Elaborating F_i^+ to record calculus λ^{rec}

Snow

September 19, 2023

Changes in $\lambda^{\mathbf{rec}}$ 0) Simplify the system, drop polymorphism, and use simple labels
1) Type soundness proved
2) Use list for record types, so the time cost of concatenation can be moved to compilation.

1 Syntax of $\lambda^{\mathbf{rec}}$

$ttyp,\ At,\ Bt,\ Ct$	$::=$		types
	$\mid$	$\top$	top type
	$\mid$	$\bot$	bottom type
	$\mid$	$\mathbb{B}$	base type
	$\mid$	$At \to Bt$	function types
	$\mid$	$\{ll : At, Bt\}$	record

$texp,\ t,\ te,\ tp,\ tv,\ tu$	$::=$		target term
	$\mid$	x	variable
	$\mid$	b	base value
	$\mid$	$\lambda x.t$	abstractions
	$\mid$	$\mathsf{fix}\, x.t$	fixpoint
	$\mid$	$t_1\ t_2$	applications
	$\mid$	$\{\}$	empty record
	$\mid$	$\{ll \Rightarrow t_1', t_2\}$	
	$\mid$	$t_1.ll$	projection
	$\mid$	$t_1; t_2$	concatenation

Notes 0) Multi-field records are nullable. 1) The whole program evaluates to a record form. 3) l stands for a string.

$\boxed{\textbf{value}\ tv}$ *(Values)*

$$\text{VALUE-UNIT}\ \frac{}{\textbf{value}\ \{\}} \qquad \text{VALUE-LIT}\ \frac{}{\textbf{value}\ b} \qquad \text{VALUE-ABS}\ \frac{}{\textbf{value}\ \lambda x.t} \qquad \text{VALUE-MERGE}\ \frac{\textbf{value}\ tv_1 \quad \textbf{value}\ tv_2}{\textbf{value}\ \{ll \Rightarrow tv_1, tv_2\}}$$

$\boxed{t \to t'}$ *(Record Calculus Structrual Reduction)*

$$\text{TS-PROJ1}\ \frac{t \to t'}{t.ll \to t'.ll} \qquad \text{TS-APPL}\ \frac{t \to t'}{t\ t_2 \to t'\ t_2} \qquad \text{TS-APPR}\ \frac{\textbf{value}\ tv \quad t \to t'}{tv\ t \to tv\ t'} \qquad \text{TS-MERGEL}\ \frac{t \to t'}{t; t_2 \to t'; t_2} \qquad \text{TS-MERGER}\ \frac{\textbf{value}\ tv \quad t \to t'}{tv; t \to tv; t'}$$

$$\text{TS-RCDHEAD}\ \frac{t \to t'}{\{ll \Rightarrow t, t_2\} \to \{ll \Rightarrow t', t_2\}} \qquad \text{TS-RCDTAIL}\ \frac{\textbf{value}\ tv \quad t \to t'}{\{ll \Rightarrow tv, t\} \to \{ll \Rightarrow tv, t'\}} \qquad \text{TS-MERGEEMPTY}\ \frac{\textbf{value}\ tv}{\{\}; tv \to tv}$$

$$\textsc{TS-MergeRcd}\quad \frac{\begin{array}{c}\textbf{value}\ tv_1\\ \textbf{value}\ tv_2 \qquad \textbf{value}\ tv_3\end{array}}{(\{ll \Rightarrow tv_1, tv_2\}); tv_3 \rightarrow \{ll \Rightarrow tv_1, (tv_2; tv_3)\}}$$

$$\textsc{TS-ProjRcd}\quad \frac{\textbf{value}\ tv \qquad \textbf{lookup}\ ll\ tv \Rightarrow t}{tv.ll \rightarrow t}$$

$$\textsc{TS-AppAbs}\quad \frac{}{(\lambda x.t)\ tv \rightarrow t[x \mapsto tv]}$$

$$\textsc{TS-Fixpoint}\quad \frac{}{\mathsf{fix}\, x.t \rightarrow t[x \mapsto \mathsf{fix}\, x.t]}$$

