A proof of correctness for the Hindley-Milner type inference algorithm

Jeff Vaughan
vaughan2@cis.upenn.edu

May 5, 2005 (Revised July 23, 2008)

1 Introduction

This report details a proof that the Hindley-Milner type inference algorithm is sound and complete with respect to a declarative formulation of let polymorphism. Let polymorphism is a decidable type system, which allows polymorphic types to be introduced to type contexts only in let expressions, and is the basis of the ML and Haskell type systems.

1.1 Road map

I will show that that the both the declarative type system are sound and complete with respect to an intermediate syntax directed system. Section 2 provides definitions for the type systems and supporting mathematical structures. The proofs and auxiliary lemmas are located in section 3.

2 Definitions

2.1 Preliminaries

2.1.1 Sequences, sets and bar notation

I use the notation $\overline{x}$ ("bar notation") to denote a finite ordered sequence of objects: $x_1 x_2 x_3 \ldots x_n$. Discussion of an arbitrary element is a shorthand for quantification over all elements (e.g. $P(x_i)$ instead of $\forall i \in \{1 \ldots n\}.P(x_i)$). Here n is a fixed number, which is potentially different for each sequence; bar notation is clearly not appropriate when we care about n's value.

I freely convert between sets and sequences. For example, I use $a \in \overline{x}$ as shorthand for $a \in \{x_j | j \in \{1 \ldots n\}\}$, or $\overline{y} = Y$ to make $\overline{y}$ an arbitrary sequence of the elements in Y.

The notation $[\overline{\tau}/\overline{\alpha}]$ is a shorthand for the substitution $[\tau_1/\alpha_1] \circ [\tau_2/\alpha_2] \circ \ldots [\tau_n/\alpha_n]$.

2.1.2 Type convention

This report includes both quantified and unquantified types. We denote unquantified types using τ and potentially quantified types with σ. The grammar of types is

$$\begin{array}{lll} \tau & ::= & \alpha \mid \tau \to \tau \\ \sigma & ::= & \tau \mid \forall \alpha.\sigma \end{array}$$

Type equality is syntactic up to alpha renaming. For monotypes, equality is purely structural. Polymorphic equality is defined using substitution (section 2.1.3). Types $\forall \overline{\alpha}.\tau$ and $\forall \overline{\beta}.\tau'$ are equal iff $\tau = [\overline{\alpha_i}/\overline{\beta_j}]\tau'$. This relation is obviously transitive, reflexive and symmetric.

The ftv function evaluates to the set of free type variables of its argument. Formally,

$$\begin{array}{lcl} \text{ftv}(\alpha) & = & \{\alpha\} \\ \text{ftv}(\tau \to \tau') & = & \text{ftv}(\tau) \cup \text{ftv}(\tau') \\ \text{ftv}(\forall\alpha.\sigma) & = & \text{ftv}(\sigma) \setminus \{\alpha\} \end{array}$$

2.1.3 Substitution

Substitution is defined as usual. Substitution over quantified types is capture avoiding. We write τ replaces α as $[\tau/\alpha]$ and the null substitution as $[-]$. Substitutions R and S are defined to be equal iff $\forall x.R(x) = S(x)$.

As functions, substitutions can be composed and applied. I use $RS\tau$ as shorthand for $(R \circ S)(\tau)$.

When $\overline{\alpha} = \alpha_1\alpha_2 \ldots \alpha_n$, I use notation $S\overline{\alpha}$ to denote mapping S over $\overline{\alpha}$, i.e. $(S\alpha_1)(S\alpha_2)\ldots(S\alpha_n)$.

The domain and range of a substitution are defined as follows

$$\begin{array}{lcl} \text{dom}(S) & = & \{\alpha : \alpha \neq S(\alpha)\} \\ \text{range}(S) & = & \{S(\alpha) : \alpha \in \text{dom}(S)\} \end{array}$$

These definitions allow us to prove $S = S'$ implies $\text{dom}(S) = \text{dom}(S')$ and $\text{range}(S) = \text{range}(S')$. Additionally, we define the derived form $\text{vars}(S) = \text{dom}(S) \cup \text{ftv}(\text{range}(S))$. We say type variable α is *fresh* for S when $\alpha \notin \text{vars}(S)$.

I define a domain restriction operator, $|_\neg$, as

$$S|_{\neg X}(\beta) = \begin{cases} \beta & \beta \notin X \\ S(\beta) & \text{otherwise} \end{cases}$$

Domain restriction binds less tightly than composition, and $S' \circ S|_{\neg \text{X}}$ reads as $(S' \circ S)|_{\neg \text{X}}$. We will use the following domain restriction properties:

- $S \circ T|_{\neg X} = S \circ (T|_{\neg X})|_{\neg X}$
- $S \circ T|_{\neg X} = (S|_{\neg X}) \circ (T|_{\neg X})$ where $X \cap \text{range}(T) = \emptyset$
- $[\tau/\alpha]|_{\neg\{\alpha\}} = [-]$
- $S|_{\neg X} = S$ where $X \cap \text{dom}(S) = \emptyset$
- $S|_{\neg X \cup Y} = (S|_{\neg X})|_{\neg Y}$

2.1.4 Ordering on types

We order types with the $\sqsubseteq$ operator which is defined as follows:

$$\frac{\beta_i \notin \text{ftv}(\forall\overline{\alpha}.\tau) \qquad \tau' = [\overline{\tau}/\overline{\alpha}]\tau}{\forall\overline{\alpha}.\tau \sqsubseteq \forall\overline{\beta}.\tau'}$$

Generally, $|\overline{\alpha}| \neq |\overline{\beta}|$. Intuitively $\sigma \sqsubseteq \sigma'$ means σ' is more specific than σ. For example,

$$\forall\beta.\beta \to \beta \sqsubseteq \forall\alpha.(\alpha \to \alpha) \to (\alpha \to \alpha) \sqsubseteq (\gamma \to \gamma) \to (\gamma \to \gamma)$$

Since there is only a single rule, this judgment is invertible. Additionally, since the rule is purely syntactic, for all instantiations of σ and σ', if $\sigma \sqsubseteq \sigma'$, we can prove it with a derivation of height one. Lastly, the $\sqsubseteq$ relation is transitive.

2.1.5 Contexts and generalization

We define typing contexts in the usual fashion,

$$\Gamma \quad ::= \quad \emptyset \mid \Gamma, x : \sigma$$

The free type variables of a context are defined as the union of the free type variables in all bindings.

$$\begin{array}{lcl} \mathrm{ftv}(\emptyset) & = & \emptyset \\ \mathrm{ftv}(\Gamma, x : \sigma) & = & \mathrm{ftv}(\Gamma) \cup \mathrm{ftv}(\sigma) \end{array}$$

Substitution over a context is defined by

$$\begin{array}{lcl} S(\emptyset) & = & \emptyset \\ S(\Gamma, x : \sigma) & = & S(\Gamma), x : S(\sigma) \end{array}$$

We also define a generalization operator which is a function from a context and a type to a new type. Intuitively, type $\overline{\Gamma}(\tau)$ is τ with all free variables quantified. Formally,

$$\overline{\Gamma}(\tau) = \forall \overline{\alpha}.\tau \quad \text{where} \quad \overline{\alpha} = \mathrm{ftv}(\tau) \setminus \mathrm{ftv}(\Gamma)$$

2.1.6 Unification

We suppose the existence of a unification algorithm, $\mathcal{U}$, with the following properties.

- $\mathcal{U}(\tau, \tau') = V$ such that $V(\tau) = V(\tau')$ or no such substitution exists and $\mathcal{U}(\tau, \tau')$ is undefined.
- $\mathrm{vars}(\mathcal{U}(\tau, \tau')) \subseteq \mathrm{ftv}(\tau) \cup \mathrm{ftv}(\tau')$
- $R(\tau) = R(\tau')$ implies there exists S such that $R = S \circ \mathcal{U}(\tau, \tau')$. That is, $\mathcal{U}(\tau, \tau')$ is the the most general unifier.

