Atomic Types	T	$::=$	$A \to B$	Function type
		$\mid$	$\bot$	Bottom type
		$\mid$	$\forall \alpha * B.\, A$	Universal quantification
Types	A, B, C, D	$::=$	α	Type variable
		$\mid$	$A \cap B$	Intersection type
		$\mid$	T	Atomic type
Expressions	e	$::=$	x	Variable
		$\mid$	$\lambda(x{:}A).\, e$	Lambda
		$\mid$	$e_1\ e_2$	Application
		$\mid$	$\Lambda \alpha * A.\, e$	Big lambda
		$\mid$	$e\ A$	Type application
		$\mid$	$e_1 ,, e_2$	Merge
Contexts	Γ	$::=$	ϵ	
		$\mid$	$\Gamma, \alpha * A$	
		$\mid$	$\Gamma, x{:}A$	
Sugar	$\Lambda \alpha.\, e$	$::=$	$\Lambda \alpha * \bot.\, e$	

Figure 1. Syntax.

$$\boxed{A <: B \hookrightarrow F}$$

$$\frac{}{\alpha <: \alpha \hookrightarrow \lambda(x{:}|\alpha|).\, x}\ \textsc{SubVar}$$

$$\frac{\tau_3 <: \tau_1 \hookrightarrow C_1 \qquad \tau_2 <: \tau_4 \hookrightarrow C_2}{\tau_1 \to \tau_2 <: \tau_3 \to \tau_4 \hookrightarrow \lambda(f{:}|\tau_1 \to \tau_2|).\, \lambda(x{:}|\tau_3|).\, C_2\ (f\ (C_1\ x))}\ \textsc{SubFun}$$

$$\frac{\tau_1 <: [\alpha_1/\alpha_2]\tau_2 \hookrightarrow C}{\forall \alpha_1 * \tau_3.\, \tau_1 <: \forall \alpha_2 * \tau_3.\, \tau_2 \hookrightarrow \lambda(f{:}|\forall \alpha_1 * \tau_3.\, \tau_1|).\, \Lambda \alpha.\, C\ (f\ \alpha)}\ \textsc{SubForall}$$

$$\frac{\tau_1 <: \tau_2 \hookrightarrow C_1 \qquad \tau_1 <: \tau_3 \hookrightarrow C_2}{\tau_1 <: \tau_2 \cap \tau_3 \hookrightarrow \lambda(x{:}|\tau_1|).\, (C_1\ x, C_2\ x)}\ \textsc{SubAnd}$$

$$\frac{\tau_1 <: \tau_3 \hookrightarrow C}{\tau_1 \cap \tau_2 <: \tau_3 \hookrightarrow \lambda(x{:}|\tau_1 \cap \tau_2|).\, C\ (\mathsf{proj}_1 x)}\ \textsc{SubAnd}_1$$

$$\frac{\tau_2 <: \tau_3 \hookrightarrow C}{\tau_1 \cap \tau_2 <: \tau_3 \hookrightarrow \lambda(x{:}|\tau_1 \cap \tau_2|).\, C\ (\mathsf{proj}_2 x)}\ \textsc{SubAnd}_2$$

1

Figure 2. Subtyping.

$$\boxed{\Gamma \vdash A \bot B}$$

$$\frac{\alpha * B \in \Gamma}{\Gamma \vdash \alpha \bot B}\ \textsc{DisjointRefl} \qquad \frac{\alpha * A \in \Gamma}{\Gamma \vdash A \bot \alpha}\ \textsc{DisjointSym}$$

$$\frac{\Gamma \vdash A \bot C \qquad \Gamma \vdash B \bot C}{\Gamma \vdash A \& B \bot C}\ \textsc{DisjointSub1}$$

$$\frac{\Gamma \vdash A \bot B \qquad \Gamma \vdash A \bot C}{\Gamma \vdash A \bot B \& C}\ \textsc{DisjointSub2}$$

$$\frac{T_1 \not<: T_2 \qquad T_2 \not<: T_1}{\Gamma \vdash T_1 \bot T_2}\ \textsc{DisjointAtomic}$$

Figure 3. Disjointness.