$\boxed{At\ \&\ Bt \Rightarrow Ct}$ *(Concatenate types)*

$$\textsc{CT-Nil}\quad \frac{\textbf{recTyp}\ Bt}{\top\ \&\ Bt \Rightarrow Bt}$$

$$\textsc{CT-Rcd}\quad \frac{\begin{array}{c}(\textbf{lookup}\ ll\ Bt_2 \Rightarrow At' \ \wedge\ At' \approx At) \ \vee\ \textbf{lookup}\ ll\ Bt_2 \not\Rightarrow\\ Bt_1\ \&\ Bt_2 \Rightarrow Ct\end{array}}{(\{ll : At, Bt_1\})\ \&\ Bt_2 \Rightarrow \{ll : At, Ct\}}$$

$\boxed{\textbf{recTyp}\ At}$ *(Valid record types)*

$$\textsc{RT-Nil}\quad \frac{}{\textbf{recTyp}\ \top}$$

$$\textsc{RT-Rcd}\quad \frac{\textbf{recTyp}\ Bt}{\textbf{recTyp}\ \{ll : At, Bt\}}$$

$\boxed{\vdash At}$ *(Type welformedness)*

$$\textsc{WF-Nil}\quad \frac{}{\vdash \top}$$

$$\textsc{WF-Bot}\quad \frac{}{\vdash \bot}$$

$$\textsc{WF-Base}\quad \frac{}{\vdash \mathbb{B}}$$

$$\textsc{WF-Rcd}\quad \frac{\begin{array}{c}\vdash At \qquad \vdash Bt \qquad \textbf{recTyp}\ Bt\\ (\textbf{lookup}\ ll\ Bt \Rightarrow At' \ \wedge\ At' \approx At) \ \vee\ \textbf{lookup}\ ll\ Bt \not\Rightarrow\end{array}}{\vdash \{ll : At, Bt\}}$$

$$\textsc{WF-Arrow}\quad \frac{\vdash At \qquad \vdash Bt}{\vdash At \rightarrow Bt}$$

$\boxed{At\ \&\ Bt \Rightarrow Ct}$ *(Concatenate types)*

$$\textsc{CT-Nil}\quad \frac{\textbf{recTyp}\ Bt}{\top\ \&\ Bt \Rightarrow Bt}$$

$$\textsc{CT-Rcd}\quad \frac{\begin{array}{c}(\textbf{lookup}\ ll\ Bt_2 \Rightarrow At' \ \wedge\ At' \approx At) \ \vee\ \textbf{lookup}\ ll\ Bt_2 \not\Rightarrow\\ Bt_1\ \&\ Bt_2 \Rightarrow Ct\end{array}}{(\{ll : At, Bt_1\})\ \&\ Bt_2 \Rightarrow \{ll : At, Ct\}}$$

$\boxed{\Gamma \vdash t : At}$ *(Target typing)*

$$\textsc{TTyping-Base}\quad \frac{\vdash \Gamma}{\Gamma \vdash b : \mathbb{B}}$$

$$\textsc{TTyping-Var}\quad \frac{\vdash \Gamma \qquad x : At \in \Gamma}{\Gamma \vdash x : At}$$

$$\textsc{TTyping-Abs}\quad \frac{\Gamma, x : At \vdash t : Bt}{\Gamma \vdash \lambda x.t : At \rightarrow Bt}$$

$$\textsc{TTyping-Fix}\quad \frac{\Gamma, x : Bt \vdash t : At \qquad At \approx Bt}{\Gamma \vdash \mathsf{fix}\, x.t : At}$$

$$\textsc{TTyping-App}\quad \frac{\begin{array}{c}\Gamma \vdash t_1 : At \rightarrow Bt\\ \Gamma \vdash t_2 : At' \qquad At \approx At'\end{array}}{\Gamma \vdash t_1\ t_2 : Bt}$$