2.2 Declarative Typing Rules

$$\frac{x : \sigma \in \Gamma}{\Gamma \vdash_D x : \sigma}\ \text{D-Var}$$

$$\frac{\Gamma \vdash_D e : \sigma' \qquad \sigma' \sqsubseteq \sigma}{\Gamma \vdash_D e : \sigma}\ \text{D-Inst}$$

$$\frac{\Gamma \vdash_D e : \sigma \qquad \alpha \notin \mathrm{ftv}(\Gamma)}{\Gamma \vdash_D e : \forall \alpha.\sigma}\ \text{D-Gen}$$

$$\frac{\Gamma \vdash_D e : \tau' \to \tau \qquad \Gamma \vdash_D e' : \tau'}{\Gamma \vdash_D e e' : \tau}\ \text{D-App}$$

$$\frac{\Gamma, x : \tau \vdash_D e : \tau'}{\Gamma \vdash_D \lambda x.e : \tau \to \tau'}\ \text{D-Abs}$$

$$\frac{\Gamma \vdash_D e : \sigma \qquad \Gamma, x : \sigma \vdash_D e' : \tau}{\Gamma \vdash_D \texttt{let}\ x = e\ \texttt{in}\ e' : \tau}\ \text{D-Let}$$

2.3 Syntax Directed Typing Rules

$$\frac{x : \sigma \in \Gamma \qquad \sigma \sqsubseteq \tau}{\Gamma \vdash_S x : \tau} \text{ SD-Var}$$

$$\frac{\Gamma \vdash_S e : \tau \qquad \Gamma, x : \overline{\Gamma}(\tau) \vdash_S e' : \tau'}{\Gamma \vdash_S \texttt{let}\, x = e\, \texttt{in}\, e' : \tau'} \text{ SD-Let}$$

$$\frac{\Gamma \vdash_S e : \tau' \to \tau \qquad \Gamma \vdash_S e' : \tau'}{\Gamma \vdash_S ee' : \tau} \text{ SD-App}$$

$$\frac{\Gamma, x : \tau \vdash_S e : \tau'}{\Gamma \vdash_S \lambda x.e : \tau \to \tau'} \text{ SD-Abs}$$

2.4 Algorithmic Typing Rules

The algorithmic typing rules are a bit trickier. The Hindley-Milner algorithm is based on unification, and occasionally needs fresh type variables. To avoid the messiness that would accompany formalizing a symbol generator, we thread a sequence, or "tape," of type variables through the judgment.

We write typing judgments in the algorithmic system as:

$$\Gamma; \mathrm{AX} \vdash_W e \uparrow (S, \tau, \mathrm{A})$$

This means that running the Hindley-Milner algorithm on expression e in context Γ with tape AX finds $e : \tau$. Additionally, the algorithm returns substitution S and tape prefix A. Substitution S is used to transmit the results of unifications performed by the algorithm to the enclosing context.

$$\frac{x : \forall \overline{\alpha}.\tau \in \Gamma}{\Gamma; \mathrm{A}\overline{\beta} \vdash_W e \uparrow (\emptyset, [\overline{\beta}/\overline{\alpha}]\tau, \mathrm{A})} \text{ WA-Var}$$

$$\frac{\Gamma; \mathrm{A}\beta\mathrm{YX} \vdash_W e_1 \uparrow (S_1, \tau_1, \mathrm{A}\beta\mathrm{Y}) \qquad S_1(\Gamma); \mathrm{A}\beta\mathrm{Y} \vdash_W e_2 \uparrow (S_2, \tau_2, \mathrm{A}\beta) \qquad V = \mathcal{U}(S_2\tau_1, \tau_2 \to \beta)}{\Gamma; \mathrm{A}\beta\mathrm{YX} \vdash_W e_1 e_2 \uparrow (V \circ S_2 \circ S_1, V(\beta), \mathrm{A})} \text{ WA-App}$$

$$\frac{\Gamma, x : \beta; \mathrm{AX} \vdash_W e \uparrow (S, \tau, \mathrm{AX})}{\Gamma; \mathrm{AX}\beta \vdash_W \lambda x.e \uparrow (S, S(\beta) \to \tau, \mathrm{A})} \text{ WA-Abs}$$

$$\frac{\Gamma; \mathrm{AYX} \vdash_W e_0 \uparrow (S_0, \tau_0, \mathrm{AY}) \qquad S_0(\Gamma), x : \overline{S_0(\Gamma)}(\tau_0); \mathrm{AY} \vdash_W e_1 \uparrow (S_1, \tau_1, \mathrm{A})}{\Gamma; \mathrm{AYX} \vdash_W \texttt{let}\, x = e_0\, \texttt{in}\, e_1 \uparrow (S_1 \circ S_0, \tau_1, \mathrm{A})} \text{ WA-Let}$$

3 Theorems

3.1 Soundness of the syntax directed declarative rules

The statement of soundness for the syntax directed system is simple: $\Gamma \vdash_S e : \tau \implies \Gamma \vdash_D e : \tau$. This statement is strong enough to be proved by induction. However, first will need the following lemma:

3.1.1 Generalization lemma

Lemma: $\Gamma \vdash_D e : \tau$ and $\sigma = \overline{\Gamma}(\tau)$ implies $\Gamma \vdash_D e : \sigma$.

Proof: I will show that $\Gamma \vdash_D e : \forall\alpha_1 \ldots \alpha_n.\tau$ where $\alpha_i \in \text{ftv}(\tau) \setminus \text{ftv}(\Gamma)$ by induction on the length of the sequence of unique quantification variables. Case analysis on the number of quantification variables:

- $n = 0$: $\forall\alpha_1 \ldots \alpha_n.\tau$ is identical to τ, and $\Gamma \vdash_D e : \forall\alpha_1 \ldots \alpha_n.\tau$ trivially.
- $n > 0$: By the inductive hypothesis, $\Gamma \vdash_D e : \forall\alpha_1 \ldots \alpha_{n-1}.\tau$

  Recall $\alpha_i \in \text{ftv}(\tau) \setminus \text{ftv}(\Gamma)$.

  From these, rule D-Gen gives

  $$\Gamma \vdash_D e : \forall\alpha_1 \ldots \alpha_n.\tau$$

Taking $\overline{\alpha} = \text{ftv}(\tau) \setminus \text{ftv}(\Gamma)$ shows $\Gamma \vdash_D e : \overline{\Gamma}(\tau)$.

QED

3.1.2 Proof of soundness for the syntax directed system

We will show $\Gamma \vdash_S e : \tau \implies \Gamma \vdash_D e : \tau$. Proof is by induction on the height of the typing judgment $\Gamma \vdash_S e : \tau$. We do case analysis on the final step of the syntax directed proof.

- SD-Var: $e = x$

  By the premises of SD-Var, $x : \sigma \in \Gamma$ and $\sigma \sqsubseteq \tau$. In the declarative system we derive:

  $$\dfrac{\dfrac{x : \sigma \in \Gamma}{\Gamma \vdash_D x : \sigma}\ \text{D-Var} \qquad \sigma \sqsubseteq \tau}{\Gamma \vdash_D x : \tau}\ \text{D-Inst}$$

- SD-Let: $e = \texttt{let}\ x = e_0\ \texttt{in}\ e_1$ and $\tau = \tau_1$

  The premises of SD-Let give $\Gamma \vdash_S e_0 : \tau_0$ and $\Gamma, x : \overline{\Gamma}(\tau_0) \vdash_S e_1 : \tau_1$

  By the induction hypothesis, we know $\Gamma \vdash_D e_0 : \tau_0$ and $\Gamma, x : \sigma \vdash_D e_1 : \tau_1$. By the generalization lemma proved in section 3.1.1, $\Gamma \vdash_D e_0 : \overline{\Gamma}(\tau_0)$.

