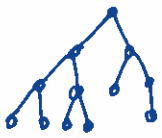


Truth Trees

Recap - Puzzles
(Knight - Knave)

Like truth table, truth trees are used to determine all truth assignments ~~that~~ (of the propositional variables) that make a compound proposition true.

(Unlike truth tables, truth trees, have sizes that are polynomial in size of the proposition.)

- Atomic proposition :- no logical connectives
- Literals :- atomic propositions and their negations.

Complex proposition :- contains logical connectives other than negations.

(P is a complex proposition on $p_1, p_2, \ldots, p_n$ variables and have $\{\neg, \wedge, \vee, \rightarrow, \leftrightarrow\}$

A Truth Tree for P has

- Complex proposition P at the root (on top*)
- Only one literal at each ~~leave~~ leaf.
- There is a unique path from each leaf to the root (this is a property of any tree).
- each node has at most 2 children.

* unlike nature in CS trees are upside down :)

- The path from any leaf to the root can be
  - either INACTIVE (contains p_i & $\neg p_i$ i.e. a contradiction.)
  - or ACTIVE (No contradiction)

- Any active path is complete if it contains no unchecked complex propositions.

- A tree is complete if all active paths are complete.

Algorithm to construct a truth tree for Complex proposition P

1. Write P at the root

2. a) Apply branching rule to the first unchecked propositions in the tree that lies on some active path.

   b) Extend every active path of the tree that continues below this proposition.

   c) Check off that proposition — its status is now 'checked'.

3. Check which path contains contradiction and put a cross X at the end.

4. Repeat step from step 2 if any active path contains unchecked complex propositions.

Branching Rules

$\neg\neg p$ $\vert$ p	$p \wedge q$ $\vert$ p q	$p \vee q$ $p \quad q$	$p \to q$ $\neg p \quad q$	$p \leftrightarrow q$ $p \quad \neg p$ $q \quad \neg q$
	$\neg(p \wedge q)$ $\neg p \quad \neg q$	$\neg(p \vee q)$ $\vert$ $\neg p$ $\neg q$	$\neg(p \to q)$ $\vert$ p $\neg q$	$\neg(p \leftrightarrow q)$ $p \quad \neg p$ $\neg q \quad q$

$p \to q \equiv \neg p \vee q$

$\neg(p \wedge q) \equiv \neg p \vee \neg q$
$\neg(p \vee q) \equiv \neg p \wedge \neg q$

$\neg(p \to q) \equiv \neg(\neg p \vee q) \equiv p \wedge \neg q$

$p \leftrightarrow q \equiv (p \to q) \wedge (q \to p)$
$\equiv (\neg p \vee q) \wedge (\neg q \vee p)$

$p \leftrightarrow q$ ✓
$\vert$
$(\neg p \vee q)$ ✓
$(\neg q \vee p)$ ✓

$\neg(p \leftrightarrow q) \equiv \neg((\neg p \wedge q) \vee (\neg q \wedge p))$

$\equiv \neg(\neg p \wedge q) \wedge \neg(\neg q \wedge p)$

$\equiv (p \vee \neg q) \wedge (q \vee \neg p)$

$\neg(p \leftrightarrow q)$ ∧
$(p \vee \neg q) \quad (q \vee \neg p)$
$\vert \qquad \vert$
$p, \neg q \qquad q, \neg p$

What does a truth tree for a complex proposition P yield ?

- A complete active path represents one or more truth assignment for P.
  - If p_i occurs on that path, p_i has value T.
  - If $\neg p_i$ occurs on that path, p_i has value F.
  - If p_i does not occur it can have both T or F.

- Distinct complete active paths may represent same truth assignment to $p_1, p_2, \ldots, p_n$ that makes P true.

- ~~If the tree has no active paths, P is a 'contradiction'.~~
- If the tree for P has no active paths, P is called 'contradiction'.

- Thus if the tree for $\neg P$ has no active paths, P is called 'Tautology'.

