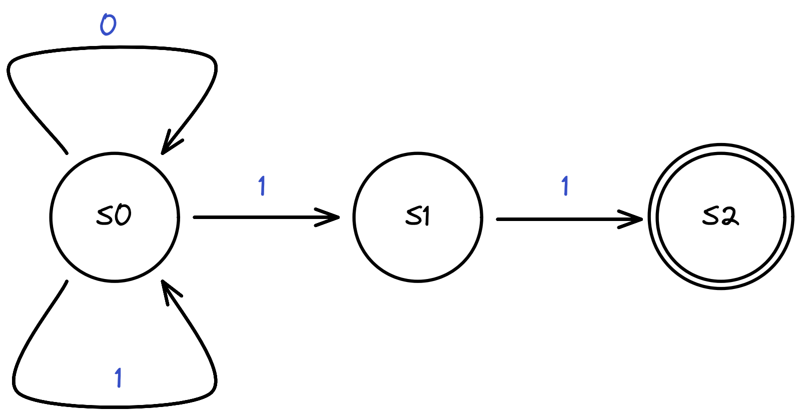
Homework 2

CS314

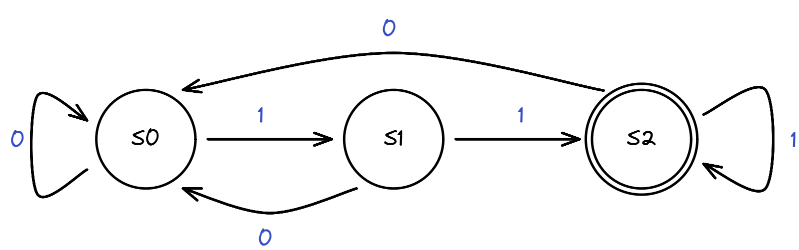
1 Problem — Finite State Automaton (FSA) - 20 pts

1. Specify the state transition graph of (1) a NFA (which is not DFA as well) without ϵ transitions and (2) a DFA that recognizes the following language: "All strings of 0's and 1's that end with 11

NFA



DFA



2. In addition to the state transition graphs (diagram), give the state transition table and the formal specification of an automaton as the quadruple $< S, s, F, T >$ for both, your NFA and DFA. Do not include "error" states.

S	s	F
S0	0	S0
S0	1	S1
S0	1	S1
S1	1	S2

NFA
State transition table

	0	1
S0	S0	S1, S2
S1	-	S2
S2	-	-

Quadruple <S, s, F T> where...

- S is the set of states **{S0, S1, S2}**
- s is the start state **S0**
- F is the final state **S2**
- T is the set of labeled transitions
  - (S0, 0) $\rightarrow$ S0
  - (S0, 1) $\rightarrow$ S0
  - (S0, 1) $\rightarrow$ S1
  - (S1, 1) $\rightarrow$ S2

DFA
State transition table

	0	1
S0	S0	S1

	0	1
S1	S0	S2
S2	S0	S2

Quadruple <S, s, F T> where...

- S is the set of states **{S0, S1, S2}**
- s is the start state **S0**
- F is the final state **S2**
- T is the set of labeled transitions
  - (S0, 0) $\rightarrow$ S0
  - (S0, 1) $\rightarrow$ S1
  - (S1, 0) $\rightarrow$ S0
  - (S1, 1) $\rightarrow$ S2
  - (S2, 0) $\rightarrow$ S0
  - (S2, 1) $\rightarrow$ S2

2 Problem — Regular and Context-Free Languages - 20 pts

Are the following languages context-free or not? If yes, specify a context-free grammar in BNF notation that generates the language. If not, give an informal argument. Furthermore, which of the languages are also regular languages, i.e., can be expressed by a regular expression? Prove it by giving the regular expression that specifies the language.

