Branching Types

Joe Wells and Christian Haack

February 26, 2002

Intersection Types

$$M, N \in \mathsf{UntypedTerm} \quad ::= \quad x \mid \lambda x.M \mid M\ N$$

$$\sigma, \tau \in \mathsf{IntTy} \quad ::= \quad \alpha \mid \sigma \to \tau \mid \sigma \wedge \tau$$

$$(\wedge_{\mathsf{i}}) \qquad \frac{E \vdash M : \sigma; \quad E \vdash M : \tau}{E \vdash M : \sigma \wedge \tau}$$

$$(\wedge_{\mathsf{e}}) \qquad \frac{E \vdash M : \tau_1 \wedge \tau_2}{E \vdash M : \tau_i} \quad \text{for } i \in \{1, 2\}$$

Explicit Types: λ^{CIL}

Virtual tuples and projections:

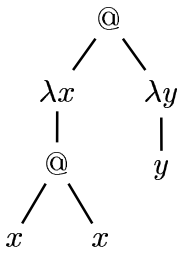
$$M, N \;::=\; \ldots \;\mid\; \wedge(M, N) \mid\; \pi_1^{\wedge} M \;\mid\; \pi_2^{\wedge} M \mid\; \ldots$$

$$(\wedge_{\mathsf{i}}) \qquad \frac{E \vdash M : \sigma; \quad E \vdash N : \tau}{E \vdash \wedge(M, N) : \sigma \wedge \tau} \quad \text{if } |M| \equiv |N|$$

$$(\wedge_{\mathsf{e}}) \qquad \frac{E \vdash M : \tau_1 \wedge \tau_2}{E \vdash \pi_i^{\wedge} M : \tau_i} \quad \text{for } i \in \{1, 2\}$$

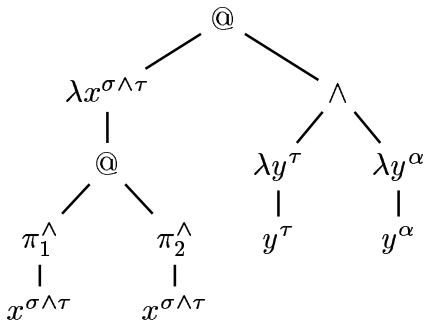
Example: An Untyped Term

$$(\lambda x.x\ x)\ (\lambda y.y)$$



Example: A Corresponding λ^{CIL}-term

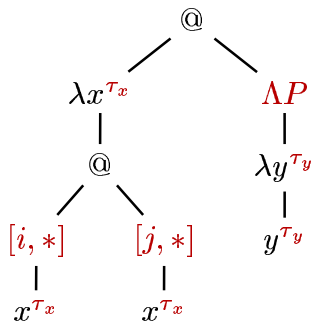
$(\lambda x.x\ x)\ (\lambda y.y)$



where $\tau = \alpha \to \alpha$ and $\sigma = \tau \to \tau$

Example: A Corresponding λ^{B}-term

$(\lambda x.x\ x)\ (\lambda y.y)$

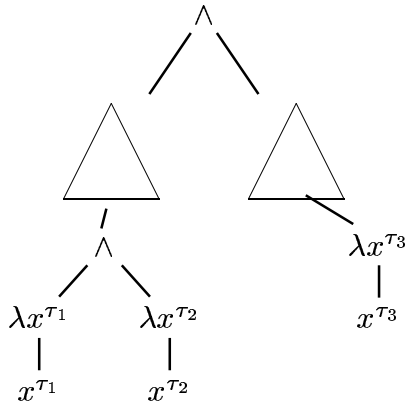


where

$$
\begin{aligned}
\tau &= \alpha \to \alpha \\
\sigma &= \tau \to \tau \\
\tau_y &= \{i = \tau,\ j = \alpha\} \\
type(\lambda y^{\tau_y}.y^{\tau_y}) &= \tau_y \to \tau_y \\
&\simeq \{i = \tau \to \tau,\ j = \alpha \to \alpha\} \\
P &= \mathsf{join}\{i = *, j = *\} \\
type(\Lambda P.\lambda y^{\tau_y}.\tau_y) &= \forall P.(\tau_y \to \tau_y) \\
\tau_x &= \forall P.(\tau_y \to \tau_y)
\end{aligned}
$$

