

THE SINGLETON KIND CALCULUS

ABSTRACT. In this section we develop the singleton kind calculus. A singleton kind $S(c)$ is the kind of all constructors that are equivalent to c . The addition of these new kinds will be useful to explain module signatures later on.

1. SYNTAX

The singleton kind calculus is built on top of souped-up F^ω .

$$\begin{aligned} k &::= \text{type} \mid k \rightarrow k \mid k * k \mid S(c) \mid \Pi\alpha : k. k \mid \Sigma\alpha : k. k \\ c &::= \alpha \mid c \rightarrow c \mid \forall\alpha : k. c \mid \lambda\alpha : k. c \mid c \ c \mid \langle c, c \rangle \mid \pi_1 \ c \mid \pi_2 \ c \\ e &::= x \mid \lambda x : c. e \mid e \ e \mid \Lambda\alpha : k. e \mid e[c] \\ \Gamma &::= \cdot \mid \Gamma, x : c \mid \Gamma, \alpha : k \end{aligned}$$

2. MOTIVATION

Consider the following ML signature.

```
sig
  type t
  type 'a u
  type ('a, 'b) v
  type w = int
end
```

The first three types can be assigned kinds in F^ω in a straight forward way.

$$\begin{aligned} t &: \text{type} \\ u &: \text{type} \rightarrow \text{type} \\ v &: \text{type} \times \text{type} \rightarrow \text{type} \end{aligned}$$

But how do we kind w ? Remember, `int` is not a kind, so it doesn't make sense to say $w : \text{int}$. But it's not quite right to say $w : \text{type}$ either, because w cannot stand for arbitrary types. We therefore write $w : S(\text{int})$: $S(\text{int})$ is the kind containing exactly `int` and all those types equivalent to `int`, such as $(\lambda\alpha : \text{type}. \alpha) \text{int}$. The other new kind constructs, $\Pi\alpha : k. k$ and $\Sigma\alpha : k. k$ (which are called dependent function spaces and dependent sums respectively) similar problems. We use Π kinds to kind assignments of parametric types. Specifically, in the signature

```
sig
  type 'a t = 'a list
end
```

we can say that

$$t : \Pi\alpha : \text{type}. S(\alpha)$$

Σ kinds are used to kind types that are assigned to abstract types. In the signature

type t
type $s = t$

we say that

$t : \mathbf{type}$
 $s : \Sigma \alpha : \mathbf{type}. \mathcal{S}(\alpha)$

3. DEFINITIONS

In this section the following judgements will be defined.

Judgement	Description
$\Gamma \vdash k : \mathbf{kind}$	k is a kind
$\Gamma \vdash k \equiv k' : \mathbf{kind}$	kind equivalence
$\Gamma \vdash k \leq k'$	subkinding
$\Gamma \vdash c : k$	c has kind k
$\Gamma \vdash c \equiv c' : k$	constructor equivalence
$\Gamma \vdash e : \tau$	e has type τ

A complete list would also include the judgement $\Gamma \vdash \tau : \mathbf{type}$ but these rules are exactly the same as in F^ω so we will omit them. Beginning with the rules for well-formed kinds:

3A $\frac{}{\Gamma \vdash \tau : \mathbf{kind}}$	3B $\frac{\Gamma \vdash c : \tau}{\Gamma \vdash S(c) : \mathbf{kind}}$	3C $\frac{\Gamma \vdash k_1 : \mathbf{kind} \quad \Gamma, k_1 : \mathbf{kind} \vdash k_2 : \mathbf{kind}}{\Gamma \vdash \Pi \alpha : k_1. k_2 : \mathbf{kind}}$
	3D $\frac{\Gamma \vdash k_1 : \mathbf{kind} \quad \Gamma, k_1 : \mathbf{kind} \vdash k_2 : \mathbf{kind}}{\Gamma \vdash \Sigma \alpha : k_1. k_2 : \mathbf{kind}}$	

Definitional equality of kinds:

3E $\frac{\Gamma \vdash k : \mathbf{kind}}{\Gamma \vdash k \equiv k : \mathbf{kind}}$	3F $\frac{\Gamma \vdash k_1 \equiv k_2 : \mathbf{kind}}{\Gamma \vdash k_2 \equiv k_1 : \mathbf{kind}}$
3G $\frac{\Gamma \vdash k_1 \equiv k_2 : \mathbf{kind} \quad \Gamma \vdash k_2 \equiv k_3 : \mathbf{kind}}{\Gamma \vdash k_1 \equiv k_3 : \mathbf{kind}}$	3H $\frac{\Gamma \vdash c \equiv c' : \mathbf{type}}{\Gamma \vdash S(c) \equiv S(c') : \mathbf{kind}}$
3I $\frac{\Gamma \vdash k_1 \equiv k'_1 : \mathbf{kind} \quad \Gamma, \alpha : k_1 \vdash k_2 \equiv k'_2 : \mathbf{kind}}{\Gamma \vdash \Pi \alpha : k_1. k_2 \equiv \Pi \alpha : k'_1. k_2 : \mathbf{kind}}$	
3J $\frac{\Gamma \vdash k_1 \equiv k'_1 : \mathbf{kind} \quad \Gamma, \alpha : k_1 \vdash k_2 \equiv k'_2 : \mathbf{kind}}{\Gamma \vdash \Sigma \alpha : k_1. k_2 \equiv \Sigma \alpha : k'_1. k_2 : \mathbf{kind}}$	