1. $\{a^n b^m c^o \mid m > 0, n \geq 0, o > 0\}$, with alphabet $\Sigma = \{a, b, c\}$

This **is a context-free language**.
<start> ::= <A>b<B>c<C>
<A> ::= a<A> | ϵ
<B> ::= b<B> | ϵ
<C> ::= c<C> | ϵ

It can be expressed with the regular expression, $a^* b^+ c^+$.

2. $\{a^n b^n c^n \mid n > 0\}$, with alphabet $\Sigma = \{a, b, c\}$

This **is not a context-free language**. a^nb^n and b^nc^n can be expressed using context-free languages. However, when considering $a^nb^nc^n$, context is necessary to determine how many occurences of c are required.

Furthermore, it can not be expressed with a regular expression.

3. $\{0^{2n}1^{4n} \mid n > 0\}$, with alphabet $\Sigma = \{0, 1\}$

This **is a context-free language**, but can not be expressed by regular expression.
<start> ::= 00<start>1111 | ϵ

4. $\{wcw^R \mid w \in \Sigma^* \text{ and } w^R \text{ is w in reverse }\}$, with alphabet $\Sigma = \{a, b, c\}$

This **is a context-free language**, but can not be expressed by regular expression.
<start> ::= a<start>a | b<start>b | c

5. $\{a^nb^mc^md^n \mid n \geq 0, m \geq 0\}$, with alphabet $\Sigma = \{a, b, c, d\}$

This **is a context-free language**, but can not be expressed as a regular expression.
<start> ::= <AD>
<AD> ::= a<AD>d | b<BC>c | ϵ
<BC> ::= b <BC> c | ϵ

6. $\{a^nb^mc^nd^m \mid n \geq 0, m \geq 0\}$, with alphabet $\Sigma = \{a, b, c, d\}$

This **is not a context-free language**. Individually, the pairs a^nb^m and c^nd^m can be defined using context-free languages but together, the number of repetitions n and m can not be generated without context from the other pair. Context is required to determine how many occurences of a, which is necessary to determine how many occurences of c are required. The same goes for b and d.

Furthemore, it can not be expressed with a regular expression.

7. $\{a^na^nb^nb^n \mid n \geq 0\}$, with alphabet $\Sigma = \{a, b\}$
   Note that $a^na^nb^nb^n = a^{2n}b^{2n}$
   This **is a context-free language**, but can not be expressed as a regular expression.

<start> ::= aa<start>bb | ϵ

8. $\{w \mid w \text{ has more than 3 symbols}\}$, with alphabet $\Sigma = \{a, b\}$

This **is a context-free language**.

<start> ::= <AB><AB><AB><AB><repeat>

<AB> ::= A | B

<repeat> ::= <AB><repeat> | ϵ

It can expressed with the regular expression, $(a|b)(a|b)(a|b)(a|b)^{+}$

3 Problem — Derivation, Parse Tree, Ambiguity, Precedence & Associativity - 60 pts

A language that is a subset of the language of propositional logic may defined as follows:

$$\begin{aligned}
<\text{start}> ::= &\quad <\text{expr}> \\
<\text{expr}> ::= &\quad <\text{expr}> \vee <\text{expr}> \quad | \\
&\quad <\text{expr}> \wedge <\text{expr}> \quad | \\
&\quad <\text{expr}> \leftrightarrow <\text{expr}> \quad | \\
&\quad <\text{const}> | <\text{var}> \\
<\text{const}> ::= &\ \text{true} \mid \text{false} \\
<\text{var}> ::= &\ \text{a} \mid \text{b} \mid \text{c}
\end{aligned}$$

1. Give a leftmost and a rightmost derivation for the sentence

$$\text{a} \wedge \text{true} \wedge \text{b} \leftrightarrow \text{false} \vee \text{true}$$