Nested Branching Types

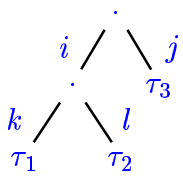
Consider λ^{CIL}-term of this shape:



In λ^{B}, the variable x is annotated by this branching type:

$$\{i = \{k = \tau_1, l = \tau_2\}, j = \tau_3\}$$

This type may be viewed as a tree:



Notational Conventions

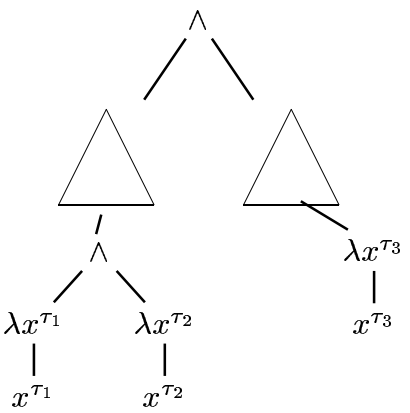
We assume that $\mathcal{I}$ is some fixed set of labels.

We use I to range over non-empty, finite subsets of $\mathcal{I}$.

We use i, j, k, l to range over elements of $\mathcal{I}$.

We use $\{i = v_i\}^I$ to denote the non-empty, finite function $\{\, (i, v_i) \mid i \in I \,\}$.

Kinds



Typing judgments:

$$E \vdash M : \tau \text{ at } \kappa$$

Kinds (or branching shapes):

$$\kappa \in \mathsf{Kind} \quad ::= \quad * \;\mid\; \{i = \kappa_i\}^I$$

κ provides a "duplication environment" for M.

$\lambda x.x$'s duplication environment:

$$\{i = \{k = *,\, l = *\},\, j = *\}$$

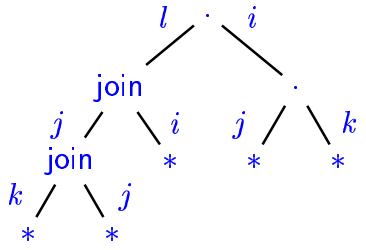
Type Selection Parameters

$$\bar{P} \in \mathsf{IndParameter} \quad ::= \quad * \;\mid\; \mathsf{join}\{i = \bar{P}_i\}^I$$
$$P \in \mathsf{Parameter} \quad ::= \quad \bar{P} \;\mid\; \{i = P_i\}^I$$

Example:

$$\{l = \mathsf{join}\{j = \mathsf{join}\{k = *,\, j = *\},\, i = *\},\, i = \{j = *,\, k = *\}\}$$

Its tree representation:



Branching Types

$$\sigma, \tau \in \mathsf{Type} \quad ::= \quad \alpha \mid \sigma \to \tau \mid \{i = \tau_i\}^I \mid \forall P.\tau$$

Type equivalence:

$$\begin{aligned} \{i = \sigma_i\}^I \to \{i = \tau_i\}^I &\succ \{i = \sigma_i \to \tau_i\}^I \\ \forall \{i = P_i\}^I . \{i = \tau_i\}^I &\succ \{i = \forall P_i.\tau_i\}^I \\ \forall *.\tau &\succ \tau \end{aligned}$$

$\succeq$: reflexive and transtitive closure of $\succ$

$\simeq$: reflexive, transitive and symmetric closure of $\succ$

The type reduction rules "bring a type's branching shape to the surface".

Branching Types (continued)

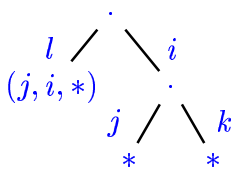
Proposition. Type reduction is confluent and terminating.

Proposition. If $(E \vdash M : \tau \ \mathsf{at}\ \kappa)$ and τ is in normal form, then τ's outer structure matches κ.

