket \theta(t'_2[e'/x_2]) \rrbracket)\} \quad (9)$$

Fix $\theta \in \llbracket \Gamma \rrbracket$ and let e be a term in set (8) and let $e' \in \llbracket \theta(s_2) \rrbracket$. Then by induction, $e' \in \llbracket \theta(s_1) \rrbracket$. So $(e e') \in \llbracket \theta(t'_1[e'/x_1]) \rrbracket$. Let $\theta' = (\theta, x_2 \mapsto e')$. From (7) we also have that $\llbracket \theta'(t'_1[x_2/x_1]) \rrbracket \subseteq \llbracket \theta'(t'_2) \rrbracket$. 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 $(e e') \in \llbracket \theta(t'_1[e'/x_2]) \rrbracket \subseteq \llbracket \theta(t'_2[e'/x_2]) \rrbracket$ 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

$$\forall \theta. \theta \in \llbracket \Gamma \rrbracket \Rightarrow \llbracket \theta(t_1) \rrbracket \subseteq \llbracket \theta(t'_2[e_x/x]) \rrbracket \quad (10)$$

and by mutual induction we also have

$$\forall \theta. \theta \in \llbracket \Gamma \rrbracket \Rightarrow \theta(e_x) \in \llbracket \theta(t_x) \rrbracket.$$

We need to show that $\forall \theta$, if $\theta \in \llbracket \Gamma \rrbracket$, then $\llbracket \theta(t_1) \rrbracket \subseteq \llbracket \theta(\exists x:t_x. t'_2) \rrbracket$. Fix some $\theta \in \llbracket \Gamma \rrbracket$. Then

$$\llbracket \theta(\exists x:t_x. t'_2) \rrbracket = \{e \mid \emptyset \vdash_B e : [\theta(t'_2)] \wedge (\exists e' \in \llbracket \theta(t_x) \rrbracket. e \in \llbracket \theta(t'_2)[e'/x] \rrbracket)\} \quad (11)$$

because $\theta(\exists x:t_x. t'_2) = \exists x:\theta(t_x). \theta(t'_2)$. Let $e \in \llbracket \theta(t_1) \rrbracket$ and set $e' = \theta(e_x) \in \llbracket \theta(t_x) \rrbracket$. Then by (10), $e \in \llbracket \theta(t'_2[e_x/x]) \rrbracket$ and by definition of the denotation of a type, in every case $\emptyset \vdash_B e : [\theta(t'_2[e_x/x])] = [\theta(t'_2)]$. 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

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

By the inductive hypothesis, we have

$$\forall \theta. \theta \in \llbracket \Gamma, x:t_x \rrbracket \Rightarrow \llbracket \theta(t'_1) \rrbracket \subseteq \llbracket \theta(t_2) \rrbracket. \quad (12)$$

We need to show that for every $\theta \in \llbracket \Gamma \rrbracket$ that it holds that $\llbracket \theta(\exists x:t_x. t'_1) \rrbracket \subseteq \llbracket \theta(t_2) \rrbracket$. Fix some $\theta \in \llbracket \Gamma \rrbracket$ and let $e \in \llbracket \theta(\exists x:t_x. t'_1) \rrbracket$. By definition, $\theta(\exists x:t_x. t'_1) = \exists x:\theta(t_x). \theta(t'_1)$ so

$$\llbracket \theta(\exists x:t_x. t'_1) \rrbracket = \{e \mid \emptyset \vdash_B e : [\theta(t'_1)] \wedge (\exists e' \in \llbracket \theta(t_x) \rrbracket. e \in \llbracket \theta(t'_1)[e'/x] \rrbracket)\}. \quad (13)$$

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

t_x . Then $e \in \llbracket \theta'(t'_1) \rrbracket$, so from (12) we can conclude $e \in \llbracket \theta'(t_2) \rrbracket = \llbracket \theta(t_2) \rrbracket$ 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 $\theta \in \llbracket \Gamma \rrbracket$. In one case $t \equiv b\{x : p\}$; then by Lemma 1 on constants, $\theta(c) = c \in \llbracket ty(c) \rrbracket = \llbracket \theta(ty(c)) \rrbracket$. In the other case, $ty(c) \equiv x:t_x \rightarrow t'$; by Lemma 1, $\delta(c, v) \in \llbracket t'[v/x] \rrbracket$ for any $v \in \llbracket t_x \rrbracket$. There are no free variables in c or t so again $\theta(c) \in \llbracket \theta(ty(c)) \rrbracket$.

