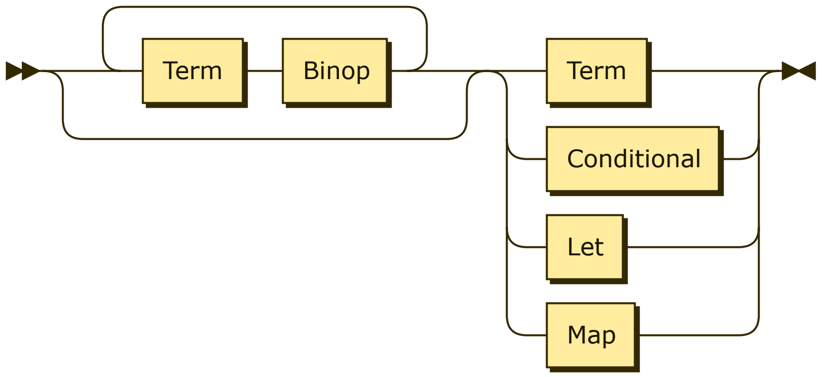
Exp:

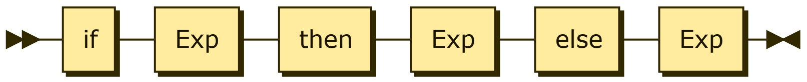


```
Exp      ::= ( Term Binop )* ( Term | Conditional | Let | Map )
```

referenced by:

- Conditional
- Def
- ExpList
- Factor
- Let
- Map

Conditional:

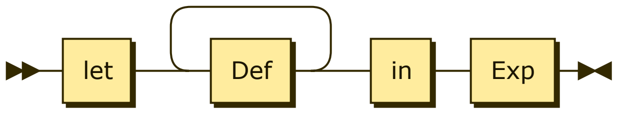


```
Conditional
         ::= if Exp then Exp else Exp
```

referenced by:

- Exp

Let:

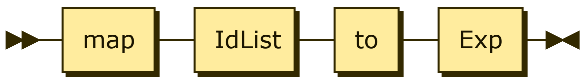


```
Let      ::= let Def+ in Exp
```

referenced by:

- Exp

Map:

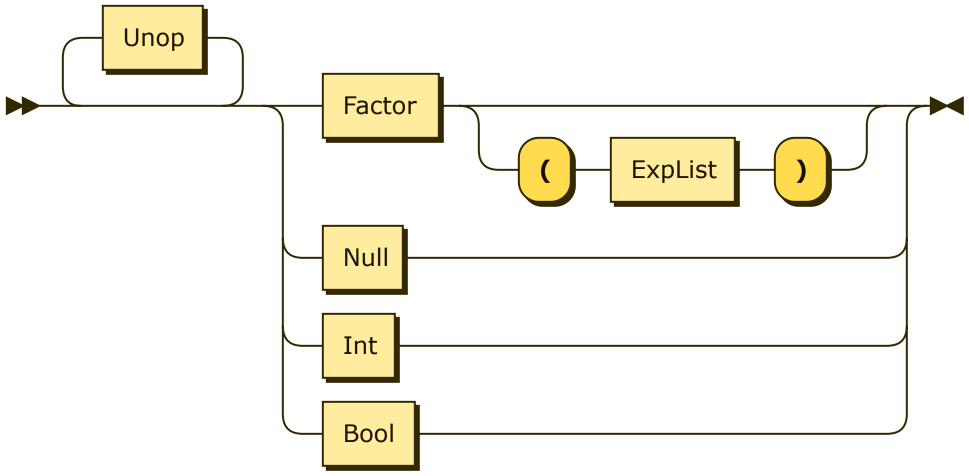


```
Map      ::= map IdList to Exp
```

referenced by:

- Exp

Term:

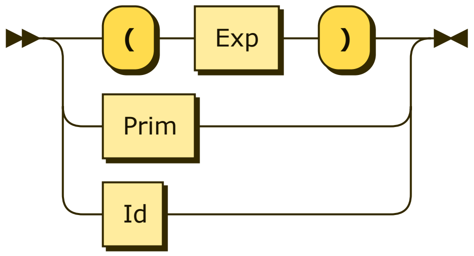


```
Term     ::= Unop* ( Factor ( '(' ExpList ')' )? | Null | Int | Bool )
```

referenced by:

- Exp

Factor:

