# Truth Trees

Computability and Logic

# Drawback of the Short Truth Table Method

- A drawback of the short truth table method is that you are not always forced to assign any further truth values:

| | | | |
|---|---|---|---|
| $R \to (Q \to \neg P)$ | $(Q \land \neg R) \to \neg P$ | $R \leftrightarrow P$ | $Q$ |
| $T_1$ $T_2$ | $T_2$ $T_1$ | $T_1$ | $T_1$ |

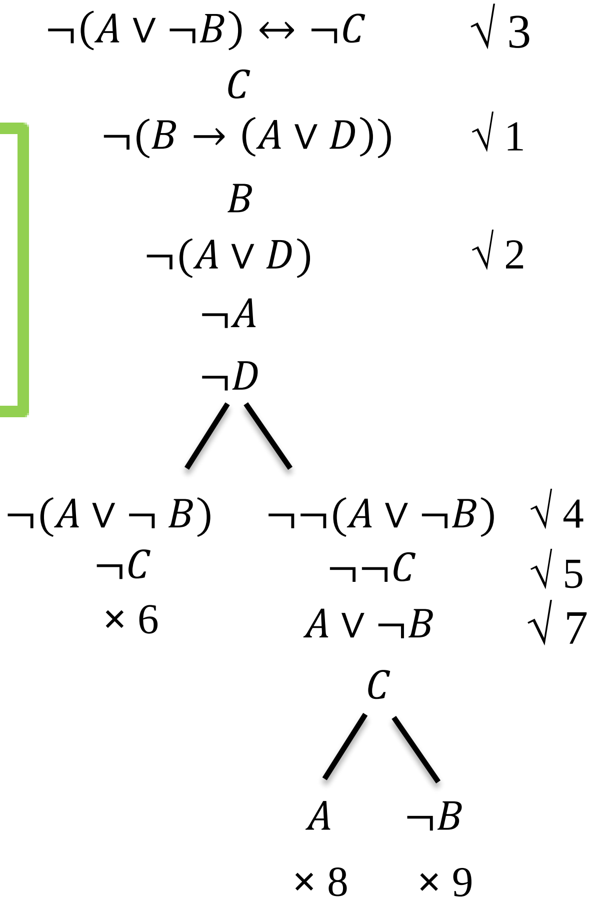
- At this point, you can choose to assign certain truth values, but if your choice does not lead to the row you are looking for, then you need to try a different option, and the short truth table method has no tools to do go through all of your options in a systematic way.

## Truth Trees

- The obvious solution to the drawback of the short truth table method is to incorporate tools to *systematically* keep track of multiple options.
- One method that does so is the truth tree method:
  - The truth tree method tries to systematically derive a contradiction from the assumption that a certain set of statements is true.
  - Like the short table method, it infers which other statements are forced to be true under this assumption.
  - When nothing is forced, then the tree branches into the possible options.

# Truth Tree Example

$\neg(A \lor \neg B) \leftrightarrow \neg C$
$C$
$\therefore B \rightarrow (A \lor D)$



# Decomposition Rules for Truth Trees

$$\begin{array}{l} \neg\neg P \;\; \surd \\ P \end{array}$$

$$\begin{array}{l} P\wedge Q \;\; \surd \\ P \\ Q \end{array}$$

$$\begin{tikzcd}[column sep=small] & \neg(P\wedge Q) \;\; \surd \arrow[dl, dash] \arrow[dr, dash] & \\ \neg P & & \neg Q \end{tikzcd}$$

$$\begin{tikzcd}[column sep=small] & P\vee Q \;\; \surd \arrow[dl, dash] \arrow[dr, dash] & \\ P & & Q \end{tikzcd}$$

$$\begin{tikzcd}[column sep=small] & P\rightarrow Q \;\; \surd \arrow[dl, dash] \arrow[dr, dash] & \\ \neg P & & Q \end{tikzcd}$$

