

# PFPL Syntax Package\*

Robert Harper

December 23, 2022

## Overview

Write `\usepackage{pfpl-syntax}` to define syntax macros for PFPL. The package provides an integrated, systematic treatment of abstract and concrete syntax for a wide range of languages.

## Abstract Binding Trees

To typeset an abstract binding tree yourself, provide the package with the `[abt]` option, and use the following commands:

- `\Opn{kwd}`: Format an operator as a keyword, or `\Opn*{symbol}` for an operator whose display is in math mode. For example, `\Opn{fun}` produces `fun` and `\Opn*{\lambda}` produces  $\lambda$ .
- `\Abt<parameters>[optional](arguments)`: produce abstract syntax, with all arguments typeset, and the starred form to omit the optional arguments. Omitted arguments are omitted. For example, the command

`\Abt{\Opn{get}}<a>[\tau]`

produces `get` $\langle a \rangle[\tau]$ , and the command

`\Abt{\Opn{set}}[\tau]<a>(e)`

produces `set` $\langle a \rangle[\tau](e)$ . The starred forms produce `get` $\langle a \rangle$ , and `set` $\langle a \rangle(e)$ , respectively.

---

\*© 2022 Robert Harper. All Rights Reserved.

- `\Abs<symbols>(variables){abt}`: produce an abstractor over symbols and/or variables within an ABT. For example, the command

`\Abt{\Opn*\lambda}{\tau}(\Abs(x){e})`

produces  $\lambda[\tau](x . e)$  and the starred form produces  $\lambda(x . e)$ . Similarly, the command

`\Abt{\Opn{dcl}}{\tau;\rho}(e;\Abs<a>{m})`

produces  $\text{dcl}[\tau;\rho](e; a . m)$  and the starred form produces  $\text{dcl}(e; a . m)$ .

These general forms are not ordinarily used directly, they are rather for authors to define their own syntax macros, as illustrated for PFPL in the next section.

## Language-Specific Definitions

The commands for formatting the many languages considered in PFPL are formulated in a series of figures organized around type structure. The tables display the abstract and concrete syntax, and the literal command used to create them (with starred forms for the concrete syntax).

$\text{fun}(\tau_1 ; \tau_2)$	$\tau_1 \rightarrow \tau_2$	$\backslash\text{arrTy}\{\backslash\tau_1\}\{\backslash\tau_2\}$
$\text{lam}[\tau_1 ; \tau_2](x . e)$	$\lambda(x . e)$	$\backslash\text{lamEx}\{x\}\{e\}$
$\text{ap}[\tau_1 ; \tau_2](e_1 ; e_2)$	$\text{ap}(e_1 ; e_2)$	$\backslash\text{appEx}\{e_1\}\{e_2\}$

Figure 1: Function Types

$\text{unit}$	$\mathbf{1}$	$\backslash\text{unitTy}$
$\text{null}$	$\langle \rangle$	$\backslash\text{unitEx}$
$\text{prod}(\tau_1 ; \tau_2)$	$\tau_1 \times \tau_2$	$\backslash\text{prodTy}\{\backslash\tau_1\}\{\backslash\tau_2\}$
$\text{proj}\langle i \rangle[\tau_1 ; \tau_2](e)$	$e \cdot i$	$\backslash\text{projEx}\langle i \rangle\{e\} \quad (i = 1, 2)$
$\text{pair}[\tau_1 ; \tau_2](e_1 ; e_2)$	$\langle e_1, e_2 \rangle$	$\backslash\text{pairEx}[\backslash\tau_1][\backslash\tau_2]\{e_1\}\{e_2\}$
$\text{vprod}\langle I \rangle(\tau_I)$	$\times_{i \in I}(i \hookrightarrow \tau_i)$	$\backslash\text{vprodTy}\langle I \rangle\langle i \rangle\{\backslash\tau\}$
$\text{vtuple}\langle I \rangle[\tau_I](e_I)$	$\langle i \hookrightarrow e_i \mid i \in I \rangle$	$\backslash\text{vtupleEx}\langle I \rangle[\backslash\tau]\{e\}$
$\text{vproj}\langle I ; i \rangle[\tau_I](e)$	$e \cdot i$	$\backslash\text{vprojEx}\langle I \rangle\langle i \rangle[\backslash\tau]\{e\} \quad (i \in I)$

The notation  $\tau_I$  stands for the finite map  $i \hookrightarrow \tau_i \mid i \in I$  and  $e_I$  stands for  $i \hookrightarrow e_i \mid i \in I$ .

