第 6 讲: 程序设计语言的语法与语义

姓名： 林凡琪　　**学号：** 211240042

评分： _____　　**评阅：** _____

2021 年 11 月 11 日

请独立完成作业，不得抄袭。
若得到他人帮助, 请致谢。
若参考了其它资料，请给出引用。
鼓励讨论，但需独立书写解题过程。

- 逻辑系统有语法、语义之分; 程序设计语言亦如是
- 欢迎进入程序设计语言理论的广阔世界

1　作业（必做部分）

题目 1 (IMP)
考虑程序设计语言 **IMP** (Imperative), 它包含如下元素:

- 整数集合 $\mathbb{Z}$ (可用 m, n 表示其中的元素)
- 真值集合 $\mathbb{T} = \{T, F\}$
- 变量集合 $\mathbb{V}$ (可用 v 表示其中的元素)
- 算法表达式 **Aexp**, 支持 “$+, -, \times$” 三则运算
- 布尔表达式 **Bexp**, 支持 “$=, \leq$” 比较操作与基本的逻辑操作
- 语句 **St**, 包括赋值语句 (:=)、顺序语句 (;)、选择语句 (if-then-else) 与循环语句 (while-do)

请为 **IMP** 设计语法，并使用 BNF 描述。
例如:

< 字母 >::=< 大写字母 >< 小写字母 >

< 大写字母 >::=A|B|C|D|E|F|G|H|I|J|K|L|M|N|O|P|Q|R|S|T|U|V|W|X|Y|Z

< 小写字母 >::=a|b|c|d|e|f|g|h|i|j|k|l|m|n|o|p|q|r|s|t|u|v|w|x|y|z

解答:
< 整数集合 $\mathbb{Z}$>::=(”1”|”2”|”3”|”4”|”5”|”6”|”7”|”8”|”9”)[{”1”|”2”|”3”|”4”|”5”|”6”|”7”|”8”|”9”|”0”}]

< 真值集合 $\mathbb{T}$>::=T|F

< 变量集合 $\mathbb{V}$>::= < 真值集合 $\mathbb{T}$>|< 整数集合 $\mathbb{Z}$>

<Aexp>::=< 整数集合 $\mathbb{Z}$> | < 变量集合 $\mathbb{V}$> | <Aexp> "+" <Aexp>|<Aexp> "-" <Aexp>|<Aexp> "×" <Aexp>

<Bexp>::=<Aexp>"="<Aexp>|<Aexp>"≤"<Aexp>|<Bexp> "and" <Bexp>|<Bexp> "or" <Bexp>|"not"<Bexp>

< 语句 St>::=< 赋值语句 >|< 顺序语句 >|< 选择语句 >|< 循环语句 >

< 赋值语句 >::=< 变量集合 $\mathbb{V}$>":=" < 整数集合 $\mathbb{Z}$>|< 变量集合 $\mathbb{V}$>":="< 真值集合 $\mathbb{T}$>

< 顺序语句 >::=< 语句 St>";"

< 选择语句 >::=if < 真值集合 $\mathbb{T}$> then < 语句 St> else < 语句 St>

< 循环语句 >::=while < 真值集合 $\mathbb{T}$> do < 语句 St>

题目 2 (Prolog 逻辑推理)

学习 Prolog 相关知识 ① ，并且完成下列程序 ② 。

这个程序定义了一个 parent 关系和两个谓词 isAncestor 和 isCollaterialRelative。其中，两个谓词的定义并不完整，因此也不能通过下方附带的测试样例。请修改 isAncestor 和 isCollaterialRelative 的定义，使其符合注释要求的功能。