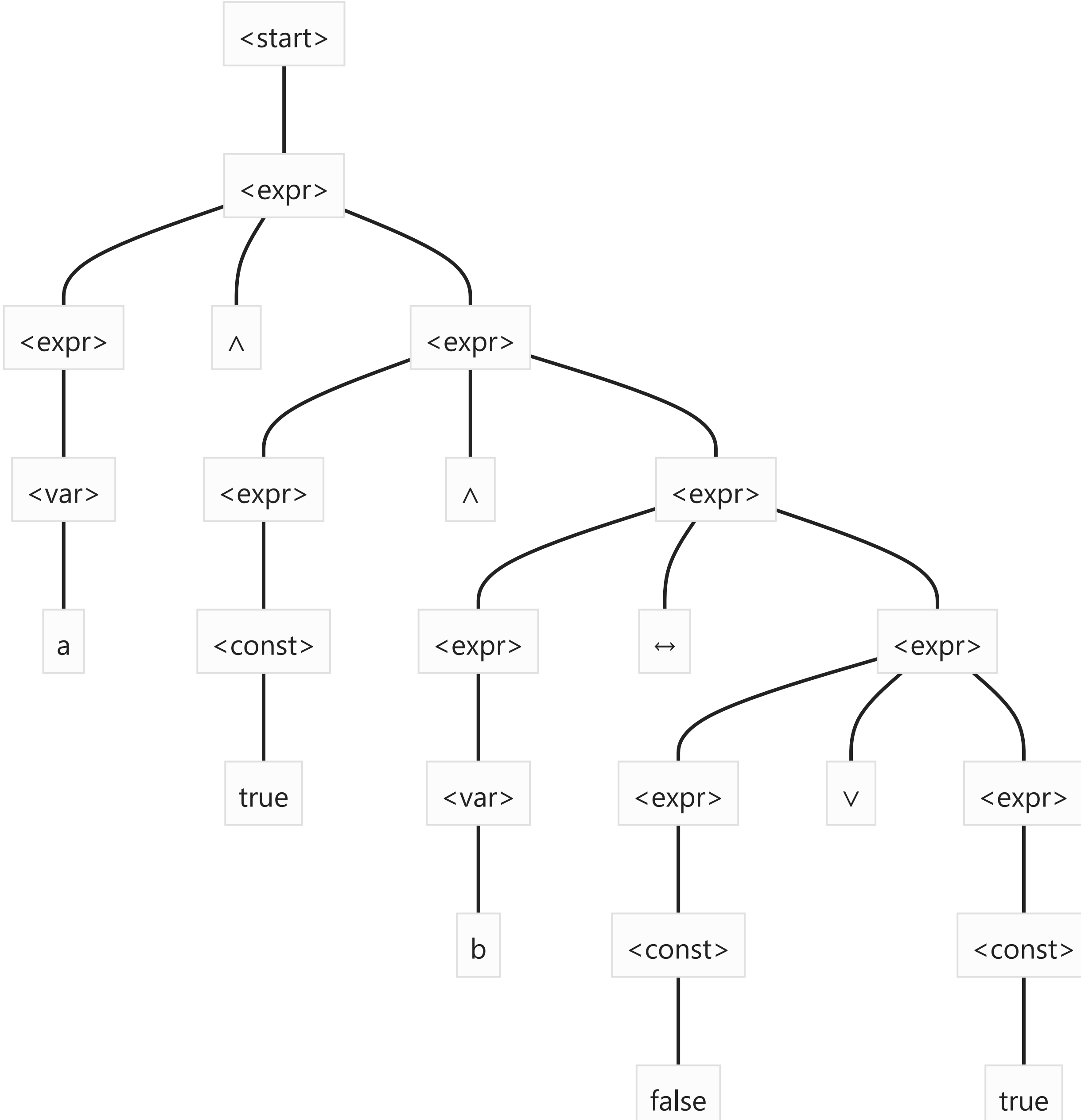
Leftmost	Rule applied
<expr>	1
<expr> ∧ <expr>	2b
<var> ∧ <expr>	2e
a ∧ <expr>	4a
a ∧ <expr> ∧ <expr>	2b
a ∧ <const> ∧ <expr>	2d
a ∧ true ∧ <expr>	3a
a ∧ true ∧ <expr> ↔ <expr>	2c
a ∧ true ∧ <var> ↔ <expr>	2e
a ∧ true ∧ b ↔ <expr>	4b
a ∧ true ∧ b ↔ <expr> ∨ <expr>	2a