  We can now apply D-Let

  $$\frac{\Gamma \vdash_D e_0 : \sigma \qquad \Gamma, x : \sigma \vdash_D e_1 : \tau_1}{\Gamma \vdash_D \texttt{let}\ x = e_0\ \texttt{in}\ e_1 : \tau_1}\ \text{D-Let}$$

- SD-App and SD-Abs. These rules are identical to their declarative counterparts.

QED

3.2 Completeness of the syntax directed system

While the declarative system can assign polytypes to terms, the syntax directed one cannot. Therefore, it's clearly not the case that $\Gamma \vdash_D e : \sigma$ implies $\Gamma \vdash_S e : \sigma$. We state a completeness differently, and require that the syntax directed system can assign a type which, when generalized, is at least as general as σ. That is, $\Gamma \vdash_S e : \tau$ with $\overline{\Gamma}(\tau) \sqsubseteq \sigma$.

Fortunately this is provable using straightforward induction and the following lemma.

3.2.1 Context generalization lemma

Lemma: $\sigma \sqsubseteq \sigma'$ and $\Gamma, x : \sigma' \vdash_S e : \tau$ implies $\Gamma, x : \sigma \vdash_S e : \tau$ with a derivation no larger than the first.

Proof: By induction on the height of the first derivation. We apply case analysis of the final step this derivation.

- SD-Var

  The premises of SD-Var give $x : \sigma' \in \Gamma$, $x : \sigma'$ and $\sigma' \sqsubseteq \tau$. Additionally we use transitivity of $\sqsubseteq$ to realize, $\sigma \sqsubseteq \tau$. By the form of the definition of $\sqsubseteq$ (section 2.1.4) that sub derivation can be written with height one.

  We can write the following proof:

$$\frac{x : \sigma \in \Gamma, x : \sigma \qquad \sigma \sqsubseteq \tau}{\Gamma,\, x : \tau}\ \text{SD-Var}$$

- SD-Let, SD-App, and SD-Abs do not make specific reference to the contents of their context. Each conclusion follows directly from the inductive hypotheses.

QED

3.2.2 Proof of completeness of the syntax directed system

We will show $\Gamma \vdash_D e : \sigma \implies \Gamma \vdash_S e : \tau$ where $\overline{\Gamma}(\tau) \sqsubseteq \sigma$. Proof is by induction on the height of the declarative typing judgment and uses case analysis on its final step.

- D-Inst

  The premises of D-Inst give $\Gamma \vdash_D e : \sigma'$.

  By the inductive hypothesis, $\Gamma \vdash_S e : \tau$ and $\overline{\Gamma}(\tau) \sqsubseteq \sigma'$. By the transitivity of the type ordering relation, we have $\overline{\Gamma}(\tau) \sqsubseteq \sigma$.

- D-Var: $e = x$ and $\sigma = \forall \overline{\alpha}.\tau$ where, by alpha renaming, $\alpha_i \notin \text{ftv}(\Gamma)$

  The premise of D-Var gives $x : \sigma \in \Gamma$.

  As $\sigma \sqsubseteq \tau$, we can use SD-Var to find $\Gamma \vdash_S x : \tau$.

  Now we want to show $\overline{\Gamma}(\tau) \sqsubseteq \sigma$.

$$\text{ftv}(\overline{\Gamma}(\tau)) = \text{ftv}(\Gamma) \cap \text{ftv}(\tau)$$

  Therefore $\alpha_i \notin \text{ftv}(\overline{\Gamma}(\tau))$, and

$$\frac{\alpha_i \notin \text{ftv}(\overline{\Gamma}(\tau)) \qquad \tau = [-]\tau}{\overline{\Gamma}(\tau) \sqsubseteq \sigma}\ \text{Defn. of } \sqsubseteq$$

- D-Gen: $\sigma = \forall \beta'.\sigma = \forall \beta', \overline{\beta}.\tau'$

  The derivation ends with

$$\frac{\Gamma \vdash_D e : \sigma \qquad \beta' \notin \text{ftv}(\Gamma)}{\Gamma \vdash_D e : \forall \beta'.\sigma}\ \text{D-Gen}$$

  By the inductive hypothesis we know $\Gamma \vdash_S e : \tau$ where $\overline{\Gamma}(\tau) = \forall \overline{\alpha}.\tau \sqsubseteq \sigma$.

  Inverting the definition of $\sqsubseteq$ we have $\beta_i \notin \text{ftv}(\overline{\Gamma}(\tau))$ and $\tau' = [\overline{\tau}/\overline{\alpha}]\tau$.

As $\beta' \notin \text{ftv}(\Gamma)$, we know $\beta' \notin \text{ftv}(\overline{\Gamma}(\tau))$.

$$\frac{\beta', \beta_i \notin \text{ftv}(\overline{\Gamma}(\tau)) \qquad \tau' = [\overline{\tau}/\overline{\alpha}]\tau}{\forall \overline{\alpha} \sqsubseteq \forall \beta', \overline{\beta}.\tau'} \text{ Defn. of } \sqsubseteq$$

Rewriting the final statement in the derivation gives us $\overline{\Gamma}(\tau) \sqsubseteq \sigma$.

- D-Let: $e = \texttt{let}\ x = e_0\ \texttt{in}\ e_1$, $\tau = \tau_1$, and $\sigma = \sigma_1$

  Premises give $\Gamma \vdash_D e_0 : \sigma_0$ and $\Gamma, x : \sigma_0 \vdash_D e_1 : \sigma_1$.

  Invoking the the inductive hypothesis we have $\Gamma \vdash_S e_0 : \tau_0$ and $\Gamma, x : \sigma_0 \vdash_S e_1 : \tau_1$, where $\overline{\Gamma}(\tau_0) \sqsubseteq \sigma_0$ and $\overline{\Gamma}(\tau_1) \sqsubseteq \sigma_1$.

  Applying the context generalization lemma (3.2.1), yields $\Gamma, x : \overline{\Gamma}(\tau_0) \vdash_S e_1 : \tau_1$. Putting this together:

$$\frac{\Gamma \vdash_S e_0 : \tau_0 \qquad \Gamma, x : \overline{\Gamma}(\tau_0) \vdash_S e_1 : \tau_1}{\Gamma \vdash_S \texttt{let}\ x = e_0\ \texttt{in}\ e_1 : \tau_1} \text{ SD-LET}$$

- Rules D-App and D-Abs are identical to their syntax directed counterparts. These cases are trivial.

QED

3.3 Soundness of the algorithmic relation

As for the syntax directed system, soundness of the algorithmic relation is easy to state: $\Gamma; \text{A} \vdash_W e \uparrow (S, \tau, \text{A}')$ implies $S\Gamma \vdash_S \tau$. We will perform induction directly on this statement, and find the following two lemmas helpful.

3.3.1 Renaming trick lemma

Lemma: $\forall \Gamma, S, \tau. \exists S'. \overline{S\Gamma}(S \circ S'\tau) \sqsubseteq S\overline{\Gamma}(\tau)$ and $S'(\Gamma) = \Gamma$.

Proof: By constructing S'.

Let $\overline{\gamma} = \text{ftv}(\tau) \setminus \text{ftv}(\Gamma)$

Pick disjoint sets of fresh variables $\overline{\delta}$ and $\overline{\zeta}$ such that $\delta_i, \zeta_i \notin \text{vars}(S) \cup \text{ftv}(\tau) \cup \text{ftv}(\Gamma)$ and $\delta_i \neq \zeta_j$.