① 推荐阅读 教程 3.1 至 3.9 章的内容。
② 你可以使用 在线运行环境 完成程序。

```
parent(a1, b1).
parent(a1, b2).
parent(b1, c1).

parent(a2, b3).
parent(a3, b3).
parent(b3, c4).
parent(b3, c5).

/* 判断 X 是否是 Y 的祖先。 */
isAncestor(X, Y) :-
    true.

/* 判断 X 是否是 Y 的旁系血亲，即 X 和 Y 有一个共同祖先，但他们互相不是对方的祖先 */
isCollaterialRelative(X, Y) :-
    true.

test(Description, Assertion) :-
    Assertion, write(Description), write(' passed.'), nl;
    \+ Assertion, write(Description), write(' failed!'), nl.

main :-
    test("Ancestor Test 1", isAncestor(a1, c1)),
    test("Ancestor Test 2", \+ isAncestor(b1, a1)),
    test("Ancestor Test 3", \+ isAncestor(b1, b2)),
```

```
26      test("Ancestor Test 4", isAncestor(a2, c4)),
27      test("Ancestor Test 5", \+ isAncestor(a2, a2)),
28      test("Ancestor Test 6", \+ isAncestor(a1, b3)),
29
30      test("Collateral Relative Test 1", isCollaterialRelative
            (c1, b2)),
31      test("Collateral Relative Test 2", isCollaterialRelative
            (c4, c5)),
32      test("Collateral Relative Test 3", \+
            isCollaterialRelative(b3, c4)),
33      test("Collateral Relative Test 4", \+
            isCollaterialRelative(c4, b3)),
34      test("Collateral Relative Test 5", \+
            isCollaterialRelative(c1, c4)).
```

解答:

③

```
1  parent(a1, b1).
2  parent(a1, b2).
3  parent(b1, c1).
4
5  parent(a2, b3).
6  parent(a3, b3).
7  parent(b3, c4).
8  parent(b3, c5).
9
10 /* 判断 X 是否是 Y 的祖先。 */
11 isAncestor(X, Y):-
12     parent(X, Y).
13 isAncestor(X, Y) :-
14     parent(X, T1),
15     parent(T1, Y).
16
17 /* 判断 X 是否是 Y 的旁系血亲, 即 X 和 Y 有一个共同祖先, 但他
       们互相不是对方的祖先 */
18 isCollaterialRelative(X, Y) :-
19     isAncestor(T, X),
20     isAncestor(T, Y),
21     parent(T1, X),
22     parent(T2, Y),
23     X \= T2,
24     Y \= T1.
25
26 test(Description, Assertion) :-
27     Assertion, write(Description), write(' passed.'), nl;
28     \+ Assertion, write(Description), write(' failed!'), nl.
29
30 main :-
31     test("Ancestor Test 1", isAncestor(a1, c1)),
32     test("Ancestor Test 2", \+ isAncestor(b1, a1)),
```

③ 作业压缩包中含有上述程序的源代码 prolog1-sol.pl, 你可以取消下面一行的注释并将修改后的程序放入 prolog1-sol.pl 中。

```
33      test("Ancestor Test 3", \+ isAncestor(b1, b2)),
34      test("Ancestor Test 4", isAncestor(a2, c4)),
35      test("Ancestor Test 5", \+ isAncestor(a2, a2)),
36      test("Ancestor Test 6", \+ isAncestor(a1, b3)),
37
38      test("Collateral Relative Test 1", isCollaterialRelative
            (c1, b2)),
39      test("Collateral Relative Test 2", isCollaterialRelative
            (c4, c5)),
40      test("Collateral Relative Test 3", \+
            isCollaterialRelative(b3, c4)),
41      test("Collateral Relative Test 4", \+
            isCollaterialRelative(c4, b3)),
42      test("Collateral Relative Test 5", \+
            isCollaterialRelative(c1, c4)).
```

题目 3 (Prolog 逻辑推理)