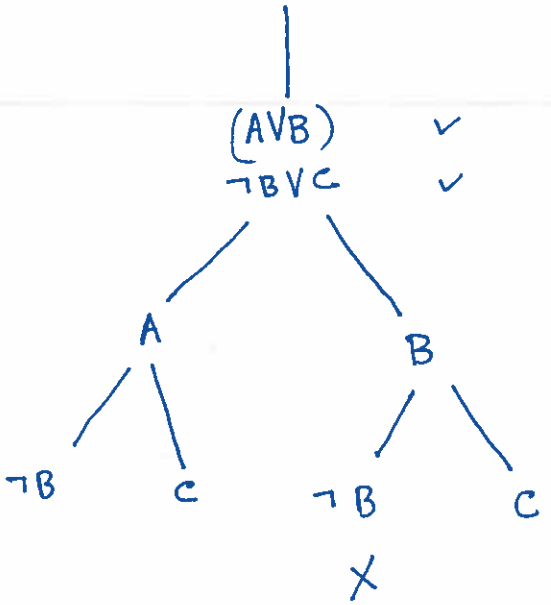
- If the trees for both P and $\neg P$ have active paths, then P is a 'contingency'.

How useful is truth tree ?

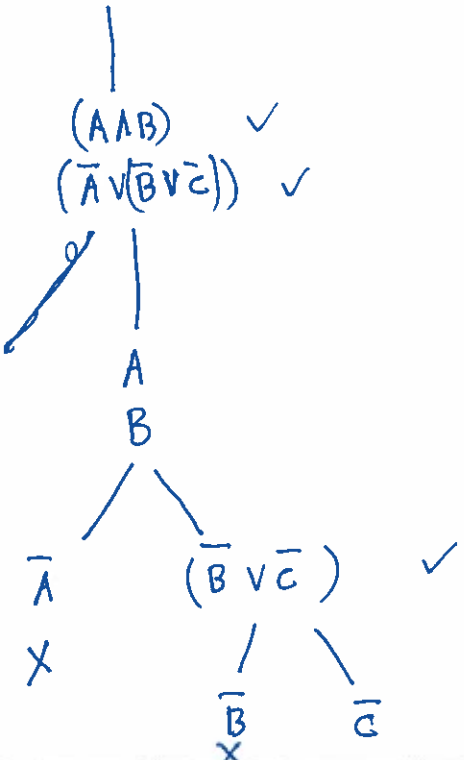
- **Contradiction** : To show that a complex proposition P is a contradiction, Construct 'Truth Tree' for P.
  P is contradiction iff the complete tree has no active paths.
- **Tautology** : Construct truth tree for $\neg P$
  P is a tautology iff the complete tree for $\neg P$ has no active paths.
- **Contingency** : Construct two trees (one for P & one for $\neg P$)
  P is a contingency iff both trees have complete tree have at least one active path each.
- **P & Q Equivalent** : $P \equiv Q$ is same as asking whether $(P \Leftrightarrow Q)$ is a tautology ?
  → Construct truth tree for $\neg(P \Leftrightarrow Q)$.
  P & Q are equivalent iff the complete tree for $\neg(P \Leftrightarrow Q)$ has no active paths.
- **$\{P, Q, R, \ldots\}$ Consistent** : Same as asking whether $(P \wedge Q \wedge R \wedge \cdots)$ is satisfiable.
  Construct truth tree for $(P \wedge Q \wedge R \wedge \ldots)$ and look for a complete active path. If such path is found the set is consistent.
- **Find DNF for P** : — Construct a truth tree for P.
  — From all complete active paths, obtain all truth assignments for propositional variables that make P true.
  — Convert each such assignments into conjunctive clauses
  — DNF formula is the disjunction of the conjunctive clauses.
- **Find CNF for P** — construct truth tree for $\neg P$.
  — For all any active complete paths construct a disjunctive clause with the negation of the appearing literals.
  — CNF formula is the conjunction of those disjunctive clauses.
- **Satisfiability** : Construct P and look for complete active path. If such a path is found P is satisfiable.