Leftmost	Rule applied
a ∧ true ∧ b ↔ <const> ∨ <expr>	2d
a ∧ true ∧ b ↔ false ∨ <expr>	3b
a ∧ true ∧ b ↔ false ∨ <const>	2d
a ∧ true ∧ b ↔ false ∨ true	3a

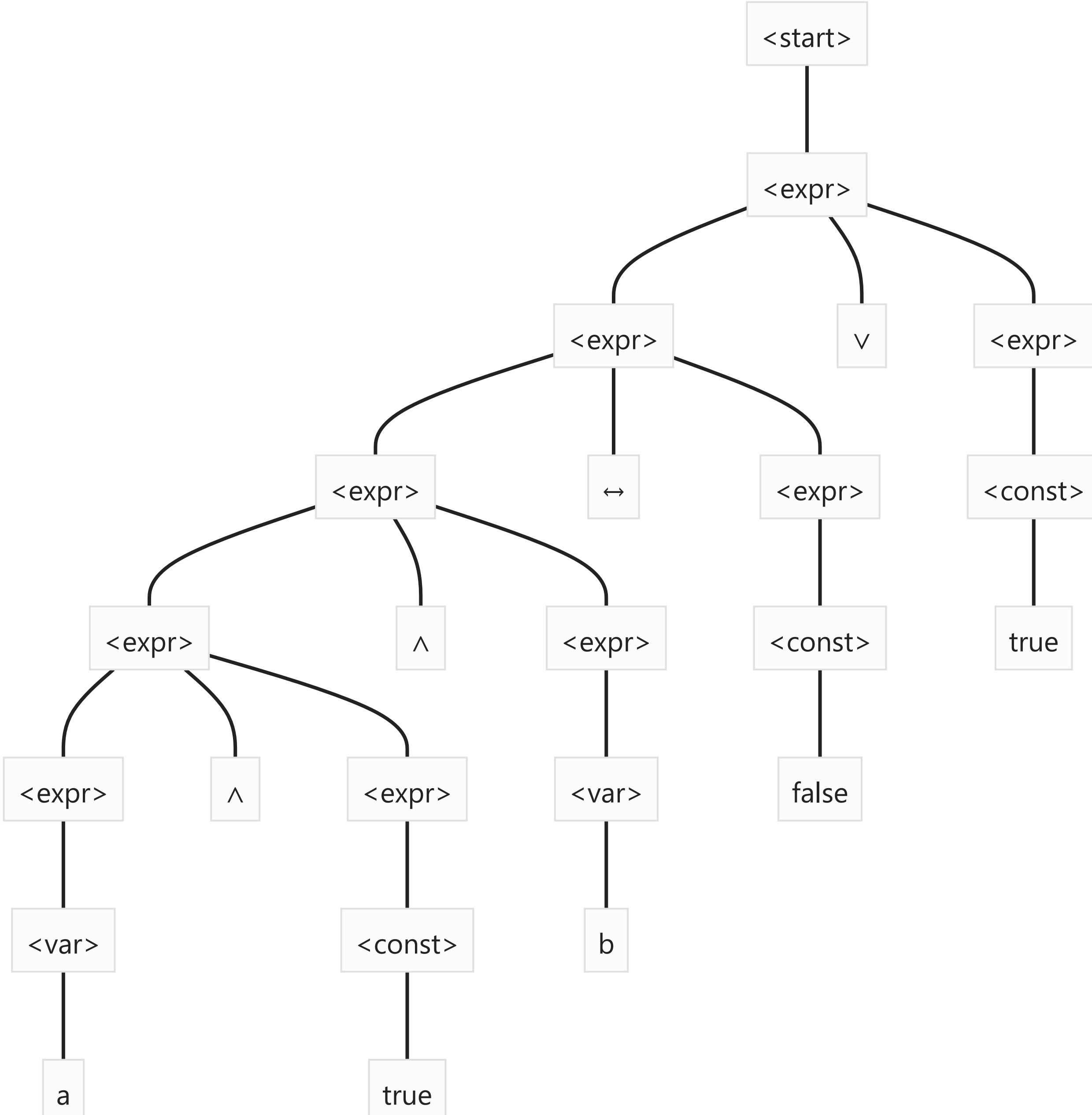
Rightmost	Rule applied
<expr>	1
<expr> ∨ <expr>	2a
<expr> ∨ <const>	2d
<expr> ∨ true	3a
<expr> ↔ <expr> ∨ true	2c
<expr> ↔ <const> ∨ true	2d
<expr> ↔ false ∨ true	3b
<expr> ∧ <expr> ↔ false ∨ true	2b
<expr> ∧ <var> ↔ false ∨ true	2e
<expr> ∧ b ↔ false ∨ true	4b
<expr> ∧ <expr> ∧ b ↔ false ∨ true	2b
<expr> ∧ <const> ∧ b ↔ false ∨ true	2d
<expr> ∧ true ∧ b ↔ false ∨ true	3a
<var> ∧ true ∧ b ↔ false ∨ true	2e
a ∧ true ∧ b ↔ false ∨ true	4a