Figure 2: Product Types

$\text{void}$	$\mathbf{0}$	$\backslash\text{voidTy}$
$\text{absurd}[\rho](e)$	$\text{absurd}(e)$	$\backslash\text{absurdEx}[\backslash\rho]\{e\}$
$\text{sum}(\tau_1 ; \tau_2)$	$\tau_1 + \tau_2$	$\backslash\text{sumTy}\{\backslash\tau_1\}\{\backslash\tau_2\}$
$\text{inj}\langle i \rangle[\tau_1 ; \tau_2](e)$	$i \cdot e$	$(i = 1, 2)$
$\text{case}[\tau_1 ; \tau_2 ; \rho](e ; x . e_1 ; x . e_2)$	$\text{case } e \{ x . e_1 \mid x . e_2 \}$	$\backslash\text{caseEx}[\backslash\tau][\backslash\rho]\{e\}\{x\}\{e\}$
$\text{bool}$	$\mathbf{2}$	$\backslash\text{boolTy}$
$\text{true}$	$\text{true}$	$\backslash\text{trueEx}$
$\text{false}$	$\text{false}$	$\backslash\text{falseEx}$
$\text{if}[\rho](e ; e_1 ; e_2)$	$\text{if}(e ; e_1 ; e_2)$	$\backslash\text{ifEx}[\backslash\rho]\{e\}\{e_1\}\{e_2\}$
$\text{vsum}\langle I \rangle(\tau_I)$	$+_{i \in I}(i \hookrightarrow \tau_i)$	$\backslash\text{vsumTy}\langle I \rangle\langle i \rangle\{\backslash\tau\}$
$\text{vinj}\langle I ; i \rangle[\tau_I](e)$	$i \cdot e$	$\backslash\text{vinjEx}\langle I \rangle\langle i \rangle[\backslash\tau]\{e\} \quad (i \in I)$
$\text{vcase}\langle I \rangle[\tau_I ; \rho](e ; x . e'_I)$	$\text{vcase } e \{ i \hookrightarrow x . e'_i \mid i \in I \}$	$\backslash\text{vcaseEx}\langle I \rangle[\backslash\rho][\backslash\tau]\{e\}\{x\}\{e'\}$

Figure 3: Sum Types

$\text{ind}(t . \tau)$	$\text{ind}(t . \tau)$	$\backslash\text{indTy}\{t\}\{\tau\}$
$\text{in}[t . \tau](e)$	$\text{in}(e)$	$\backslash\text{inEx}[t][\tau]\{e\}$
$\text{rec}[t . \tau; \rho](e; x . e')$	$\text{rec}(e; x . e')$	$\backslash\text{recEx}[t][\tau][\rho]\{e\}\{x\}\{e'\}$
$\text{nat}$	$\text{nat}$	$\backslash\text{natTy}$
$\text{zero}$	$\text{zero}$	$\backslash\text{zeroEx}$
$\text{succ}(e)$	$\text{succ}(e)$	$\backslash\text{succEx}\{e\}$
$\text{natit}[\rho](e; e_0; x . e_1)$	$\text{natit } e \{ e_0 \mid x . e_1 \}$	$\backslash\text{natitEx}[\rho]\{e\}\{e_0\}\{x\}\{e_1\}$
$\text{ifz}[\rho](e; e_0; x . e_1)$	$\text{ifz } e \{ e_0 \mid x . e_1 \}$	$\backslash\text{ifzEx}[\rho]\{e\}\{e_0\}\{x\}\{e_1\}$
$\text{list}(\tau)$	$\tau \text{ list}$	$\backslash\text{listTy}\{\tau\}$
$\text{nil}[\tau]$	$\text{nil}$	$\backslash\text{nilEx}[\tau]$
$\text{cons}[\tau](e_1; e_2)$	$\text{cons}(e_1; e_2)$	$\backslash\text{consEx}[\tau]\{e_1\}\{e_2\}$
$\text{listrec}[\rho; \tau](e; e_1; x, y . e_2)$	$\text{listrec } e \{ e_1 \mid x, y . e_2 \}$	$\backslash\text{listrecEx}[\rho][\tau]\{e\}\{e_1\}\{x\}\{y\}\{e_2\}$
$\text{listcase}[\rho; \tau](e; e_1; x, y . e_2)$	$\text{listcase } e \{ e_1 \mid x, y . e_2 \}$	$\backslash\text{listcaseEx}[\rho][\tau]\{e\}\{e_1\}\{x\}\{y\}\{e_2\}$
$\text{coi}(t . \tau)$	$\text{coi}(t . \tau)$	$\backslash\text{coiTty}\{t\}\{\tau\}$
$\text{out}[t . \tau](e)$	$\text{out}(e)$	$\backslash\text{outEx}[t][\tau]\{e\}$
$\text{gen}[t . \tau; \sigma](e; x . e')$	$\text{gen}(e; x . e')$	$\backslash\text{genEx}[t][\tau][\sigma]\{e\}\{x\}\{e'\}$
$\text{conat}$	$\text{conat}$	$\backslash\text{conatTy}$
$\text{stream}(\tau)$	$\tau \text{ stream}$	$\backslash\text{streamTy}\{\tau\}$

