1 Introduction

!!! These rules are WIP - the ones that come directly from *Complete and Easy* should be right, but the rest have not been proven !!!

We present a type inference and checking system based off of the one in *Complete and Easy Bidirectional Typechecking for Higher-Rank Polymorphism* by Dunfield and Krishnaswami. This system is extended to encompass record types.

Future work may include extending to full(er) subtyping, as well as variant types.

2 Conventions

We sometimes use β and γ for type variables in addition to α. We adopt the same conventions Dunfield and Krishnaswami use for their typing rules.

In record types, we elide the final empty row type $(\cdot)$; e.g. we write $\{\ell_0 : A_0, \ell_1 : A_1\}$ instead of $\{\ell_0 : A_0, \ell_1 : A_1, \cdot\}$.

3 Extended Declarative Type System

An extension of *Complete and Easy*'s declarative type system.

3.1 Terms

Terms	e	$::=$	x	Variable
		$\mid$	$()$	1
		$\mid$	$\{\}$	Empty record
		$\mid$	$\lambda x \to e$	Lambda
		$\mid$	$e\ e$	Application
		$\mid$	$(e : A)$	Type annotation
		$\mid$	$\{\ell_1 : e_1, \ldots, \ell_k : e_k\}$	Record
		$\mid$	$e.\ell$	(Record) Projection
		$\mid$	(e)	Parentheses
Labels	ℓ	$::=$	$\{abcdefghijklmnopqrstuvwxyz\}^+$	

3.2 Types

$$
\begin{array}{llcl}
\text{Base types} & \mathcal{B} & ::= & 1 \\
 & & | & \texttt{Num} \\
 & & | & \texttt{Bool} \\
 & & | & \ldots \\
\text{Types} & A, B, C & ::= & \mathcal{B} \\
 & & | & \alpha \\
 & & | & \forall \alpha . A \\
 & & | & \text{Ɐ} \alpha_\rho . A \\
 & & | & A \to B \\
 & & | & \{R\} \\
\text{Monotypes} & \tau, \sigma & ::= & \mathcal{B} \\
 & & | & \alpha \\
 & & | & \tau \to \sigma \\
 & & | & \{\rho\} \\
\text{Rows} & R & ::= & \cdot \\
 & & | & \alpha_\rho \\
 & & | & \ell : A, R \\
\text{Monorows} & \rho & ::= & \cdot \\
 & & | & \ell : \tau, \rho
\end{array}
$$

3.2.1 A note on quantifiers

We distinguish $\forall$ from Ɐ, where the former is a quantifier over type variables and the latter is a quantifier over row variables.

3.3 Contexts

$$
\begin{array}{lcl}
\Psi & ::= & \cdot \\
 & | & \Psi, \alpha \\
 & | & \Psi, \alpha_\rho \\
 & | & \Psi, x : A
\end{array}
$$

3.4 Well-formedness

The judgement $\Psi \vdash A$ asserts that in the context Ψ, A is well-formed. We will make an abuse of notation and have $\Psi \vdash R$ apply to rows as well, even though they are not really types.

$$\frac{\alpha \in \Psi}{\Psi \vdash \alpha}\ (\text{DeclUVarWF}) \qquad \frac{}{\Psi \vdash \mathcal{B}}\ (\text{DeclBaseWF}) \qquad \frac{\Psi \vdash A \qquad \Psi \vdash B}{\Psi \vdash A \to B}\ (\text{DeclArrowWF})$$

$$\frac{\Psi, \alpha \vdash A}{\Psi \vdash \forall \alpha . A}\ (\text{DeclForallWF}) \qquad \frac{\Psi, \alpha_\rho \vdash A}{\Psi \vdash \text{Ɐ} \alpha_\rho . A}\ (\text{DeclForallRowWF}) \qquad \frac{\Psi \vdash R}{\Psi \vdash \{R\}}\ (\text{DeclRecordWF})$$

$$\frac{}{\Psi \vdash \cdot}\ (\text{DeclRowNilWF}) \qquad \frac{\alpha_\rho \in \Psi}{\Psi \vdash \alpha_\rho}\ (\text{DeclRowVarWF}) \qquad \frac{\Psi \vdash A \qquad \Psi \vdash R}{\Psi \vdash \ell : A, R}\ (\text{DeclRowWF})$$

3.5 Subtyping

We'll also make an abuse of notation and have the judgement $\Psi \vdash R_1 \leq R_2$ be valid for subtyping.
TODO: add the rules from complete and easy.

$$\frac{}{\Psi \vdash \cdot \leq \cdot}\ (\leq \text{RowNil}) \qquad \frac{\alpha_\rho \in \Psi}{\Psi \vdash \alpha_\rho \leq \alpha_\rho}\ (\leq \text{RowVar})$$

We treat the rows sort of as sets and reorder the labels appropriately to match them.