Type Selectors

$$\begin{aligned} \bar{A} \in \mathsf{IndArgument} &::= * \mid i, \bar{A} \\ A \in \mathsf{Argument} &::= \bar{A} \mid \{i = A_i\}^I \end{aligned}$$

Example:



Type Selection

$$\mathsf{select}^{\mathsf{i}} : \mathsf{Type} \times \mathsf{Argument} \to \mathsf{Type} \qquad \text{(partial function)}$$

If type argument is normal:

$$\begin{aligned} \mathsf{select}^{\mathsf{i}}(\tau, *) &= \tau, \quad \text{if } \tau \text{ is individual} \\ \mathsf{select}^{\mathsf{i}}(\forall(\mathsf{join}\{i = \bar{P}_i\}^I).\{i = \tau_i\}^I, (j, \bar{A})) &= \mathsf{select}^{\mathsf{i}}(\forall \bar{P}_j.\tau_j, \bar{A}), \quad \text{if } (j \in I) \\ \mathsf{select}^{\mathsf{i}}(\{i = \tau_i\}^I, \{i = A_i\}^I) &= \{i = \mathsf{select}^{\mathsf{i}}(\tau_i, A_i)\}^I \end{aligned}$$

If τ not normal:

$$\mathsf{select}^{\mathsf{i}}(\tau, A) = \mathsf{select}^{\mathsf{i}}(\mathsf{nf}(\tau), A)$$

Terms and Typing Rules

$$M, N \in \mathsf{Term} \quad ::= \quad \Lambda P.M \;\mid\; M[A] \;\mid\; \lambda x^{\tau}.M \;\mid\; M\ N \;\mid\; x^{\tau}$$

Conversion rule:

$$(\simeq) \qquad \frac{E \vdash M : \tau \ \mathsf{at}\ \kappa}{E \vdash M : \tau' \ \mathsf{at}\ \kappa} \quad \text{if } (\tau \simeq \tau')$$

Standard rules:

$$(\mathsf{ax}) \qquad \frac{}{E \vdash x^{\tau} : \tau \ \mathsf{at}\ \kappa} \quad \text{if } (E : \kappa) \text{ and } (\tau \simeq E(x))$$

$$(\rightarrow_{\mathsf{i}}) \qquad \frac{E[x \mapsto \sigma] \vdash M : \tau \ \mathsf{at}\ \kappa}{E \vdash \lambda x^{\sigma}.M : \sigma \rightarrow \tau \ \mathsf{at}\ \kappa}$$

$$(\rightarrow_{\mathsf{e}}) \qquad \frac{E \vdash M : \sigma \rightarrow \tau \ \mathsf{at}\ \kappa; \quad E \vdash N : \sigma \ \mathsf{at}\ \kappa}{E \vdash M\ N : \tau \ \mathsf{at}\ \kappa}$$

Terms and Typing Rules (cont.)

$\forall$-rules:

$$(\forall_{\mathsf{e}}) \qquad \frac{E \vdash M : \tau \ \mathsf{at}\ \kappa}{E \vdash M[A] : \tau' \ \mathsf{at}\ \kappa} \quad \text{if } (\mathsf{select}^{\mathsf{i}}(\tau, A) = \tau')$$

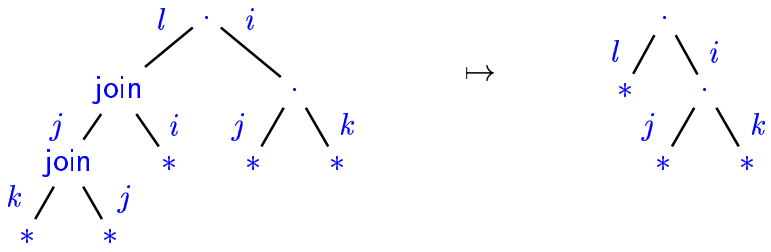
$$(\forall_{\mathsf{i}}) \qquad \frac{\mathsf{expand}(E, P) \vdash M : \tau \ \mathsf{at}\ \lceil P \rceil}{E \vdash \Lambda P.M : \forall P.\tau \ \mathsf{at}\ \lfloor P \rfloor}$$

Outer Kinds of Parameters

$\lfloor\cdot\rfloor$: Parameter $\to$ Kind

$$\lfloor * \rfloor = *, \qquad \lfloor \mathsf{join}\{i = \bar{P}_i\}^I \rfloor \;=\; *, \qquad \lfloor \{i = P_i\}^I \rfloor = \{i = \lfloor P_i \rfloor\}^I$$