Figure 4: Inductive and Coinductive Types

$\text{All}(t . \tau)$	$\forall(t . \tau)$	$\backslash\text{AllTy}\{t\}\{\tau\}$
$\text{Lam}(t . e)$	$\Lambda(t . e)$	$\backslash\text{LamEx}\{t\}[\tau]\{e\}$
$\text{Ap}[t . \tau](e; \sigma)$	$\text{Ap}(e; \sigma)$	$\backslash\text{AppEx}[t][\tau]\{e\}\{\sigma\}$
$\text{Some}(t . \tau)$	$\exists(t . \tau)$	$\backslash\text{SomeTy}\{t\}\{\tau\}$
$\text{Pack}[t . \tau](\rho; e)$	$\text{Pack}(\rho; e)$	$\backslash\text{PackEx}[t][\tau]\{\rho\}\{e\}$
$\text{Open}[t . \tau; \rho](e; t, x . e')$	$\text{Open}(e; t, x . e')$	$\backslash\text{OpenEx}[t][\tau]\{e\}[\rho]\{x\}\{e'\}$

Figure 5: Polymorphic Types

<b>ans</b>	<b>ans</b>	<b>\ansTy</b>
<b>yes</b>	<b>yes</b>	<b>\yesEx</b>
<b>no</b>	<b>no</b>	<b>\noEx</b>
<b>cont</b> ( $\tau$ )	$\tau$ <b>cont</b>	<b>\contTy</b> {\tau}
<b>cont</b> [ $\tau$ ]( $k$ )	<b>cont</b> ( $k$ )	<b>\contEx</b> {k}
<b>letcc</b> [ $\tau$ ]( $x.e$ )	<b>letcc</b> ( $x.e$ )	<b>\letccEx</b> [\tau]{x}{e}
<b>throw</b> [ $\tau; \rho$ ]( $e_1; e_2$ )	<b>throw</b> ( $e_1; e_2$ )	<b>\throwEx</b> [\tau][\rho]{e_1}{e_2}
<b>emp</b> [ $\tau$ ]	<b>•</b>	<b>\empStk</b> [\tau]
<b>ext</b> [ $\tau_1; \tau_2$ ]( $k; f$ )	$k; f$	<b>\extStk</b> [\tau_1][\tau_2]{k}{f}