$$\textsc{TTyping-RcdNil}\quad \frac{\vdash \Gamma}{\Gamma \vdash \{\} : \top}$$

$$\textsc{TTyping-RcdCons}\quad \frac{\begin{array}{c}\textbf{recTyp}\ Bt\\ (\textbf{lookup}\ ll\ Bt \Rightarrow At' \ \wedge\ At' \approx At) \ \vee\ \textbf{lookup}\ ll\ Bt \not\Rightarrow\\ \Gamma \vdash t_1 : At \qquad \Gamma \vdash t_2 : Bt\end{array}}{\Gamma \vdash (\{ll \Rightarrow t_1, t_2\}) : (\{ll : At, Bt\})}$$

$$\textsc{TTyping-RcdProj}\quad \frac{\begin{array}{c}\Gamma \vdash t : At\\ \textbf{lookup}\ ll\ At \Rightarrow Bt\end{array}}{\Gamma \vdash t.ll : Bt}$$

$$\textsc{TTyping-RcdMerge}\quad \frac{\begin{array}{c}\textbf{recTyp}\ At\\ \textbf{recTyp}\ Bt \qquad \Gamma \vdash t_1 : At\\ \Gamma \vdash t_2 : Bt \qquad At\ \&\ Bt \Rightarrow Ct\end{array}}{\Gamma \vdash t_1; t_2 : Ct}$$

2 Syntax of F_i^+

$$
\begin{array}{llll}
\mathit{typ},\ A,\ B,\ C & ::= & & \text{types} \\
 & | & \top & \text{top type} \\
 & | & \bot & \text{bottom type} \\
 & | & \mathbb{B} & \text{base type} \\
 & | & A \to B & \text{function types} \\
 & | & A \,\&\, B & \text{intersection} \\
 & | & \{l : A\} & \text{record}
\end{array}
$$

$$
\begin{array}{llll}
\mathit{exp},\ e & ::= & & \text{expressions} \\
 & | & \top & \text{top value} \\
 & | & b & \text{base literal} \\
 & | & x & \text{variables} \\
 & | & \lambda x.\, e : A \to B & \text{abstractions} \\
 & | & \mathsf{fix}\ x.\, e : A & \text{fixpoint} \\
 & | & e_1\ e_2 & \text{applications} \\
 & | & e_1 \,,,\, e_2 & \text{merge} \\
 & | & e : A & \text{annotation} \\
 & | & \{l = e\} & \text{record} \\
 & | & e.l & \text{projection}
\end{array}
$$

Notes 0) Fixpoint typing rule updated.
1) Type translation (from source type to string) modeled.
2) Current focus: converting source type to target type.

2.1 Elaboration rules

1) All top-like values are translated into empty lists.

$$
\textsc{Ela-Top}\quad \frac{}{\Gamma \vdash \top \ \Rightarrow\ \top \rightsquigarrow \{\}}
\qquad
\textsc{Ela-TopAbs}\quad \frac{\rceil B \lceil}{\Gamma \vdash \lambda x.\, e : A \to B \ \Rightarrow\ A \to B \rightsquigarrow \{\}}
\qquad
\textsc{Ela-TopRcd}\quad \frac{\Gamma \vdash e \ \Rightarrow\ A \rightsquigarrow t \qquad \rceil A \lceil}{\Gamma \vdash \{l = e\} \ \Rightarrow\ \{l : A\} \rightsquigarrow \{\}}
$$

2) All basic values (that are not merges) are translated into single-field records.

