記号処理・中野・木 4 講時

2012 年 8 月 2 日

I　拡張 BNF 記法で書かれた次の文法について考える。

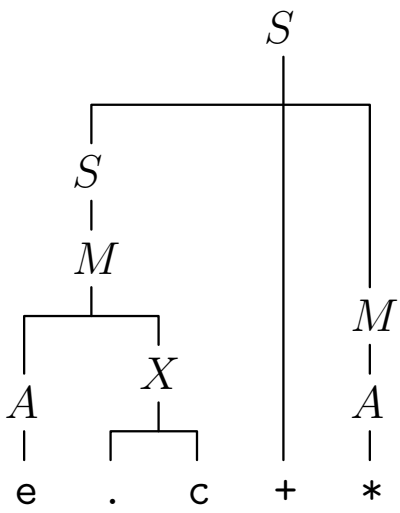
$\langle A \rangle$ ::= “`*`” | “`e`” .　　$\langle X \rangle$ ::= “`.`” “`c`” | “`#`” “`i`” .

$\langle M \rangle$ ::= $\langle A \rangle$ [$\langle X \rangle$] | $\langle X \rangle$.　　$\langle S \rangle$ ::= $\langle M \rangle$ | $\langle S \rangle$ [“`>`” | “`+`”] $\langle M \rangle$.

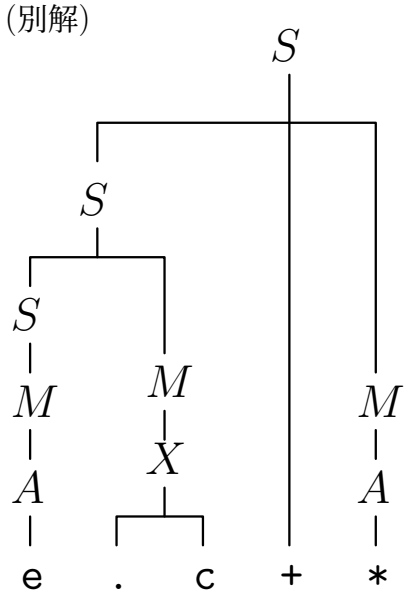
(1) 次の表に挙げた 5 つの終端記号列について、非終端記号 X、M、S から導出できる場合は○を、できない場合は×を、表の空欄にそれぞれ書き込みなさい。(10 点)

終端記号列	X	M	S
`*`	×	○	○
`.c`	○	○	○
`e#i.c`	×	×	○
`e>+*`	×	×	×
`*e+#i`	×	×	○

(2) 終端記号列 `e.c+*` の非終端記号 S に対する構文木を書きなさい。そのような構文木が存在しない場合は「構文木なし」と書きなさい。(4 点)



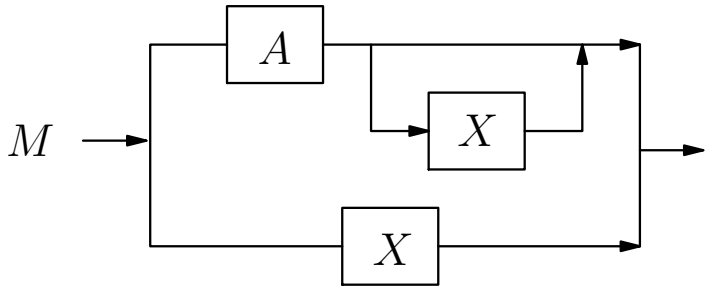
(別解)



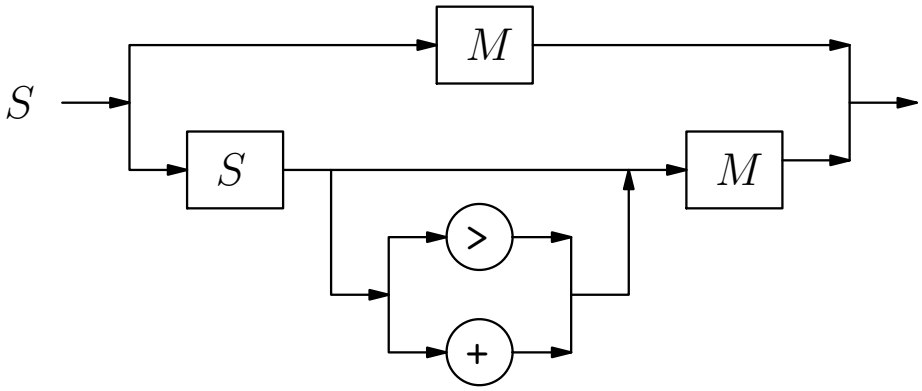
学籍番号　　氏名

(裏面に続く)

