Subtle Points

James, Fred and Conor, probably

March 11, 2016

We have a system of *kinds*, kind checking and kind synthesis. Contexts assign kinds to free variables. We write $\dashv x : K$ to assert such an assignment. We write context lookups and context extensions in rules, but never the contexts themselves.

$$\frac{e \in J \quad J \leq K}{K \ni \underline{e}} \qquad \frac{\dashv x : K}{x \in K} \qquad \frac{\mathsf{KIND} \ni K \quad K \ni t}{t{:}K \in K} \qquad \underline{t{:}K} \rightsquigarrow t$$

Subkinding is certainly reflexive.

$$\frac{}{K \leq K}$$

We shall ensure that transitivity is admissible, and that so is subsumption.

$$\textbf{Admissibly,} \qquad \frac{I \leq J \quad J \leq K}{I \leq K} \qquad \frac{J \ni t \quad J \leq K}{K \ni t}$$

We also allow kinds to compute before checking and after synthesis.

$$\frac{J \rightsquigarrow K \quad K \ni t}{J \ni t} \qquad \frac{e \in J \quad J \rightsquigarrow K}{e \in K}$$

1 type theory as we know it

Types are a kind. Each type gives rise to the kind of its elements.

$$\frac{}{\mathsf{KIND} \ni \mathsf{TYPE}} \qquad \frac{\mathsf{TYPE} \ni T}{\mathsf{KIND} \ni \mathsf{EL}\, T}$$

Function types work like this.

$$\frac{x : \mathsf{EL}\, S \vdash \mathsf{TYPE} \ni T}{\mathsf{TYPE} \ni (x{:}S) \to T} \qquad \frac{x : \mathsf{EL}\, S \vdash \mathsf{EL}\, T \ni t}{\mathsf{EL}\, (x{:}S) \to T \ni \lambda\, x \to t}$$

$$\frac{f \in \mathsf{EL}\, (x{:}S) \to T[x] \quad \mathsf{EL}\, S \ni s}{f\ s \in \mathsf{EL}\, T[s{:}S]}$$

$$(\lambda\, x \to t[x] : \mathsf{EL}\, (x{:}S) \to T[x])\ s \rightsquigarrow t(s{:}\mathsf{EL}\, S) : \mathsf{EL}\, T(s{:}\mathsf{EL}\, S)$$

$$\frac{\mathsf{EL}\, S' \leq \mathsf{EL}\, S \quad x : \mathsf{EL}\, S' \vdash \mathsf{EL}\, T \leq \mathsf{EL}\, T'}{\mathsf{EL}\, (x{:}S) \to T \leq \mathsf{EL}\, (x{:}S') \to T'}$$

Pair types work like this.

$$\frac{x : \mathsf{EL}\ S \vdash \mathsf{TYPE} \ni T}{\mathsf{TYPE} \ni (x{:}S) \times T} \qquad \frac{\mathsf{EL}\ S \ni s \qquad \mathsf{EL}\ T[s{:}\mathsf{EL}\ S] \ni t}{\mathsf{EL}\ (x{:}S) \times T[x] \ni s, t}$$

$$\frac{p \in \mathsf{EL}\ (x{:}S) \times T}{p\ \mathsf{car} \in \mathsf{EL}\ S} \qquad \frac{p \in \mathsf{EL}\ (x{:}S) \times T[x]}{p\ \mathsf{cdr} \in \mathsf{EL}\ T[s{:}S]}$$

$$(s, t : \mathsf{EL}\ (x{:}S) \times T)\ \mathsf{car} \rightsquigarrow s{:}\mathsf{EL}\ S$$

$$(s, t : \mathsf{EL}\ (x{:}S) \times T[x])\ \mathsf{cdr} \rightsquigarrow t{:}\mathsf{EL}\ T[p\ \mathsf{car}]$$

$$\frac{\mathsf{EL}\ S \leq \mathsf{EL}\ S' \quad x : \mathsf{EL}\ S \vdash \mathsf{EL}\ T \leq \mathsf{EL}\ T'}{\mathsf{EL}\ (x{:}S) \times T \leq \mathsf{EL}\ (x{:}S') \times T'}$$

There will probably be more stuff.

