1 . “Left-most” derivation:

atom : NAME LPAREN args RPAREN

: NAME (args)

: P (args) (Using NAME →P):

: P (args, arg) (Using args → args COMMA arg)

: P (args, arg. arg) (Using args → args COMMA arg)

: P (args, arg, arg, arg) (Using args → args COMMA arg)

: P (args, arg, arg, arg, arg) (Using args → args COMMA arg)

: P (arg , arg, arg, arg, arg) (Using args →arg)

: P (NAME, arg, arg, arg, arg) (Using args → NAME)

: P(x, arg, arg, arg, arg) (Using NAME → x)

: P (x, NUMBER, arg, arg, arg) (Using arg → NUMBER)

: P (x, 20, arg, arg, arg) (Using NUMBER → 20)

: P (x,20, NAME, arg, arg) (Using arg → NAME)

: P (x, 20 , x, arg, arg) (Using NAME → x)

: P (x,20, NAME, arg) (Using arg → NAME)

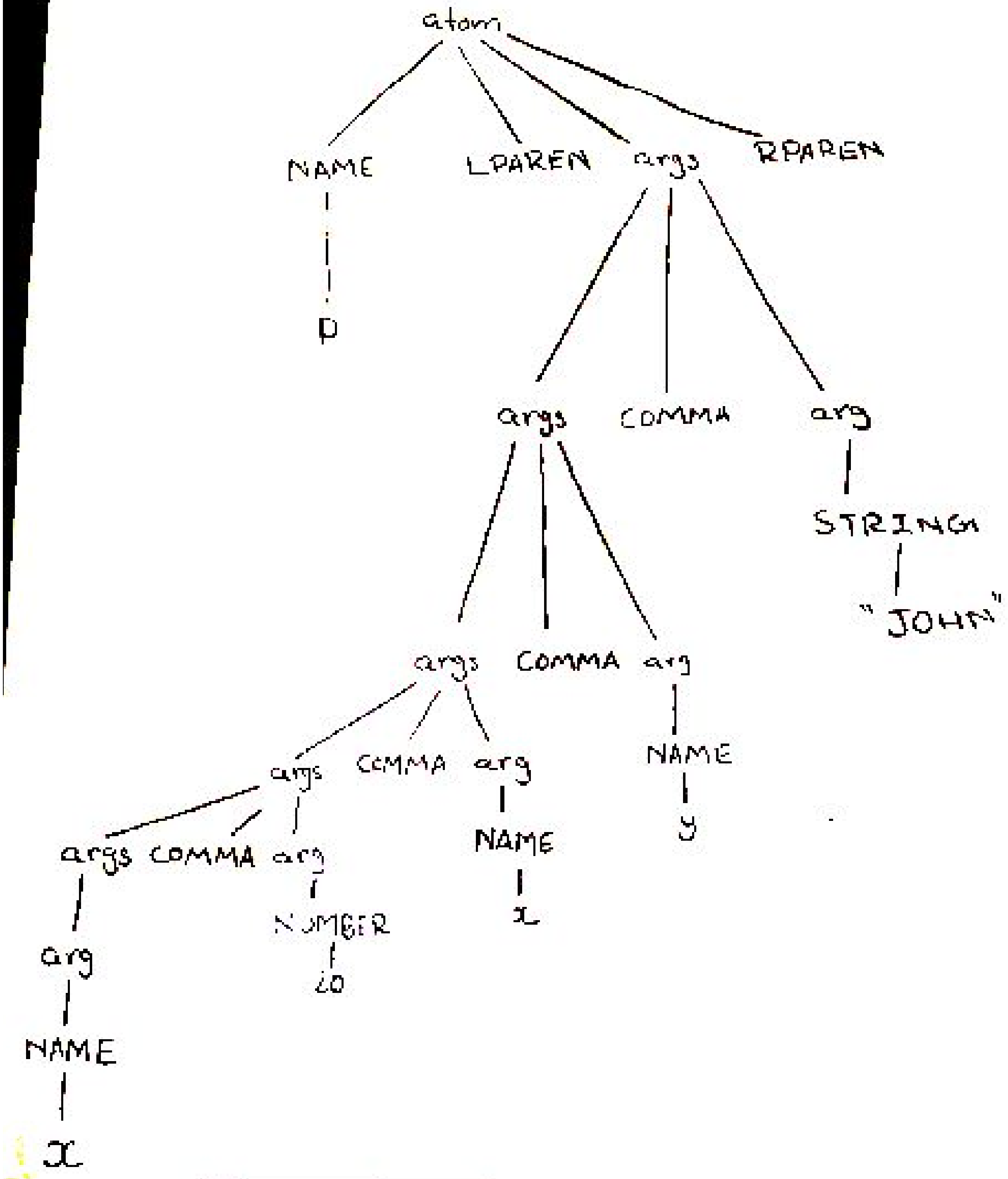
: P (x, 20, x,y, arg) (Using NAME → y)

: P (x,20,x,y,STRING) (Using arg → STRING)

: P (x, 20,x,y,“john”) (Using STRING → “john”)

Parse tree :

1. Parse tree :



3a.

```
alist : LBRACE ID wae RBRACE
P[0] = { 'id': p[2], 'value': p[3] }
alist : LBRACE ID wae RBRACE alist

        If p[2] == p[5] ['id']:

          P[0] = False

        Else P[0] = [{ 'id' : p[2], 'value': p[3] },P[5]]
def p_alist_1(p): 'alist : LBRACE ID wae RBRACE
p[0] = { 'id': p[2], 'num': p[3] }
def p_alist_2(p):'alist : LBRACE ID wae RBRACE alist' if p[5]['id'] == p[2]:
p[0] = False
Else P[0] = [{ 'id' : p[2], 'num' : p[3] }, p[5]]
```

3b.

```
alist : LBRACE ID wae RBRACE
P[0] = { 'id': p[2], 'value': p[3] }
alist : LBRACE ID wae RBRACE alist
def p_alist_1(p): 'alist : LBRACE ID wae RBRACE
p[0] = { 'id': p[2], 'num': p[3] }
def p_alist_2(p):'alist : LBRACE ID wae RBRACE alist' if p[4]['id'] == p[2]:
p[0] = False
Else P[0] = [{ 'id' : p[2], 'num' : p[3] }, p[4]]
```

4.

M_C(if-then-else,s) = a denotational function that defines the meaning of if-then-else, that maps either from bool (maps to true, false or error) to stmt - list (stmt) or stmt list (stmt SEMI stmt-list)

M_C(stmt-list, s) = a denotational function that defines the meaning of stmt-list, stmt C maps to new states resulting from the execution of the stmt-SEMI stmt-list.