$$\frac{\Psi \vdash A \leq B \qquad \Psi \vdash R_1 \leq R_2}{\Psi \vdash \ell : A, R_1 \leq \ell : B, R_2}\ (\leq \text{Row}) \qquad \frac{\Psi \vdash R \leq_1 R_2}{\Psi \vdash \{R_1\} \leq \{R_2\}}\ (\leq \text{Rcd})$$

$$\frac{\Psi \vdash \rho \qquad \Psi \vdash [\alpha_\rho \mapsto \rho]\{R_1\} \leq \{R_2\}}{\Psi \vdash \text{ꓤ}\alpha_\rho.\{R_1\} \leq \{R_2\}}\ (\leq \text{ꓤL Rcd}) \qquad \frac{\Psi \vdash \{R_1\} \qquad \Psi, \beta \vdash \{R_1\} \leq \{R_2\}}{\Psi \vdash \{R_1\} \leq \text{ꓤ}\alpha_\rho.\{R_2\}}\ (\leq \text{ꓤR Rcd})$$

3.6 Declarative Typing

3.6.1 Judgements

$\Psi \vdash e \Leftarrow A$, Under context Ψ, e checks against type A.
$\Psi \vdash e \Rightarrow A$, Under context Ψ, e synthesizes type A.
$\Psi \vdash A \bullet e \mathrel{\Rightarrow\!\!\!\Rightarrow} C$, Under context Ψ type A applied to e synthesizes type C.
$\Psi \vdash A\#\ell \longrightarrow C$, under context Ψ type A projects type C at ℓ.

3.6.2 Rules

!!! All of the declarative rules from *Complete and Easy* still apply, though they have yet to be added. !!!

$$\frac{\Psi \vdash e \Leftarrow A \qquad \Psi \vdash r \Leftarrow \{R\}}{\Psi \vdash \{\ell : e, r\} \Leftarrow \{\ell : A, R\}}\ (\text{Decl RcdI}) \qquad \frac{\Psi \vdash e \Rightarrow A \qquad \Psi \vdash r \Rightarrow \{R\}}{\Psi \vdash \{\ell : e, r\} \Rightarrow \{\ell : A, R\}}\ (\text{Decl RcdI} \Rightarrow)$$

$$\frac{\Psi, \alpha_\rho \vdash e \Leftarrow A}{\Psi \vdash e \Leftarrow \text{ꓤ}\alpha_\rho.A}\ (\text{Decl ꓤI}) \qquad \frac{\Psi \vdash e \Rightarrow A \qquad \Psi \vdash A\#\ell \longrightarrow C}{\Psi \vdash e.\ell \Rightarrow C}\ (\text{DeclPrjI})$$

$$\frac{}{\Psi \vdash \{\ell : C, R\}\#\ell \longrightarrow C}\ (\text{Decl Lookup}) \qquad \frac{\Psi \vdash \rho \qquad \Psi \vdash [\alpha_\rho \mapsto \rho]A\#\ell \longrightarrow C}{\Psi \vdash \text{ꓤ}\alpha_\rho.A\#\ell \longrightarrow C}\ (\text{Decl ꓤ Lookup})$$

$$\frac{\Psi \vdash \tau \qquad \Psi \vdash [\alpha \mapsto \tau]A\#\ell \longrightarrow C}{\Psi \vdash \forall\alpha.A\#\ell \longrightarrow C}\ (\text{Decl } \forall \text{ Lookup})$$

4 Algorithmic Typing

4.1 Notes

The algorithmic typing, unlike the declarative typing, presently doesn't distinguish between rowvars and evars. The two are very similar in nature, and so we conflate them in the typing rules. It's just that rowvars can only be resolved to rows. This subtle difference is partially reflected in the implementaiton of the language: rowvars and evars are separate; however, quantifiers are not.
For convenience's sake, these rules will not presently distinguish between rowvars and evars, although this may change.

4.2 Terms

Terms are the same as in the declarative system.

4.3 Types

The main difference in types for the Algorithmic System is the addition of evars ($\hat{\alpha}$). There are some notational differences too, as a result.

$$
\begin{array}{rcl}
\mathcal{B} & ::= & \texttt{Num} \mid \texttt{Bool} \mid \cdots \\
A & ::= & \mathcal{B} \mid \hat{\alpha} \mid \alpha \mid \forall \alpha . A \mid A_0 \to A_1 \mid \{R\} \\
\tau & ::= & \mathcal{B} \mid \hat{\alpha} \mid \alpha \mid \tau_0 \to \tau_1 \mid \{\rho\} \\
R & ::= & \cdot \mid \hat{\alpha} \mid \ell : A, R \\
r & ::= & \cdot \mid \ell : A, r \\
\rho & ::= & \cdot \mid \hat{\alpha} \mid \ell : \tau, \rho
\end{array}
$$

4.4 Typing Rules

We define algorithmic typing with the following judgments:

$$\Gamma \vdash e \Leftarrow A \dashv \Delta \qquad \Gamma \vdash e \Rightarrow A \dashv \Delta$$

which respectively represent type checking (inputs: Γ, e, A; output: Δ) and type synthesis (inputs: Γ, e; outputs: A, Δ).
We also define a binary algorithmic judgement:

$$\Gamma \vdash X \Box Y \Rrightarrow Z \dashv \Delta$$

which represents a binary judgement $\Box$ under the context Γ on values X and Y that synthesizes Z with output context Δ. For example, the syntax that Dunfield and Krishnaswami use for function application synthesis judgements would be

$$\Gamma \vdash A \bullet e \Rrightarrow C \dashv \Delta$$

which means that under context Γ, A applied to the term e synthesizes output type C and context Δ.

