

# 1 Bytecode Abstract Syntax

$b$	$::= v P \bar{i} \bar{M}$	<i>bytecode file</i>
$v$	$::= \text{magic major.minor}$	<i>magic+version number</i>
$P$	$::= \text{fully qualified path}$	<i>path to module</i>
$i$	$::= \text{import } \mu \text{ URI} : \tau \text{ as } x$	<i>module import</i>
$\mu$	$::= [\text{metadata}] [\text{type}]$	
$M$	$::= \text{module } P : \tau = e$ $  \text{ type } P = T [\delta]$	<i>top level modules</i>
$e$	$::= x$ $  \text{ new } \tau \{x \Rightarrow \bar{d}\}$ $  e.m(\bar{e})$ $  e.f$ $  e.f = e$ $  \text{ let } x = e \text{ in } e$ $  \mathcal{L}$ $  \overline{e.\text{match } x : p.L \Rightarrow e} [\text{else } e]$ $  \overline{e} \mid \bar{d}$	<i>expressions</i>
$\mathcal{L}$	$::= \text{string}$ $  \text{integer}$	<i>literals</i>

$d$	$::= \text{val } f : \tau = e$ $  \text{var } f : \tau = e$ $  \text{def } m(\overline{x} : \overline{\tau}) : \tau = e$ $  \text{type } L = T [\delta]$	<i>declarations</i>
$T$	$::= c$ $  \text{extag } c$ $  \text{datatag } \overline{p.L} \ c$	<i>type desc.</i>
$c$	$::= \tau$ $  \text{extends } p.L \ \tau$	<i>case desc.</i>
$\tau$	$::= \tau \{x \Rightarrow \bar{\sigma}\}_s$ $  p.L$ $  \top$ $  \perp$ $  ?$	<i>type</i>
$p$	$::= x$ $  p.f$	<i>paths</i>
$s$	$::= \text{stateful} \mid \text{pure}$	
$\sigma$	$::= \text{val } f : \tau$ $  \text{var } f : \tau$ $  \text{def } m : \Pi \overline{x} : \overline{\tau}. \tau$ $  \text{type } L = T [\delta]$ $  \text{type}_s L [\delta]$	<i>decl type</i>
$\delta$	$::= \text{metadata } e$	<i>metadata</i>

Notation: overbar means a list of elements, as in Java