Figure 6: Continuation Types

<b>parr</b> ( $\tau_1; \tau_2$ )	$\tau_1 \multimap \tau_2$	<b>\parrTy</b> {\tau_1}{\tau_2}
<b>fun</b> [ $\tau_1; \tau_2$ ]( $f, x.e$ )	<b>fun</b> ( $f, x.e$ )	<b>\funEx</b> {f}{x}{e}
<b>ap</b> [ $\tau_1; \tau_2$ ]( $e_1; e_2$ )	<b>ap</b> ( $e_1; e_2$ )	<b>\appEx</b> {e_1}{e_2}
<b>fix</b> [ $\tau$ ]( $x.e$ )	<b>fix</b> ( $x.e$ )	<b>\fixEx</b> [\tau]{x}{e}
<b>rect</b> .( $\tau$ )	<b>rect</b> .( $\tau$ )	<b>\recTy</b> {t}{\tau}
<b>fold</b> [ $t.\tau$ ]( $e$ )	<b>fold</b> ( $e$ )	<b>\foldEx</b> [t][\tau]{e}
<b>unfold</b> [ $t.\tau$ ]( $e$ )	<b>unfold</b> ( $e$ )	<b>\unfoldEx</b> [t][\tau]{e}
<b>self</b> ( $\tau$ )	<b>self</b> ( $\tau$ )	<b>\selfTy</b> {\tau}
<b>roll</b> [ $\tau$ ]( $e$ )	<b>roll</b> ( $e$ )	<b>\rollEx</b> [\tau]{e}
<b>self</b> [ $\tau$ ]( $x.e$ )	<b>self</b> ( $x.e$ )	<b>\selfEx</b> [\tau]{x}{e}
<b>lam</b> ( $x.M$ )	$\lambda(x.M)$	<b>\ulamEx</b> {x}{M}
<b>app</b> ( $M_1; M_2$ )	$M_1(M_2)$	<b>\uapEx</b> {M_1}{M_2}
<b>I</b>	<b>I</b>	<b>\uIEx</b>
<b>K</b>	<b>K</b>	<b>\uKEx</b>
<b>S</b>	<b>S</b>	<b>\uSEx</b>
<b>B</b>	<b>B</b>	<b>\uBEx</b>

Figure 7: Recursive Types

$\text{cmd}(\tau)$	$\tau \text{ cmd}$	$\backslash\text{cmdTy}\{\backslash\tau\}$
$\text{cmd}[\tau](m)$	$\text{cmd}(m)$	$\backslash\text{cmdEx}[\backslash\tau]\{m\}$
$\text{ret}[\tau](e)$	$\text{ret}(e)$	$\backslash\text{retCmd}[\backslash\tau]\{e\}$
$\text{bnd}[\tau; \rho](e; x.m)$	$\text{bnd } x \leftarrow e; m$	$\backslash\text{bndCmd}\{e\}\{x\}\{m\}$
$\text{dcl}[\tau; \rho](e; a.m)$	$\text{dcl } a := e; m$	$\backslash\text{dclCmd}\{e\}\{a\}\{m\}$
$\text{ref}(\tau)$	$\tau \text{ ref}$	$\backslash\text{refTy}\{\backslash\tau\}$
$\text{ref}(a)$	$\& a$	$\backslash\text{refEx}\{a\}$
$\text{get}(a)$	$! a$	$\backslash\text{getCmd}\{a\}$
$\text{getref}[\tau](e)$	$*e$	$\backslash\text{getrefCmd}[\backslash\tau]\{e\}$
$\text{set}(a)(e)$	$a := e$	$\backslash\text{setCmd}\{a\}\{e\}$
$\text{setref}[\tau](e_1; e_2)$	$e_1 * = e_2$	$\backslash\text{setrefCmd}[\backslash\tau]\{e_1\}\{e_2\}$

Symbols  $a$  are constants of sort  $\text{loc}$ .

Figure 8: Command Types

$\text{st}$	$\text{st}$	$\backslash\text{staticLock}$
$\text{type}$	$\text{type}$	$\backslash\text{univSg}$
$\text{val}(\tau)$	$\text{val}(\tau)$	$\backslash\text{valSg}\{\backslash\tau\}$
$\text{Ext}(S; M)$	$\{S \mid M\}$	$\backslash\text{extSg}\{S\}\{M\}$
$\text{in}[S; M](M')$	$\text{in}(M')$	$\backslash\text{inMd}[S][M]\{M'\}$
$\text{out}[S; M](M')$	$\text{out}(M')$	$\backslash\text{outMd}[S][M]\{M'\}$
$\text{Comp}(S)$	$S \text{ Comp}$	$\backslash\text{compSg}\{S\}$
$\text{Pi}(S_1; X.S_2)$	$X:S_1 \rightarrow S_2$	$\backslash\text{piSg}\{S_1\}\{X\}\{S_2\}$
$\text{fun}[S_1](X.S_2)$	$\text{fun } X : S_1 \text{ in } M_2$	$\backslash\text{funMd}\{S_1\}\{X\}\{M_2\}$
$\text{inst}[S_1; X.S_2](M_1; M_2)$	$M_1(M_2)$	$\backslash\text{instMd}[S_1][X][S_2]\{M_1\}\{M_2\}$
$\text{Sig}(S_1; X.S_2)$	$X:S_1 \times S_2$	$\backslash\text{sigSg}\{S_1\}\{X\}\{S_2\}$
$\text{str}[S_1; X.S_2](M_1; M_2)$	$\text{str } M_1 \text{ in } M_2$	$\backslash\text{strMd}[S_1][X][S_2]\{M_1\}\{M_2\}$
$\text{proj}\langle i \rangle[S_1; X.S_2](M)$	$M \cdot i$	$\backslash\text{projMd}\langle i \rangle[S_1][X][S_2]\{M\} \quad (i = 1, 2)$