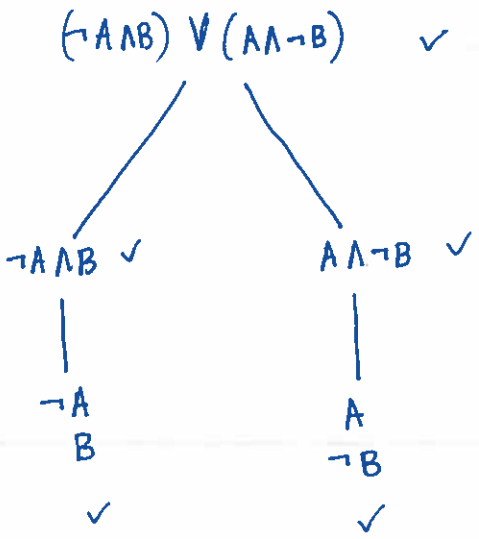
1. $A \vee B$, $\neg B \vee C$ Are there consistent ?

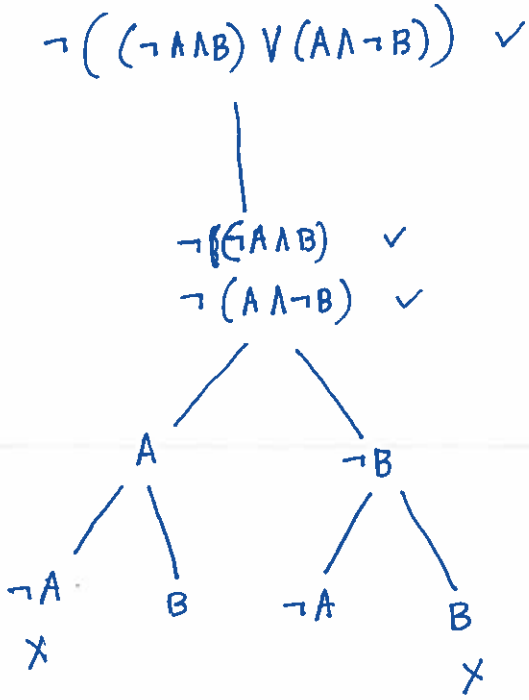
$P \equiv (A \vee B) \wedge (\neg B \vee C)$ ✓

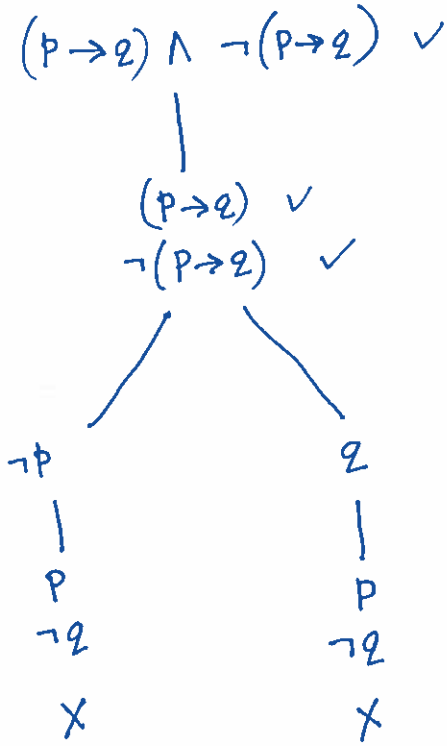


2. $(A \wedge B) \wedge (\bar{A} \vee (\bar{B} \vee \bar{C}))$ ✓




(¬A∧B) ∨ (A∧¬B) ✓
¬A∧B ✓
¬A
B
✓
A∧¬B ✓
A
¬B
✓


¬((¬A∧B) ∨ (A∧¬B)) ✓
¬(¬A∧B) ✓
¬(A∧¬B) ✓
A
¬B
¬A
B
¬A
B
X
X


(P→Q) ∧ ¬(P→Q) ✓
(P→Q) ✓
¬(P→Q) ✓
¬P
P
¬Q
X
Q
P
¬Q
X

Page 98