$$\begin{array}{l} \neg(P\rightarrow Q) \;\; \surd \\ P \\ \neg Q \end{array}$$

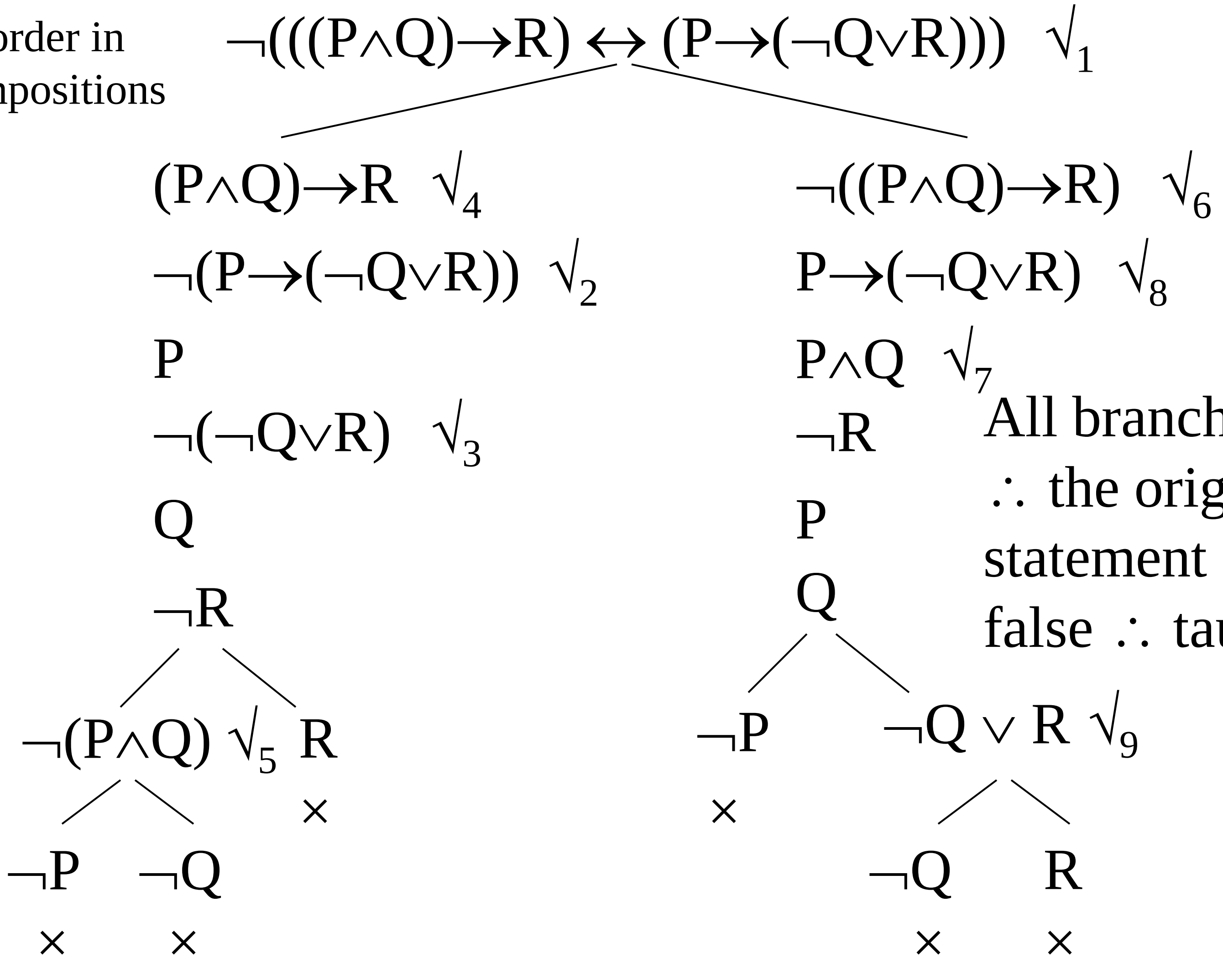
$$\begin{array}{l} \neg(P\vee Q) \;\; \surd \\ \neg P \\ \neg Q \end{array}$$

$$\begin{tikzcd}[column sep=small, row sep=small] & P\leftrightarrow Q \;\; \surd \arrow[dl, dash] \arrow[dr, dash] & \\ P & & \neg P \\ Q & & \neg Q \end{tikzcd}$$

$$\begin{tikzcd}[column sep=small, row sep=small] & \neg(P\leftrightarrow Q) \;\; \surd \arrow[dl, dash] \arrow[dr, dash] & \\ P & & \neg P \\ \neg Q & & Q \end{tikzcd}$$

# Truth Tree Example

(The numbers indicate the order in which decompositions were made)



$\neg(((P\wedge Q)\rightarrow R)\leftrightarrow(P\rightarrow(\neg Q\vee R)))\ \surd_1$
$(P\wedge Q)\rightarrow R\ \surd_4$
$\neg(P\rightarrow(\neg Q\vee R))\ \surd_2$
$P$
$\neg(\neg Q\vee R)\ \surd_3$
$Q$
$\neg R$
$\neg(P\wedge Q)\ \surd_5$
$R$
$\times$
$\neg P$
$\neg Q$
$\times$
$\times$
$\neg((P\wedge Q)\rightarrow R)\ \surd_6$
$P\rightarrow(\neg Q\vee R)\ \surd_8$
$P\wedge Q\ \surd_7$
$\neg R$
$P$
$Q$
$\neg P$
$\times$
$\neg Q\vee R\ \surd_9$
$\neg Q$
$R$
$\times$
$\times$

All branches close $\therefore$ the original statement cannot be false $\therefore$ tautology!