2 segmentations and points

Segmentations are a kind, and so are points in segmentations.

$$\frac{}{\mathsf{KIND} \ni \mathsf{SEG}} \qquad \frac{\mathsf{SEG} \ni \sigma}{\mathsf{KIND} \ni \mathsf{POINT}\ \sigma}$$

There is a trivial segmentations. All segmentations have endpoints.

$$\frac{}{\mathsf{SEG} \ni -} \qquad \frac{}{\mathsf{POINT}\ \sigma \ni 0} \qquad \frac{}{\mathsf{POINT}\ \sigma \ni 1}$$

But there are also nontrivial segmentations.

$$\frac{\mathsf{SEG} \ni \sigma \quad \mathsf{TYPE} \ni T \quad \mathsf{SEG} \ni \tau}{\mathsf{SEG} \ni \{\sigma T \tau\}}$$

These have inner points.

$$\frac{\mathsf{POINT}\ \sigma \ni p}{\mathsf{POINT}\ \{\sigma T \tau\} \ni \blacktriangleleft p} \qquad \frac{\mathsf{POINT}\ \tau \ni p}{\mathsf{POINT}\ \{\sigma T \tau\} \ni \blacktriangleright p}$$

The equational theory of points admits

$$\blacktriangleleft 0 \rightsquigarrow 0 \qquad \blacktriangleleft 1 \rightsquigarrow \blacktriangleright 0 \qquad \blacktriangleright 1 \rightsquigarrow 1$$

Each segmentation has its opposite. There are no neutral segmentations, so this involutive operation computes fully.

$$-^\circ = - \qquad (\{\sigma T \tau\})^\circ = \{\tau^\circ T \sigma^\circ\}$$

However, there will be variables which stand for points, so opposite is an eliminator for points.

$$\frac{e \in \mathsf{POINT}\ \sigma}{e^\circ \in \mathsf{POINT}\ \sigma^\circ}$$

It computes as follows.

$$(0 : \mathsf{POINT}\ \sigma)^\circ \rightsquigarrow 1 : \mathsf{POINT}\ \sigma^\circ$$
$$(\blacktriangleleft p : \mathsf{POINT}\ \{\sigma T \tau\})^\circ \rightsquigarrow \blacktriangleright (p : \mathsf{POINT}\ \sigma) : \mathsf{POINT}\ \{\tau^\circ T \sigma^\circ\}$$
$$(\blacktriangleright p : \mathsf{POINT}\ \{\sigma T \tau\})^\circ \rightsquigarrow \blacktriangleleft (p : \mathsf{POINT}\ \tau) : \mathsf{POINT}\ \{\tau^\circ T \sigma^\circ\}$$
$$(1 : \mathsf{POINT}\ \sigma)^\circ \rightsquigarrow 0 : \mathsf{POINT}\ \sigma^\circ$$
$$(i^\circ)^\circ \rightsquigarrow i$$

Note that a neutral point is either a point variable or its opposite.

Segmentations are ordered by *subtlety*, which induces subkinding.

$$\frac{}{- \le \sigma} \qquad \frac{\sigma \le \sigma' \quad \mathsf{TYPE} \ni T \equiv T' \quad \tau \le \tau'}{\{\sigma T \tau\} \le \{\sigma' T' \tau'\}} \qquad \frac{\sigma \le \tau}{\mathsf{POINT}\ \sigma \le \mathsf{POINT}\ \tau}$$

3 type paths

A type path type is given as a segmentation between two types.

$$\frac{\mathsf{TYPE} \ni S \quad \mathsf{SEG} \ni \sigma \quad \mathsf{TYPE} \ni T}{\mathsf{TYPE} \ni S \sigma T}$$

A type path is an abstraction over the points in the segmentation which connects with the segmentation at all points.

$$\frac{i : \mathsf{POINT}\ \sigma \vdash \mathsf{TYPE} \ni T[i] \quad \mathsf{TYPE} \ni S \equiv T[0] \quad T[\cdot] \bowtie \sigma \quad \mathsf{TYPE} \ni T[1] \equiv U}{\mathsf{EL}\ S \sigma U \ni \lambda\, i \to T}$$

Connection is given as follows:

$$\frac{}{T[\cdot] \bowtie -} \qquad \frac{T[\blacktriangleleft \cdot] \bowtie \sigma \quad \mathsf{TYPE} \ni S \equiv T[\blacktriangleright 0] \quad T[\blacktriangleright \cdot] \bowtie \tau}{T[\cdot] \bowtie \{\sigma S \tau\}}$$

Subtlety induces subtyping contravariantly, meaning that you can forget the existence of (hence connection with) intermediate points.

$$\frac{\sigma \le \tau}{\mathsf{EL}\ S \tau T \le \mathsf{EL}\ S \sigma T}$$

The elimination behaviour is by substitution.

$$(\lambda\, i \to T[i] : \mathsf{EL}\ S \sigma U)\ p \rightsquigarrow T[p : \mathsf{POINT}\ \sigma] : \mathsf{TYPE}$$

The type path type tells us the types at all its *canonical* points.

$$(S \sigma U).0 = S \qquad (S\{\sigma T \tau\}U).\blacktriangleleft p = (S \sigma T).p$$
$$(S\{\sigma T \tau\}U).\blacktriangleright p = (T \tau U).p \qquad (S \sigma U).1 = U$$

So, given that type reconstruction is easy for neutrals, we may have

$$Q \in \mathsf{EL}\ S \sigma U \wedge (S \sigma U).p = T \quad \Rightarrow \quad Q\, p \rightsquigarrow T : \mathsf{TYPE}$$

If we know that a path has nontrivial segmentation, we can grab its segments. Let's write $Q\blacktriangleleft$ for the path $\lambda\, i \to Q\ (\blacktriangleleft i)$ which is just the left segment of Q, and similarly $Q\blacktriangleright$ for the right segment.