$$
\textsc{Ela-Base}\quad \frac{}{\Gamma \vdash b \ \Rightarrow\ \mathbb{B} \rightsquigarrow \{|\mathbb{B}| \Rightarrow b\}}
\qquad
\textsc{Ela-Abs}\quad \frac{\neg\rceil B \lceil \qquad \Gamma, x : A \vdash e \ \Leftarrow\ B \rightsquigarrow t}{\Gamma \vdash \lambda x.\, e : A \to B \ \Rightarrow\ A \to B \rightsquigarrow \{|A \to B| \Rightarrow \lambda x.t\}}
$$

$$
\textsc{Ela-Rcd}\quad \frac{\neg\rceil A \lceil \qquad \Gamma \vdash e \ \Rightarrow\ A \rightsquigarrow t}{\Gamma \vdash \{l = e\} \ \Rightarrow\ \{l : A\} \rightsquigarrow \{|\{l : A\}| \Rightarrow t\}}
$$

3) Variables are directly used (unwrap the thunk) and merged terms are simply concatenated, with the expectation that $\cdot \vdash e \ \Rightarrow\ A \rightsquigarrow t$ implies $t \to^* \{|A| \Rightarrow t'\}$ (or a multi-field record) (i.e. the variable will be substituted by a term of this exact type).

$$
\textsc{Ela-Var}\quad \frac{x : A \in \Gamma}{\Gamma \vdash x \ \Rightarrow\ A \rightsquigarrow x}
\qquad
\textsc{Ela-Merge}\quad \frac{\Gamma \vdash e_1 \ \Rightarrow\ A \rightsquigarrow t_1 \qquad \Gamma \vdash e_2 \ \Rightarrow\ B \rightsquigarrow t_2 \qquad A * B}{\Gamma \vdash e_1 \,,,\, e_2 \ \Rightarrow\ A \,\&\, B \rightsquigarrow t_1; t_2}
$$

4) Projection, application, and type application make use of the property that every translated subterms are records. Application is lazy.

$$\textsc{Ela-Proj}\quad \frac{\Gamma \vdash e \Rightarrow A \rightsquigarrow t_1 \qquad t_1 : A; \{l\} \rightsquigarrow t_2 : B}{\Gamma \vdash e.l \Rightarrow B \rightsquigarrow t_2}$$

$$\textsc{Ela-App}\quad \frac{\Gamma \vdash e_1 \Rightarrow A \rightsquigarrow t_1 \qquad \Gamma \vdash e_2 \Rightarrow B' \rightsquigarrow t_2 \qquad t_1 : A; t_2 : B' \rightsquigarrow t_3 : C}{\Gamma \vdash e_1\, e_2 \Rightarrow C \rightsquigarrow t_3}$$

5) Annotated expressions, like fixpoints, rely on the subsumption rule to insert coercions. The coerced expression always evaluates to a record.

$$\textsc{Ela-Anno}\quad \frac{\Gamma \vdash e \Leftarrow A \rightsquigarrow t}{\Gamma \vdash e : A \Rightarrow A \rightsquigarrow t}$$

$$\textsc{Ela-Fix}\quad \frac{\Gamma, x : A \vdash e \Leftarrow A \rightsquigarrow t}{\Gamma \vdash \mathsf{fix}\ x.\, e : A \Rightarrow A \rightsquigarrow \mathsf{fix}\, x.t}$$

$$\textsc{Ela-Sub}\quad \frac{\Gamma \vdash e \Rightarrow A \rightsquigarrow t_1 \qquad t_1 : A <: B \rightsquigarrow t_2}{\Gamma \vdash e \Leftarrow B \rightsquigarrow t_2}$$

