

Languages-beta: OC-L-06-Patterns *

The P_{LAN}CompS Project

OC-L-06-Patterns.cbs | PLAIN | PRETTY

OUTLINE

6 Patterns

Pattern evaluation

Pattern sequence evaluation

Language "OCaml Light"

6 Patterns

Syntax $P : \text{pattern} ::= \text{value-name}$
| `'_'`
| `constant`
| `pattern 'as' value-name`
| `(' pattern)`
| `(' pattern ':' typexpr)`
| `pattern '|' pattern`
| `constr pattern`
| `pattern comma-pattern+`
| `{' field '=' pattern semic-field-pattern* ';' ? }`
| `[' pattern semic-pattern* ';' ?]`
| `pattern '::' pattern`

$CP : \text{comma-pattern} ::= \text{' , ' pattern}$

$SP : \text{semic-pattern} ::= \text{' ; ' pattern}$

$SFP : \text{semic-field-pattern} ::= \text{' ; ' field '=' pattern}$

Rule $\llbracket \text{'(' } P \text{ ')} \rrbracket : \text{pattern} = \llbracket P \rrbracket$

Rule $\llbracket \text{'(' } P \text{ ':' } T \text{ ')} \rrbracket : \text{pattern} = \llbracket P \rrbracket$

Rule $\llbracket \text{'{' } F \text{ '=' } P \text{ SFP* ';' ? '}} \rrbracket : \text{pattern} = \llbracket \text{'{' } F \text{ '=' } P \text{ SFP* '}} \rrbracket$

Rule $\llbracket \text{'[' } P \text{ SP* ';' ? ']} \rrbracket : \text{pattern} = \llbracket \text{'[' } P \text{ SP* ']} \rrbracket$

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

Pattern evaluation

Semantics $\text{evaluate-pattern}[_ : \text{pattern}] : \Rightarrow \text{patterns}$

Rule $\text{evaluate-pattern}[VN] = \text{pattern-bind}(\text{value-name}[VN])$

Rule $\text{evaluate-pattern}[_] = \text{pattern-any}$

Rule $\text{evaluate-pattern}[CNST] = \text{value}[CNST]$

Rule $\text{evaluate-pattern}[P \text{ 'as' } VN] =$
 $\text{pattern-unite}(\text{evaluate-pattern}[P], \text{pattern-bind}(\text{value-name}[VN]))$

Rule $\text{evaluate-pattern}[P_1 \text{ 'I' } P_2] =$
 $\text{pattern-else}(\text{evaluate-pattern}[P_1], \text{evaluate-pattern}[P_2])$

Rule $\text{evaluate-pattern}[CSTR P] =$
 $\text{variant}(\text{constr-name}[CSTR], \text{evaluate-pattern}[P])$

Rule $\text{evaluate-pattern}[P_1 \text{ ', ' } P_2 CP^*] =$
 $\text{tuple}(\text{evaluate-comma-pattern-sequence}[P_1 \text{ ', ' } P_2 CP^*])$

Rule $\text{evaluate-pattern}[\{ F '=' P SFP^* \}] =$
 $\text{pattern closure}(\text{match-loosely}(\text{given}, \text{record}(\text{map-unite}(\text{evaluate-field-pattern-sequence}[F '=' P SFP^*])))$

Rule $\text{evaluate-pattern}[\text{'[' } P SP^* \text{ ']}] =$
 $[\text{evaluate-semic-pattern-sequence}[P SP^*]]$

Rule $\text{evaluate-pattern}[P_1 \text{ '::' } P_2] =$
 $\text{pattern closure}(\text{if-true-else}(\text{is-equal}(\text{given}, []), \text{fail}, \text{collateral}(\text{match}(\text{head}(\text{given}), \text{evaluate-pattern}[P_1]), \text{match}(\text{tail}(\text{given}), \text{evaluate-pattern}[P_2]))))$

Pattern sequence evaluation

Semantics $\text{evaluate-comma-pattern-sequence}[_ : (\text{pattern comma-pattern}^*)] : (\Rightarrow \text{patterns})^+$

Rule $\text{evaluate-comma-pattern-sequence}[P_1 \text{ ', ' } P_2 CP^*] =$
 $\text{evaluate-pattern}[P_1], \text{evaluate-comma-pattern-sequence}[P_2 CP^*]$

Rule $\text{evaluate-comma-pattern-sequence}[P] = \text{evaluate-pattern}[P]$

Semantics $\text{evaluate-semic-pattern-sequence}[_ : (\text{pattern semic-pattern}^*)] : (\Rightarrow \text{patterns})^+$

Rule $\text{evaluate-semic-pattern-sequence}[P_1 \text{ ';' } P_2 SP^*] =$
 $\text{evaluate-pattern}[P_1], \text{evaluate-semic-pattern-sequence}[P_2 SP^*]$

Rule $\text{evaluate-semic-pattern-sequence}[P] = \text{evaluate-pattern}[P]$

Semantics $\text{evaluate-field-pattern-sequence}[_ : (\text{field '=' pattern semic-field-pattern}^*)]]$
 $: \Rightarrow (\text{maps}(\text{ids}, \text{patterns}))^+$

Rule $\text{evaluate-field-pattern-sequence}[F_1 '=' P_1 ';' F_2 '=' P_2 \text{ SFP}^*] =$
 $($
 $\{ \text{field-name}[F_1] \mapsto \text{evaluate-pattern}[P_1] \},$
 $\text{evaluate-field-pattern-sequence}[F_2 '=' P_2 \text{ SFP}^*])$

Rule $\text{evaluate-field-pattern-sequence}[F '=' P] =$
 $\{ \text{field-name}[F] \mapsto \text{evaluate-pattern}[P] \}$