# 1 Kinds

Kinds are constraints over types. These constraints are expressed via kind assignments akin to type assignments.

**Definition 1** (Kinds)**.** *$\mathbb{K}$ is the space of kinds given as follows:*

$\mathbb{K} \Rightarrow \texttt{Star} \mid \texttt{Constraint} \mid \mathbb{K} \to \mathbb{K}$

*Just like with types, we will assume right associativity of the "$\to$" operator.*

`Star` is the kind of all data objects; namely runtime variables, constants and functions.

`Constraint` is the kind of class constraints. For example: given a type class $C$ with type parameters $a$ and $b$, then $C\ a\ b$ is a class constraint of the kind `Constraint`.

**Definition 2** (Kind assignments)**.** *$\Gamma \vdash \tau : k$ means that given the context $\Gamma$ (a set of assignments), the type $\tau$ has the kind $k$. If this hold for every $\Gamma$, we write $\vdash \tau : k$. If $\Gamma$ is obvious or does not change throughout a proof, we write just $\tau : k$.*

**Definition 3** (Kind arity)**.** *We call a kind $k$ $n$-ary, if there exist $n$ kinds $l_1, \dots l_n$ such that given types $\tau : k$, $\sigma_1 : l_1, \dots \sigma_n : l_n$ the type $\tau\ \sigma_1 \dots \sigma_n$ is valid and its kind is either* `Star` *or* `Constraint`*.*

*Kind $k$ has arity of 0 if and only if it is either* `Star` *or* `Constraint`*.*

**Definition 4** (Context as a partial function)**.** *In the scope of this thesis, we assume all kinds be mono-kinds (if $\Gamma \vdash \tau : k$ and $\Gamma \vdash \tau : k'$, then $k \equiv k'$). We will define rules, for which this will hold. And because of this, we can define applying $\Gamma$ to the type $\tau$ as $\Gamma(\tau) := k \Leftrightarrow \Gamma \vdash \tau : k$.*

**Definition 5** (Kinding rules (incomplete))**.** $\dfrac{\tau : k \in \Gamma}{\Gamma \vdash \tau : k}$

$$\frac{\Gamma \vdash x : \tau}{\Gamma \vdash \tau : \texttt{Star}}$$

$$\frac{\Gamma \vdash \tau : k \to l \qquad \Gamma \vdash \sigma : k}{\Gamma \vdash \tau\sigma : l}$$

$$\frac{\textit{class } C \textit{ has parameters } \tau_1, \dots \tau_n, \textit{ superclasses } D_1, \dots D_m \textit{ within a global context } \Gamma}{\textit{class context } \Gamma_C = \Gamma \cup \{\tau_1 : k_1, \dots \tau_n : k_n, D_1 : \Gamma_{D_1}(D_1), \dots D_m : \Gamma_{D_m}(D_m), \textit{method typings}\},\ \Gamma_C \vdash C : k_1 \to k_2 \to \dots \to k_n \to \texttt{Constraint}}$$

$$\frac{\textit{struct } S \textit{ has parameters } \tau_1, \dots \tau_n \textit{ within a global context } \Gamma}{\begin{array}{c}\textit{struct context } \Gamma_S = \Gamma \cup \{\tau_1 : k_1, \dots \tau_n : k_n, \textit{field typings}\}, \\ \Gamma_S \vdash S : k_1 \to k_2 \to \dots \to k_n \to \texttt{Star}\end{array}}$$

$$\frac{\textit{function } f \textit{ has parameters } x_1 : \tau_1, \dots x_n : \tau_n, \textit{ return type } \sigma, \textit{ class constraints } D_1, \dots D_m \textit{ within } \textit{(}}{\begin{array}{c}\textit{function context } \Gamma_f = \Gamma \cup \{x_1 : \tau_1, \dots x_n : \tau_n, \sigma : \texttt{Star}, D_1 : \Gamma_{D_1}(D_1), \dots D_m : \Gamma_{D_m}(L \\ \Gamma_f \vdash f : (x_1, \dots x_n) \to \sigma\end{array}}$$

$$\frac{\Gamma_x \vdash x : \Gamma_x(x)}{\Gamma \vdash x : \Gamma_x(x)}$$

**Definition 6** (Kind ordering (completing the kinding rules))**.** *We define the ordering of kinds as the transitive closure generated by the following rules:*

- $\texttt{Star} < \texttt{Star} \to \texttt{Star}$
- $k \to l \wedge k' < k \Rightarrow k' \to l < k \to l$
- $k \to l \wedge l' < l \Rightarrow k \to l' < k \to l$

*Then, if the inference for kinds is inconclusive (within the scope of type classes and structs), we assume the infimal kind assignments for which the kinding rules hold.*