By the definition of generalization:

$$S\overline{\Gamma}(\tau) = S\forall\overline{\gamma}.\tau = S\forall\overline{\zeta}.[\overline{\zeta}/\overline{\gamma}]\tau = \forall\overline{\zeta}.S \circ [\overline{\zeta}/\overline{\gamma}]\tau$$

Defining $S' = [\overline{\delta}/\overline{\gamma}]$ and applying the definition of generalization:

$$\overline{S\Gamma}(S \circ S'\tau) = \forall\overline{\delta\beta}.S \circ S'\tau$$

where $\beta_i \neq \delta_j$.

Because $\text{ftv}(\overline{S\Gamma}(SS'\tau)) \subseteq \overline{\delta} \cup \text{range}(S) \cup \text{ftv}(\tau)$,

$$\zeta_i \notin \text{ftv}(\overline{S\Gamma}(S \circ S'\tau))$$

and

$$\frac{\zeta_i \notin \text{ftv}(\overline{S\Gamma}(S \circ S'\tau)) \qquad S \circ [\overline{\zeta}/\overline{\gamma}]\tau = [\overline{\zeta}/\overline{\delta}] \circ S \circ [\overline{\delta}/\overline{\gamma}]\tau}{\overline{S\Gamma}(S \circ S'\tau) \sqsubseteq S\overline{\Gamma}(\tau)} \text{ Defn. of } \sqsubseteq$$

As $\text{vars}(S') \cap \text{ftv}(\Gamma) = \emptyset$, we also conclude $S'(\Gamma) = \Gamma$.

QED

3.3.2 Substitution Lemma

Lemma: $\Gamma \vdash_S e : \tau$ implies $S\Gamma \vdash_S e : S\tau$

Proof: By induction on the height of the judgment.

- SD-Var

$$\frac{x : \sigma \in \Gamma \qquad \sigma \sqsubseteq \tau}{\Gamma \vdash_S x : \tau}\ \text{SD-Var}$$

Noting $S\sigma \in S\Gamma$ and $S\sigma \sqsubseteq S\tau$, we use SD-Var to conclude $S\Gamma \vdash_S x : S\tau$.

- SD-Let

$$\frac{\Gamma \vdash_S e : \tau \qquad \Gamma,\, x : \overline{\Gamma}(\tau) \vdash_S e' : \tau'}{\Gamma \vdash_S \texttt{let}\ x = e\ \texttt{in}\ e' : \tau'}\ \text{SD-Let}$$

By the induction hypothesis:

$$S\Gamma,\, x : S\overline{\Gamma}(\tau) \vdash_S e' : S\tau'$$

However, we need a judgment of the form $S\Gamma,\, x : \overline{S\Gamma}(\tau_b) \vdash_S \tau'$

Picking S' as the existential witness to lemma 3.3.1 gives $\overline{S\Gamma}(S \circ S'\tau) \sqsubseteq S\overline{\Gamma}(\tau)$ and $S'(\Gamma) = \Gamma$.

Thus, by the context generalization lemma (3.2.1),

$$S\Gamma,\, \overline{S\Gamma}(S \circ S'\tau) \vdash_S S\tau'$$

By the induction hypothesis and $\Gamma = S'\Gamma$:

$$SS'\Gamma \vdash_S e : S \circ S'\tau$$

$$S\Gamma \vdash_S e : S \circ S'\tau$$

Applying SD-Let gives $S\Gamma \vdash_S \texttt{let}\ x = e\ \texttt{in}\ e' : S\tau'$

- SD-App and SD-Abs: These cases are trivial. The conclusions follow directly from the induction hypotheses.

QED

3.3.3 Proof of soundness of the algorithmic relation

We will show $\Gamma; \mathrm{A} \vdash_W e \uparrow (S, \tau, \mathrm{A}')$ implies $S\Gamma \vdash_S \tau$. Proof is by induction on the derivation of the algorithmic judgment. Case analysis on the final rule used:

- WA-Var: $e = x$, $\tau = [\overline{\beta}/\overline{\alpha}]\tau_0$ and $S = [-]$.

By the premise of WA-Var, we find $x : \forall\overline{\alpha}.\tau_0 \in \Gamma$

$$\frac{x : \forall\overline{\alpha}.\tau_0 \in \Gamma \qquad \dfrac{[\overline{\beta}/\overline{\alpha}]\tau_0 = [\overline{\beta}/\overline{\alpha}]\tau_0}{\forall\overline{\alpha}.\tau_0 \sqsubseteq [\overline{\beta}/\overline{\alpha}]\tau_0}\ \text{Defn. of } \sqsubseteq}{\Gamma \vdash_S [\overline{\beta}/\overline{\alpha}]\tau_0}\ \text{SD-Var}$$

As $\Gamma = [-]\Gamma$, this case is concluded.

- WA-Abs: $e = \lambda x.e_0$ and $\tau = S(\beta) \to \tau_0$

  By the premise of WA-Abs,

  $$\Gamma, x : \beta; \mathrm{AX} \vdash_W \lambda x.e_0 \uparrow (S, \tau_0, \mathrm{A})$$

  By the induction hypothesis and definition of substitution,

  $$S\Gamma, x : S\beta \vdash_S e_0 : \tau_0$$

  Applying SD-Abs to the above judgment, we derive:

  $$S\Gamma \vdash_S \lambda x.e_0 : S(\beta) \to \tau_0$$

- WA-Let: $e = \texttt{let}\ x = e_0\ \texttt{in}\ e_1$, S=$S_1 \circ S_2$

  The algorithmic derivation ends with

  $$\frac{\Gamma; \mathrm{AYX} \vdash_W e_0 \uparrow (S_0, \tau_0, \mathrm{AY}) \qquad S_0(\Gamma), x : \overline{S_0(\Gamma)}(\tau_0); \mathrm{AY} \vdash_W e_1 \uparrow (S_1, \tau_1, \mathrm{A})}{\Gamma; \mathrm{AYX} \vdash_W \texttt{let}\ x = e_0\ \texttt{in}\ e_1 \uparrow (S_1 \circ S_0, \tau_1, \mathrm{A})}\ \text{WA-Let}$$

  We want to show:

  $$S_1 S_0 \Gamma \vdash_S \texttt{let}\ x = e_0\ \texttt{in}\ e_1 : \tau_1$$

  This requires the following two judgments (for some τ_b):

  $$S_1 S_0 \Gamma \vdash_S e_0 : \tau_b$$

  $$S_1 S_0 \Gamma, x : \overline{S_1 S_0 \Gamma}(\tau_b) \vdash_S e_1 : \tau_1$$

  Observe that the inductive hypothesis gives

  $$S_0 \Gamma \vdash_S e_0 : \tau_0$$

  and

  $$S_1 S_0 \Gamma, x : S_1 \overline{S_0 \Gamma}(\tau_0) \vdash_S e_1 : \tau_1$$

  By lemma 3.3.1 we can pick $S^\diamond$ such that

  $$\overline{S_1 S_0 \Gamma}(S_1 S^\diamond \tau_0) \sqsubseteq S_1 \overline{S_0 \Gamma}(\tau_0)$$

  and $S^\diamond(S_0\Gamma) = S_0\Gamma$

  By the substitution lemma (lemma 3.3.2) and the first invocation of the induction hypothesis:

  $$S_1 S^\diamond S_0 \Gamma \vdash_S e_0 : S_1 S^\diamond \tau_0$$

  and

  $$S_1 S_0 \Gamma \vdash_S e_0 : S_1 S^\diamond \tau_0$$

  Applying context generalization (lemma 3.2.1) to the induction hypothesis's second claim gives

  $$S_1 S_0 \Gamma, x : \overline{S_1 S_0 \Gamma}(S_1 S^\diamond \tau_0) \vdash_S e : \tau_1$$

  Putting this all together gives

  $$\frac{S_1 S_0 \Gamma \vdash_S e_0 : S_1 S^\diamond \tau_0 \qquad S_1 S_0 \Gamma, x : \overline{S_1 S_0 \Gamma}(S_1 S^\diamond \tau_0) \vdash_S e : \tau_1}{S_1 S_0 \Gamma \vdash_S \texttt{let}\ x = e_0\ \texttt{in}\ e_1 \tau_1}\ \text{SD-Let}$$