2. Give the corresponding parse trees for the derivations.

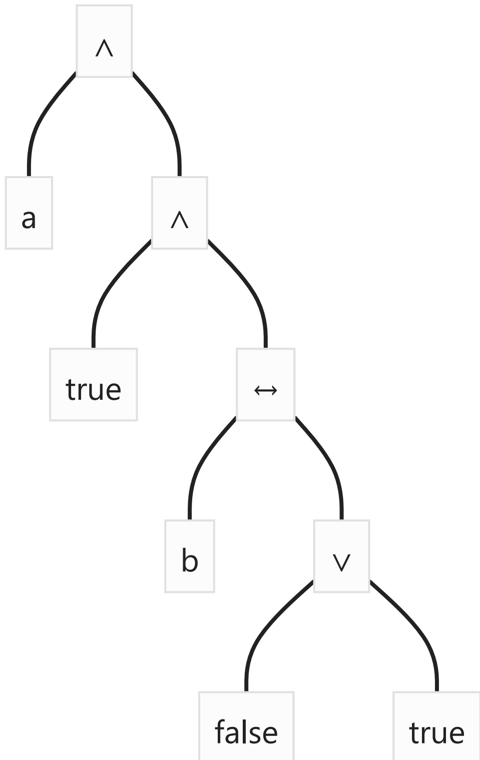
Leftmost


<start>
<expr>
<expr>
∧
<expr>
<var>
<expr>
∧
<expr>
a
<const>
<expr>
↔
<expr>
true
<var>
<expr>
∨
<expr>
b
<const>
<const>
false
true

