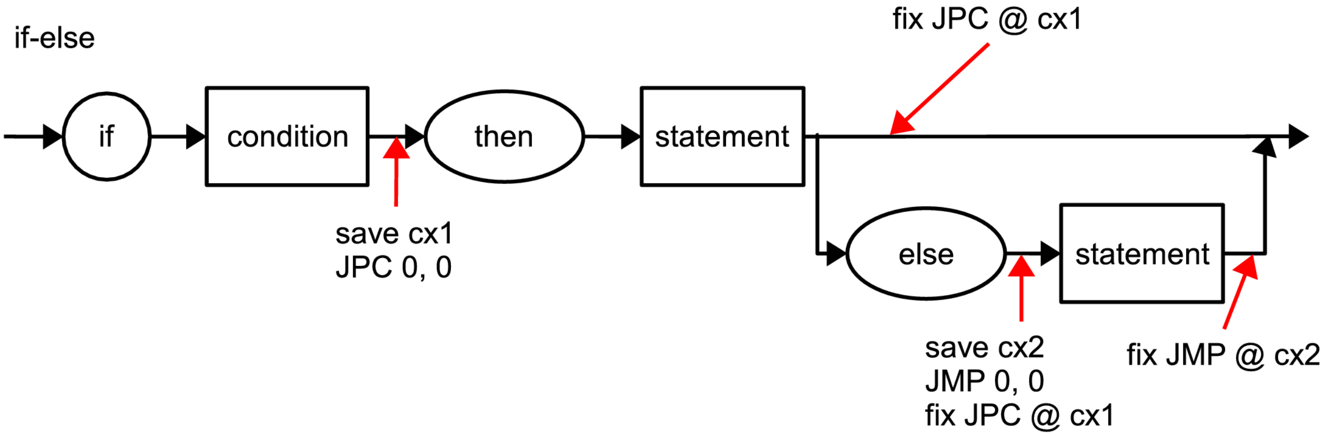
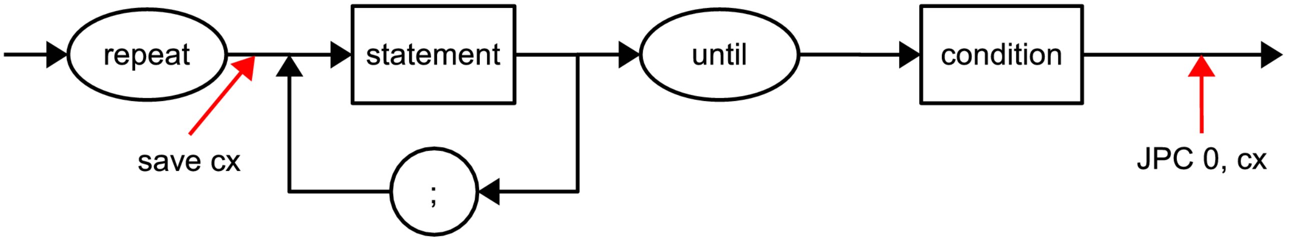
Syntax Diagrams (adding p-Code for the new constructs)

