ML Syntax

Phrase ::= [Exp | SimpleDecl | Module | "use" String "monitor"] "."

Module ::= "module" ModuleName ["includes" (ModuleName)*] "body" Decl "end".

SimpleExp ::= ["op"] (Exp | Con) ["\"" (Exp | Con) | ("\"" (Exp | Con) "]

Exp ::= SimpleExp | Exp SimpleExp | Exp Typ e | Exp id e Exp | (Exp / "\"" | "\"" Exp | "escape" Exp |

if Exp "then" Exp "else" Exp "while" Exp "do" Exp |

let Decl in Exp "end" |

Exp "\"" Decl "end" |

case Exp of Match |

Exp "]" | Exp "?" Exp |

Exp "\"" id e "\"" Exp |

"fun" Match.

SimpleDecl ::= ["val" | "type" | "abstype" TypeBind |

["export" ExportList] "with" Decl "end" |

["import" ModuleName] |

["infix" (id e) 1 | "nonfix" (id e) 1]

Decl ::= SimpleDecl | Decl ";" Decl |

("Decl")

ExportList ::= ("abstype" | "type" | "val") (id e / ";") 1

ValBind ::= Pat ::= Exp |

["op"] id e SimplePat ["\"" TypePat] "=" Exp |

[end] id e SimplePat["type" TypePat] "=" Exp |

["\"" (Decl | Typ eBind | "\"" id e | "\""

ValBind "and" ValBind |

["rec" | "val"] ValBind.

TypeBind ::= [Param] TypeId ["of" Type] |

["\"" (Decl | Typ eBind | "\""

Param ::= Typ eVar | ("\"" (Typ eVar / "\"" | "\""

Keywords

? | ?? | "_" : abstype and body case do else export escape from fun import if in includes infix let local module nonfix of op rec then type use val where while

Operations

applicati on L L L (L div mod) L (L + | L | L | < | <= | = | = | > | > | = | = |

Digit ::= 0 | 1 | 9.

Symbol ::= # | & | $ | @ | \" " |

Character ::= h | i | o | e | l |

id e ::= Letter (Letter | Digit | "\"" | Symbol | 1.

Integer ::= (Digit | 1.

String ::= "\"" (Character | "\""

Con ::= (""

Integer | String | id e.

Typ eId ::= id e.

Typ eVar ::= "\"" id e.

ModuleName ::= id e | String.

Constructors

integers strings tuples () | true | false | nil :: ref user-defined

Metasyntax

Strings between quotes " " are terminals.

Identifiers are non-terminals.

Juxtaposition is syntactic concatenation.

["_" | "\""

is syntactic alternative.

["_" | "\""

is zero or one times (i.e. optionally) "_".

["_" | "\""

is n (default 0) or more times "_".

["_" | "\""

is n or more times "_" separated by "_".

Parentheses ("_

are used for precedence.

Syntactic alternatives and infix operators are listed in order of decreasing precedence.