Kind membership — which constructors belong to a given kind:

$$\begin{array}{c}
\begin{array}{c} \text{3K} \\ \frac{\alpha : k \in \Gamma}{\Gamma \vdash \alpha : k} \end{array} \quad
\begin{array}{c} \text{3L} \\ \frac{\Gamma \vdash c_1 : \mathbf{type} \quad \Gamma \vdash c_2 : \mathbf{type}}{\Gamma \vdash c_1 \rightarrow c_2 : \mathbf{type}} \end{array} \quad
\begin{array}{c} \text{3M} \\ \frac{\Gamma \vdash k : \mathbf{kind} \quad \Gamma, \alpha : k \vdash c : \mathbf{type}}{\Gamma \vdash \forall \alpha : k. c : \mathbf{type}} \end{array} \\
\\
\begin{array}{c} \text{3N} \\ \frac{\Gamma \vdash k_1 : \mathbf{kind} \quad \Gamma, \alpha : k_1 \vdash c : k_2}{\Gamma \vdash \lambda \alpha : k_1. c : \Pi \alpha : k_1. k_2} \end{array} \quad
\begin{array}{c} \text{3O} \\ \frac{\Gamma \vdash c_1 : \Pi \alpha : k. k' \quad \Gamma \vdash c_2 : k}{\Gamma \vdash c_1 c_2 : [c_2/\alpha]k'} \end{array} \\
\\
\begin{array}{c} \text{3P} \\ \frac{\Gamma \vdash c_1 : k_1 \quad \Gamma \vdash c_2 : [c_1/\alpha]k_2 \quad \Gamma, \alpha : k_1 \vdash k_2 : \mathbf{kind}}{\Gamma \vdash \langle c_1, c_2 \rangle : \Sigma \alpha : k_1. k_2} \end{array} \quad
\begin{array}{c} \text{3Q} \\ \frac{\Gamma \vdash c : \Sigma \alpha : k_1. k_2}{\Gamma \vdash \pi_1 c : k_1} \end{array} \\
\\
\begin{array}{c} \text{3R} \\ \frac{\Gamma \vdash c : \Sigma \alpha : k_1. k_2}{\Gamma \vdash \pi_2 c : [\pi_1 c/\alpha]k_2} \end{array} \quad
\begin{array}{c} \text{3S} \\ \frac{\Gamma \vdash c : \mathbf{type}}{\Gamma \vdash c : S(c)} \end{array}
\end{array}$$

Notice that even though $\Sigma \alpha : k_1. k_2$ is called a dependent sum, it behaves like a product. Now, the rules for subkinding:

$$\begin{array}{c}
\begin{array}{c} \text{3T} \\ \frac{\Gamma \vdash k : \mathbf{kind}}{\Gamma \vdash k \leq k} \end{array} \quad
\begin{array}{c} \text{3U} \\ \frac{\Gamma \vdash k_1 \leq k_2 \quad \Gamma \vdash k_2 \leq k_3}{\Gamma \vdash k_1 \leq k_3} \end{array} \quad
\begin{array}{c} \text{3V} \\ \frac{\Gamma \vdash c : \mathbf{type}}{\Gamma \vdash S(c) \leq \mathbf{type}} \end{array} \\
\\
\begin{array}{c} \text{3W} \\ \frac{\Gamma \vdash c \equiv c' : \mathbf{type}}{\Gamma \vdash S(c) \leq S(c') : \mathbf{type}} \end{array} \quad
\begin{array}{c} \text{3X} \\ \frac{\Gamma \vdash k'_1 \leq k_1 \quad \Gamma \alpha : k_1 \vdash k_2 : \mathbf{kind} \quad \Gamma, \alpha : k'_1 \vdash k_2 \leq k'_2}{\Gamma \vdash \Pi \alpha : k_1. k_2 \leq \Pi \alpha : k'_1. k'_2} \end{array} \\
\\
\begin{array}{c} \text{3Y} \\ \frac{\Gamma \vdash k_1 \leq k'_1 \quad \Gamma, \alpha : k_1 \vdash k_2 \leq k'_2 \quad \Gamma, \alpha : k'_1 \vdash k'_2 : \mathbf{kind}}{\Gamma \vdash \Sigma \alpha : k_1. k_2 \leq \Sigma \alpha : k'_1. k'_2} \end{array} \quad
\begin{array}{c} \text{3Z} \\ \frac{\Gamma \vdash c : k \quad \Gamma \vdash k \leq k'}{\Gamma \vdash c : k'} \end{array} \\
\\
\begin{array}{c} \text{3AA} \\ \frac{\Gamma \vdash k \equiv k'}{\Gamma \vdash k \leq k'} \end{array}
\end{array}$$