2.2 Coercions

$\boxed{t_1 : A <: B \rightsquigarrow t_2}$ *(Coercive subtyping)*

$$\textsc{S-Top}\quad \frac{B\ \mathsf{Ordinary} \qquad \rceil B \lceil}{t : A <: B \rightsquigarrow \{\}}$$

$$\textsc{S-Bot}\quad \frac{B\ \mathsf{Ordinary} \qquad \neg\rceil B \lceil}{t : \bot <: B \rightsquigarrow \{|B| \Rightarrow \mathsf{fix}\, x.x\}}$$

$$\textsc{S-Base}\quad \frac{}{t : \mathbb{B} <: \mathbb{B} \rightsquigarrow \{|\mathbb{B}| \Rightarrow t.|\mathbb{B}|\}}$$

$$\textsc{S-Arrow}\quad \frac{B_2\ \mathsf{Ordinary} \qquad \neg\rceil B_2 \lceil \qquad x : B_1 <: A_1 \rightsquigarrow t_1 \ \wedge\ (t.|A_1 \to A_2|)\, t_1 : A_2 <: B_2 \rightsquigarrow t_2}{t : A_1 \to A_2 <: B_1 \to B_2 \rightsquigarrow \{|B_1 \to B_2| \Rightarrow \lambda x.t_2\}}$$

$$\textsc{S-Rcd}\quad \frac{B\ \mathsf{Ordinary} \qquad \neg\rceil B \lceil \qquad t.|\{l : A\}| : A <: B \rightsquigarrow t_2}{t : \{l : A\} <: \{l : B\} \rightsquigarrow \{|\{l : B\}| \Rightarrow t_2\}}$$

$$\textsc{S-AndL}\quad \frac{C\ \mathsf{Ordinary} \qquad t : A <: C \rightsquigarrow t'}{t : A \,\&\, B <: C \rightsquigarrow t'}$$

$$\textsc{S-AndR}\quad \frac{C\ \mathsf{Ordinary} \qquad t : B <: C \rightsquigarrow t'}{t : A \,\&\, B <: C \rightsquigarrow t'}$$

$$\textsc{S-Split}\quad \frac{B_1 \lhd B \rhd B_2 \qquad t : A <: B_1 \rightsquigarrow t_1 \qquad t : A <: B_2 \rightsquigarrow t_2 \qquad t_1 : B_1 \rhd B \lhd t_2 : B_2 \rightsquigarrow t_3}{t : A <: B \rightsquigarrow t_3}$$

$\boxed{t_1 : A \rhd C \lhd t_2 : B \rightsquigarrow t}$ *(Coercive merging)*

$$\textsc{M-Top}\quad \frac{\rceil C \lceil \qquad A \lhd C \rhd B}{t_1 : A \rhd C \lhd t_2 : B \rightsquigarrow \{\}}$$

$$\textsc{M-And}\quad \frac{\neg\rceil A \,\&\, B \lceil}{t_1 : A \rhd A \,\&\, B \lhd t_2 : B \rightsquigarrow t_1; t_2}$$

$$\textsc{M-Arrow}\quad \frac{\neg\rceil B_1 \lceil \qquad \neg\rceil B_2 \lceil \qquad (t_1.|A \to B_1|)\, x : B_1 \rhd B \lhd (t_2.|A \to B_2|)\, x : B_2 \rightsquigarrow t}{t_1 : A \to B_1 \rhd A \to B \lhd t_2 : A \to B_2 \rightsquigarrow \{|A \to B| \Rightarrow \lambda x.t\}}$$

$$\textsc{M-ArrowL}\quad \frac{\neg\rceil B_1 \lceil \qquad \rceil B_2 \lceil \qquad (t_1.|A \to B_1|)\, x : B_1 \rhd B \lhd \{\} : B_2 \rightsquigarrow t}{t_1 : A \to B_1 \rhd A \to B \lhd t_2 : A \to B_2 \rightsquigarrow \{|A \to B| \Rightarrow \lambda x.t\}}$$

M-ArrowR

$$\frac{\rceil B_1 \lceil \qquad \neg\rceil B_2 \lceil \qquad \{\} : B_1 \rhd B \lhd (t_2.|A \to B_2|)\, x : B_2 \rightsquigarrow t}{t_1 : A \to B_1 \rhd A \to B \lhd t_2 : A \to B_2 \rightsquigarrow \{|A \to B| \Rightarrow \lambda x.t\}}$$