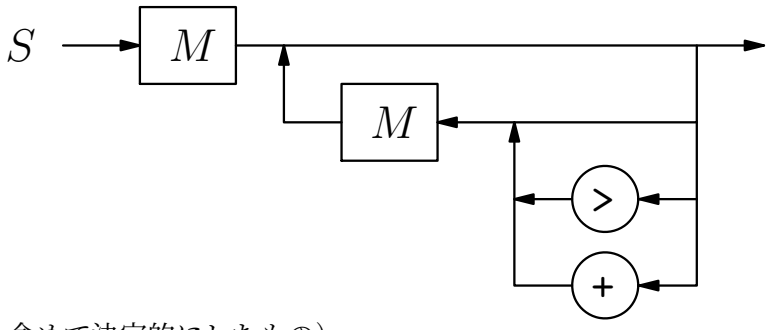
(3) 非終端記号 M の導出規則を構文図で表しなさい。(4 点)



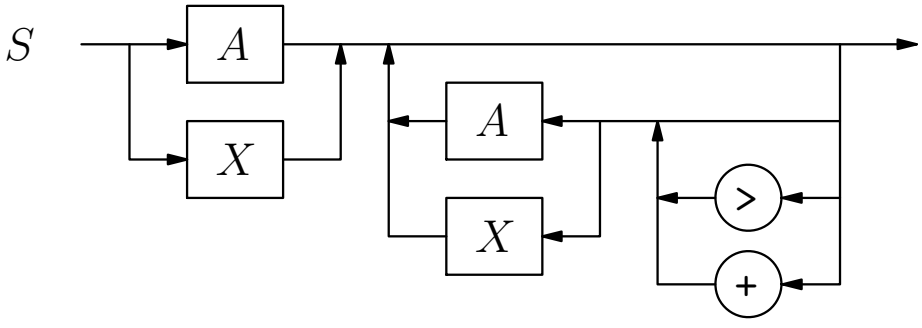
(4) 非終端記号 S の導出規則を構文図で表しなさい。(4 点)



(5) (4) の構文図を決定的なものに書き換えなさい。(4 点)



(別解 — M の内部も含めて決定的にしたもの)



(次ページに続く)

(6) 以下の C プログラムは、標準入力 (キーボード) から、文字列と改行文字を入力すると、入力された文字列が非終端記号 S から導出できるかどうかを判定するプログラムの一部である。関数 X、M、S の定義を補って、このプログラムを完成しなさい。(25 点)

```
#include <stdio.h>
#include <stdlib.h>

int t;

void gettoken(void)
{
    t = getchar();
}

void error(void)
{
    printf("Error!\n");
    exit(1);
}

extern void X(void);
extern void M(void);
```

```
extern void S(void);

void A(void)
{
    if (t != '*' && t != 'e')
        error();
    gettoken();
}

int main(void)
{
    gettoken();
    S();
    if (t != '\n')
        error();
    printf("OK!\n");
    return 0;
}
```

```
void X(void)
{
        switch (t) {
        case '.':
            gettoken();
            if (t != 'c')
                error();
            gettoken();
            break;
        case '#':
            gettoken();
            if (t != 'i')
                error();
            gettoken();
            break;
        default:
                error();
        }









}
```

学籍番号　　氏名

(裏面に続く)

(問題 I (6) の解答欄の続き)

```
void M(void)
{
        switch (t) {
        case '*': case 'e':
            A();
            if (t == '.' || t == '#')
                X();
            break;
        case '.' case '#'
            X();
            break;
        default:
            error();
        }

}

void S(void)
{
        M();
        while (t == '>' || t == '+' || t == '*'
                        || t == 'e' || t == '.' || t == '#') {
            if (t == '>' || t == '+')
                gettoken();
            M();
        }

}
```