$$\frac{(x:A)\in\Gamma}{\Gamma\vdash x\Rightarrow A\dashv\Gamma}\ (\mathsf{Var})\qquad \frac{\Gamma\vdash e\Rightarrow A\dashv\Theta\qquad \Theta\vdash[\Theta]\,A<:[\Theta]\,B\dashv\Delta}{\Gamma\vdash e\Leftarrow B\dashv\Delta}\ (\mathsf{Sub})$$

$$\frac{\Gamma\vdash A\qquad \Gamma\vdash e\Leftarrow A\dashv\Delta}{\Gamma\vdash(e:A)\Rightarrow A\dashv\Delta}\ (\mathsf{Annotation})\qquad \frac{\Gamma,\alpha\vdash e\Leftarrow A\dashv\Delta,\alpha,\Theta}{\Gamma\vdash e\Leftarrow\forall\alpha.A\dashv\Delta}\ (\forall\ \mathsf{I})$$

$$\frac{\Gamma,x:A\vdash e\Leftarrow B\dashv\Delta,x:A,\Theta}{\Gamma\vdash\lambda x.e\Leftarrow A\to B\dashv\Delta}\ (\to\!\mathsf{I})\qquad \frac{\Gamma,\hat\alpha,\hat\beta,x:\hat\alpha\vdash e\Leftarrow\hat\beta\dashv\Delta,x:\hat\alpha,\Theta}{\Gamma\vdash\lambda x.e\Rightarrow\hat\alpha\to\hat\beta\dashv\Delta}\ (\to\!\mathsf{I}\!\Rightarrow)$$

$$\frac{\Gamma\vdash e_1\Rightarrow A\dashv\Theta\qquad \Theta\vdash[\Theta]\,A\bullet e_2\Rrightarrow C\dashv\Delta}{\Gamma\vdash e_1\ e_2\Rightarrow C\dashv\Delta}\ (\to\!\mathsf{E})\qquad \frac{\Gamma,\hat\alpha\vdash A[\alpha:=\hat\alpha]\bullet e\Rrightarrow C\dashv\Delta}{\Gamma\vdash\forall\alpha.A\bullet e\Rrightarrow C\dashv\Delta}\ (\forall\ \mathsf{App})$$

$$\frac{\Gamma\vdash e\Leftarrow A\dashv\Delta}{\Gamma\vdash A\to C\bullet e\Rrightarrow C\dashv\Delta}\ (\to\!\mathsf{App})$$

$$\frac{\Gamma[\hat\alpha_2,\hat\alpha_1,\hat\alpha=\hat\alpha_1\to\hat\alpha_2]\vdash e\Leftarrow\hat\alpha_1\dashv\Delta}{\Gamma[\hat\alpha]\vdash\hat\alpha\bullet e\Rrightarrow\hat\alpha_2\dashv\Delta}\ (\hat\alpha\mathsf{App})$$

$$\frac{}{\Gamma\vdash\{\}\Leftarrow\{\}\dashv\Gamma}\ (\{\}\mathsf{I})\qquad \frac{}{\Gamma\vdash\{\}\Rightarrow\{\}\dashv\Gamma}\ (\{\}\mathsf{I}\!\Rightarrow)$$

$$\frac{\Gamma\vdash e\Leftarrow A\dashv\Theta\qquad \Theta\vdash[\Theta]\,r\Leftarrow[\Theta]\,\rho\dashv\Delta}{\Gamma\vdash\{\ell:e,r\}\Leftarrow\{\ell:A,\rho\}\dashv\Delta}\ (\mathsf{RcdI})\qquad \frac{\Gamma\vdash e\Rightarrow A\dashv\Theta\qquad \Theta\vdash[\Theta]\,r\Rightarrow[\Theta]\,\rho\dashv\Delta}{\Gamma\vdash\{\ell:e,r\}\Rightarrow\{\ell:A,\rho\}\dashv\Delta}\ (\mathsf{RcdI}\!\Rightarrow)$$

$$\frac{\Gamma\vdash e\Rightarrow A\dashv\Theta\qquad \Theta\vdash[\Theta]\,A\,.\,\ell\Rrightarrow C\dashv\Delta}{\Gamma\vdash e\#\ell\Rightarrow C\dashv\Delta}\ (\mathsf{Prj})\qquad \frac{\Gamma,\hat\alpha\vdash A[\alpha:=\hat\alpha]\,.\,\ell\Rrightarrow C\dashv\Delta}{\Gamma\vdash\forall\alpha.A\,.\,\ell\Rrightarrow C\dashv\Delta}\ (\forall\ \mathsf{Prj})$$

$$\frac{\Gamma\vdash R\#l\longrightarrow C\dashv\Delta}{\Gamma\vdash R\,.\,\ell\Rrightarrow C\dashv\Delta}\ (\mathsf{RcdPrjR})$$