Definitional equality of constructors:

$$\begin{array}{c}
\begin{array}{ccc}
\frac{3AB}{\Gamma \vdash c : k} & \frac{3AC}{\Gamma \vdash c_1 \equiv c_2 : k} & \frac{3AD}{\Gamma \vdash c_2 : k \quad \Gamma, \alpha : k \vdash c_1 : k'} \\
\Gamma \vdash c \equiv c : k & \Gamma \vdash c_2 \equiv c_1 : k & \Gamma \vdash (\lambda \alpha : k. c_1) c_2 \equiv [c_2/\alpha]c_1 : [c_2/\alpha]k'
\end{array} \\
\\
\begin{array}{ccc}
\frac{3AE}{\Gamma \vdash c_1 : k_1 \quad \Gamma \vdash c_2 : k_2} & \frac{3AF}{\Gamma \vdash c_1 : k_1 \quad \Gamma \vdash c_2 : k_2} & \frac{3AG}{\Gamma \vdash c_1 \equiv c_2 : k \quad \Gamma \vdash k \leq k'} \\
\Gamma \vdash \pi_1 \langle c_1, c_2 \rangle \equiv c_1 & \Gamma \vdash \pi_2 \langle c_1, c_2 \rangle \equiv c_2 & \Gamma \vdash c_1 \equiv c_2 : k'
\end{array} \\
\\
\begin{array}{ccc}
\frac{3AH}{\Gamma \vdash c \equiv c' : \mathbf{type}} & \frac{3AI}{\Gamma \vdash c : S(c')} & \frac{3AJ}{\Gamma \vdash k_1 \equiv k_2 : \mathbf{kind} \quad \Gamma, \alpha : k_1 \vdash c \equiv c' : k_2} \\
\Gamma \vdash c \equiv c' : S(c) & \Gamma \vdash c \equiv c' : \mathbf{type} & \Gamma \vdash \lambda \alpha : k_1. c \equiv \lambda \alpha : k'_1. c' : \Pi \alpha : k_1. k_2
\end{array} \\
\\
\frac{3AK}{\Gamma \vdash c \equiv c' : \Pi \alpha : k. k' \quad \Gamma \vdash c_2 \equiv c'_2 : k} \\
\Gamma \vdash c_1 c_2 \equiv c'_1 c'_2 : [c_2/\alpha]k' \\
\\
\frac{3AL}{\Gamma \vdash c_1 \equiv c'_1 : k_1 \quad \Gamma \vdash c_2 \equiv c'_2 : [c_1/\alpha]k_2 \quad \Gamma, \alpha : k_1 \vdash k_2 : \mathbf{kind}} \\
\Gamma \vdash \langle c_1, c_2 \rangle \equiv \langle c'_1, c'_2 \rangle : \Sigma \alpha : k_1. k_2 \\
\\
\begin{array}{cc}
\frac{3AM}{\Gamma \vdash c \equiv c' : \Sigma \alpha : k_1. k_2} & \frac{3AN}{\Gamma \vdash c \equiv c' : \Sigma \alpha : k_1. k_2} \\
\Gamma \vdash \pi_1 c \equiv \pi_1 c' : k_1 & \Gamma \vdash \pi_2 c \equiv \pi_2 c' : [\pi_1 c/\alpha]k_2
\end{array} \\
\\
\frac{3AN}{\Gamma \vdash c_1 \equiv c'_1 : \mathbf{type} \quad \Gamma \vdash c_2 \equiv c'_2 : \mathbf{type}} \\
\Gamma \vdash c_1 \rightarrow c_2 \equiv c'_1 \rightarrow c'_2 : \mathbf{type} \\
\\
\frac{3AO}{\Gamma \vdash k \equiv k' : \mathbf{kind} \quad \Gamma, \alpha : k \vdash c \equiv c' : \mathbf{type}} \\
\Gamma \vdash \forall \alpha : k. c \equiv \forall \alpha : k'. c' : \mathbf{type} \\
\\
\frac{3AP}{\Gamma, \alpha : k_1 \vdash c \alpha \equiv c' \alpha : k_2 \quad \Gamma \vdash c : \Pi \alpha : k_1. k_2 \quad \Gamma \vdash c' : \Pi \alpha : k_1. k_2''} \\
\Gamma \vdash c \equiv c' : \Pi \alpha : k_1. k_2 \\
\\
\frac{3AQ}{\Gamma \vdash \pi_1 c \equiv \pi_1 c' \quad \Gamma, \alpha : k_1 \vdash k_2 : \mathbf{kind} \quad \Gamma \vdash \pi_2 c \equiv \pi_2 c' : [\pi_1 c/\alpha]k_2} \\
\Gamma \vdash c \equiv c' : \Sigma \alpha : k_1. k_2
\end{array}$$