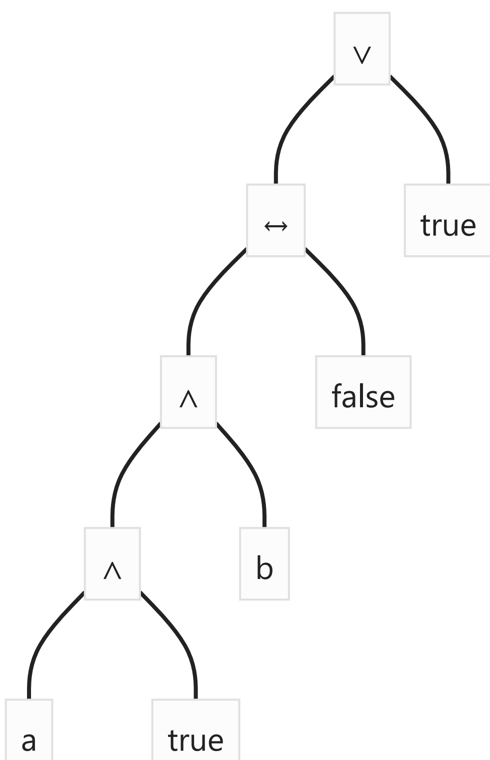
Rightmost



3. Give the corresponding abstract syntax tree (AST)
   Leftmost



Rightmost



4. Show that the above grammar is ambiguous.
   Take the sentence, $\mathrm{a} \wedge \mathrm{true} \wedge \mathrm{b}$, for example.

With the above grammar, we may generate two distinct derivations for this sentence (either two leftmost, or two rightmost), which shows that the grammar is ambiguous.

Derivation 1.

Leftmost	**Rule applied**
<expr>	1
<expr> ∧ <expr>	2b
<var> ∧ <expr>	2e
a ∧ <expr>	4a
a ∧ <expr> ∧ <expr>	2b
a ∧ <const> ∧ <expr>	2d
a ∧ true ∧ <expr>	3a
a ∧ true ∧ <var>	2e
a ∧ true ∧ b	4b

Derivation 2.

Leftmost	**Rule applied**
<expr>	1
<expr> ∧ <expr>	2b
<expr> ∧ <expr> ∧ <expr>	2c
<var> ∧ <expr> ∧ <expr>	2e
a ∧ <expr> ∧ <expr>	5a
a ∧ <const> ∧ <expr>	2d
a ∧ true ∧ <expr>	3a
a ∧ true ∧ <var>	2e
a ∧ true ∧ b	4b

5. Give an unambiguous grammar for the same language that enforces the following precendence and associativity:

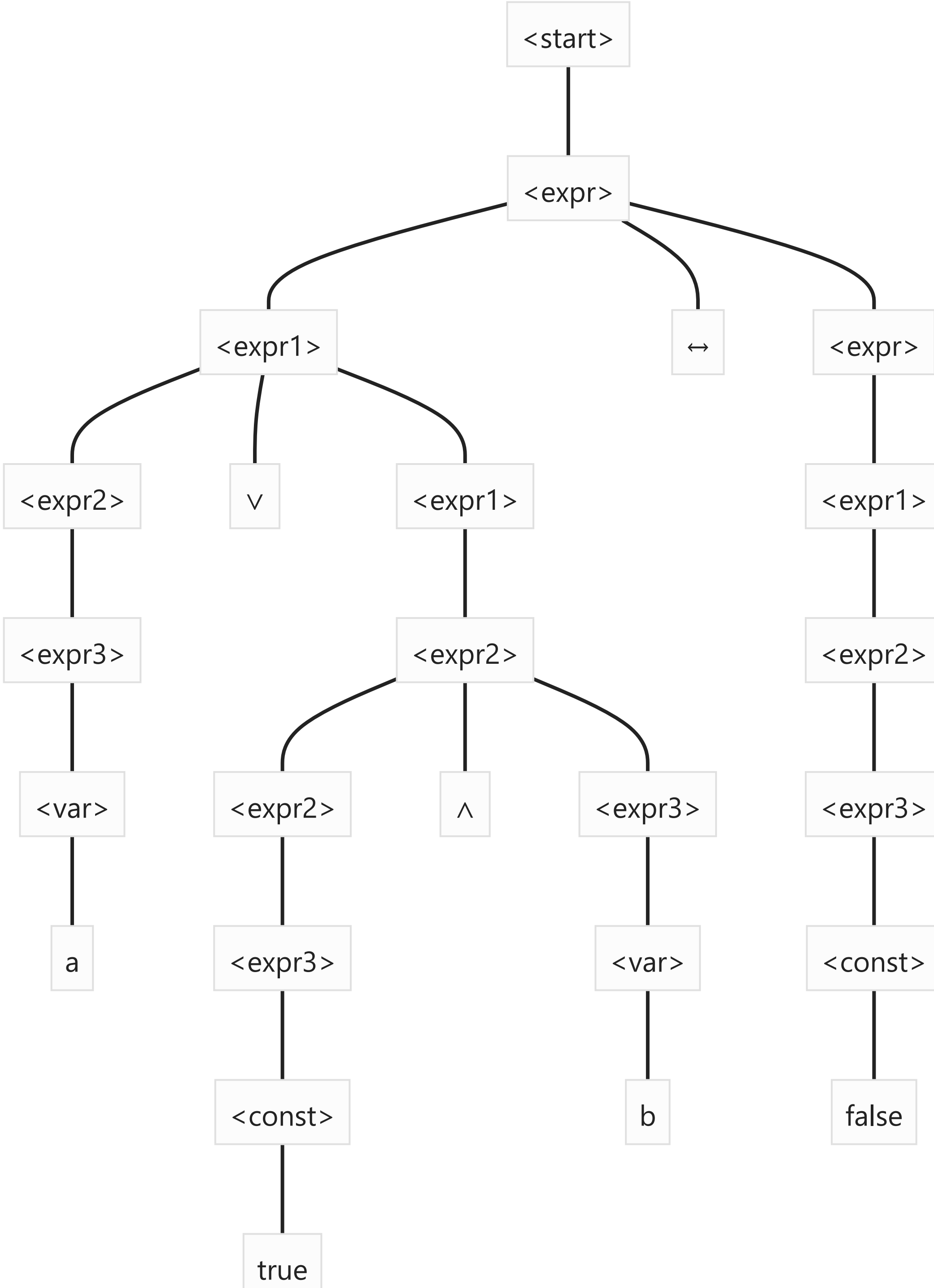
- $\wedge$ has the highest precedence (binds strongest), followed by a $\vee$, and then $\leftrightarrow$
- $\vee$ is left associative, and $\leftrightarrow$ and $\vee$ are right associative