We define record lookup $\Gamma\vdash\rho\#\ell\longrightarrow A\dashv\Delta$ as follows (inputs: Γ, ρ, l; outputs: A, Δ):

$$\frac{}{\Gamma\vdash\{\ell:A,R\}\#\ell\longrightarrow A\dashv\Gamma}\ (\mathsf{lookupYes})\qquad \frac{\ell\neq\ell'\qquad \Gamma\vdash\{R\}\#\ell\longrightarrow A\dashv\Delta}{\Gamma\vdash\{\ell':A',R\}\#\ell\longrightarrow A\dashv\Delta}\ (\mathsf{lookupNo})$$

$$\frac{}{\Gamma[\hat\alpha]\vdash\hat\alpha\#\ell\longrightarrow\hat\alpha_0\dashv\Gamma[\hat\alpha_0,\hat\alpha_1,\hat\alpha=\{\ell:\hat\alpha_0,\hat\alpha_1\}]}\ (\mathsf{Lookup}\ \hat\alpha)$$

$$\frac{}{\Gamma[\hat\alpha]\vdash\{\hat\alpha\}\#\ell\longrightarrow\hat\alpha_0\dashv\Gamma[\hat\alpha_0,\hat\alpha_1,\hat\alpha=(\ell:\hat\alpha_0,\hat\alpha_1)]}\ (\mathsf{Lookup\ RowTail})$$

4.5 Subsumption

We define the algorithmic subsumption:

$$\Gamma \vdash A_0 <: A_1 \dashv \Delta$$

which represents A_0 subsumes A_1 with input context Γ and output context Δ. Subsumption is like subtyping, but only applies to quantifiers. Everything else must be strict equality (for now, this also means records, so you can't use $\{\ell_1 : \texttt{Bool}, \ell_2 : \texttt{Num}\}$ in place of $\{\ell_1 : \texttt{Bool}\}$ even though you really *should* be able to).

$$\frac{}{\Gamma[\hat{\alpha}] \vdash \hat{\alpha} <: \hat{\alpha} \dashv \Gamma[\hat{\alpha}]}\ \text{(EVar)} \qquad \frac{}{\Gamma[\alpha] \vdash \alpha <: \alpha \dashv \Gamma[\alpha]}\ \text{(Var)} \qquad \frac{}{\Gamma \vdash \mathcal{B} <: \mathcal{B} \dashv \Gamma}\ \text{(Const)}$$

$$\frac{\Gamma, \blacktriangleright_{\hat{\alpha}}, \hat{\alpha} \vdash A[\alpha := \hat{\alpha}] <: B \dashv \Delta, \blacktriangleright_{\hat{\alpha}}, \Theta}{\Gamma \vdash \forall \alpha. A <: B \dashv \Delta}\ (\forall\text{L}) \qquad \frac{\Gamma, \alpha \vdash A <: B \dashv \Delta, \alpha, \Theta}{\Gamma \vdash A <: \forall \alpha. B \dashv \Delta}\ (\forall\text{R})$$

$$\frac{\Gamma \vdash B_1 <: A_1 \dashv \Theta \qquad \Theta \vdash [\Theta]\, A_2 <: [\Theta]\, B_2 \dashv \Delta}{\Gamma \vdash A_1 \to A_2 <: B_1 \to B_2 \dashv \Delta}\ (\to)$$

$$\frac{\hat{\alpha} \notin FV(A) \qquad \Gamma[\hat{\alpha}] \vdash \hat{\alpha} \mathrel{:\leqq} A \dashv \Delta}{\Gamma[\hat{\alpha}] \vdash \hat{\alpha} <: A \dashv \Delta}\ \text{InstantiateL} \qquad \frac{\hat{\alpha} \notin FV(A) \qquad \Gamma[\hat{\alpha}] \vdash A \mathrel{\leqq:} \hat{\alpha} \dashv \Delta}{\Gamma[\hat{\alpha}] \vdash A <: \hat{\alpha} \dashv \Delta}\ \text{InstantiateR}$$

$$\frac{\Gamma \vdash R_0 <: R_1 \dashv \Delta}{\Gamma \vdash \{R_0\} <: \{R_1\} \dashv \Delta}\ \text{Record}$$

For this rule, we treat the rows as sets and assume they are reordered so that the matching labels are at the front of the row. An algorithmic implementation would want to deal with the recursive and base cases by looking at the set intersection and difference of the rows.