4 kinky paths

Next, we give you form of conditional expression to compute types from points in a piecewise continuous way.

$$\frac{\begin{array}{cccccc} p \in \mathsf{POINT}\ \pi & \mathsf{TYPE} \ni S & \mathsf{SEG} \ni \sigma & \mathsf{TYPE} \ni T & \mathsf{SEG} \ni \tau & \mathsf{TYPE} \ni U \\ \pi \leq \{\sigma T \tau\} & & \mathsf{EL}\ S\sigma T \ni P & & \mathsf{EL}\ T\tau U \ni Q & \end{array}}{p\ \mathsf{kink}(S\sigma T\tau U | P\ Q) \in \mathsf{TYPE}}$$

The conditional computes when we learn which half the point is in. Of course, the two halves meet in the middle.

$$\begin{array}{rl} (0 : \mathsf{POINT}\ \pi)\ \mathsf{kink}(S\_\_\_\_ | P\ Q) & \rightsquigarrow S : \mathsf{TYPE} \\ (\blacktriangleleft p : \mathsf{POINT}\ \sigma T \tau)\ \mathsf{kink}(S\_\_\_U | P\ Q) & \rightsquigarrow (P : \mathsf{EL}\ S\sigma T)\ p \\ (\blacktriangleright p : \mathsf{POINT}\ \sigma T \tau)\ \mathsf{kink}(S\_\_\_U | P\ Q) & \rightsquigarrow (Q : \mathsf{EL}\ T\tau U)\ p \\ (1 : \mathsf{POINT}\ \pi)\ \mathsf{kink}(\_\_\_\_U | P\ Q) & \rightsquigarrow U : \mathsf{TYPE} \end{array}$$

Now, the subtlety is that the kinky construction can be more subtle than the point being projected from it: that just means we actually know more types at more points than we can be asked about. We can allow points that we're definitely not going to get asked for to wobble about a bit.

$$\begin{array}{l} p \in \mathsf{POINT}\ - \wedge i : \mathsf{POINT}\ \sigma \vdash \mathsf{TYPE} \ni S \equiv (P : \mathsf{EL}\ S\sigma T)\ i \\ \quad \Rightarrow \quad p\ \mathsf{kink}(S\sigma T\tau U | P\ Q) \rightsquigarrow (Q : \mathsf{EL}\ T\tau U)\ p \end{array}$$

If p can only refer to an *endpoint* and P is *constant* (and equal to both S and T) we can effective yank the midpoint all the way left, taking Q as the whole path. Similarly,

$$\begin{array}{l} p \in \mathsf{POINT}\ - \wedge i : \mathsf{POINT}\ \sigma \vdash \mathsf{TYPE} \ni (Q : \mathsf{EL}\ T\tau U)\ i \equiv U \\ \quad \Rightarrow \quad p\ \mathsf{kink}(S\sigma T\tau U | P\ Q) \rightsquigarrow (P : \mathsf{EL}\ S\sigma T)\ p \end{array}$$

Moreover, if we're sure we're only going to project an endpoint, we can standardise the segmentation structure by shuffling kinks to the right.

$$\begin{array}{l} p \in \mathsf{POINT}\ - \wedge \\ i : \mathsf{POINT}\ \sigma \vdash \mathsf{TYPE} \ni (P : \mathsf{EL}\ S\sigma T)\ i \equiv i\ \mathsf{kink}(S\sigma_0 S' \sigma_1 T | P_0\ P_1) \\ \quad \Rightarrow \quad p\ \mathsf{kink}(S\sigma T\tau U | P\ Q) \\ \quad\quad \rightsquigarrow \quad p\ \mathsf{kink}(S\sigma_0 S' \{\sigma_1 T \tau\} U | P_0\ \lambda\, i \to i\ \mathsf{kink}(S' \sigma_1 T \tau U | P_1\ Q)) \end{array}$$

So we have acquired categorical structure for trivially segmented paths $S{-}T$. If we have $\mathsf{EL}\ S{-}T \ni P$ and $\mathsf{EL}\ T{-}U \ni Q$, we get their composite

$$\mathsf{EL}\ S{-}U \geq \mathsf{EL}\ S\{-T-\}U \ni \lambda\, i \to i\ \mathsf{kink}(S{-}T{-}U | P\ Q)$$

where this composition absorbs identities (constant paths) and is associative.

