```
1. (1) :: a | b | c | ... | ?
 (dig) ::= 0 | 1 | 2 | ... | 9
  (expr)::= () 1 (dig) 1 (U)
          1 let (1d) = (expr) in (expr)
          1 (expry; (expr)
          1 besin lexpit end
use example: let x = 5 in begin x; x end
         two different derivations
1 (expi)
       * let (1d) = (cxpc) in (expi)
        let x = <expi7 in (expr)
       let x = < digy in (exp)
        let x = 5 in Lexpit
        let x = 5 in begin lexpert and
        Itt x = & in begin x's seypir end
        lel k= 5 in begin x', x end
   (expir
       let (1d7 = cexp+7 1, cexp+7
       let x = cexpr7 'n cexpi7
        let & - (dig > 11 Stype)
       Izt x = 5 in Leypi7
 There are two different ways to derive the same strings,
 displaying ambiguity.
2. <id>>:= a|b)c|...|z
    (dy) ::= 011121...19
    (expr> := < tom7 | (1+7
   Germ 7: = 01 (dig7 | de7 | Ksemi7 125egin7
   Uet7: = let (1d) = cexpr7 in (term?
    LSEMIT :: = Kterny; Kleiny
   Loeginy ::= besin Kerpiy
```

```
3, (expi7

(let)

let(ld) = (exp) in (leim)

1

9

6

(tem)

(tem)

(tem)

(ld)

(ld)

1

4

6

6
```