$$\boxed{\Gamma \vdash A\ \mathsf{type}}$$

$$\Gamma \vdash \bot\ \mathsf{type}\ \textsc{WFBot} \qquad \frac{\Gamma \vdash A\ \mathsf{type} \qquad \Gamma \vdash B\ \mathsf{type}}{\Gamma \vdash A \to B\ \mathsf{type}}\ \textsc{WFFun}$$

$$\frac{\Gamma, \alpha * B \vdash A\ \mathsf{type}}{\Gamma \vdash \forall \alpha * B.\, A\ \mathsf{type}}\ \textsc{WFForall} \qquad \frac{\alpha * A \in \Gamma}{\Gamma \vdash \alpha\ \mathsf{type}}\ \textsc{WFVar}$$

$$\frac{\Gamma \vdash A\ \mathsf{type} \qquad \Gamma \vdash B\ \mathsf{type} \qquad \Gamma \vdash A \bot B}{\Gamma \vdash A \cap B \mathsf{type}}\ \textsc{WFInter}$$

Figure 4. Well-formed types.

$$\boxed{\Gamma \vdash e : A \hookrightarrow E}$$

$$\frac{x{:}A \in \Gamma}{\Gamma \vdash x : A \hookrightarrow x}\ \textsc{TyVar}$$

$$\frac{\Gamma \vdash A\ \mathsf{type} \qquad \Gamma, x{:}A \vdash e : B \hookrightarrow E}{\Gamma \vdash \lambda(x{:}A).\, e : A \to B \hookrightarrow \lambda(x{:}|A|).\, E}\ \textsc{TyLam}$$

$$\frac{\begin{array}{c}\Gamma \vdash e_1 : A_1 \to A_2 \hookrightarrow E_1 \\ \Gamma \vdash e_2 : A_3 \hookrightarrow E_2 \qquad A_3 <: A_1 \hookrightarrow C\end{array}}{\Gamma \vdash e_1\ e_2 : A_2 \hookrightarrow E_1\ (C\ E_2)}\ \textsc{TyApp}$$

$$\frac{\Gamma, \alpha * B \vdash e : A \hookrightarrow E \qquad \Gamma \vdash B\ \mathsf{type}}{\Gamma \vdash \Lambda \alpha * B.\, e : \forall \alpha * B.\, A \hookrightarrow \Lambda \alpha.\, E}\ \textsc{TyBLam}$$

$$\frac{\Gamma \vdash e : \forall \alpha * C.\, B \hookrightarrow E \qquad \Gamma \vdash A \bot C \qquad \Gamma \vdash A\ \mathsf{type}}{\Gamma \vdash e\ A : [A/\alpha]B \hookrightarrow E\ |A|}\ \textsc{TyTApp}$$

$$\frac{\begin{array}{c}\Gamma \vdash e_1 : A \hookrightarrow E_1 \\ \Gamma \vdash e_2 : B \hookrightarrow E_2 \qquad \Gamma \vdash A \bot B\end{array}}{\Gamma \vdash e_1 ,, e_2 : A \cap B \hookrightarrow (E_1, E_2)}\ \textsc{TyMerge}$$

Figure 5. Typing.

0.1 "Testsuite" of examples

1. $\lambda(x : \mathsf{Int}{*}\mathsf{Int}).(\lambda(z : \mathsf{Int}).z)\ x$: This example should not type-check because it leads to an ambigous choice in the body of the lambda. In the current system the well-formedness checks forbid such example.
2. $\Lambda A.\Lambda B.\lambda(x : A).\lambda(y : B).(\lambda(z : A).z)(x,,y)$: This example should not type-check because it is not guaranteed that the instantiation of A and B produces a well-formed type. The TyMerge rule forbids it with the disjointness check.

3. $\Lambda A.\Lambda B * A.\lambda(x : A).\lambda(y : B).(\lambda(z : A).z)(x,,y)$: This example should type-check because B is guaranteed to be disjoint with A. Therefore instantiation should produce a well-formed type.

$$\epsilon \vdash 1,,2 : (\mathsf{Int} * \mathsf{Int}) \Rightarrow \mathsf{Int} \cap \mathsf{Int}$$

Definition 1. (Disjointness) Two sets S and T are *disjoint* if there does not exist an element x, such that $x \in S$ and $x \in T$.

Definition 2. (Disjointness) Two types A and B are *disjoint* if there does not exist an expression e, which is not a merge, such that $\epsilon \vdash e : A'$, $\epsilon \vdash e : B'$, $A' <: A$, and $B' <: B$.

Definition 3. (Disjointness) $A \perp B = \nexists C. A <: C \wedge B <: C$

Two types A and B are *disjoint* if their least common supertype is $\top$.