下面这个 Prolog 程序试图解决一个经典的逻辑推理题。请修改谓词 xxxGuess 的定义，使其能够对 X, Pos 变量返回正确的结果。

> 赵钱孙李周吴郑王八位将军外出打猎，有一位将军的箭射中了一只鹿。在拔出箭头前，众人饶有兴致的猜起来是谁射中了鹿。
>
> 1. 赵说：“或者是王将军射中的，或者是吴将军射中的。”
> 2. 钱说：“如果这只箭正好射中了鹿的头部，那么肯定是我射中的。”
> 3. 孙说：“我确定是郑将军射中的。”
> 4. 李说：“即使箭正好在鹿的头上，也不可能是钱将军射中的。”
> 5. 周说：“赵将军猜错了。”
> 6. 吴说：“不是我射中的，也不是王将军射中的。”
> 7. 郑说：“不是孙将军射中的。”
> 8. 王说：“赵将军没有猜错。”
>
> 最后，大家把鹿身上的箭拔出来查看，八位将军中正好有三个人猜对了。

```
1  /* 定义八个将军 */
2  general(zhao).
3  general(qian).
4  general(sun).
5  general(li).
6  general(zhou).
7  general(wu).
8  general(zheng).
9  general(wang).
10
11 /* 射中鹿的位置 */
12 position(head).
13 position(other).
14
15 /* 赵的猜测, X 代表射中鹿的人 */
16 zhaoGuess(X) :-
17     false.
```

```
18
19 /* 钱的猜测, X 代表射中鹿的人, Pos 代表射中鹿的位置 */
20 qianGuess(X, Pos) :-
21     false.
22
23 sunGuess(X) :-
24     false.
25
26 liGuess(X, Pos) :-
27     false.
28
29 zhouGuess(X) :-
30     false.
31
32 wuGuess(X) :-
33     false.
34
35 zhengGuess(X) :-
36     false.
37
38 wangGuess(X) :-
39     false.
40
41 /* 计算列表中为“真”的谓词数量 */
42 count([], 0).
43 count([X|L], Count) :-
44     X,
45     count(L, Count1),
46     Count is Count1 + 1.
47 count([X|L], Count) :-
48     \+ X,
49     count(L, Count).
50
51 writeAnswer(X, Pos) :-
52     write('射死鹿的人是'), write(X), nl,
53     (zhaoGuess(X), write('赵猜对了'), nl ; true),
54     (qianGuess(X, Pos), write('钱猜对了'), nl ; true),
55     (sunGuess(X), write('孙猜对了'), nl ; true),
56     (liGuess(X, Pos), write('李猜对了'), nl ; true),
57     (zhouGuess(X), write('周猜对了'), nl ; true),
58     (wuGuess(X), write('吴猜对了'), nl ; true),
59     (zhengGuess(X), write('郑猜对了'), nl ; true),
60     (wangGuess(X), write('王猜对了'), nl ; true).
61
62 /* 判定射死鹿的人为 X, 射中部位为 Pos 是否满足题目条件 */
63 ans(X, Pos) :-
64     general(X),
65     position(Pos),
66     count([zhaoGuess(X),
67            qianGuess(X, Pos),
68            sunGuess(X),
69            liGuess(X, Pos),
```

```
70              zhouGuess(X),
71              wuGuess(X),
72              zhengGuess(X),
73              wangGuess(X)],
74             3),
75      writeAnswer(X, Pos).
76
77  /*
78   * 在交互模式中使用 ans(X, Pos) 输出所有满足条件的解。
79   */
```

解答:

程序计算出的结果为:

X 将军射中了鹿头/鹿身, 且将军 A、B、C 猜对了。④

修改后的程序:

④ 作业压缩包中含有上述程序的源代码 prolog2-sol.pl, 你可以取消下面一行的注释并将修改后的程序放入 prolog2-sol.pl 中。

```
1   /* 定义八个将军 */
2   general(zhao).
3   general(qian).
4   general(sun).
5   general(li).
6   general(zhou).
7   general(wu).
8   general(zheng).
9   general(wang).
10
11  /* 射中鹿的位置 */
12  position(head).
13  position(other).
14
15  /* 赵的猜测, X 代表射中鹿的人 */
16  zhaoGuess(X) :-
17      X = wu.
18  zhaoGuess(X) :-
19      X = wang.
20
21  /* 钱的猜测, X 代表射中鹿的人, Pos 代表射中鹿的位置 */
22  qianGuess(X, Pos) :-
23      Pos = head,
24      X = qian.
25  qianGuess(X, Pos) :-
26      Pos = other.
27
28  sunGuess(X) :-
29      X = zheng.
30
31  liGuess(X, Pos) :-
32      Pos = head,
33      X \= qian.
34
```

```
35 zhouGuess(X) :-
36     X \= wu,
37     X \= wang.
38
39 wuGuess(X) :-
40     X \= wu,
41     X \= wang.
42
43 zhengGuess(X) :-
44     X \= sun.
45
46 wangGuess(X) :-
47     X = wu.
48 wangGuess(X) :-
49     X = wang.
50
51 /* 计算列表中为“真”的谓词数量 */
52 count([], 0).
53 count([X|L], Count) :-
54     X,
55     count(L, Count1),
56     Count is Count1 + 1.
57 count([X|L], Count) :-
58     \+ X,
59     count(L, Count).
60
61 writeAnswer(X, Pos) :-
62     write('射死鹿的人是'), write(X), nl,
63     (zhaoGuess(X), write('赵猜对了'), nl ; true),
64     (qianGuess(X, Pos), write('钱猜对了'), nl ; true),
65     (sunGuess(X), write('孙猜对了'), nl ; true),
66     (liGuess(X, Pos), write('李猜对了'), nl ; true),
67     (zhouGuess(X), write('周猜对了'), nl ; true),
68     (wuGuess(X), write('吴猜对了'), nl ; true),
69     (zhengGuess(X), write('郑猜对了'), nl ; true),
70     (wangGuess(X), write('王猜对了'), nl ; true).
71
72 /* 判定射死鹿的人为 X, 射中部位为 Pos 是否满足题目条件 */
73 ans(X, Pos) :-
74     general(X),
75     position(Pos),
76     count([zhaoGuess(X),
77            qianGuess(X, Pos),
78            sunGuess(X),
79            liGuess(X, Pos),
80            zhouGuess(X),
81            wuGuess(X),
82            zhengGuess(X),
83            wangGuess(X)],
84           3),
85     writeAnswer(X, Pos).
86
```

```
/*
 * 在交互模式中使用 ans(X, Pos) 输出所有满足条件的解。
 */
```