- WA-App: $e = e_1 e_2$, S=$V \circ S_2 \circ S_1$ and $\tau = V(\beta)$

  From the premises of WA-App we have

  $$\Gamma; \mathrm{A}\beta\mathrm{YX} \vdash_W e_1 \uparrow (S_1, \tau_1, \mathrm{A}\beta\mathrm{Y})$$

  $$S_1\Gamma,\ \mathrm{A}\beta\mathrm{Y}; \vdash_W e_2 \uparrow (S_2, \tau_2, A\beta)$$

  and

  $$V = \mathcal{U}(S_2(\tau_1), \tau_2 \to \beta)$$

  Applying the induction hypothesis to the first two premises gives $S_1\Gamma \vdash_S e_1 : \tau_1$ and $S_2 S_1 \Gamma \vdash_S e_2 : \tau_2$

  Using the substitution lemma (3.3.2) on the above judgments yields: $VS_2S_1\Gamma \vdash_S e_1 : VS_2\tau_1$ and $VS_2S_1\Gamma \vdash_S e_2 : V\tau_2$

  From the definition of unification and the third WA-App premise:

  $$VS_2(\tau_1) = V(\tau_2 \to \beta) = V(\tau_2) \to V(\beta)$$

  Hence

  $$VS_2S_1\Gamma \vdash_S e_1 : V(\tau_2) \to V(\beta)$$

  and

  $$\frac{VS_2S_1\Gamma \vdash_S e_1 : V(\tau_2) \to V(\beta) \qquad VS_2S_1\Gamma \vdash_S e_2 : V(\tau_2)}{V \circ S_2 \circ S_1\Gamma \vdash_S e_1 e_2 : V(\beta)}\ \text{SD-App}$$

  QED

3.4 Completeness of the algorithmic system

We want to show that the algorithmic system is complete with respect to the syntax directed type system. Completeness here does not mean that we can algorithmically find any type valid in the syntax directed system. As the syntax directed system can assign many types to an expression and the algorithmic system can only assign one, this is clearly not possible. Instead, I claim that the Hindley-Milner algorithm assigns principal types; that is, any type given by the syntax directed system is a substitution of the type found algorithmically.

The claim above is too weak to use directly for induction, so I will prove that, given some Γ and e, for all substitutions Q,

$$\begin{aligned} Q\Gamma \vdash_S e : \tau' \implies \exists R.\forall \mathrm{AX}.\mathrm{AX} \cap (\mathrm{ftv}(\Gamma) \cup \mathrm{ftv}(\tau') \cup \mathrm{vars}(Q)) = \emptyset \qquad &\text{implies} \\ \text{(i)} \quad \Gamma; \mathrm{AX} \vdash_W e \uparrow (S, \tau, A) \\ \text{(ii)} \quad (R \circ S)|_{\neg \mathrm{X}} = Q \\ \text{(iii)} \quad \tau' = R(\tau) \end{aligned}$$

Although this statement is complicated, it makes sense. Given $Q\Gamma \vdash_S e : \tau$, we expect (but do not prove directly) $\Gamma \vdash_S e : \tau$, as $\Gamma \sqsubseteq Q\Gamma$. Thus claim (i) is reasonable. Claim (ii) is a technical detail required for the SD-Let case, and claim (iii) is the statement we actually want.

Additionally the main proof will require new lemmas. Lemmas 3.4.1 and 3.4.4 are used directly, while 3.4.2 and 3.4.3 needed to prove 3.4.4.

3.4.1 Algorithmic substitution variables lemma

Lemma: $\Gamma; \mathrm{A}\mathrm{X} \vdash_W e \uparrow (S, \tau, A)$ implies $\mathrm{ftv}(\tau) \cup \mathrm{vars}(S) \subseteq \mathrm{ftv}(\Gamma) \cup \mathrm{X}$

Proof: By induction on the height of the algorithmic derivation.

- WA-Var:

$$\frac{x : \forall \overline{\alpha}.\tau \in \Gamma}{\Gamma; \mathrm{A}\overline{\beta} \vdash_W e \uparrow (\emptyset, [\overline{\beta}/\overline{\alpha}]\tau, \mathrm{A})}\ \text{WA-Var}$$

$$\mathrm{vars}([-]) \cup \mathrm{ftv}([\overline{\beta}/\overline{\alpha}]\tau) = (\mathrm{ftv}(\tau) \setminus \overline{\alpha}) \cup \overline{\beta} = \mathrm{ftv}(\forall \overline{\alpha}.\tau) \cup \overline{\beta} \subseteq \mathrm{ftv}(\Gamma) \cup \overline{\beta}$$

- WA-App:

$$\frac{\Gamma; \mathrm{A}\beta\mathrm{Y}\mathrm{X} \vdash_W e_1 \uparrow (S_1, \tau_1, \mathrm{A}\beta\mathrm{Y}) \qquad S_1(\Gamma); \mathrm{A}\beta\mathrm{Y} \vdash_W e_2 \uparrow (S_2, \tau_2, \mathrm{A}\beta) \qquad V = \mathcal{U}(S_2\tau_1, \tau_2 \to \beta)}{\Gamma; \mathrm{A}\beta\mathrm{Y}\mathrm{X} \vdash_W e_1 e_2 \uparrow (V \circ S_2 \circ S_1, V(\beta), \mathrm{A})}\ \text{WA-App}$$

By the induction hypothesis,

$$\mathrm{ftv}(\tau_1) \cup \mathrm{vars}(S_1) \subseteq \mathrm{ftv}(\Gamma) \cup \mathrm{X}$$

$$\mathrm{ftv}(\tau_2) \cup \mathrm{vars}(S_2) \subseteq \mathrm{ftv}(S_1\Gamma) \cup \mathrm{Y} \subseteq \mathrm{ftv}(\Gamma) \cup \mathrm{XY}$$

Hence $\mathrm{ftv}(S_2(\tau_1)) \subseteq \mathrm{ftv}(\Gamma) \cup \mathrm{XY}$.

By the definition of unification, $\mathrm{vars}(V) \subseteq \mathrm{ftv}(S_2(\tau_1)) \cup \mathrm{ftv}(\tau_2) \cup \{\beta\}$.

Therefore $\mathrm{ftv}(V(\beta)) \cup \mathrm{vars}(V \circ S_2 \circ S_1) \subseteq \mathrm{ftv}(\Gamma) \cup \mathrm{XY}\beta$.

- WA-Abs:

$$\frac{\Gamma,\ x : \beta; \mathrm{A}\mathrm{X} \vdash_W e \uparrow (S, \tau, \mathrm{A}\mathrm{X})}{\Gamma; \mathrm{A}\mathrm{X}\beta \vdash_W \lambda x.e \uparrow (S, S(\beta) \to \tau, \mathrm{A})}\ \text{WA-Abs}$$

By the induction hypothesis,

$$\mathrm{vars}(S) \cup \mathrm{ftv}(\tau) \subseteq \mathrm{ftv}(\Gamma,\ x : \beta) \cup \mathrm{X} \subseteq \mathrm{ftv}(\Gamma) \cup \mathrm{X}\beta$$

Therefore

$$\mathrm{vars}(S) \cup \mathrm{ftv}(S(\beta) \to \tau) \subseteq \mathrm{ftv}(\Gamma) \cup \mathrm{X}\beta$$

- WA-Let:

$$\frac{\Gamma; \mathrm{A}\mathrm{Y}\mathrm{X} \vdash_W e_0 \uparrow (S_0, \tau_0, \mathrm{A}\mathrm{Y}) \qquad S_0(\Gamma), x : \overline{S_0(\Gamma)}(\tau_0); \mathrm{A}\mathrm{Y} \vdash_W e_1 \uparrow (S_1, \tau_1, \mathrm{A})}{\Gamma; \mathrm{A}\mathrm{Y}\mathrm{X} \vdash_W \mathtt{let}\ x = e_0\ \mathtt{in}\ e_1 \uparrow (S_1 \circ S_0, \tau_1, \mathrm{A})}\ \text{WA-Let}$$