$$\frac{\Gamma \vdash A <: B \dashv \Theta \qquad \Theta \vdash [\Theta] R_1 <: [\Theta] R_2 \dashv \Delta}{\Gamma \vdash \ell : A, R_1 <: \ell : B, R_2 \dashv \Delta}\ \text{(Row)} \qquad \frac{}{\Gamma \vdash \cdot <: \cdot \dashv \Gamma}\ \text{(Row Nil)}$$

$$\frac{\Gamma[\hat{\alpha}] \vdash \hat{\alpha} \mathrel{:\leqq} r \dashv \Delta}{\Gamma[\hat{\alpha}], \cdot \vdash \hat{\alpha} <: r \dashv \Delta}\ \text{(RowMissingL)} \qquad \frac{\Gamma[\hat{\alpha}] \vdash r \mathrel{\leqq:} \hat{\alpha} \dashv \Delta}{\Gamma[\hat{\alpha}], \cdot \vdash r <: \hat{\alpha} \dashv \Delta}\ \text{(RowMissingR)}$$

This rule also treats the rows as sets and assumes that there are no equal labels between the two rows. Assume an analagous rule for a different order of EVars (the order doesn't matter).

$$\frac{\Gamma \vdash \hat{\alpha} <: m_1 : B_1, m_2 : B_2, \ldots \dashv \Theta \qquad \Theta \vdash \ell_1 : [\Theta]\, A_1, \ell_2 : [\Theta]\, A_2, \ldots <: \hat{\beta} \dashv \Delta}{\Gamma[\hat{\alpha}][\hat{\beta}] \vdash \ell_1 : A_1, \ell_2 : A_2, \ldots, \hat{\alpha} <: m_1 : B_1, m_2 : B_2, \ldots, \hat{\beta} \dashv \Delta}\ \text{(RowMissingLR)}$$

$$\frac{}{\Gamma[\hat{\alpha}][\hat{\beta}] \vdash \hat{\alpha} <: \hat{\beta} \dashv \Gamma[\hat{\alpha}][\hat{\beta} = \hat{\alpha}]}\ \text{(RowTailReach)}$$

4.6 Instantiation

Instantiation is a judgement that solves an EVar, either on the left or right side of the judgement. It is important to recurisvely assign "helper" EVars to match the shape of whatever the EVar is being instantiated to instead of blindly assigning it, as there may be EVars inside of the type it is being assigned to.
The EVar is instantiated so that it subsumes or is subsumed by the type, depending on the type of instantiation (left or right, respectively).
!!! Need an instantiation for rows that is like InstRcd !!!

4.6.1 Left Instantiation

$$\frac{\Gamma \vdash \mathcal{B}}{\Gamma[\hat{\alpha}] \vdash \hat{\alpha} \underset{:=}{\leq} \mathcal{B} \dashv \Gamma[\hat{\alpha} = \mathcal{B}]}\ \textsf{InstLSolve} \qquad \frac{}{\Gamma[\hat{\alpha}][\hat{\beta}] \vdash \hat{\alpha} \underset{:=}{\leq} \hat{\beta} \dashv \Gamma[\hat{\alpha}][\hat{\beta} = \hat{\alpha}]}\ \textsf{InstLReach}$$

$$\frac{\Gamma[\hat{\alpha}_2, \hat{\alpha}_1, \hat{\alpha} = \hat{\alpha}_1 \to \hat{\alpha}_2] \vdash A_1 \underset{=:}{\leq} \hat{\alpha}_1 \dashv \Theta \qquad \Theta \vdash \hat{\alpha}_2 \underset{:=}{\leq} [\Theta]\, A_2 \dashv \Delta}{\Gamma[\hat{\alpha}] \vdash \hat{\alpha} \underset{:=}{\leq} A_1 \to A_2 \dashv \Delta}\ \textsf{InstLArr}$$

$$\frac{\begin{array}{l}\Gamma_0[(\hat{\alpha}_{k+1}), \hat{\alpha}_k, \ldots, \hat{\alpha}_1, \hat{\alpha} = \{\ell_1 : \hat{\alpha}_1, \ell_2 : \hat{\alpha}_2, \ldots, \ell_k : \hat{\alpha}_k, (\hat{\alpha}_{k+1})\}] \vdash \hat{\alpha}_1 \underset{:=}{\leq} A_1 \dashv \Gamma_1 \\ \Gamma_1 \vdash \hat{\alpha}_2 \underset{:=}{\leq} [\Gamma_1]\, A_2 \dashv \Gamma_2 \qquad \cdots \qquad (\Gamma_k \vdash \hat{\alpha}_{k+1} \underset{:=}{\leq} [\Gamma_k]\, \hat{\beta} \dashv \Delta)\end{array}}{\Gamma_0[\hat{\alpha}] \vdash \hat{\alpha} \underset{:=}{\leq} \{\ell_1 : A_1, \ell_2 : A_2, \ldots, \ell_{k-1} : A_{k-1}, (\hat{\beta})\} \dashv \Delta}\ \textsf{InstLRcd}$$

In the above rule, the parentheticals only come into play if there is a row tail in the record. If there isn't, assume that $\Delta = \Gamma_k$.

$$\frac{\Gamma[\hat{\alpha}], \beta \vdash \hat{\alpha} \underset{:=}{\leq} B \dashv \Delta, \beta, \Delta'}{\Gamma[\hat{\alpha}] \vdash \hat{\alpha} \underset{:=}{\leq} \forall \beta. B \dashv \Delta}\ \textsf{InstLAllR}$$