Case T-VAR: We have $\Gamma \vdash e : t$ where $e \equiv x$. By inversion, $(x:t) \in \Gamma$. Then for any $\theta \in \llbracket \Gamma \rrbracket$, we have by definition $\theta(x) \in \llbracket \theta(t) \rrbracket$ as desired.

Case T-ABS: We have $\Gamma \vdash e : t$ where $e \equiv \lambda x.e'$ and $t \equiv x:t_x \rightarrow t'$. By inversion, $\Gamma, x:t_x \vdash e' : t'$ and by the inductive hypothesis,

$$\forall \theta'. \theta' \in \llbracket \Gamma, x:t_x \rrbracket \Rightarrow \theta'(e') \in \llbracket \theta'(t') \rrbracket. \quad (14)$$

Let $\theta \in \llbracket \Gamma \rrbracket$ and let $e_x \in \llbracket \theta(t_x) \rrbracket$. Then let

$$\theta' := (\theta, x \mapsto e_x) \in \llbracket \Gamma, x:t_x \rrbracket.$$

Then from (14),

$$\theta(e')[e_x/x] = \theta'(e') \in \llbracket \theta'(t') \rrbracket = \llbracket \theta(t')[e_x/x] \rrbracket. \quad (15)$$

We need to show that for every $\theta \in \llbracket \Gamma \rrbracket$, it holds that

$$\begin{aligned} \theta(e) \in \llbracket \theta(x:t_x \rightarrow t') \rrbracket &= \llbracket x:\theta(t_x) \rightarrow \theta(t') \rrbracket \\ &= \{ \hat{e} \mid (\emptyset \vdash_B \hat{e} : [t_x] \rightarrow [t']) \wedge (\forall \hat{e}_x \in \llbracket \theta(t_x) \rrbracket. \hat{e} \hat{e}_x \in \llbracket \theta(t')[\hat{e}_x/x] \rrbracket) \} \end{aligned}$$

We have $\emptyset \vdash_B \theta(e) : [t_x] \rightarrow [t']$ 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 ?? we have that $\theta(e) e_x \in \llbracket t'[e_x/x] \rrbracket$. In the other case suppose that there exists some value such that $e_x \hookrightarrow^* v_x$. Then $\theta(e) e_x = (\lambda x.\theta(e')) e_x \hookrightarrow^* (\lambda x.\theta(e')) v_x \hookrightarrow \theta(e')[v_x/x] \in \llbracket \theta(t')[v_x/x] \rrbracket$. By a forthcoming lemma (TODO: this), $\llbracket \theta(t')[v_x/x] \rrbracket \subseteq \llbracket \theta(t')[e_x/x] \rrbracket$. Then by Lemma ?? we conclude that $\theta(e) e_x \in \llbracket \theta(t')[e_x/x] \rrbracket$ and so $\theta(e) \in \llbracket \theta(t') \rrbracket$.

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, $\Gamma \vdash e' : x:t_x \rightarrow t'$ and $\Gamma \vdash e_x : t_x$. By the inductive hypothesis we have

$$\forall \theta. \theta \in \llbracket \Gamma \rrbracket \Rightarrow \theta(e') \in \llbracket \theta(x:t_x \rightarrow t') \rrbracket \quad (16)$$

and

$$\forall \theta. \theta \in \llbracket \Gamma \rrbracket \Rightarrow \theta(e_x) \in \llbracket \theta(t_x) \rrbracket. \quad (17)$$

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

$$\theta(e) = \theta(e') \theta(e_x) \in \llbracket \exists x:\theta(t_x). \theta(t') \rrbracket = \llbracket \theta(\exists x:t_x. t') \rrbracket. \quad (18)$$

Case T-LET: We have $\Gamma \vdash e : t$ where $e \equiv \text{let } x = e_x \text{ in } e'$. 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

$$\forall \theta. \theta \in \llbracket \Gamma \rrbracket \Rightarrow \theta(e_x) \in \llbracket \theta(t_x) \rrbracket$$

and

$$\forall \theta'. \theta' \in \llbracket \Gamma, x:t_x \rrbracket \Rightarrow \theta'(e') \in \llbracket \theta'(t) \rrbracket. \quad (19)$$