(次ページに問題 II)

Ⅱ　次は Minimum C のソースプログラムと、それをコンパイルして得られた MVM の機械語プログラムである。

ソースプログラム

```
int base;

foo(n, x, y)
{
    if ( [ base < n ] )
        x = foo(n-1, y, x+y);
    return x;
}

main()
{
    int sum, k;

    input base;
    k = 1;
    sum = 0;
    while (sum <= 100) {
        int n;

        input n;
        sum = sum + foo(n, k+1, k+n);
        k = [ n ];
    }
    print [ sum ];
}
```

コンパイル結果

```
  0  ADD R13,R14,R0
  4  LDI R11,284
  8  [ CALL R0,112 ]
 12  EXIT R1
 16  PUSH R13
 20  ADD R13,R14,R0
 24  LD R1,R11,0
 28  LD R2,R13,16
 32  SUB R0,R1,R2
 36  JPGE R0,92
 40  LD R1,R13,16
 44  LDI R2,1
 48  SUB R1,R1,R2
 52  PUSH R1
 56  LD R1,R13,8
 60  PUSH R1
 64  LD R1,R13,12
 68  LD R2,R13,8
 72  ADD R1,R1,R2
 76  PUSH R1
 80  CALL R0,16
 84  ADDI R14,R14,12
 88  [ ST R1,R13,12 ]
 92  LD R1,R13,12
 96  POP R13
100  POP R15
104  POP R13
108  POP R15
112  PUSH R13
116  ADD R13,R14,R0
120  SUBI R14,R14,8
124  IN R1
128  ST R1,R11,0
132  LDI R1,1
136  ST R1,R13,-8
140  LDI R1,0
144  ST R1,R13,-4
148  LD R1,R13,-4
152  LDI R2,100
156  SUB R0,R1,R2
160  [ JPGT R0,264 ]
164  [ SUBI R14,R14,4 ]
168  IN R1
172  ST R1,R13,-12
176  [ LD R1,R13,-4 ]
180  PUSH R1
184  LD R1,R13,-12
188  PUSH R1
192  LD R1,R13,-8
196  LDI R2,1
200  ADD R1,R1,R2
204  PUSH R1
208  LD R1,R13,-8
212  [ LD R2,R13,-12 ]
216  [ ADD R1,R1,R2 ]
220  PUSH R1
224  CALL R0,16
228  [ ADDI R14,R14,12 ]
232  ADD R2,R1,R0
236  [ POP R1 ]
240  ADD R1,R1,R2
244  ST R1,R13,-4
248  LD R1,R13,-12
252  ST R1,R13,-8
256  [ ADDI R14,R14,4 ]
260  [ JP R0,148 ]
264  LD R1,R13,-4
268  OUT R1
272  [ ADDI R14,R14,8 ]
276  POP R13
280  POP R15
```

(1) ソースプログラムとコンパイル結果が対応するように、空欄部分を補いなさい。(30 点)

学籍番号　　氏名

(裏面に続く)

(2) この機械語プログラムを起動して、キーボードから整数 7 と 8 を順に入力してみると、92 番地の機械語命令が**最初に**実行された直後の時点で、MVM の スタックポインタ (R14) の値は 1048512 であった。この時点でのスタック中のデータ (整数値) を下図の空欄に書き込みなさい。ただし、下図の 1 つの欄は、それぞれ 32bit (4byte) のデータを表すものとし、プログラムの実行開始時には、スタックポインタ (R14) は 2^{20} ($=$ 1048576) で初期化されるものとする。(15 点)

MVM のメモリ (スタック) の内容

(アドレス)	⋮
1048512	1048532
1048516	84
1048520	11
1048524	9
1048528	7
1048532	1048568
1048536	228
1048540	9
1048544	2
1048548	8
1048552	0
1048556	8
1048560	1
1048564	0
1048568	1048576
1048572	12
1048576	

(3) (2) の時点での R11 と R13 (フレームポインタ) の値を下の欄にそれぞれ書きなさい。(4 点)

R11	284
R13	1048512

(定期試験問題終り)