Figure 9: Module Types

<b>one</b>	1	<code>\unitPr</code>
<b>par</b> ( $p_1 ; p_2$ )	$p_1 \otimes p_2$	<code>\parPr{p_1}{p_2}</code>
<b>null</b>	0	<code>\nullPr</code>
<b>or</b> ( $p_1 ; p_2$ )	$p_1 \oplus p_2$	<code>\choosePr{p_1}{p_2}</code>
<b>que</b> ( $\langle a \rangle(p)$ )	$? \langle a \rangle(p)$	<code>\quePr{a}{p}</code>
<b>sig</b> ( $\langle a \rangle(p)$ )	$! \langle a \rangle(p)$	
<b>rcv</b> ( $\langle a \rangle(x.p)$ )	$? \langle a \rangle(x.p)$	<code>\sigPr{a}{p}</code>
<b>snd</b> ( $\langle a \rangle(e ; p)$ )	$! \langle a \rangle(e ; p)$	<code>\sndPr{a}{e}{p}</code>
<b>asnd</b> ( $\langle a \rangle(e)$ )	$! \langle a \rangle(e)$	<code>\asndPr{a}{e}</code>
<b>sync</b> ( $\varepsilon$ )	<b>sync</b> ( $\varepsilon$ )	<code>\syncPr{\varepsilon}</code>
<b>new</b> [ $\tau$ ]( $a.p$ )	$\nu(a.p)$	<code>\newPr{a}{p}</code>
<b>sil</b>	$\epsilon$	<code>\silAc</code>
<b>sig</b> ( $\langle a \rangle$ )	$a!$	<code>\sigAc{a}</code>
<b>que</b> ( $\langle a \rangle$ )	$a?$	<code>\queAc{a}</code>
<b>snd</b> ( $\langle a \rangle(e)$ )	$a!e$	<code>\sndAc{a}{e}</code>
<b>rcv</b> ( $\langle a \rangle(e)$ )	$a?e$	
<b>any</b> ( $e$ )	$?e$	<code>\rcvAc{a}{e}</code>
<b>emit</b> ( $e$ )	$!e$	<code>\emitAc{e}</code>

Figure 10: Processes

<b>F</b> ( $\tau^+$ )	$\uparrow(\tau^+)$	<code>\freeTy{\posTy{\tau}}</code>
<b>ret</b> [ $\tau^+$ ]( $v$ )	<b>ret</b> ( $v$ )	<code>\freeEx[\posTy{\tau}]{v}</code>
<b>let</b> [ $\tau^- ; \rho^-$ ]( $e ; x.e'$ )	<b>let</b> ( $e ; x.e'$ )	<code>\fletEx[\negTy{\tau}][\negTy{\rho}]{e}{x}{e'}</code>
<b>U</b> ( $\tau^-$ )	$\downarrow(\tau^-)$	<code>\thunkTy{\negTy{\tau}}</code>
<b>thunk</b> [ $\tau^-$ ]( $e$ )	<b>thunk</b> ( $e$ )	<code>\thunkEx[\negTy{\tau}]{e}</code>
<b>force</b> ( $v$ )	<b>force</b> ( $v$ )	<code>force(v)</code>