Let $\theta \in \llbracket \Gamma \rrbracket$. 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 $\theta' = (\theta, x \mapsto v_x) \in \llbracket \Gamma, x:t_x \rrbracket$ because we chose $\theta'(x) = v_x \in \llbracket \theta(t_x) \rrbracket$. From the operational semantics $\theta(\text{let } x=e_x \text{ in } e') = \text{let } x=\theta(e_x) \text{ in } \theta(e') \hookrightarrow^* \text{let } x=v_x \text{ in } \theta(e') \hookrightarrow \theta(e')[v_x/x]$. Then from (19),

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

where the last equality follows from the fact that the judgement $\Gamma \vdash_w t$ implies $\emptyset \vdash_w \theta(t)$ 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 5, $\theta(e) \in \llbracket \theta(t) \rrbracket$.

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

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, $\theta(e') \in \llbracket \theta(t) \rrbracket$. By the operational semantics of type annotations, $\theta(e) = (\theta(e') : \theta(t)) \hookrightarrow \theta(e') \in \llbracket \theta(t) \rrbracket$, so we conclude that $\theta(e) \in \llbracket \theta(t) \rrbracket$ by Lemma 5.

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, $\forall \theta. \theta \in \llbracket \Gamma \rrbracket \Rightarrow \theta(e) \in \llbracket \theta(s) \rrbracket$ and by mutual induction, part 1 of the Lemma gives us that $\forall \theta. \theta \in \llbracket \Gamma \rrbracket \Rightarrow \llbracket \theta(s) \rrbracket \subseteq \llbracket \theta(t) \rrbracket$. Then we conclude that $\forall \theta. \theta \in \llbracket \Gamma \rrbracket \Rightarrow \theta(e) \in \llbracket \theta(t) \rrbracket$.

Case T-BOT: We have $\Gamma \vdash \perp : t$ where $t \equiv b\{x : p\}$. Then because \perp cannot evaluate, we have that $\perp \in \llbracket b\{x : \theta(p)\} \rrbracket = \llbracket \theta(t) \rrbracket$. \square

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

$$\Gamma, x:t_x, \Gamma' \vdash \forall x_1:b. p_1 \Rightarrow p_2[x_1/x_2]. \quad (20)$$

By Lemma ??, we have

$$\forall \theta^*. \theta^* \in \llbracket \Gamma, x:t_x, \Gamma' \rrbracket \Rightarrow \emptyset \vdash_e \theta^*(\forall x_1:b. p_1 \Rightarrow p_2[x_1/x_2]) \quad (21)$$

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

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

$$\forall \theta, \theta'. (\theta, x \mapsto \theta(e_x), \theta') \in \llbracket \Gamma, x:t_x, \Gamma' \rrbracket \Rightarrow (\emptyset \vdash_e \forall x_1:b. \theta'(\theta(p_1)[\theta(e_x)/x]) \Rightarrow \theta'(\theta(p_2)[\theta(e_x)/x]))[x_1/x_2])$$

or equivalently,

$$\forall \theta, \theta'. (\theta, \theta') \in \llbracket \Gamma, \Gamma'[e_x/x] \rrbracket \Rightarrow (\emptyset \vdash_e \forall x_1:b. (\theta, \theta')(p_1[e_x/x]) \Rightarrow (\theta, \theta')(p_2[e_x/x][x_1/x_2])) \quad (23)$$

TODO: can't finish this one yet. \square

MOST RECENTLY UPDATED

Lemma 7. (*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

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

3. If $\Gamma, x:t_x, \Gamma' \vdash_w t$ then

$$\Gamma, \Gamma'[e_x/x] \vdash_w t[e_x/x]$$

Proof. We proceed by mutual induction □

Lemma 8. (*Well-formedness of types in judgements*) If $\Gamma \vdash e : t$ and $\vdash_w \Gamma$ then $\Gamma \vdash_w t$.

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

Case T-PRIM: We have $e \equiv c$. By inversion, $t = ty(c)$ and by Lemma 1 we have $\emptyset \vdash_w ty(c)$. By repeated application of Lemma 3, we have $\Gamma \vdash_w ty(c)$.

Case T-VAR: We have $\Gamma \vdash e : t$ where $e \equiv x$. By inversion, $x:t \in \Gamma$, so we can write $\Gamma \equiv \Gamma', x:t, \Gamma''$ and by repeated inversion of WFE-BIND, $\vdash_w \Gamma', x:t$ and inverting again we get $\Gamma' \vdash_w t$. Inductively applying Lemma 3 gives us $\Gamma \vdash_w t$.