By the induction hypothesis,

$$\mathrm{ftv}(\tau_0) \cup \mathrm{vars}(S_0) \subseteq \mathrm{ftv}(\Gamma) \cup \mathrm{X}$$

$$\mathrm{ftv}(\tau_1) \cup \mathrm{vars}(S_1) \subseteq \mathrm{ftv}(S_0\Gamma,\ x : \overline{S_0\Gamma}(\tau_0)) \cup \mathrm{XY} \subseteq \mathrm{ftv}(\Gamma) \cup \mathrm{vars}(S_0) \cup \mathrm{ftv}(\tau_0) \cup \mathrm{XY} \subseteq \mathrm{ftv}(\Gamma) \cup \mathrm{XY}$$

Therefore

$$\mathrm{vars}(S_0 \circ S_1) \cup \mathrm{ftv}(\tau_1) \subseteq \mathrm{ftv}(\Gamma) \cup \mathrm{XY}$$

QED

3.4.2 Lemma: Substitution of renamed variables

Lemma: Assume $\overline{\delta}$ is contains fresh, pairwise-disjoint type variables—that is $\delta_i \notin \text{vars}(S) \cup \text{ftv}(\tau)$ and $\delta_i = \delta_j$ only when $i = j$. Then

$$S\tau = [S(\overline{\alpha})/\overline{\delta}] \circ S \circ [\overline{\delta}/\overline{\alpha}]\tau.$$

Proof: Proof is by induction on structure of τ.

- Case $\tau = x$
  - Subcase $x = \alpha_i \in \overline{\alpha}$

$$\begin{aligned}
S\tau &= S\alpha_i \\
&= [S(\alpha_i)/\delta_i]\delta_i \\
&= [S(\overline{\alpha})/\overline{\delta}]\delta_i && \text{as elements } \overline{\delta} \text{ disjoint} \\
&= [S(\overline{\alpha})/\overline{\delta}] \circ S\delta_i && \text{as } \delta_i \notin \text{vars}(S) \\
&= [S(\overline{\alpha})/\overline{\delta}] \circ S \circ [\overline{\delta}/\overline{\alpha}]\alpha_i \\
&= [S(\overline{\alpha})/\overline{\delta}] \circ S \circ [\overline{\delta}/\overline{\alpha}]\tau
\end{aligned}$$

  - Subcase $x \notin \overline{\alpha}$

$$\begin{aligned}
S\tau &= Sx \\
&= [S(\overline{\alpha})/\overline{\delta}] \circ Sx && \text{by freshness of } \overline{\delta} \\
&= [S(\overline{\alpha})/\overline{\delta}] \circ S \circ [\overline{\delta}/\overline{\alpha}]x && \text{as } x \notin \overline{\alpha}
\end{aligned}$$

- Case $\tau = \tau_1 \to \tau_2$

  Immediate from the induction hypotheses.

QED

3.4.3 Lemma: Free variables of an applied substitution

Lemma: $\text{ftv}(S\tau) = \cup\{\text{ftv}(Sx) \mid x \in \text{ftv}(\tau)\}$.

Proof: Proof is by induction on the structure of τ.

- Case $\tau = y$

$$\text{ftv}(S\tau) = \text{ftv}(Sy) = \cup\{\text{ftv}(Sy)\} = \cup\{\text{ftv}(Sx) \mid x \in \{y\}\} = \cup\{\text{ftv}(Sx) \mid x \in \text{ftv}(\tau)\}$$

- Case $\tau = \tau_1 \to \tau_2$

$$\begin{aligned}
\text{ftv}(S(\tau_1 \to \tau_2)) &= \text{ftv}(S\tau_1 \to S\tau_2) \\
&= \text{ftv}(S\tau_1) \to \text{ftv}(S\tau_2) \\
&= (\cup\{\text{ftv}(Sx) \mid x \in \text{ftv}(\tau_1)\}) \cup (\cup\{\text{ftv}(Sx) \mid x \in \text{ftv}(\tau_2)\}) && \text{by the induction hypothesis} \\
&= \cup\{\text{ftv}(Sx) \mid x \in \text{ftv}(\tau_1) \cup \text{ftv}(\tau_2)\} \\
&= \cup\{\text{ftv}(Sx) \mid x \in \text{ftv}(\tau_1 \to \tau_2)\}
\end{aligned}$$

QED

3.4.4 Lemma: Substitution through generalization

Lemma: $S\overline{\Gamma}(\tau) \sqsubseteq \overline{S\Gamma}(S\tau)$

Proof: Begin by rewriting the above types.

$$\begin{aligned} S\overline{\Gamma}(\tau) &= S(\forall\overline{\alpha}.\tau) \\ &= \forall\overline{\delta}.S \circ [\overline{\delta}/\overline{\alpha}]\tau \end{aligned}$$

$$\overline{S\Gamma}(S\tau) = \forall\overline{\beta}.S\tau$$

where $\overline{\alpha} = \text{ftv}(\tau) \setminus \text{ftv}(\Gamma)$ and $\overline{\beta} = \text{ftv}(S\tau) \setminus \text{ftv}(S\Gamma)$. The $\overline{\delta}$ are defined to be fresh and pairwise disjoint—$\delta_i \notin \text{ftv}(\tau) \cup \text{vars}(S) \cup \text{ftv}(\Gamma)$ and $\delta_i = \delta_j$ only when $i = j$.

By the definition of $\sqsubseteq$ it suffices to show the following.

(i) $\beta_i \notin \text{ftv}(\forall\overline{\delta}.S \circ [\overline{\delta}/\overline{\alpha}]\tau)$, for every $\beta_i \in \overline{\beta}$

(ii) $S\tau = [\overline{\tau''}/\overline{\delta}] \circ S \circ [\overline{\delta}/\overline{\alpha}]\tau$, for some sequence, $\overline{\tau''}$.

Proposition (ii) is immediate from Lemma 3.4.2.

We now demonstrate (i).

Suppose $\overline{\beta} = \emptyset$. Then (i) is vacuously true and we're done. Otherwise consider an arbitrary $\beta_i \in \overline{\beta}$. We can conservatively approximate the free variables of $\forall\overline{\delta}.S \circ [\overline{\delta}/\overline{\alpha}]\tau$ as follows.

$$\begin{aligned} \text{ftv}(\forall\overline{\delta}.S \circ [\overline{\delta}/\overline{\alpha}]\tau) &= \text{ftv}(S \circ [\overline{\delta}/\overline{\alpha}]\tau) \setminus \overline{\delta} \\ &= \text{ftv}(S([\overline{\delta}/\overline{\alpha}]\tau)) \setminus \overline{\delta} \\ &= \cup\{\text{ftv}(Sx) \mid x \in \text{ftv}([\overline{\delta}/\overline{\alpha}]\tau)\} \setminus \overline{\delta} && \text{by Lemma 3.4.3} \\ &\subseteq \cup\{\text{ftv}(Sx) \mid x \in (\text{ftv}(\tau) \setminus \overline{\alpha}) \cup \overline{\delta}\} \setminus \overline{\delta} \\ &= (\cup\{\text{ftv}(Sx) \mid x \in (\text{ftv}(\tau) \setminus \overline{\alpha})\}) \cup (\cup\{\text{ftv}(Sx) \mid x \in \overline{\delta}\}) \setminus \overline{\delta} \\ &= \cup\{\text{ftv}(Sx) \mid x \in (\text{ftv}(\tau) \setminus \overline{\alpha})\} \cup \overline{\delta} \setminus \overline{\delta} && \text{as } \overline{\delta} \text{ fresh from } S \\ &\subseteq \cup\{\text{ftv}(Sx) \mid x \in (\text{ftv}(\tau) \setminus \overline{\alpha})\} \end{aligned}$$

It suffices to show $\beta_i \notin \cup\{\text{ftv}(Sx) \mid x \in (\text{ftv}(\tau) \setminus \overline{\alpha})\}$.