题目 4 (Prolog 斐波那契)

这个程序定义了两个谓词 fib 和 find_fib。其中，两个谓词的定义并不完整，因此也不能通过下方附带的测试样例。请修改 fib 和 find_fib 的定义，使其符合注释要求的功能。

```
/*
 * 在C++中，我们可以这样定义阶乘函数
 int factorial(int n)
 {
    if (n == 1) return 1;
    int tmp = factorial(n - 1);
    return tmp * m;
 }

 * 在prolog的视角下，我们可以将其中函数看成一组关系：
   - 在谓词factorial下，(1, 1)构成一组关系。
   - 在谓词factorial下，(n - 1, tmp)构成一组关系。
   - 在谓词factorial下，(n, tmp * m)构成一组关系。

 *  因此，我们可以写出以下的代码：
 */

factorial(0, 1).
factorial(1, 1).

factorial(N, M) :-
    N > 1,
    N1 is N - 1,
    factorial(N1, M1),
    M is M1 * N.

% 上述定义为：
% 在 N > 1时，谓词factorial采用递归定义
% 条件 (N > 1)
% 定义N1 (其值为N - 1)
% 寻找与N1在谓词factorial下构成关系的变量：M1
% 推出与N在谓词factorial下构成关系的变量：M1 * N

% 注意，此时用 N1 = N - 1是错的
/*
 int fibonacci(int n)
 {
    if (n == 1) return 1;
    if (n == 2) return 1;
    return fibonacci(n - 1) + fibonacci(n - 2);
 }

```

```
 练习: 请给出斐波那契数列, 在prolog下的递归定义

*/

fibonacci(1, 1).

fibonacci(2, 1).

fibonacci(N, M) :-
    true.

/*
 练习:
 findFibonacci(N, M, T)代表第T个斐波那契数为N, 第T + 1个斐波
    那契数为M的一组关系。
 请给出findFibonacci的定义。
*/

findFibonacci(1, 1, 1).
findFibonacci(1, 2, 2).

findFibonacci(N, M, T) :-
    true.

test(Description, Assertion) :-
    Assertion, write(Description), write(' passed.'), nl;
    \+ Assertion, write(Description), write(' failed!'), nl.

main:-
    test("fibonacci Tset 1", \+ fibonacci(1, 2)),
    test("fibonacci Tset 2", fibonacci(5, 5)),
    test("fibonacci Tset 3", \+ fibonacci(10, 66)),
    test("fibonacci Tset 4", fibonacci(20, 6765)),
    test("fibonacci Tset 5", fibonacci(25, 75025)),
    test("findFibonacci Tset 1", \+ findFibonacci(2, 3, 4)),
    test("findFibonacci Tset 2", findFibonacci(3, 5, 4)),
    test("findFibonacci Tset 3", \+ findFibonacci(5, 8, 8)),
    test("findFibonacci Tset 4", findFibonacci(8, 13, 6)),
    test("findFibonacci Tset 5", findFibonacci(13, 21, 7)).
```

解答:

⑤

```
/*
 * 在C++中, 我们可以这样定义阶乘函数
 int factorial(int n)
 {
    if (n == 1) return 1;
    int tmp = factorial(n - 1);
    return tmp * m;
 }
```

⑤ 作业压缩包中含有上述程序的源代码 prolog3-sol.pl, 你可以取消下面一行的注释并将修改后的程序放入 prolog3-sol.pl 中。

```
9
10  * 在prolog的视角下, 我们可以将其中函数看成一组关系:
11    - 在谓词factorial下, (1, 1)构成一组关系。
12    - 在谓词factorial下, (n - 1, tmp)构成一组关系。
13    - 在谓词factorial下, (n, tmp * m)构成一组关系。
14
15  *  因此, 我们可以写出以下的代码:
16  */
17
18 factorial(0, 1).
19 factorial(1, 1).
20
21 factorial(N, M) :-
22     N > 1,
23     N1 is N - 1,
24     factorial(N1, M1),
25     M is M1 * N.
26
27 % 上述定义为:
28 % 在 N > 1时, 谓词factorial采用递归定义
29 % 条件 (N > 1)
30 % 定义N1 (其值为N - 1)
31 % 寻找与N1在谓词factorial下构成关系的变量: M1
32 % 推出与N在谓词factorial下构成关系的变量: M1 * N
33
34 % 注意, 此时用 N1 = N - 1是错的
35 /*
36  int fibonacci(int n)
37  {
38      if (n == 1) return 1;
39      if (n == 2) return 1;
40      return fibonacci(n - 1) + fibonacci(n - 2);
41  }
42
43  练习: 请给出斐波那契数列, 在prolog下的递归定义
44
45 */
46
47 fibonacci(1, 1).
48
49 fibonacci(2, 1).
50
51 fibonacci(N, M) :-
52     N > 2,
53     N1 is N - 1,
54     N2 is N - 2,
55     fibonacci(N1, M1),
56     fibonacci(N2, M2),
57     M is M1 + M2.
58
59 /*
60  练习:
```

```
61  findFibonacci(N, M, T)代表第T个斐波那契数为N, 第T + 1个斐波
        那契数为M的一组关系。
62  请给出findFibonacci的定义。
63 */
64
65 findFibonacci(1, 1, 1).
66 findFibonacci(1, 2, 2).
67
68 findFibonacci(N, M, T) :-
69     T > 2,
70     fibonacci(T, A1),
71     N is A1,
72     fibonacci(T + 1, A2),
73     M is A2.
74
75
76 test(Description, Assertion) :-
77     Assertion, write(Description), write(' passed.'), nl;
78     \+ Assertion, write(Description), write(' failed!'), nl.
79
80 main:-
81     test("fibonacci Tset 1", \+ fibonacci(1, 2)),
82     test("fibonacci Tset 2", fibonacci(5, 5)),
83     test("fibonacci Tset 3", \+ fibonacci(10, 66)),
84     test("fibonacci Tset 4", fibonacci(20, 6765)),
85     test("fibonacci Tset 5", fibonacci(25, 75025)),
86     test("findFibonacci Tset 1", \+ findFibonacci(2, 3, 4)),
87     test("findFibonacci Tset 2", findFibonacci(3, 5, 4)),
88     test("findFibonacci Tset 3", \+ findFibonacci(5, 8, 8)),
89     test("findFibonacci Tset 4", findFibonacci(8, 13, 6)),
90     test("findFibonacci Tset 5", findFibonacci(13, 21, 7)).
```

2 Open Topics

Open Topics 1 (正则表达式)
请介绍正则表达式 (Regualr Expression) 的语法、语义与用例等。
基本要求:

- 循序渐进
- 使用有趣而实用的例子

参考资料:

- Regular expression @ wiki
- regex101

Open Topics 2 (程序设计语言的语义)
阅读并介绍经典论文 "CACM1969 (Hoare) An Axiomatic Basis for Computer Progamming" @ problem-solving-class-paperswelove:

- 作者简介 Tony Hoare @ wiki
- 概览文章的结构与贡献
- 重点介绍第三节 “Program Execution” 的内容
- 介绍 Table III 中的证明示例

3 订正

4 反馈

你可以写 ⑥：

- 对课程及教师的建议与意见
- 教材中不理解的内容
- 希望深入了解的内容
- ···

⑥ 优先推荐 ProblemOverflow