## Further Rules for Truth Trees

- A decomposable statement is any statement that is not a literal.
- A sentence belongs to every branch below it.
- You can close a branch if an atomic statement and its negation both belong to that branch.
- Statements need to be decomposed in every open branch it belongs to.
- A branch is finished if all of its decomposable statements have been decomposed.

# How to Use Truth Trees

- At the ‘root’ of the tree, write down all statements that you try and make true according to the combination of truth values you are interested in.
- Decompose according to the rules until you have a finished open branch or until all branches close.
  - If there is a finished open branch, then that means that it is possible for all statements at the root of the tree to be true.
  - If all branches close, then that means that it is not possible for all statements at the root of the tree to be true.
  - It is up to you to draw the appropriate conclusion from this.

# Example: Testing for Validity

- To use a tree to test for validity:
  - 1. Write down at the root of the tree all premises and the negation of the conclusion
  - 2. Work through the tree until you find an open and completed branch or all branches are closed
  - 3a. If you found an open and completed branch, then that means that it is possible for all statements in the root of the tree to be true, which in turn means that it is possible for all premises to be true while the conclusion is false. Hence, the argument is invalid.
  - 3b. If all branches closed, the opposite is true, i.e. the argument is valid.

# Example: Testing for Tautology

- To use a tree to see whether some statement is a tautology:
  - 1. Write down at the root the *negation* of the statement
  - 2.  Work out tree
  - 3a. If you find an open and completed branch, then that means that it is possible for the statement in the root of the tree to be true, which in turn means that it is possible for the original statement to be false. Hence, the original statement is not a tautology.
  - 3b. If all branches close, then the original statement cannot be false, and hence *is* a tautology.

# How to Avoid Bushy Trees

- Since at any point there can be multiple undecomposed decomposable statements, the tree is going to look different based on which statement you choose to decompose.
- Since more branches means more work, you want to avoid branching as much as possible. So, as a heuristic:
  - Choose statements that don’t branch
  - If you have to branch, choose those that you know will quickly lead to a closed branch.
  - If you don’t know which one leads quickly to a closed branch, choose a large statement (why?)

# Possible Features to Improve Tree Method

- Allow branch to be closed whenever it contains *any* statement and its negation (i.e. not just for an atomic statement and its negation)
- Add short-cuts when you know some branch will immediately close (e.g. when you have P and $P \to Q$, the $\neg P$ branch for the decomposition of $P \to Q$ will immediately close with P, leaving just one branch with Q … so why not do this immediately)
- Add ‘Law of Excluded Middle’: at any time, two branches can be created for any statement $\varphi$: one with $\varphi$, and one with $\neg\varphi$.
- The ‘KE Calculus’ does implement the last two ideas (see next slide). Others? (research project …)

# Rules of KE Calculus

$$\dfrac{\neg\neg P}{P}$$

DN

$$\dfrac{\neg(P \rightarrow Q)}{\begin{array}{l} P \\ \neg Q \end{array}} \qquad \dfrac{P \wedge Q}{\begin{array}{l} P \\ Q \end{array}} \qquad \dfrac{\neg(P \vee Q)}{\begin{array}{l} \neg P \\ \neg Q \end{array}}$$

Alpha

$$\begin{tikzcd} & \cdot \arrow[dl, dash] \arrow[dr, dash] & \\ \varphi & & \neg\varphi \end{tikzcd}$$

Branch

$$\dfrac{\begin{array}{l} P \vee Q \\ \neg P \end{array}}{Q} \qquad \dfrac{\begin{array}{l} P \rightarrow Q \\ P \end{array}}{Q}$$

$$\dfrac{\begin{array}{l} \neg(P \wedge Q) \\ P \end{array}}{\neg Q} \qquad \dfrac{\begin{array}{l} P \rightarrow Q \\ \neg Q \end{array}}{\neg P}$$

Beta

$$\dfrac{\begin{array}{l} P \leftrightarrow Q \\ P \end{array}}{Q} \qquad \dfrac{\begin{array}{l} \neg(P \leftrightarrow Q) \\ P \end{array}}{\neg Q}$$

$$\dfrac{\begin{array}{l} P \leftrightarrow Q \\ \neg P \end{array}}{\neg Q} \qquad \dfrac{\begin{array}{l} \neg(P \leftrightarrow Q) \\ \neg P \end{array}}{Q}$$

Eta