For a contradiction, assume β_i is in the set. Then, for some x, $\beta_i \in \text{ftv}(Sx)$ and $x \in (\text{ftv}(\tau) \setminus \overline{\alpha})$. Expanding the definition of $\overline{\alpha}$ gives $x \in \text{ftv}(\tau) \setminus (\text{ftv}(\tau) \setminus \text{ftv}(\Gamma))$. So $x \in \text{ftv}(\Gamma)$, and $\text{ftv}(Sx) \subseteq \text{ftv}(S\Gamma)$. As $\beta_i \in \text{ftv}(Sx)$, we see $\beta_i \in \text{ftv}(S\Gamma)$. This is a contradiction with the definition of $\overline{\beta}$.

QED

3.4.5 Proof of completeness for the algorithmic system

$$\forall\Gamma, e.\forall Q.Q\Gamma \vdash_S e : \tau' \implies \exists R.\forall \text{AX}.\text{AX} \cap (\text{ftv}(\Gamma) \cup \text{ftv}(\tau') \cup \text{vars}(Q)) = \emptyset \qquad \text{implies}$$

(i) $\Gamma; \text{AX} \vdash_W e \uparrow (S, \tau, A)$

(ii) $(R \circ S)|_{\neg \text{X}} = Q$

(iii) $\tau' = R(\tau)$

Proof will be by induction on the height of the syntax directed judgment. We proceed with case analysis on the final step of the proof tree.

- SD-Var Given $e = x$, $S = [-]$ and $\Gamma = \Gamma_0, x : \forall\overline{\alpha'}.\tau_0'$

  The syntax directed judgment looks like

$$\frac{x : Q\forall\overline{\alpha'}.\tau_0' \in Q\Gamma \qquad Q\forall\overline{\alpha'}.\tau_0' \sqsubseteq \tau'}{Q\Gamma \vdash_S x : \tau'} \text{ SD-Var}$$

Let $\forall\overline{\alpha}.\tau_0 = \forall\overline{\alpha'}.\tau_0'$ where $\alpha_i \notin \text{vars}(Q)$. Therefore $\forall\overline{\alpha}.Q\tau_0 = Q\forall\overline{\alpha'}$ and

$$x : \forall\overline{\alpha}.Q\tau_0 \in Q\Gamma$$

$$\forall\overline{\alpha}.Q\tau_0 \sqsubseteq \tau'$$

(i) Algorithm succeeds

$$\frac{x : \forall\overline{\alpha}.\tau_0 \in \Gamma}{\Gamma; \text{A}\overline{\beta} \vdash_W x \uparrow ([-], [\overline{\beta}/\overline{\alpha}], \text{A})} \text{ SD-Var}$$

(ii) $\exists R.R \circ S|_{\neg\overline{\beta}} = Q$
We need to find a substitution R, to fulfill both the above condition and requirement (iii).
Inverting the proof that $\forall\overline{\alpha}.Q\tau_0 \sqsubseteq \tau'$ gives $\tau' = [\overline{\tau}/\overline{\alpha}]\tau_0$.
Propose $R = [\overline{\tau}/\overline{\beta}] \circ Q$
Noting $\beta_i \notin \text{vars}(Q)$:

$$R \circ S|_{\neg\overline{\beta}} = [\overline{\tau}/\overline{\beta}] \circ Q \circ [-]|_{\neg\overline{\beta}} = Q \circ [\overline{\tau}/\overline{\beta}]|_{\neg\overline{\beta}} = Q$$

(iii) Want to show $R[\overline{\beta}/\overline{\alpha}]\tau_0 = \tau'$

$$\begin{aligned} R[\overline{\beta}/\overline{\alpha}]\tau_0 &= [\overline{\tau}/\overline{\beta}]Q[\overline{\beta}/\overline{\alpha}]\tau_0 \\ &= [\overline{\tau}/\overline{\beta}][\overline{\beta}/\overline{\alpha}]Q\tau_0 \\ &= [\overline{\tau}/\overline{\alpha}]Q\tau_0 \\ &= \tau' \end{aligned}$$

- SD-Abs

The syntax directed judgment ends in

$$\frac{Q\Gamma,\, x : \tau \vdash_S e : \tau'}{Q\Gamma \vdash_S \lambda x.e : \tau \to \tau'} \text{ SD-Abs}$$

(i) We want to show the algorithm succeeds for term $\lambda x.e$ in context Γ.
Pick a tape satisfying the assumptions, $\text{AX}\beta$, and let $Q' = [\tau/\beta] \circ Q$. As $\beta \notin \text{range}(Q)$, we know $Q'(\Gamma,\, x : \beta) = \Gamma, x : \tau$ and

$$Q'(\Gamma,\, x : \beta) \vdash_S e : \tau'$$

with a derivation identical to that in the premise of the given instance of SD-Abs.
Apply the induction hypothesis to find:

$$\Gamma,\, x : \beta; \text{AX} \vdash_W (S, \tau'^{\text{w}}, \text{A})$$

as well as $\exists R.R \circ S|_{\neg\text{X}}$ and $\tau' = R\tau'^{\text{w}}$.
Hence, by WA-Abs,

$$\Gamma; \text{AX}\beta \vdash_W \lambda x.e \uparrow (S, S(\beta) \to \tau'^{w}, \text{A})$$

(ii) Want to show $\exists R.R \circ S|_{\neg\text{X}\beta} = Q$. Let R be a witness to the existential statement in (i).
By the I.H. we have

$$R \circ S|_{\neg\text{X}} = Q' = [\tau/\beta] \circ Q$$

Therefore

$$R \circ S|_{\neg\text{X}\beta} = Q$$

(iii) Want to show $R(S(\beta) \to \tau'^{\mathrm{w}}) = \tau \to \tau'$
From $R \circ S|_{\neg \mathrm{X}} = Q'$, we see $RS(\beta) = Q'(\beta) = \tau$
By the inductive hypothesis, $R(\tau'^{\mathrm{w}}) = \tau'$
Therefore $R(S(\beta) \to \tau'^{\mathrm{w}}) = \tau \to \tau'$

- SD-Let: The syntax directed judgment ends with:

$$\frac{Q\Gamma \vdash_S e_0 : \tau_0 \qquad Q\Gamma,\, x : \overline{Q\Gamma}(\tau_0) \vdash_S e_1 : \tau_1}{Q\Gamma \vdash_S \mathtt{let}\, x = e_0\, \mathtt{in}\, e_1 : \tau_1}\ \text{SD-Let}$$

(i) Algorithm succeeds
Applying the inductive hypothesis to the left hand premise gives

$$\Gamma; A\mathrm{XY} \vdash_W e_0 \uparrow (S_0, \tau_0^{\mathrm{w}}, \mathrm{AY})$$

where $R \circ S_0|_{\neg Y} = Q$ and $\tau_0 = R(\tau_0^{\mathrm{w}})$.
We want to use WA-Let:

$$\frac{\Gamma; \mathrm{AXY} \vdash_W e_0 \uparrow (S_0, \tau_0^{\mathrm{w}}, \mathrm{AY}) \qquad S_0\Gamma,\, x : \overline{S_0\Gamma}; \mathrm{AX} \vdash_W e_1 \uparrow (S_1, \tau_1^{\mathrm{w}}, \mathrm{A})}{\Gamma; \mathrm{AXY} \vdash_W \mathtt{let}\, x = e_0\, \mathtt{in}\, e_1 \uparrow (S_1 \circ S_0, \tau_1^{\mathrm{w}}, \mathrm{A})}\ \text{WA-Let}$$