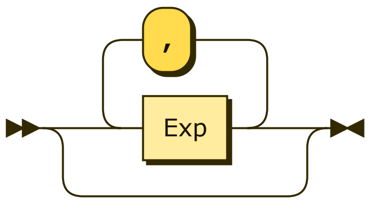


```
Factor   ::= '(' Exp ')'
           | Prim
           | Id
```

referenced by:

- Term

ExpList:

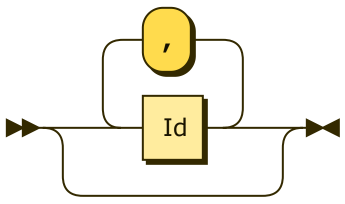


```
ExpList  ::= ( Exp ( ',' Exp )* )?
```

referenced by:

- Term

IdList:

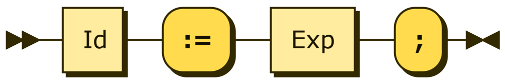


```
IdList   ::= ( Id ( ',' Id )* )?
```

referenced by:

- Map

Def:

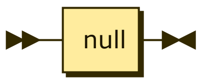


```
Def      ::= Id ':=' Exp ';'
```

referenced by:

- Let

Null:

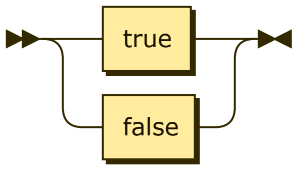


```
Null     ::= null
```

referenced by:

- Term

Bool:

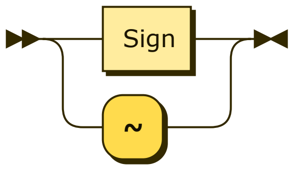


```
Bool     ::= true
           | false
```

referenced by:

- Term

Unop:

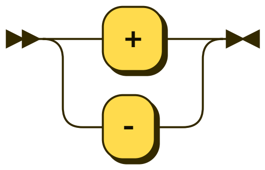


```
Unop     ::= Sign
           | '~'
```

referenced by:

- Term

Sign:

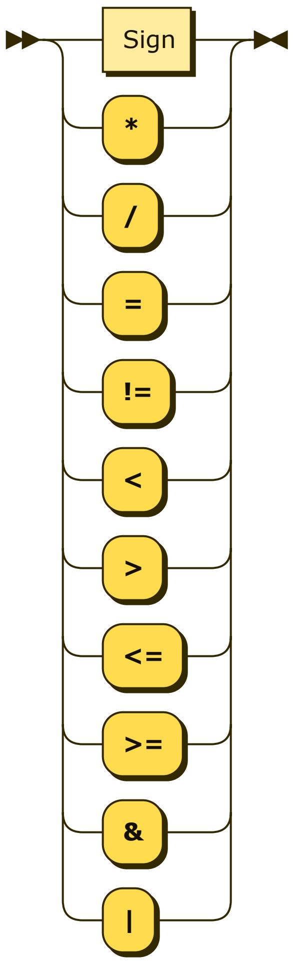


```
Sign     ::= '+'
           | '-'
```

referenced by:

- Binop
- Unop

Binop:

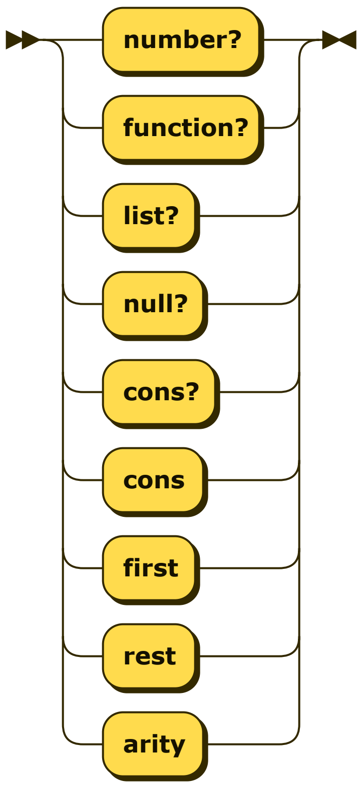


```
Binop    ::= Sign
           | '*'
           | '/'
           | '='
           | '!='
           | '<'
           | '>'
           | '<='
           | '>='
           | '&'
           | '|'
```

referenced by:

- Exp

Prim:



```
Prim     ::= 'number?'
           | 'function?'
           | 'list?'
           | 'null?'
           | 'cons?'
           | 'cons'
           | 'first'
           | 'rest'
           | 'arity'
```

referenced by:

- Factor