4.6.2 Right Instantiation

$$\frac{\Gamma \vdash \mathcal{B}}{\Gamma[\hat{\alpha}] \vdash \mathcal{B} \underset{=:}{\leq} \hat{\alpha} \dashv \Gamma[\hat{\alpha} = \mathcal{B}]}\ \textsf{InstRSolve} \qquad \frac{}{\Gamma[\hat{\alpha}][\hat{\beta}] \vdash \hat{\beta} \underset{=:}{\leq} \hat{\alpha} \dashv \Gamma[\hat{\alpha}][\hat{\beta} = \hat{\alpha}]}\ \textsf{InstRReach}$$

$$\frac{\Gamma[\hat{\alpha}_2, \hat{\alpha}_1, \hat{\alpha} = \hat{\alpha}_1 \to \hat{\alpha}_2] \vdash \hat{\alpha}_1 \underset{:=}{\leq} A_1 \dashv \Theta \qquad \Theta \vdash [\Theta]\, A_2 \underset{=:}{\leq} \hat{\alpha}_2 \dashv \Delta}{\Gamma[\hat{\alpha}] \vdash A_1 \to A_2 \underset{=:}{\leq} \hat{\alpha} \dashv \Delta}\ \textsf{InstRArr}$$

$$\frac{\begin{array}{l}\Gamma_0[(\hat{\alpha}_{k+1}), \hat{\alpha}_k, \ldots, \hat{\alpha}_1, \hat{\alpha} = \{\ell_1 : \hat{\alpha}_1, \ell_2 : \hat{\alpha}_2, \ldots, \ell_k : \hat{\alpha}_k, (\hat{\alpha}_{k+1})\}] \vdash A_1 \underset{=:}{\leq} \hat{\alpha}_1 \dashv \Gamma_1 \\ \Gamma_1 \vdash [\Gamma_2]\, A_2 \underset{=:}{\leq} \hat{\alpha}_2 \dashv \Gamma_3 \qquad \cdots \qquad (\Gamma_k \vdash \hat{\beta} \underset{=:}{\leq} \hat{\alpha}_{k+1} \dashv \Delta)\end{array}}{\Gamma_0[\hat{\alpha}] \vdash \{\ell_1 : A_1, \ell_2 : A_2, \ldots, \ell_{k-1} : A_{k-1}, (\hat{\beta})\} \underset{=:}{\leq} \hat{\alpha} \dashv \Delta}\ \textsf{InstRRcd}$$

In the above rule, the parentheticals only come into play if there is a row tail in the record. If there isn't, assume that $\Delta = \Gamma_k$.

$$\frac{\Gamma[\hat{\alpha}], \blacktriangleright_{\hat{\alpha}}, \hat{\beta} \vdash A[\beta := \hat{\beta}] <: \hat{\alpha} \dashv \Delta, \blacktriangleright_{\hat{\beta}}, \Delta'}{\Gamma[\hat{\alpha}] \vdash \forall \beta. B <: \hat{\alpha} \dashv \Delta}\ \textsf{InstRAllL}$$

5 Proof Extensions

We only provide the cases pertaining to the extensions, as the rest of the proof remains the same as in *Complete and Easy*.
Note: we'll be sort of conflating row variables (α_ρ) with type variables (α) because they are symmetric and also disjoint. We do our best to visually distinguish the two, but when we prove something involving type variables, we mean for both type and row variables unless otherwise noted. The same is true of τ and ρ.

5.1 Properties of Well-Formedness

Proposition 1 (Weakening). *If $\Psi \vdash A$, then $\Psi, \Psi' \vdash A$ by a derivation of the same size.*

This still follows by straightforward induction on the cases we have added.

Proposition 2 (Substitution). *If $\Psi \vdash A$ and $\Psi, \alpha, \Psi' \vdash B$, then $\Psi, [\alpha \mapsto A]\Psi' \vdash [\alpha \mapsto A]B$.*

Proof. The proof is by induction on B.

- Case $B = R$.

  If α_ρ isn't the row tail of R (i.e. is not at the "surface level" of the row), then it follows from the inductive hypothesis that $\Psi, \Psi' \vdash R$.

  If α_ρ does occur in R, then the substitution extends R with the row A. Suppose $A = \ell_1 : A_1, \ldots, \ell_k : A_k$. Because $\Psi \vdash A$, we know that by DeclRowWF, $\Psi \vdash A_i$ for $1 \leq i \leq k$. By proposition 1, $\Psi, \Psi' \vdash A_i$ for $1 \leq i \leq k$, and thus by DeclRowWF $\Psi, \Psi' \vdash [\alpha_\rho \mapsto A]B$.

- Case $B = \forall\alpha_\rho.A$.

  This follows immediately by the inductive hypothesis after applying DeclForall-RowWF.

- Case $B = \{R\}$.

  This follows from Case $B = R$.

$\square$

Lemma 2.5 (Row Formedness). *If $\Psi \vdash \ell_1 : A_1, \ldots, \ell_k : A_k$, then $\Psi \vdash \ell_1 : A_1, \ldots, \ell_i : A_i$ and $\Psi \vdash A_i$ for all $1 \leq i \leq k$.*

This follows from DeclRowWF by induction.