Case T-ABS: We have $\Gamma \vdash e : t$ where $e \equiv \lambda x. e'$ and $t \equiv x:t_x \rightarrow t'$. By inversion, we have $\Gamma, x:t_x \vdash e' : t'$ and $\Gamma \vdash_w t_x$. By the inductive hypothesis, we have $\Gamma, x:t_x \vdash_w t'$. By rule WF-FUNC we have $\Gamma \vdash_w x:t_x \rightarrow t'$.

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, $\Gamma \vdash e_1 : x:t_x \rightarrow t'$ and $\Gamma \vdash e_2 : t_x$. By the inductive hypothesis we have $\Gamma \vdash_w x:t_x \rightarrow t'$ and $\Gamma \vdash_w t_x$. By inverting rule WF-FUNC (on the former judgement), we have $\Gamma \vdash_w t_x$ and $\Gamma, x:t_x \vdash_w t'$. By rule WF-EXIS, $\Gamma \vdash_w \exists x:t_x. t'$.

Case T-LET: We have $\Gamma \vdash e : t$ where $e \equiv \text{let } x=e_x \text{ in } e'$. By inversion we have, in particular, that $\Gamma \vdash_w t$.

Case T-ANN: We have $\Gamma \vdash e : t$ where $e \equiv e' : t$. By inversion we have $\Gamma \vdash e' : t$ and by the inductive hypothesis we conclude $\Gamma \vdash_w t$.

Case T-SUB: We have $\Gamma \vdash e : t$. By inversion we have, in particular, $\Gamma \vdash_w t$. □

Lemma 9. (*Witnesses and subtyping*) If $\Gamma \vdash e_x : t_x$ and $\Gamma, x:t_x \vdash_w t'$ then $\Gamma \vdash t'[e_x/x] <: \exists x:t_x. t'$.

Proof. By Lemma 4, we have that $\Gamma \vdash t' <: t'$ and by the Substitution Lemma we have $\Gamma \vdash t'[e_x/x] <: t'[e_x/x]$. Applying rule S-WITN (with $t \equiv \exists x:t_x. t'$), we get $\Gamma \vdash t'[e_x/x] <: \exists x:t_x. t'$. □

Theorem 10. (*The Progress Theorem*) If $\emptyset \vdash e : t$ then either e is a value or there exists a term e' such that $e \hookrightarrow e'$.

TODO: Do I need crash?.

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

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

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

Case T-ABS: This case holds trivially because $e \equiv \lambda x.e'$ is a value.

Case T-APP: We have $\emptyset \vdash e : t$ where $e \equiv e_1 e_2$ and $t \equiv \exists x:t_x.t'$. By inversion, $\emptyset \vdash e_1 : x:t_x \rightarrow t'$ and $\emptyset \vdash e_2 : t_x$ for some type s . 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 $\emptyset \vdash e_2 : t_x$), 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-LET: We have $\emptyset \vdash e : t$ where $e \equiv (\text{let } x=e_1 \text{ in } e_2)$. By inversion, $\emptyset \vdash e_1 : t_x$ and $x:t_x \vdash e_2 : t$. First, suppose that $e_1 \equiv v$. Then by rule T-LETV, $\text{let } x=v \text{ in } e_2 \hookrightarrow e_2[v/x]$. Second, suppose that e_1 is not a value. Then by the inductive hypothesis (applied to judgement $\emptyset \vdash e_1 : t_x$), there exists a term e'_1 such that $e_1 \hookrightarrow e'_1$. Then by rule E-LET we have $\text{let } x=e_1 \text{ in } e_2 \hookrightarrow \text{let } x=e'_1 \text{ in } e_2$.

Case T-ANN: We have $\emptyset \vdash e : t$ where $e \equiv (e_1 : t)$. By inversion, $\emptyset \vdash e_1 : t$. 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 $\emptyset \vdash e : t$. By inversion, $\emptyset \vdash e : s$, $\emptyset \vdash s <: t$, and $\emptyset \vdash_w t$ 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. \square

Theorem 11. (*The Preservation Theorem*) If $\emptyset \vdash e : t$ and $e \hookrightarrow e'$, then $\emptyset \vdash e' : t$.