While the left premise followed directly from our induction hypothesis, the right premise will require a little more work.
By the premises

$$Q\Gamma, x : \overline{Q\Gamma}(\tau_0) \vdash_S e_1 : \tau_1$$

Using the inductive hypothesis, we substitute $R \circ S_0$ for Q and $R\tau_0^{\mathrm{w}}$ for τ_0

$$R \circ S_0\Gamma, x : \overline{R \circ S_0\Gamma}(R\tau_0^{\mathrm{w}}) \vdash_S e_1 : \tau_1$$

By lemma 3.4.4, $R\overline{S_0\Gamma}(\tau_0^{\mathrm{w}}) \sqsubseteq \overline{R \circ S_0\Gamma}(R\tau_0^{\mathrm{w}})$ and by context generalization (lemma 3.2.1) we have

$$R \circ S_0\Gamma, x : R\overline{S_0\Gamma}(\tau_0^{\mathrm{w}}) \vdash_S e_1 : \tau_1$$

Applying the inductive hypothesis (with context $S_0\Gamma$ and substitution R) gives

$$S_0\Gamma, x : \overline{S_0\Gamma}(\tau_0^{\mathrm{w}}); \mathrm{AX} \vdash_W e_1 \uparrow (S_1, \tau_1^{\mathrm{w}}, \mathrm{A})$$

and R' such that $R' \circ S_1|_{\neg \mathrm{X}} = R$ and $\tau_1 = R'\tau_1^{\mathrm{w}}$
Immediately we use WA-Let to conclude

$$\Gamma; \mathrm{AXY} \vdash_W \mathtt{let}\, x = e_0\, \mathtt{in}\, e_1 \uparrow (S_1 \circ S_0, \tau_1^{\mathrm{w}}, \mathrm{A})$$

(ii) Picking R' as our existential witness, want to show $R' \circ (S_1 \circ S_0)|_{\neg \mathrm{XY}} = Q$.
Note that by lemma 3.4.1, $\mathrm{vars}(S_0) \subseteq \mathrm{ftv}(\Gamma) \cup \mathrm{Y}$, so $((R' \circ S_1) \circ S_0)|_{\neg \mathrm{X}} = (R' \circ S_1)|_{\neg \mathrm{X}} \circ S_0$.

$$\begin{aligned}
(R' \circ (S_1 \circ S_0))|_{\neg \mathrm{XY}} &= ((R' \circ S_1) \circ S_0)|_{\neg \mathrm{XY}} \\
&= ((R' \circ S_1)|_{\neg \mathrm{X}} \circ S_0)|_{\neg \mathrm{Y}} \\
&= (R \circ S_0)|_{\neg Y} \\
&= Q
\end{aligned}$$

(iii) $\tau_1 = R'(\tau_1^{\mathrm{w}})$ by the inductive hypothesis.

- SD-App

$$\frac{Q\Gamma \vdash_S e_1 : \tau_2 \to \tau \qquad Q\Gamma \vdash_S e_2 : \tau_2}{Q\Gamma \vdash_S e_1 e_2 : \tau}\ \text{SD-App}$$

(i) Want to show that the algorithm succeeds
Applying the induction hypothesis to left premise gives

$$\Gamma; \mathrm{A}\beta\mathrm{YX} \vdash_W e_1 \uparrow (S_1, \tau_1^{\mathrm{w}}, \mathrm{A}\beta\mathrm{Y})$$

with $R_1 \circ S_1|_{\neg \mathrm{X}} = Q$ and $\tau_2 \to \tau = R_1\tau_1^{\mathrm{w}}$.
Now we can rewrite the right premise as $R_1 S_1 \Gamma \vdash_S e_2 : \tau_2$ and use the inductive hypothesis to find

$$S_1\Gamma; \mathrm{A}\beta\mathrm{Y} \vdash_W e_2 \uparrow (S_2, \tau_2^{\mathrm{w}}, \mathrm{A}\beta)$$

with $R_2 \circ S_2|_{\neg \mathrm{Y}} = R_1$ and $\tau_2 = R_2\tau_2^{\mathrm{w}}$.
If we can unify $S_2\tau_1^{\mathrm{w}}$ and $\tau_2^{\mathrm{w}} \to \beta$, we will be able to use WA-App. Note that (by lemma 3.4.1), $\mathrm{ftv}(\tau_1^{\mathrm{w}}) \cap \beta\mathrm{Y} = \emptyset$ and $\beta \notin \mathrm{ftv}(\tau_2^{\mathrm{w}})$.
As $R_2 \circ S_2|_{\neg \mathrm{Y}} \circ S_1|_{\neg \mathrm{X}}(\beta) = Q(\beta) = \beta$ and $\beta \notin \mathrm{X} \cup \mathrm{Y} \cup \mathrm{vars}(S_1) \cup \mathrm{vars}(S_2)$, we know $R_2(\beta) = \beta$.
Using the above, we derive:

$$[\tau/\beta]R_2 S_2 \tau_1^{\mathrm{w}} = [\tau/\beta]R_1\tau_1^{\mathrm{w}} = [\tau/\beta](\tau_2 \to \tau) = \tau_2 \to \tau$$

and

$$[\tau/\beta]R_2(\tau_2^{\mathrm{w}} \to \beta) = [\tau/\beta]R_2\tau_2^{\mathrm{w}} \to [\tau/\beta]R_2\beta = [\tau/\beta]\tau_2 \to [\tau/\beta]\beta = \tau_2 \to \tau$$

Therefore these types can be unified. As unification returns a most general unifier, $V = \mathcal{U}(S\tau_1^{\mathrm{w}}, \tau_2^{\mathrm{w}} \to \beta)$ implies there exists R^* such that $[\tau/\beta]R_2 = R^* \circ V$.
Finally, we conclude

$$\Gamma; \mathrm{A}\beta\mathrm{YX} \vdash_W e_1 e_2 \uparrow (S_1, V(\beta), \mathrm{A})$$

(ii) Picking R^* as the existential witness for claims (i) and (ii), we want to show $R^* \circ V \circ S_2 \circ S_1|_{\neg\beta\mathrm{XY}} = Q$.

$$\begin{aligned} R^* \circ V \circ S_2 \circ S_1|_{\neg\beta\mathrm{XY}} &= [\tau/\beta] \circ R_2 \circ S_2 \circ S_1|_{\neg\beta\mathrm{XY}} \\ &= [\tau/\beta] \circ (R_2 \circ S_2)|_{\neg \mathrm{Y}} \circ S_1|_{\neg\mathrm{X}\beta} \\ &= [\tau/\beta] \circ R_1 \circ S_1|_{\neg\mathrm{X}\beta} \\ &= [\tau/\beta] \circ Q|_{\neg\beta} \\ &= Q \end{aligned}$$

(iii) We want to show $R^* \circ V(\beta) = \tau$.

$$R^* \circ V(\beta) = [\tau/\beta] \circ R_2(\beta) = [\tau/\beta](\beta) = \tau$$

4 Conclusion

Combining our proofs of soundness gives

$$\Gamma; \mathrm{A} \vdash_W e \uparrow (S, \tau, \mathrm{A}') \Rightarrow \Gamma \vdash_D e : \tau$$

Likewise, chaining completeness yields

$$\Gamma; \vdash_D e : \sigma \Rightarrow \Gamma; \mathrm{A} \vdash_W e \uparrow (S, \tau, \mathrm{A}')$$

where $\overline{\Gamma}(\tau) \sqsubseteq \sigma$. Hence, up to type generalization, the Hindley-Milner and algorithmic systems are equivalent.

5 Acknowledgments

Dimitrios Vytiniotis provided invaluable guidance on substitution semantics and their roll in tricky inductive proofs. Christian Urban pointed out a bug in the original proof of Lemma 3.4.4.