5.2 Reflexivity

Lemma 3 (Reflexivity of Declarative Subtyping)**.** *Subtyping is reflexive: if* $\Psi \vdash A$ *then* $\Psi \vdash A \leq A$.

Proof. The proof is by induction on A.

- Case $A = \forall\alpha_\rho.A_0$. This follows by the same reasoning for $\forall$.
- Case $A = \ell_1 : A_1, \ldots, \ell_k : A_k$. By the inductive hypothesis, $\ell_1 : A_1, \ldots, \ell_{k-1} : A_{k-1}$ subtypes itself. Also by the inductive hypothesis, $\ell_k : A_k \leq \ell_k : A_k$. These combined with $\leq$ Row complete the proof.
- Case $A = \{R\}$. This follows from the row case.

□

5.3 Subtyping Implies Well-Formedness

Lemma 4 (Well-Formedness)**.** *If* $\Psi \vdash A \leq B$ *then* $\Psi \vdash A$ *and* $\Psi \vdash B$.

Proof. The proof is by induction on the derivation.

- Case $\Psi \vdash \ell : A, R_1 \leq \ell : B, R_2$.

  By the hypothesis, $\Psi \vdash A \leq B$ and $\Psi \vdash R \leq_1 R_2$. From the inductive hypothesis, we get $\Psi \vdash A; B; R_1; R_2$. We may then apply DeclRowWF to conclude that $\Psi \vdash \ell : A, R_1$ and $\Psi \vdash \ell : B, R_2$, as desired.
- Case ForallRowL and ForallRowR. Symmetric to the ForallL and ForallR cases.
- Case $\Psi \vdash \{R_1\}\{R_2\} \leq$. Follows from the row case and DeclRecordWF.

□

5.4 Substitution

Lemma 5 (Substitution)**.** *If* $\Psi \vdash \tau$ *and* $\Psi, \alpha, \Psi' \vdash A \leq B$, *then* $\Psi, [\alpha \mapsto \tau]\Psi' \vdash [\alpha \mapsto \tau]A \leq [\alpha \mapsto \tau]B$.

Proof. The proof is by induction on the derivation.

- Case $\Psi \vdash \ell : A, R_1 \leq \ell : B, R_2$.

  By the inductive hypothesis, $\Psi, [\alpha \mapsto \tau]\Psi' \vdash [\alpha \mapsto \tau]R_1 \leq [\alpha \mapsto \tau]R_2$ and $\Psi, [\alpha \mapsto \tau]\Psi' \vdash [\alpha \mapsto \tau]A \leq [\alpha \mapsto \tau]B$. By $\leq$ Row, we conclude that

  $$\Psi, [\alpha \mapsto \tau]\Psi' \vdash \ell : [\alpha \mapsto \tau]A, [\alpha \mapsto \tau]R_1 \leq \ell : [\alpha \mapsto \tau]B, [\alpha \mapsto \tau]R_2$$

  which, by definition of substitution, gives the desired result.

□

5.5 Transitivity

Lemma 6 (Transitivity of Declarative Subtyping). *If* $\Psi \vdash A \leq B$ *and* $\Psi \vdash B \leq C$ *then* $\Psi \vdash A \leq C$.

Proof. We induct on a similar metric used in the proof in *Complete and Easy*.

$$\langle \#\forall(B), \quad \#\rotatebox[origin=c]{180}{$\forall$}(B), \quad \mathcal{D}_1 + \mathcal{D}_2 \rangle$$

where $\#\forall$ is the count of quantifiers and $\#\rotatebox[origin=c]{180}{$\forall$}$ is the count of row quantifiers. The last part is the size of each of the respective derviations in the hypothesis.

TODO: reconcile the new metric and add in the rest of the cases that are affected by it.

- Case $\Psi \vdash \ell : A, R_1 \leq \ell : B, R_2$ and $\Psi \vdash \ell : B, R_2 \leq \ell : C, R_3$. By the inductive hypothesis, $\Psi \vdash R \leq_1 R_2$ and $\Psi \vdash R \leq_2 R_3$ implies $\Psi \vdash R \leq_1 R_3$. Similarly, we can conclude that $\Psi \vdash A \leq C$. Using these judgements, we can apply DeclRowWF to conclude the desired.

□

5.6 Invertibility of $\leq \forall$R

Lemma 7 (Invertibility). *If* $\mathcal{D}$ *derives* $\Psi \vdash A \leq \forall \beta . B$, *then* $\mathcal{D}'$ *derives* $\Psi, \beta \vdash A \leq B$ *where* $\mathcal{D}' < \mathcal{D}$.

6 Algorithmic System Proof Extensions

6.1 Context Extension

6.2 Syntactic Properties

Most (all?) of these proofs remain unchanged due to the fact that we haven't done much to significantly alter the framework (although we have added a new kind of evar, it functions much the same way).