M-Rcd

$$\frac{\neg\rceil A_1 \lceil \qquad \neg\rceil A_2 \lceil \qquad t_1.|\{l : A_1\}| : A_1 \rhd A \lhd t_2.|\{l : A_2\}| : A_2 \rightsquigarrow t}{t_1 : \{l : A_1\} \rhd \{l : A\} \lhd t_2 : \{l : A_2\} \rightsquigarrow \{|\{l : A\}| \Rightarrow t\}}$$

M-RcdL

$$\frac{\neg\rceil A_1 \lceil \qquad \rceil A_2 \lceil \qquad t_1.|\{l : A_1\}| : A_1 \rhd A \lhd \{\} : A_2 \rightsquigarrow t}{t_1 : \{l : A_1\} \rhd \{l : A\} \lhd t_2 : \{l : A_2\} \rightsquigarrow \{|\{l : A\}| \Rightarrow t\}}$$

M-RcdR

$$\frac{\rceil A_1 \lceil \qquad \neg\rceil A_2 \lceil \qquad \{\} : A_1 \rhd A \lhd t_2.|\{l : A_2\}| : A_2 \rightsquigarrow t}{t_1 : \{l : A_1\} \rhd \{l : A\} \lhd t_2 : \{l : A_2\} \rightsquigarrow \{|\{l : A\}| \Rightarrow t\}}$$

$\boxed{t_1 : A; t_2 : B \rightsquigarrow t_3 : C}$ *(Distributive application)*

A-Top

$$\frac{\rceil A \lceil}{t_1 : A; t_2 : B \rightsquigarrow \{\} : \top}$$

A-Arrow

$$\frac{\neg\rceil B \lceil \qquad t_2 : C <: A \rightsquigarrow t_3}{t_1 : A \to B; t_2 : C \rightsquigarrow (t_1.|A \to B|)\, t_3 : B}$$

A-And

$$\frac{t_1 : A; t_2 : C \rightsquigarrow t_3 : A' \qquad t_1 : B; t_2 : C \rightsquigarrow t_4 : B'}{t_1 : A \,\&\, B; t_2 : C \rightsquigarrow t_3; t_4 : A' \,\&\, B'}$$

2.3 Auxiliary definitions

$$\frac{A_{k_1} < A_{k_2} < \cdots < A_{k_m} \qquad \neg\rceil A_k \lceil}{|A_1 \& A_2 \& \cdots \& A_n| = |A_{k_1}|\&|A_{k_2}|\& \cdots \&|A_{k_m}|}$$

$\boxed{\rceil A \lceil}$ *(Top-like Types)*

TL-top

$$\frac{}{\rceil \top \lceil}$$

TL-and

$$\frac{\rceil A \lceil \qquad \rceil B \lceil}{\rceil A \,\&\, B \lceil}$$

TL-arr

$$\frac{\rceil B \lceil}{\rceil A \to B \lceil}$$

TL-rcd

$$\frac{\rceil B \lceil}{\rceil \{l : B\} \lceil}$$

$\boxed{A \textsf{ Ordinary}}$ *(Ordinary Types)*

O-top

$$\frac{}{\top \textsf{ Ordinary}}$$

O-int

$$\frac{}{\mathbb{B} \textsf{ Ordinary}}$$

O-arrow

$$\frac{B \textsf{ Ordinary}}{A \to B \textsf{ Ordinary}}$$

O-rcd

$$\frac{B \textsf{ Ordinary}}{\{l : B\} \textsf{ Ordinary}}$$

$\boxed{A * B}$ *(Type Disjointness (Algorithmic))*

$$\textsc{D-topL}\ \frac{}{\top * A} \qquad \textsc{D-topR}\ \frac{}{A * \top} \qquad \textsc{D-andL}\ \frac{A_1 * B \qquad A_2 * B}{A_1 \,\&\, A_2 * B} \qquad \textsc{D-andR}\ \frac{A * B_1 \qquad A * B_2}{A * B_1 \,\&\, B_2} \qquad \textsc{D-BaseArr}\ \frac{}{\mathbb{B} * A_1 \to A_2} \qquad \textsc{D-ArrBase}\ \frac{}{A_1 \to A_2 * \mathbb{B}}$$

