

Languages-beta: OC-L-08-Type-and-Exception-Definitions *

The PPlanCompS Project

OC-L-08-Type-and-Exception-Definitions.cbs | PLAIN | PRETTY

OUTLINE

8 Type and exception definitions

Type definitions

Exception definitions

Language "OCaml Light"

8 Type and exception definitions

Syntax $TDS : \text{type-definition} ::= \text{'type' typedef and-typedef}^*$
 $ATD : \text{and-typedef} ::= \text{'and' typedef}$
 $TD : \text{typedef} ::= \text{type-params? typeconstr-name type-information}$
 $TI : \text{type-information} ::= \text{type-equation? type-representation? type-constraint}^*$
 $TE : \text{type-equation} ::= \text{'=' typexpr}$
 $TR : \text{type-representation} ::= \text{'=' ' | ? constr-decl bar-constr-decl}^*$
 $\quad \quad \quad | \text{'=' record-decl}$
 $BCD : \text{bar-constr-decl} ::= \text{'|' constr-decl}$
 $TPS : \text{type-params} ::= \text{type-param}$
 $\quad \quad \quad | \text{'(' type-param (',' type-param)* ')'}$
 $TP : \text{type-param} ::= \text{variance? ' ' ident}$
 $\quad \quad \quad \text{variance} ::= \text{'+' | '-'}$
 $RD : \text{record-decl} ::= \text{'{' field-decl (',' field-decl)* ';' '}'}$
 $CD : \text{constr-decl} ::= \text{(constr-name | '[' ']' | '(::)) ('of' constr-args)?}$
 $CA : \text{constr-args} ::= \text{typexpr star-typexpr}^*$
 $FD : \text{field-decl} ::= \text{field-name ':' poly-typexpr}$
 $ED : \text{exception-definition} ::= \text{'exception' constr-decl}$
 $\quad \quad \quad | \text{'exception' constr-name '=' constr}$
 $TC : \text{type-constraint} ::= \text{'constraint' ' ' ident '=' typexpr}$

*Suggestions for improvement: plancomps@gmail.com.
Reports of issues: <https://github.com/plancomps/CBS-beta/issues>.

Type definitions

Semantics $\text{define-types}[_ : \text{type-definition}] : \Rightarrow \text{environments}$
Rule $\text{define-types}[\text{'type' } TD \text{ } ATD^*] = \text{collateral}(\text{define-typedefs}[\text{ } TD \text{ } ATD^*])$

Semantics $\text{define-typedefs}[_ : (\text{typedef and-typedef}^*)] : (\Rightarrow \text{environments})^+$
Rule $\text{define-typedefs}[TD_1 \text{ 'and' } TD_2 \text{ } ATD^*] =$
 $\text{define-typedefs}[TD_2], \text{define-typedefs}[TD_2 \text{ } ATD^*]$
Rule $\text{define-typedefs}[TPS^? \text{ } TCN \text{ '=' } CD \text{ } BCD^*] =$
 $\text{define-constrs}[CD \text{ } BCD^*]$
Rule $\text{define-typedefs}[TPS^? \text{ } TCN \text{ '=' } RD] = \text{map}(\text{ })$
Rule $\text{define-typedefs}[TPS^? \text{ } TCN \text{ '=' } T] = \text{map}(\text{ })$

Semantics $\text{define-constrs}[_ : (\text{constr-decl bar-constr-decl}^*)] : (\Rightarrow \text{environments})^+$
Rule $\text{define-constrs}[CD_1 \text{ 'I' } CD_2 \text{ } BCD^*] =$
 $\text{define-constrs}[CD_1], \text{define-constrs}[CD_2 \text{ } BCD^*]$
Rule $\text{define-constrs}[CN] =$
 $\{\text{constr-name}[CN] \mapsto \text{variant}(\text{constr-name}[CN], \text{tuple}(\text{ }))\}$
Rule $\text{define-constrs}[CN \text{ 'of' } CA] =$
 $\{\text{constr-name}[CN] \mapsto$
 $\text{function closure}(\text{variant}(\text{constr-name}[CN], \text{given}))\}$

Exception definitions

Semantics $\text{define-exception}[_ : \text{exception-definition}] : \Rightarrow \text{environments}$
Rule $\text{define-exception}[\text{'exception' } CD] = \text{define-constrs}[CD]$
Rule $\text{define-exception}[\text{'exception' } CN \text{ '=' } CSTR] = \text{map}(\text{ })$