$$
\begin{array}{rcll}
t & ::= & x & \textit{Variable} \\
 & | & \lambda x.t & \textit{Abstraction} \\
 & | & (t, t) & \textit{Application}
\end{array}
$$

Grammar 1: λ-calculus syntax

1 Terms, Types and Kinds

For programming language, three levels: ***terms***, ***types***, and ***kinds***, have proved sufficient .

Recap term-level abstraction and application in the λ-calculus as shown in Grammar 1.

$\Gamma \vdash \mathrm{T} :: \mathrm{K}$ is read as "type T has kind K in context Γ".

kinding is a well-formedness relation.

To treat type-level functions, collectively called *type operators* more formally, it is required to:

1. Add a collection of rules of *kinding* which specify how type expressions can be combined to yield new type expressions.
2. Whenever a type T appears in a term $(\lambda x : \mathrm{T}.t)$, check whether T is well formed.
3. Add a collection of rules for the definitonal equivalence relations between types.