Ataabu 語言的實作

Tan, Kian-ting

目錄

1. 句法解釋

以下針對這個語言的句法 (syntax)，進行解釋。

1.1. 語法草稿

1. 該語法範例碼如下，AST以後再行生成：
2. 一些注意事項：
    1. 不支援可變變數
    2. 不支援迴圈

```
int x = 10; # 常數定義法，這是註解
flo y = 10.1 + int2flo(x) / 3.0;  # 浮點數
str name = "John"; # 字串，預設用utf-8
bool c = True; # 布林值，或是 False
int x = ({int x -> int : return x + 1;})(5); # gets 6

'''
這是多行註解
函數上面的多行註解會轉成 docString，先用 markdown 做子語言吧;
底下爲列表和陣列
'''
List(int) a_list = [1, 2, 3, 4, 5];
Array(int) a_array = array![1, 2, 3, 4, 5]);

# 以下是 doc string 的範例，前面要加 @doc：
@doc '''
# Desc
find the sqrt sum of x and y

# Args
 - x : lfs value
 - y : rhs value

# Eg
    sqrtsum(3, 4) == 25; # True
'''
fn sqrtSum = int x, int y -> int :
             int z = x ** 2 + y ** 2;
             return z;
```

```
fn isJohn = str s -> bool :
             return case {
               #print! 是巨集。!結尾是巨集名稱。 ++ 是字串相加
               s == john -> {print!("Hi, " ++ s ++ "!\n");
                            True;}
               else -> False;
               };

#不返回值(void)的時候也要標註 return;
fn printCat = void -> void :
                       print!("cat!");
                       return ;

# 多型：
# @{} vars of Type with constraints
fn map = @(A, B  : Any) # or @(A, B in Any)
              (List A) origList; ( A -> B ) aFunction -> (List B) :
          return match origList{
             [] -> origList;
             [x:xs] -> cons(aFunction(origList),
                            map(origList[1:], aFunction));
          };

# 定義自定型別：
type Person = Person(str name, int age);
type Person = Person(str , int );
type TrafficLight = Red | Yellow | Green;
type List a = @(a : Any) Nil | Cons(a, List(a));

Person peter = Person{"Peter", 17};

print!(peter.name); # print "Peter"

debug!(peter); #印出 Person{name : "Peter", age : 17}這樣。

str myNameIsPeter = match peter
{
    "Peter" _ -> "Is Peter ainm dom",
```

```
    _ _ -> "Ní Peter ainm dom"

}

str t = peter.name[5];

#分行字串
str multilineString = '''
jumps over the lazy dog.
'''

# 將peter的內容克隆一份，然後屬性name改成"John"，傳予john
Person john = peter[name <- "John"];

@lang DSL # 匯入語言
import datetime; #匯入模組
importPath "/path/to/file.toy" #匯入路徑程式檔


@doc '''
這是給模組的 @doc，表示資訊
'''
```

1.2. 語法生成文法

以下用類似 BNF 文法來表示這個玩具語言：

- `a{n}` 表示a出現n次
- `a?` 表示a出現0,1次
- `a*` 表示a出現0次以上
- `a+` 表示a出現1次以上
- `#` 註解
- `\x` 脫逸字元 x
- `ID` identifier
- `(not a [b…])` 不是 a (, b...) 的字元，方括號[]內表示非必要，可選。
    - `ASCII_SPACE`：Ascii 的半形空白字元
    - `CHAR`：任意字元
    - `$`：指1個以上空白或縮排字元。
    - `$*`：指0個以上空白或縮排字元。

```
ALL = LANG_HEADER? EMPTY_LINE* IMPORTS? EMPTY_LINE* (DOCSTRING)?  BODY # 所有的內容

#匯入
IMPORTS = IMPORT {EMPTY_LINE* IMPORT}+
IMPORT = IMPORTLIB $ NL | IMPORT_PATH $ NL
IMPORTLIB = import $ ID # 匯入自函式庫
IMPORT_PATH = importPath $  ( $ " NON_EMPTY_STRING " $ ) #匯入自檔案路徑

#定義DSL
LANG_HEADER = # $ lang $ ID #使用語言的模式

# 定義DocumentString
DOCSTRING = @ $ doc $ DOCSTR_CONTENT

#定義DocumentString的細部。本段放在另一個解析器, x = anotherParser(DOCSTRCONTENT)
DOCSTR_CONTENT = ''' NL # $ DOCSTR_SECTION $ NL DOCSTR_SECTION_EXPLANATION'''
DOCSTR_SECTION = ID

# docstring的各項解釋, 開頭不能用井號
DOCSTR_SECTION_EXPLANATION = {NON_HASH_CHAR NON_EMPTY_STRING NL}+

BODY = BODY_LINE*
BODY_LINE = INLINE_COMMENT | VAR_DEF | EXPR | TYPE_DEF | MACRO_DEF | EMPTY_LINE
INLINE_COMMENT = $* # NON_NL_CHAR+ NL

VAR_DEF = $* TYPE_ID $ VAR_ID $* = $* EXPR $* ;

# Type Definition, eg.
# type List a = @(a : Any) Nil | Cons(a, List(a));
TYPE_DEF = Type $ TYPEID $* = $* POLYMOR? $ SUM_TYPE ;

SUM_TYPE = PRODUCT_TYPE $* { | $* PRODUCT_TYPE $*}*
```

```
PRODUCT_TYPE = UNIRY_CONSTRUCT | RECORD | STRUCT;
UNIRY_CONSTRUCT = CONSTRUCT_ID;
RECORD = CONSTRUCT_ID $* ( $* CONSTRUCT_LIST $* )
CONSTRUCT_LIST = CONSTRUCT {$*, $* CONSTRUCT}*
CONSTRUCT = TYPE_VAR TYPE_AUG?
TYPE_AUG = $* ( $* TYPE_VAR TYPE_AUG_REMAINED* $* )
TYPE_AUG_REMAINED = $*, $*  TYPE_VAR

# 定義 structure 和 attribution
STRUCT = ATTR_ID $* ( $* ATTR_LIST $* )
ATTR_LIST= ATTR {$*, $* ATTR}+
ATTR = CONSTRUCT ATTR_ID
ATTR_ID = ID
TYPE_ID = ID

# 空行
EMPTY_LINE = SPACE_TAB* NL
#換行
NL = \n | \r

#非半形井號字元
NON_HASH_CHAR = (not #)

#非換行字元
NON_NL_CHAR = (not \n \r)
NON_EMPTY_STRING = CHAR+ #非空串
EMPTY_STRING = CHAR{0} #空字串
STRING = NON_EMPTY_STRING | EMPTY_STRING
SPACE_TAB  = ASCII_SPACE | \t #空白與縮排字元, 簡稱為 $
ID = [a-zA-Z_][_a-zA-Z0-9]* # identifier, in regexp
```