Mitchell Rathbun

1.

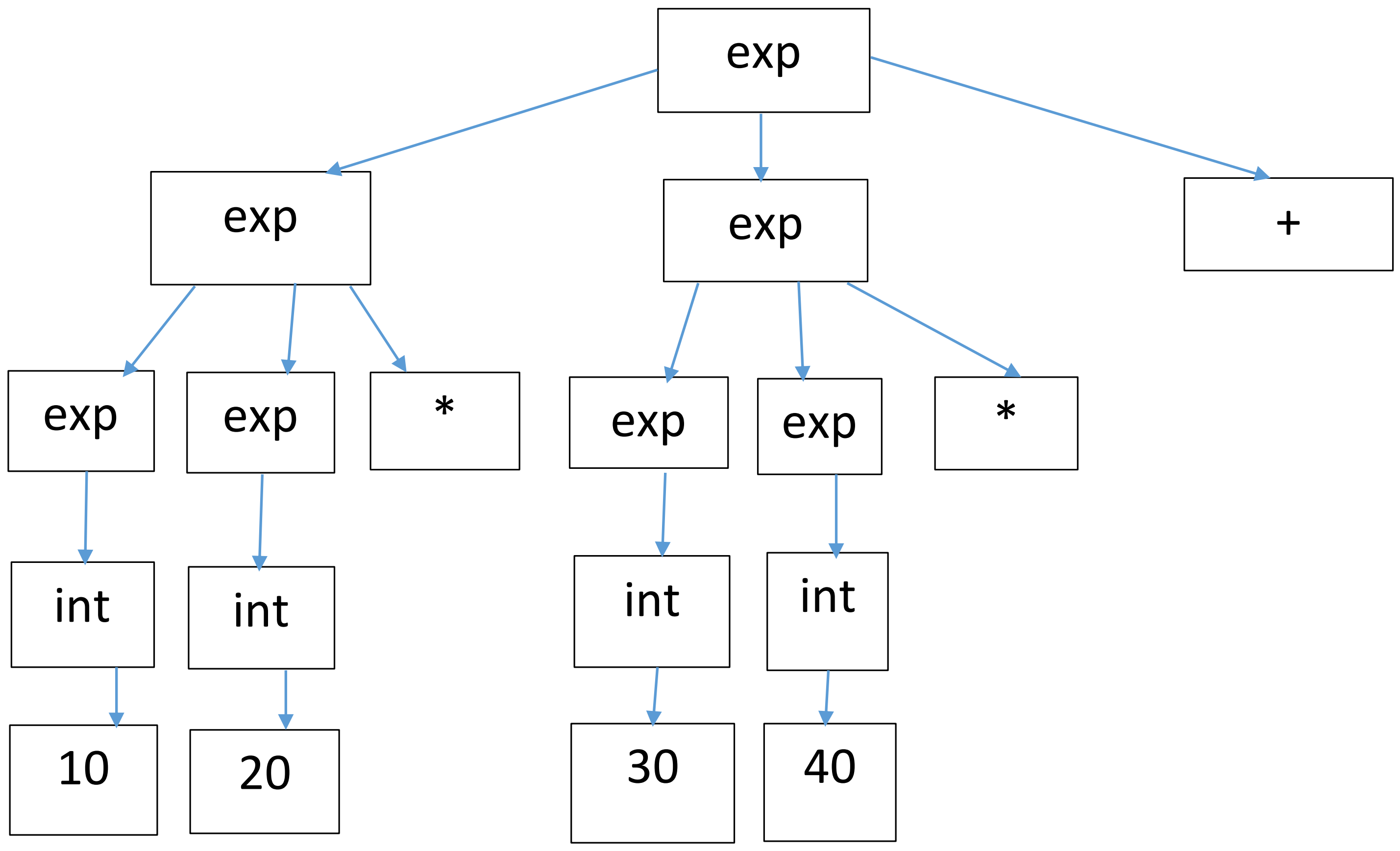
i. No

ii. Yes

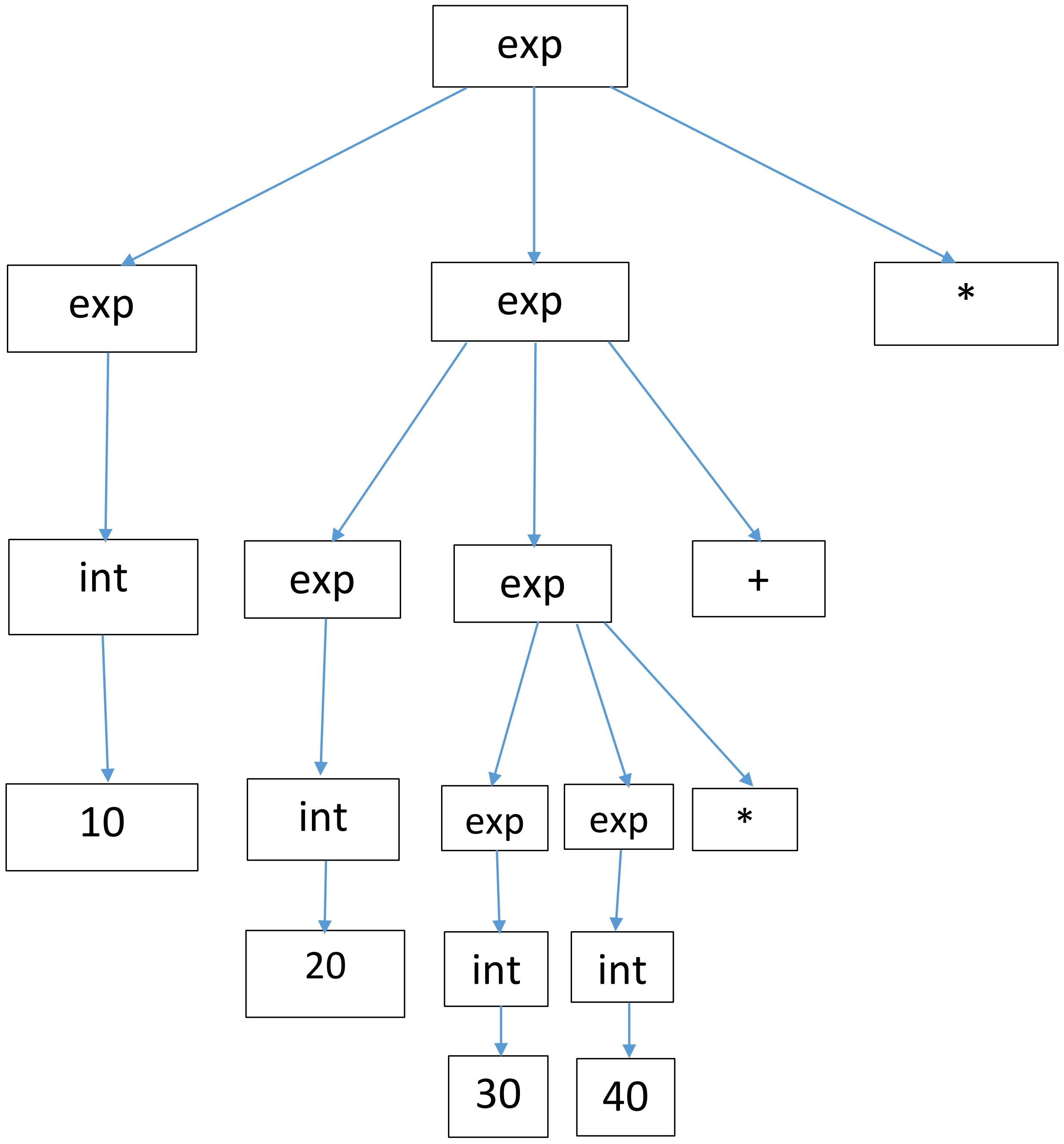
iii. Yes

iv. No

Parse tree for ii.



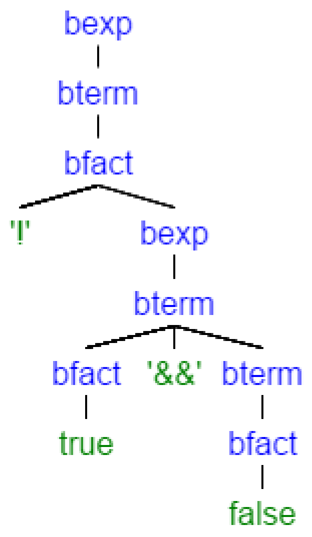
Parse tree for iii.



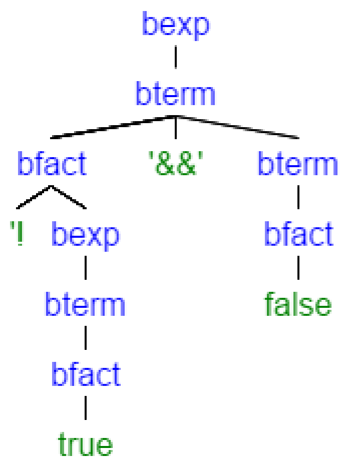
2. S -> switch '(' aexp ')' {num {, num} : stmt}$^+$ [else stmt] end

3.

a. BEXP2 is ambiguous since '!', '||' and '&&' all have the same level of precedence. To prove ambiguity, the expression "! true '&&' false" can be looked at. There are two distinctly different parse trees for this expression. The first parse tree would show that expression to be ! bexp when bfact is first reached. The second tree would instead have it as ! bexp && bfact.



1st valid parse tree



2nd valid parse tree

b.

bexp -> bterm | bterm '||' bexp

bterm -> bfact | bfact '&&' bterm

bfact -> true | false | id | '(' bexp ')' | '!' bfact

c.

bexp -> bterm | bterm '||' bexp

bterm -> bfact | bfact '&&' bterm

bfact -> true | false | id | '(' bexp ')' | aexp relop aexp

relop -> == | >= | <= | > | <

4.

a. A context-free grammar can't adequately describe an XML-like element due to the requirement that the beginning and ending tags must match. It is impossible to enforce type consistency with a context-free grammar, which is why attribute grammars are so important.

b.

element -> primitive | begin_tag$_{t1}$ element {element} end_tag$_{t2}$

Semantics rule: t1==t2

primitive -> id | num

begin_tag$_{t}$ -> '<' id$_{name}$ '>'

Semantics rule: t=name

end_tag$_{t}$ -> '</' id$_{name}$ '>'

Semantics rule: t=name