ML | A Programming Language

eqbal.me

فهرست مطالب

o1/n2o

N2O: Standard ML Application Server

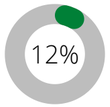
2 Contributors | 0 Issues | 19 Stars | 4 Forks

o1/n2o

N2O: Standard ML Application Server. Contribute to o1/n2o development by creating an account on GitHub.

GitHub

1. بهینه بودن
2. چند نمونه ای (Multi paradigm)
3. سادگی
4. تابع های رده بالا
5. بررسی نوع ایستا
6. چند ریختی
7. امکان پیاده سازی روابط جبری
8. ماژولار



ML
Meta Language

1400/3

ML (Meta Language)

یک زبان برنامه نویسی تابعی همه منظوره

ساخته شده توسط رابین میلر و همکاران در اواخر دهه ۱۹۷۰

کلمه ML از فوق زبان (Meta langauge) گرفته شده است.

ML برای بهبود بخشیدن به رویه اثبات در قضیه LFC طراحی شده است.

- نوشتن کامپایلر
- اثبات قضیه های ریاضی

بیشتر زبان های حال حاضر از جمله:

Rust, Scala, C++, F#, Kotlin, Haskell

کامپایلرشان بر اساس زبان ML پیاده سازی شده است.

ML
Meta Language

1400/3

1. داده
2. عملیات اصلی
3. کنترل ترتیب
4. کنترل داده
5. مدیریت حافظه
6. محیط عملیاتی

ML
Meta Language

1400/3

داده اولیه

- integer (int)
- reals (real)
- Bool (bool)
- Strings (string)

داده اولیه

- عدد صحیح (integers)

```
Attribute: int
Value: 1, 2, ~3, ~4
Operattors: +, -, *, div, mod, <,<=,>,>=,<>

   > val x = 7 : int
```

داده اولیه

- عدد حقیقی (Reals)

```
Attribute: real
Value: 3.0, 3.1415, ~3.2E2
Operation: +,-,*,/,<,<=,>,>=,<>,log,exp,sin,arctan

   > val length = 7.5 : real
```

داده اولیه

- بول (bool)

```
Attribute: bool
Value: true, false,
Operation: not, orelse(logical or), andalso(logical and)

   > val status = true : bool
```

داده اولیه

- رشته (String)

```
Attribute: String
Value: ”abc”, “hello world\n”, x^”.sml”
Operation: ^(concatenation),length,substring

   > val university = "uok" : string
```

داده های ساختار یافته

- **Tuples:** ordered collection of values
- **Records:** collection of named values
- **Lists:** list of values of homogeneous type

داده های ساخت یافته

- تاپل (Tuples)

```
>> Syntax: ( exp-list )
   - ( 1, 2, 3);
   val it = (1,2,3) : int * int * int

>> Accessing bypattern matching or by label
- val (a,b) = (2.3,"zippy");
val a = 2.3: real
val b = "zippy": string
- #2 (a,b);
val it = "zippy" : string
```

ML
Meta Language

داده های ساخت یافته

- رکورد (Records)

```
>> A collection of labeled data items:
   - val ex = { name = "eqbal", userid = 7 };
   {name:string, userid:int}

>> Accessing by pattern matching or by label
-  #name ex;
val it = "eqbal": string
```

داده های ساخت یافته

- لیست (Lists)

```
>> all elements must be of same type
>> Syntax:
   - [ 2, 6, 4, 9];
   val it = [2,6,4,9]: int list
   - [ ¨a¨, ¨b¨, ¨c¨];
   val it = [¨a¨,¨b¨,¨c¨]: string list

>> Accessing and Selection:
```

عملیات اصلی

```
-> : => variable type assign
-> | => pip
-> ' => alpha
-> . -> member access

(* comment *)
```

کنترل ترتیب

Expressions

Arithmetic expressions

$\sim$	Unary minus
$*$, $/$, div, mod	Multiplicative operators
$+$, $-$	Additive operators

Eachexpression is terminated by a semicolon (;)

$$(expression_1; expression_2; \ldots; expression_n)$$

arithmetic relationships

$$=, <>, >=, <=, <, \text{and } >$$

کنترل ترتیب

If expression

$$\textbf{if } expression \textbf{ then } true\_part \textbf{ else } false\_part$$

```
13  fun max(ls : int list)=
14      if null (tl ls)
15      then hd ls
16      else
17          let val tmp = max(tl ls)
18          in if (hd ls) < tmp then tmp else (hd ls)
19          end;
```

ML
Meta Language

کنترل ترتیب

Function Definition

$$\textbf{fun}\ function\_name\ (\ parameters\ )\ =\ expression;$$

```
13  fun max(ls : int list)=
14      if null (tl ls)
15      then hd ls
16      else
17          let val tmp = max(tl ls)
18          in if (hd ls) < tmp then tmp else (hd ls)
19          end;
```

کنترل ترتیب

```
loop expression
```

$$\textbf{while } expression_1 \textbf{ do } expression_2$$

کنترل ترتیب

Exceptions: **exception** *exception_name*

raise *exception_name*

handle by: *expression* **handle** *handler*

```
1  fun hd xs =
2    case xs of
3         [] => raise List.Empty
4       | x::_ => x;
5
6
7  (*  *)
8  exception MyException;
9
```

hd(list) return the first element of list

کنترل ترتیب

Structure:

structure *identifier* = **struct**
    $element_1$
    $element_2$
    ...
    $element_n$ **end**

```
- structure ListItem = struct
= val NewList = nil;
= fun add([ ],y)= y :: nil |
=       add(x,y) = x @ [y];
= fun size(x) = if x=[ ] then 0 else 1+size(tl(x));
= fun delete(x) = tl(x)
= end;
```

ML
Meta Language

کنترل داده

```
22  fun n_times(f, n, x)=
23    if n = 0
24    then x
25    else f(n_times(f, n - 1, x));
26
27  fun double x = x + x;
28  fun increment x = 1 + x;
29
30  (n_times(double, 4,2) = 32);
31  (n_times(double, 4,7) = 112);
32  (n_times(increment, 4,2) = 6);
33  (n_times(tl, 4, [1,2,3,4,5]) = [5]);
```

ML
Meta Language

مدیریت حافظه

ML نیز مانند بیشتر زبان های تابعی دارای مدیریت حافظه خودکار است.
مدیریت حافظه خودکار تکنیکی عمومی برای بازیافت حافظه اختصاص یافته به هر شی به صورت خودکار است. هرچند این از لحاظی غیر ممکن است که همیشه بتوان تعیین کرد که هر شی در حافظه مورد نیاز است یا خیر و به همین خاطر از روش های garbage collector برای رفع این مشکل استفادھ شدھ است.

در نسخه های کامپیلر جدید این زبان قابلیت پیشرفته garbage collectors را نیز به خوبی پشتیبانی میکند

محیط عملیاتی

محیط عملیاتی زبان از دو نوع intractive و command line پشتیبانی میکند و میتوان با استفادھ از خط فرمان و نصف کامپایلر به اجرای دستورات زبان پرداخت.