Todo: Do I need crash?

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

Case T-CON: 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-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-APP: We have $\emptyset \vdash e : t$ 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, $\emptyset \vdash e_1 : x:t_x \rightarrow t'$ and $\emptyset \vdash e_2 : t_x$. 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 semantics $e' = \delta(c, v)$ and by Lemma 1, we have $\emptyset \vdash \delta(c, v) : t'[v/x]$ because n. By Lemma 8, we have $\emptyset \vdash_w \exists x:t_x.t'$ and by inverting WF-EXIS we have $x:t_x \vdash_w t'$. By Lemma 9, $\emptyset \vdash t'[v/x] <: \exists x:t_x.t'$, and so by rule T-SUB, $\emptyset \vdash \delta(c, v) : \exists x:t_x.t'$.

Second, consider $e_1 \equiv c$ and e_2 not a value. By Theorem 10, 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, $\emptyset \vdash e'_2 : t_x$. We conclude by T-APP that $\emptyset \vdash e' : \exists x:t_x. t'$.

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, $e' \equiv e'_1[v/x]$. By inversion of T-ABS, we have $x:t_x \vdash e'_1 : t'$, and by the substitution lemma we have $\emptyset \vdash e'_1[v/x] : t'[v/x]$. By Lemma 8, we have $\emptyset \vdash_w \exists x:t_x. t'$ and by inverting WF-EXIS we have $x:t_x \vdash_w t'$. By Lemma 9, $\emptyset \vdash t'[v/x] <: \exists x:t_x. t'$, and so by rule T-SUB, $\emptyset \vdash e' : \exists x:t_x. t'$.

Fourth, consider $e_1 \equiv \lambda x.e'_1$ and e_2 not a value. By Theorem 10, 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, $\emptyset \vdash e'_2 : t_x$. We conclude by T-APP that $\emptyset \vdash e' : \exists x:t_x. t'$.

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 10, 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, $\emptyset \vdash e'_1 : \exists x:t_x. t'$. By rule SYN-APP, $\emptyset \vdash e' : \exists x:t_x. t'$.

Case T-LET: We have $\emptyset \vdash e : t$ where $e \equiv (\text{let } x=e_1 \text{ in } e_2)$ and $t \equiv t_2$. By inversion, $\emptyset \vdash e_1 : t_1$, $x:t_1 \vdash e_2 : t_2$, and $\emptyset \vdash_w t_2$ for some type t_1 . First suppose that e_1 is not a value. Then by Theorem 10, there exists some term e'_1 such that $e_1 \hookrightarrow e'_1$. By Rule E-LET, $\text{let } x=e_1 \text{ in } e_2 \hookrightarrow \text{let } x=e'_1 \text{ in } e_2$, and by determinism of the operational semantics, $e' \equiv \text{let } x=e'_1 \text{ in } e_2$. By the inductive hypothesis, $\emptyset \vdash e_1 : t_1$. Then by T-LET, $\emptyset \vdash e' : t_2$.

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

Case T-ANN: We have $\emptyset \vdash e : t$ where $e \equiv (e_1 : t)$ and $e \hookrightarrow e'$. By inversion, $\emptyset \vdash e_1 : t$. By Theorem 10 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, $\emptyset \vdash e'_1 : t$. By rule SYN-ANN, $\emptyset \vdash (e'_1 : t) : t$.

Case T-SUB: We have $\emptyset \vdash e : t$. By inversion $\emptyset \vdash e : s$ and $\emptyset \vdash s <: t$ for some type s , and also $\emptyset \vdash_w t$. By the inductive hypothesis $\emptyset \vdash e' : s$. By rule CHK-SYN, $\emptyset \vdash e' : t$. \square

4 Algorithmic Typing

Lemma 12. (*Denotational Soundness of SMT*) If $\Gamma \vdash p$ then $\Gamma \vdash_e p$.

General Todo list:

- does having call-by-value (non-lazy) semantics affect wha the definitions of denotations of types ($\llbracket \cdot \rrbracket$) 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 $\llbracket t[v/x] \rrbracket \subseteq \llbracket t[e/x] \rrbracket$
- Lemma 5 needs a part (3): If $\Gamma \vdash_w t$ then $\forall \theta. \theta \in \llbracket \Gamma \rrbracket \Rightarrow \emptyset \vdash_w \theta(t)$.
- 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).