Example:

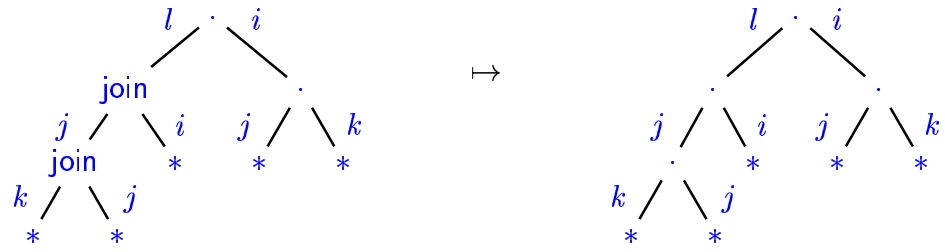


Inner Kinds of Parameters

$\lceil\cdot\rceil$: Parameter $\to$ Kind

$$\lceil * \rceil = *, \qquad \lceil \mathsf{join}\{i = \bar{P}_i\}^I \rceil \;=\; \{i = \lceil \bar{P}_i \rceil^I\}, \qquad \lceil \{i = P_i\}^I \rceil = \{i = \lceil P_i \rceil\}^I$$

Example:



Type Expansion

$$\mathsf{expand} : \mathsf{Type} \times \mathsf{Parameter} \to \mathsf{Type} \qquad \text{(partial function)}$$

$$\begin{array}{rcll}
\mathsf{expand}(\tau, *) & = & \tau, & \text{if } \mathsf{normal}(\tau) \\
\mathsf{expand}(\tau, \mathsf{join}\{i = \bar{P}_i\}^I) & = & \{i = \mathsf{expand}(\tau, \bar{P}_i)\}^I, & \text{if } \mathsf{normal}(\tau) \\
\mathsf{expand}(\{i = \tau_i\}^I, \{i = P_i\}^I) & = & \{i = \mathsf{expand}(\tau_i, P_i)\}^I, & \text{if } \mathsf{normal}(\{i = \tau_i\}^I) \\
\mathsf{expand}(\tau, P) & = & \mathsf{expand}(\mathsf{nf}(\tau), P), & \text{if } \neg\mathsf{normal}(\tau)
\end{array}$$

Second clause duplicates τ.

$\mathsf{expand}(\tau, P)$ has branching shape $\lceil P \rceil$.

Term Reduction

Substitution:

$$\begin{array}{rcl}
 & \ldots & \\
(\Lambda P.M)[x := N] & = & \Lambda P.\ (M[x := \mathsf{expand}(N, P)]) \\
 & \ldots &
\end{array}$$

Reduction rules:

$$\begin{array}{lrcl}
(\beta_\lambda) & ((\lambda x^\tau.M)N) & \to & (M[x := N]) \\
(\beta_\Lambda) & (\Lambda P.M)[A] & \to & \Lambda P'.((\mathsf{select}^{\mathsf{b}}(M, A^{\mathsf{s}}))[A^{\mathsf{a}}]), \\
 & \multicolumn{3}{l}{\text{if } \mathsf{match}(P, A) = (P', A^{\mathsf{s}}, A^{\mathsf{a}}) \text{ and neither } P \text{ nor } A \text{ is trivial}} \\
(*_\Lambda) & (\Lambda P.M) & \to & M, \quad \text{if } P \text{ is trivial} \\
(*_A) & (M[A]) & \to & M, \quad \text{if } A \text{ is trivial}
\end{array}$$

Term Reduction (cont.)

Theorem (Subject Reduction).
If $(M \to N)$ and $(E \vdash M : \tau \text{ at } \kappa)$, then $(E \vdash N : \tau \text{ at } \kappa)$.

Type erasure: $|\cdot| : \mathsf{Term} \to \mathsf{UntypedTerm}$

Theorem (Soundness of Reduction).
If $M \to N$, then $|M| \to^* |N|$.

Theorem (Completeness of Reduction).
If M is well-typed and $|M| \to |N|$, then $(M \to^* N)$.