$$\textsc{D-ArrArr}\ \frac{A_2 * B_2}{A_1 \to A_2 * B_1 \to B_2} \qquad \textsc{D-rcdEq}\ \frac{A * B}{\{l : A\} * \{l : B\}} \qquad \textsc{D-rcdNeq}\ \frac{l_1 \neq l_2}{\{l_1 : A\} * \{l_2 : B\}} \qquad \textsc{D-BaseRcd}\ \frac{}{\mathbb{B} * \{l : A\}} \qquad \textsc{D-RcdBase}\ \frac{}{\{l : A\} * \mathbb{B}}$$

$$\textsc{D-ArrRcd}\ \frac{}{A_1 \to A_2 * \{l : A\}} \qquad \textsc{D-RcdArr}\ \frac{}{\{l : A\} * A_1 \to A_2}$$

$\boxed{B \lhd A \rhd C}$ *(Split a Type into Two)*

$$\textsc{Sp-and}\ \frac{}{(A) \lhd (A \,\&\, B) \rhd (B)} \qquad \textsc{Sp-arrow}\ \frac{B_1 \lhd B \rhd B_2}{(A \to B_1) \lhd (A \to B) \rhd (A \to B_2)} \qquad \textsc{Sp-rcd}\ \frac{B_1 \lhd B \rhd B_2}{\{l : B_1\} \lhd \{l : B\} \rhd \{l : B_2\}}$$

$\boxed{At <: Bt}$ *(Target Subtyping)*

$$\textsc{TS-top}\ \frac{\vdash At}{At <: \top} \qquad \textsc{TS-refl}\ \frac{\vdash At}{At <: At} \qquad \textsc{TS-arrow}\ \frac{At_1 <: At_2 \qquad Bt_1 <: Bt_2}{At_1 \to Bt_1 <: At_2 \to Bt_2}$$

$$\textsc{TS-rcd}\ \frac{\begin{array}{c}\textbf{recTyp}\ Ct' \\ \textbf{lookup}\ ll\ Ct \Rightarrow Bt \\ Bt <: At \\ (\textbf{lookup}\ ll\ Ct' \Rightarrow At' \ \wedge\ At' \approx At) \ \vee\ \textbf{lookup}\ ll\ Ct' \not\Rightarrow \\ Ct <: Ct'\end{array}}{Ct <: \{ll : At, Ct'\}}$$

```
1) type soundness of the target calculus

2) connecting the source to the target

- type translation function (from source type to string):
intersection types are flattened, their
toplike parts are filtered, and all the conjuncts are sorted (but no deduplication).

- a relation that characterizes types that corresponds to the same
string.

3) I need a function that converts source types to target types, to prove that
elaboration generates well-typed terms. There are 3 problems:
- Source types and target types are not one-to-one mapped. The typing rule for
toplike lambda functions does not split types but the subsumption rule does
(via subtyping)
\x:Int . x : Top&Top ~> { }
(\x:Int . x : Int) : Int->Top&Top ~> { ->|Top&Top| => \ x . ... }
```

Lemma eqIndTyp_sound_complete: $A \ = B$ iff $|A| = |B|$.

Lemma disjoint_type_no_eqInd: if $A \ = B$ and $A * B$ then there is a contradiction.

Lemma translation from source type to target type is a function

Lemma cosub_well_typed: if $t_1 : A <: B \rightsquigarrow t_2$ and $G|| \vdash t1 <= |[A]|$ then $G|| \vdash t2 : |[B]|$.

Theorem elaboration_well_typed: if $\Gamma \vdash e \ \mathit{dirflag} \ A \rightsquigarrow t$ then $|[G]|| \vdash t : |[A]|$.