Figure 11: Polarized Types

<b>tens</b> ( $\tau_1 ; \tau_2$ )	$\tau_1 \otimes \tau_2$	<code>\tensorTy{\tau_1}{\tau_2}</code>
<b>with</b> [ $\tau_1 ; \tau_2$ ]( $v_1 ; v_2$ )	$v_1 \otimes v_2$	<code>\tensorEx{v_1}{v_2}</code>
<b>split</b> [ $\tau_1 ; \tau_2 ; \rho$ ]( $v ; x_1, x_2.e$ )	<b>split</b> $v \{x_1, x_2.e\}$	<code>\splitEx{v}{x_1}{x_2}{e}</code>
<b>and</b> ( $\tau_1 ; \tau_2$ )	$\tau_1 \& \tau_2$	<code>\bothTy{\tau_1}{\tau_2}</code>
<b>both</b> ( $e_1 ; e_2$ )	$e_1 \& e_2$	<code>\bothEx{e_1}{e_2}</code>
<b>par</b> $e ; x, y.(e')$	<b>par</b> $e ; x, y.(e')$	<code>\parEx{e}{x}{y}{e'}</code>

Figure 12: Parallel Types

<b>truth</b>	$\top$	<code>\trueProp</code>
<b>truthI</b>	$\top I$	<code>\trueIPf</code>
<b>and</b> ( $\phi_1 ; \phi_2$ )	$\phi_1 \wedge \phi_2$	<code>\andProp{\phi_1}{\phi_2}</code>
<b>andI</b> [ $\phi_1 ; \phi_2$ ]( $\pi_1 ; \pi_2$ )	$\wedge I(\pi_1 ; \pi_2)$	<code>\andIPf{\pi_1}{\pi_2}</code>
<b>andE</b> $\langle i \rangle$ [ $\phi_1 ; \phi_2$ ]( $\pi$ )	$\wedge E\langle i \rangle(\pi)$	<code>\andEPf[\phi_1][\phi_2]&lt;i&gt;\{\pi\}</code>
<b>falsity</b>	$\perp$	<code>\falseProp</code>
<b>falseI</b> [ $\phi$ ]( $\pi$ )	$\perp I(\pi)$	<code>\falseIPf[\phi]{\pi}</code>
<b>falseE</b> [ $\phi$ ]( $\pi$ )	$\perp E(\pi)$	<code>\falseEPf[\phi]{\pi}</code>
<b>or</b> ( $\phi_1 ; \phi_2$ )	$\phi_1 \vee \phi_2$	<code>\orProp{\phi_1}{\phi_2}</code>
<b>orI</b> $\langle i \rangle$ [ $\phi_1 ; \phi_2$ ]( $\pi$ )	$\vee I\langle i \rangle(\pi)$	<code>\orIPf[\phi_1][\phi_2]&lt;i&gt;\{\pi\}</code>
<b>orE</b> [ $\phi_1 ; \phi_2 ; \rho$ ]( $\pi ; x . \pi_1 ; x . \pi_2$ )	$\vee E(\pi ; x . \pi_1 ; x . \pi_2)$	<code>\orEPf[\phi_1][\phi_2]\{\pi\}\{x\}\{\pi_1\}\{x\}\{\pi_2\}</code>
<b>imp</b> ( $\phi_1 ; \phi_2$ )	$\phi_1 \supset \phi_2$	<code>\impProp{\phi_1}{\phi_2}</code>
<b>impI</b> [ $\phi_1 ; \phi_2$ ]( $x . \pi_2$ )	$\supset I(x . \pi_2)$	<code>\impIPf[\phi_1][\phi_2]\{x\}\{\pi_2\}</code>
<b>impE</b> [ $\phi_1 ; \phi_2$ ]( $\pi ; \pi_1$ )	$\supset E(\pi ; \pi_2)$	<code>\impEPf[\phi_1][\phi_2]\{\pi\}\{\pi_1\}</code>
<b>not</b> ( $\phi$ )	$\neg \phi$	<code>\notProp{\phi}</code>
<b>notI</b> [ $\phi$ ]( $x . \pi$ )	$\neg I(x . \pi)$	<code>\notIPf[\phi]\{x\}\{\pi\}</code>
<b>notE</b> [ $\phi$ ]( $\pi ; \pi_2$ )	$\neg E(\pi ; \pi_2)$	<code>\notEPf[\phi]\{\pi\}\{\pi_2\}</code>

Figure 13: Propositions and Proofs