6.3 Instantiation Extends

Lemma 32 (Instantiation Extends). *If* $\Gamma \vdash \hat{\alpha} \mathrel{\underset{:=}{\leq}} \tau \dashv \Delta$ *or* $\Gamma \vdash \tau \mathrel{\underset{=:}{\leq}} \hat{\alpha} \dashv \Delta$, *then* $\Gamma \longrightarrow \Delta$.

Proof.

- Case InstLRcd and InstRRcd: We'll employ a similar approach to the one used in Case InstLArr. We want to derive $\Gamma_0[\hat{\alpha}] \longrightarrow \Delta$.

  Our IH gives us, by a lot of transitivity (which requires an inductive proof), that

  $$\Gamma_0[(\hat{\alpha}_{k+1}), \hat{\alpha}_k, \ldots, \hat{\alpha}_1, \hat{\alpha} = \{\ell_1 : \hat{\alpha}_1, \ell_2 : \hat{\alpha}_2, \ldots, \ell_k : \hat{\alpha}_k, (\hat{\alpha}_{k+1})\}] \longrightarrow \Delta$$

We can employ a series of Lemma 28 (Unsolved Variable Addition for Extension), one Lemma 26 (Solution Admissibility for Extension), and transitivity to obtain

$$\Gamma_0 \longrightarrow \Gamma_0[(\hat{\alpha}_{k+1}), \hat{\alpha}_k, \ldots, \hat{\alpha}_1, \hat{\alpha} = \{\ell_1 : \hat{\alpha}_1, \ell_2 : \hat{\alpha}_2, \ldots, \ell_k : \hat{\alpha}_k, (\hat{\alpha}_{k+1})\}]$$

, which, by the above and transitivity gives the desired.

□

6.4 Subtyping Extends

Lemma 33 (Subtyping Extension)**.** *If* $\Gamma \vdash A <: B \dashv \Delta$*, then* $\Gamma \longrightarrow \Delta$*.*

Proof. By induction on the derivation.

- Case Record: Follows immediately from the IH
- Case Row: Follows from IH and transitivity. $\Gamma \longrightarrow \Theta$ and $\Theta \longrightarrow Delta$ implies the desired.
- Case Row Nil: Follows immediately from Γ being unchanged.
- Case RowMissingL and RowMissingR: Follows from instantiation extending.
- Case RowMissingLR: Follows from transitivity.
- Case RowTailReach: Follows from Lemma 28 (Solution Admissibility for Extension).

□

6.5 Decidability of Instantiation

fill in more specific details

Lemma 34 (Left Unsolvedness Preservation)**.** *If* $\underbrace{\Gamma_0, \hat{\alpha}, \Gamma_1}_{\Gamma} \vdash \hat{\alpha} \mathrel{:\leq} A \dashv \Delta$ *or* $\underbrace{\Gamma_0, \hat{\alpha}, \Gamma_1}_{\Gamma} \vdash A \mathrel{\leq:} \hat{\alpha} \dashv \Delta$*, and* $\hat{\beta} \in \mathsf{unsolved}(\Gamma_0), then \hat{\beta} \in \mathsf{unsolved}(\Delta)$*.*

Proof. The proof for the cases InstLRcd and InstRRcd follow from the same technique used for InstLArr. □

Lemma 35 (Left Free Variable Preservation)**.**

Lemma 36 (Instantiation Size Preservation)**.**

The proofs for the above are also similar.

Theorem 8 (Decidability of Instantiation)**.**

Follows from the lemmas with a similar technique to InstLArr.

7 *Complete and Easy* Errata

A collection of minor errata from *Complete and Easy* found while writing this document. We use the authors' notations in this section, if ever we differ.

$\leq \forall$ **R, Declarative**

The rule, as stated:

$$\frac{\Psi, \beta \vdash A \leq B}{\Psi \vdash A \leq \forall \beta . B} \leq \forall \mathsf{R}$$

The problem:

$$\cfrac{\cfrac{\beta \in \cdot, \beta}{\cdot, \beta \vdash \beta \leq \beta}}{\cdot \vdash \beta \leq \forall \beta . \beta}$$

By Lemma 4, this implies $\cdot \vdash \beta$ which is not true.

The fix:

$$\frac{\Psi \vdash A \qquad \Psi, \beta \vdash A \leq B}{\Psi \vdash A \leq \forall \beta . B} \leq \forall \mathsf{R}\ \text{Fixed}$$

Poposition 2, Substitution

There is a missing $[A/\alpha]$ on the context (as done in later rules, e.g. A4, Substitution).

The fix:

Write as the conclusion

$$\Psi, [A/\alpha]\Psi' \vdash [A/\alpha]B.$$

Proof of A5, Transitivity

There is a spurious $\Psi \vdash \tau$ in the $\leq \forall$ R case.

Lemma 19, Extension Equality Preservation

In case AddSolved, the line

We implicity assume that Δ' is well-formed, so $\hat{\alpha} \notin$ dom(Δ').

is not quite correct. The line should be

We implicity assume that $\Delta', \hat{\alpha} = \tau$ is well-formed, so $\hat{\alpha} \notin$ dom(Δ').