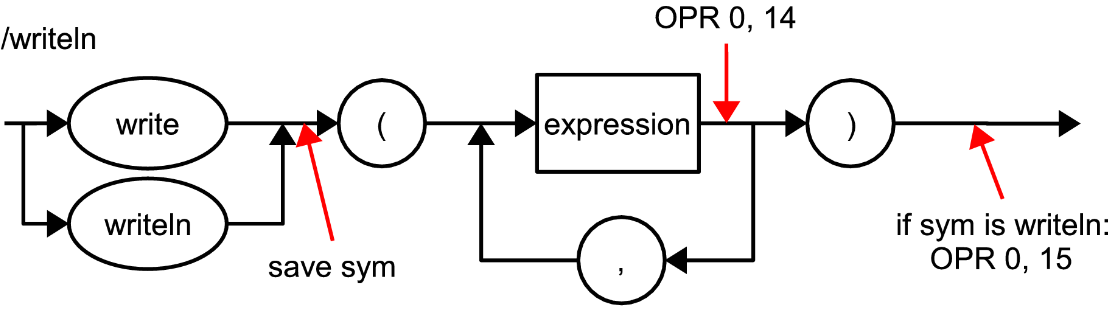
if-else



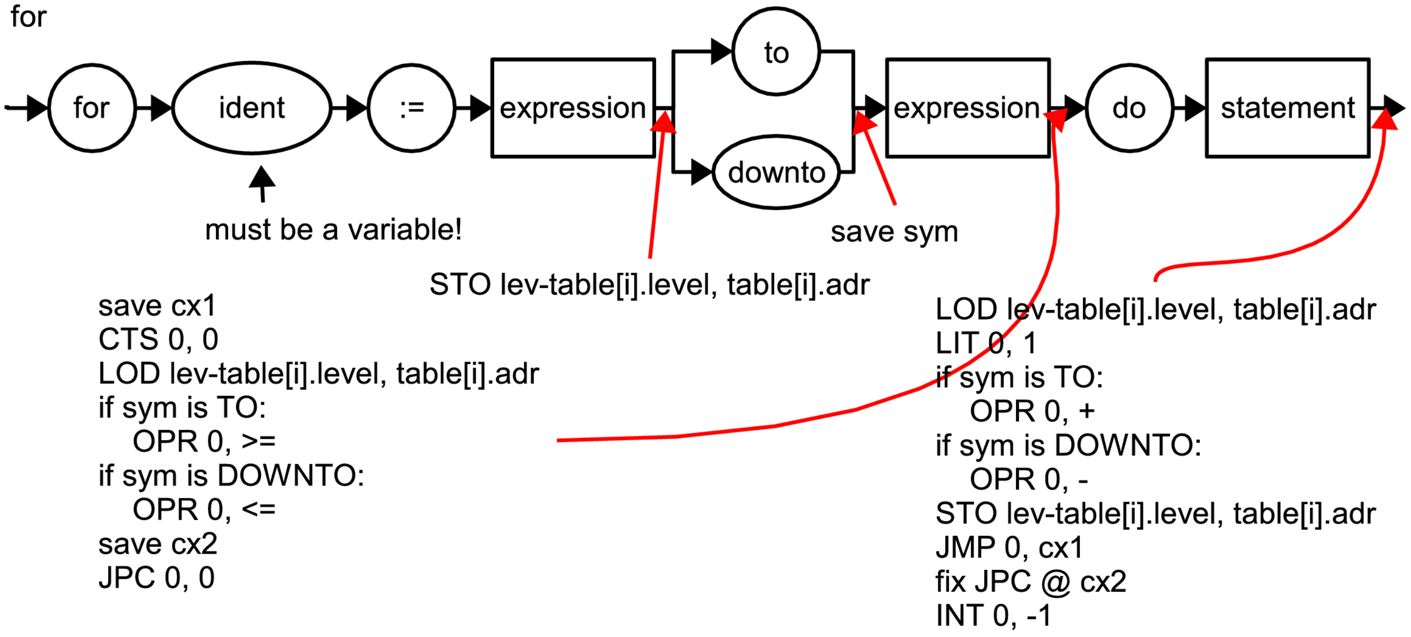
repeat-until



write/writeln



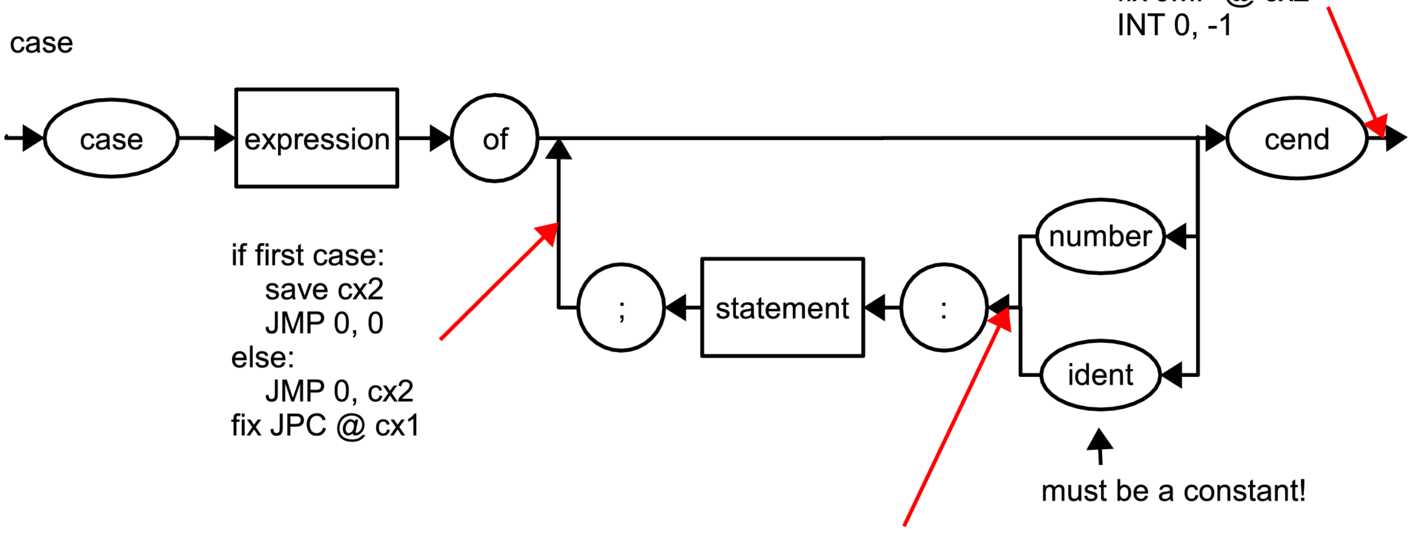
for



```
save cx1
CTS 0, 0
LOD lev-table[i].level, table[i].adr
if sym is TO:
   OPR 0, >=
if sym is DOWNTO:
   OPR 0, <=
save cx2
JPC 0, 0
```

```
LOD lev-table[i].level, table[i].adr
LIT 0, 1
if sym is TO:
   OPR 0, +
if sym is DOWNTO:
   OPR 0, -
STO lev-table[i].level, table[i].adr
JMP 0, cx1
fix JPC @ cx2
INT 0, -1
```

case



fix JMP @ cx2
INT 0, -1

if first case:
    save cx2
    JMP 0, 0
else:
    JMP 0, cx2
fix JPC @ cx1

CTS 0, 0
if number:
    LIT 0, num
if constant:
    LIT 0, table[i].value
OPR 0, =
save cx1
JPC 0, 0

we've added a new instruction to our p-Machine: copy top stack (CTS)
- CTS
    - t++;
    - s[t] = s[t-1];

note the alternate way of combining write and writeln
- we just need to save the sym
- if the sym is writeln, then generate the final OPR 0, 15 after parsing RPAREN

CTS 0, 0 is used in for-to/downto-do as follows:
- we first perform the initial assignment
- we then evaluate the expression on the right (which may be complicated and computationally intensive)
- we copy this result so we can reuse it without having to recompute it
- we load the expression on the left and perform a reverse comparison
    - normally, we execute statement of the FOR if left <= right (in the case of TO)
    - in this case, we do this if right >= left (in the case of TO)
    - the reverse is true in the case of DOWNTO
- the conditional jump destroys our copied value
- but that's fine because the original is still there
    - and we'll just copy it again when we re-evaluate the condition of the FOR
    - but when we're done, we have to remove this copy
        - hence the INT 0, -1 (decrease top stack by 1)
- don't forget the increment (TO) or decrement (DOWNTO) after the FOR statement(s)

CTS 0, 0 is used in case-of-cend as follows:
- we evaluate the initial expression and then copy it
- this way we don't have to recompute it for every case we compare to
- at the end, we'll still have the original calculation as in FOR, so we similarly remove it by INT 0, -1
- a simple boolean can be used to keep track of whether or not we're at the first case