Section A.

1. (a). it's lexer's job to determine whether a token is BOOL-VAL or ID.

My parser would regard T/F as a keyword, report error if encounter such situation.

(b)

i) IfVariants = "if" | "si" | "ob" | "gdyby"

<YYINITIAL> {

{IfVariants} { return symbol(sym.IF); }

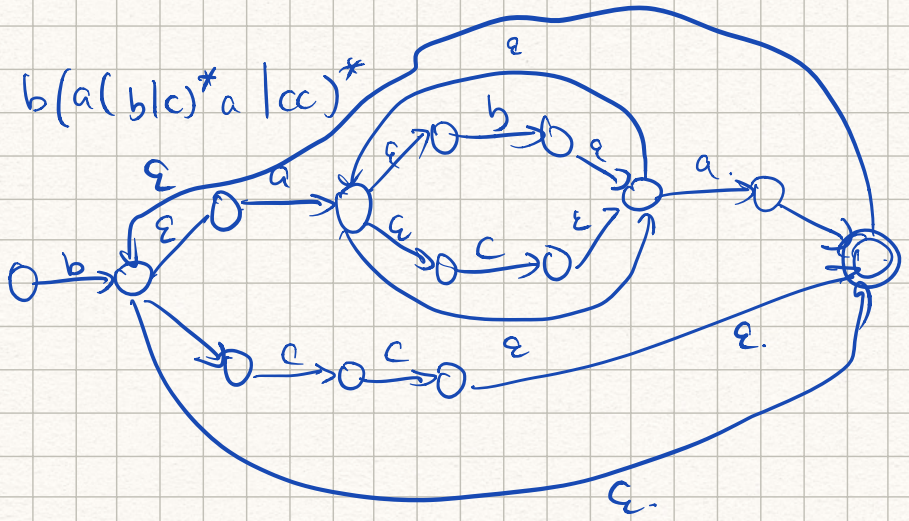
}.

ii) Introduces more keywords

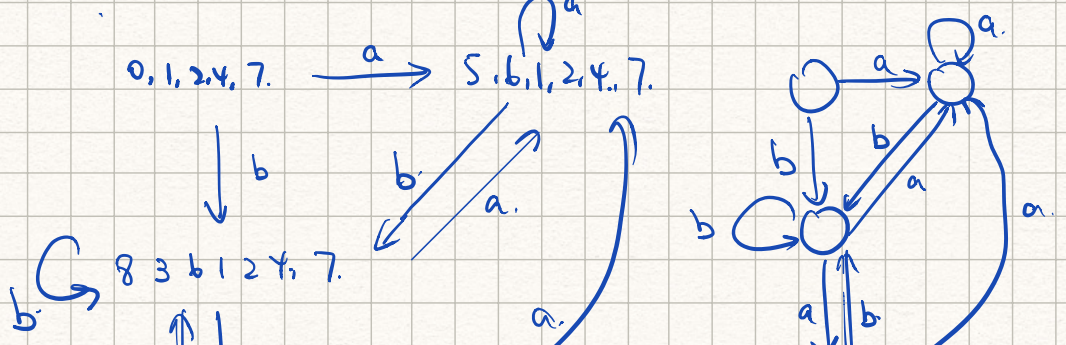
limits naming possibilities

makes parsing meaningfully more complicated

(c)



(d).



b | ↓ a.

95 67 124.

2.

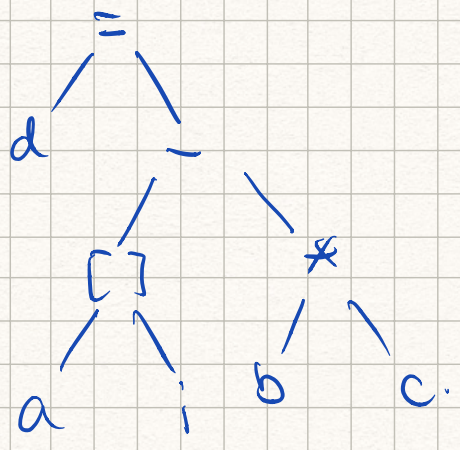
(a) * short commands / compact code.

* simpler compiler. (independent code gen)

↓ * more memory references

* harder to factor out subexpression for optimisation

(b) AST:



t1 = a[i]

t2 = b MUL c.

t3 = t1 - t2.

d = t3.

[0] ARR. a. i.

[1] MUL b c

[2] SUB [0] [1]

[3]. ASSIGN [2] d.

(c) ↑. easy to optimise (no need to modify code)

modify the references

↑. saves space (redundant temps).

(d) control link

access link.

function params

local variables / tmp variables

return vals.s

return address / machine status

(e)

```
Loop:
t1 = i LE 10
fjump t1 END
T(if --- itf).
t2 = aGEb.
fjump ELSE
   T(a+ --- i)
jump ENDIF
ELSE:
   T(a --- b).
ENDIF:
t3 = i ADD 1
i = t3.
t4 = 2 MUL i (i << 1)   strength reduction.
t5 = b ADD t4.
t6 = a ADD t5.
a = t6.
jump LOOP
END
```

inversion