5 transporting values along type paths

Let us now figure out how to transport values between points on a type path.

$$\frac{\mathsf{EL}\ S\sigma T \ni Q \quad \mathsf{POINT}\ \sigma \ni p \quad \mathsf{POINT}\ \sigma \ni r \quad \mathsf{EL}\ Q\ p \ni u}{u(Q|p{\to}r) \in \mathsf{EL}\ Q\ r}$$

How should this work? For a start, the fact that we can decide equality of open things in point kinds means we can run on the spot.

$$u(Q|i{\to}i) \rightsquigarrow u : \mathsf{EL}\ Q\ i$$

We can also zoom. Here, matching against $\blacktriangleleft$ and $\blacktriangleright$ should be liberal in the sense that $\blacktriangleleft i$ matches 0 with $i = 0$ and $\blacktriangleright 0$ with $i = 1$, and similarly on the right.

$$u(Q|\blacktriangleleft i{\to}\blacktriangleleft j) \rightsquigarrow u(\lambda\, k \to Q\ (\blacktriangleleft k)|i{\to}j)$$
$$u(Q|\blacktriangleright i{\to}\blacktriangleright j) \rightsquigarrow u(\lambda\, k \to Q\ (\blacktriangleright k)|i{\to}j)$$

If we get a thing on each side, we go via the middle.

$$u(Q|\blacktriangleleft i{\to}\blacktriangleright j) \rightsquigarrow u(Q|\blacktriangleleft i{\to}\blacktriangleleft 1)(Q|\blacktriangleright 0{\to}\blacktriangleright j)$$
$$u(Q|\blacktriangleright i{\to}\blacktriangleleft j) \rightsquigarrow u(Q|\blacktriangleright i{\to}\blacktriangleright 0)(Q|\blacktriangleleft 1{\to}\blacktriangleleft j)$$

If i and j are distinct but neutral, it's ok to be stuck. Real work must happen when i and j are a permutation of 0 and 1, and $\sigma \equiv -$. That's as far as we can get by looking at the points.

We can also look at the type $Q\,k$ we get at some arbitrary point $k : \mathsf{POINT}\ \sigma$, which amounts to η-expanding, then peeking under the $\lambda\, k \to$. Kinky paths transport in stages, e.g.:

$$s(\lambda\, k \to k\ \mathsf{kink}(S\sigma T\tau U|P\ Q)|0{\to}1) \rightsquigarrow s(P|0{\to}1)(Q|0{\to}1)$$

(What if the head isn't k? It's ok to be stuck, I think, because s can't be canonical.)

When we have handy η-laws and structural paths, we can be quite aggressive and still resolve critical pairs.

$$\begin{array}{l} u(\lambda\, k \to (x : S[k]) \times T[k,x]|i{\to}j) \rightsquigarrow \\ \quad \text{let } s[k] = (u\ \mathsf{car})(\lambda\, k \to S[k]|i{\to}k);\ t[k] = (u\ \mathsf{cdr})(\lambda\, k \to T[k, s[k]]|i{\to}k) \\ \quad \text{in } s[j], t[j] \end{array}$$

The coherence comes from the fact that we can extrude a value across the whole type path from any point on it. Functions are just as easy.

$$\begin{array}{l} u(\lambda\, k \to (x : S[k]) \to T[k,x]|i{\to}j) \rightsquigarrow \lambda\, x \to \\ \quad \text{let } s[k] = x(\lambda\, k \to S[k]|j{\to}k);\ t[k] = (u\ s[i])(\lambda\, k \to T[k, s[k]]|i{\to}k) \\ \quad \text{in } t[j] \end{array}$$

How about type paths? Composition!

$$Q(\lambda\, k \to S[k]{-}T[k]|0{\to}1) \rightsquigarrow (\lambda\, k \to S[k^\circ]) \circ Q \circ (\lambda\, k \to T[k])$$

And if we have nontrivial segmentation, that's ok, too. We can transport the segments separately, then kink them back together.

$$
\begin{array}{l}
Q(\lambda\, k \to S[k]\{\sigma[k]T[k]\tau[k]\}U[k]|0{\to}1) \rightsquigarrow \lambda\, i \to \\
\quad i\ \mathsf{kink}(S[k]\sigma[k]T[k]\tau[k]U[k]| \\
\qquad (Q{\blacktriangleleft}(\lambda\, k \to S[k]\sigma[k]T[k]|0{\to}1)) \\
\qquad (Q{\blacktriangleright}(\lambda\, k \to T[k]\tau[k]U[k]|0{\to}1)))
\end{array}
$$