$$
\begin{aligned}
<\text{start}> &::= \text{<expr>} \\
<\text{expr}> &::= \text{<expr1>} \ \leftrightarrow \ \text{<expr>} \mid \text{<expr1>} \\
<\text{expr1}> &::= \text{<expr2>} \ \vee \ \text{<expr1>} \mid \text{<expr2>} \\
<\text{expr2}> &::= \text{<expr2>} \ \wedge \ \text{<expr3>} \mid \text{<expr3>} \\
<\text{expr3}> &::= \text{<const>} \mid \text{<var>} \\
<\text{const}> &::= \text{true} \mid \text{false} \\
<\text{var}> &::= \text{a} \mid \text{b} \mid \text{c}
\end{aligned}
$$

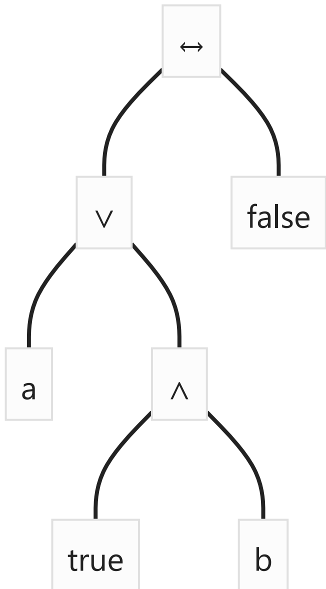
6. Give the parse tree and AST for your new, unambiguous grammar for the sentence

$$\text{a} \vee \text{true} \wedge \text{b} \leftrightarrow \text{false}$$

Parse tree


<start>
<expr>
<expr1>
↔
<expr>
<expr2>
∨
<expr1>
<expr1>
<expr3>
<expr2>
<expr2>
<var>
<expr2>
∧
<expr3>
<expr3>
a
<expr3>
<var>
<const>
<const>
b
false
true

AST


↔
∨
false
a
∧
true
b