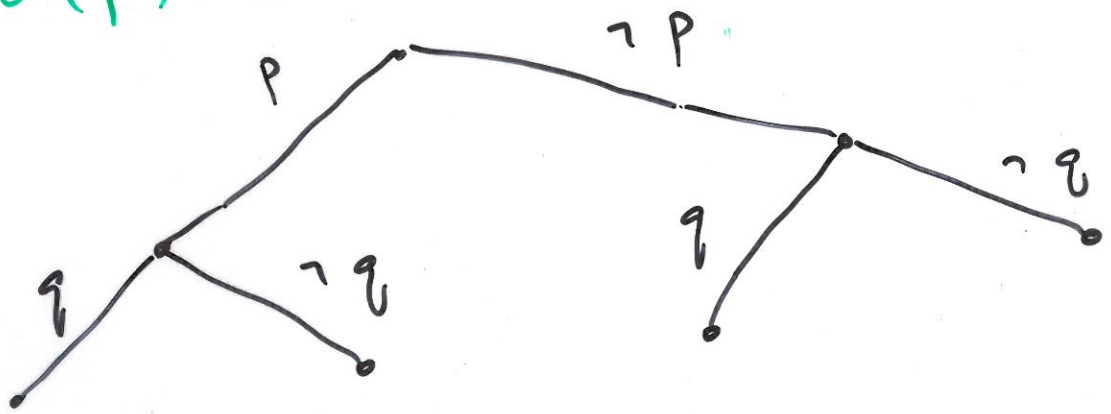
Backtracking and Resolution

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

nsider the tree representing
ll possible truth assignments.

Here $T(p)=0$ is represented as $\neg p$



Observations:

(1) If Γ is satisfiable, at least one branch of this tree has an assignment that satisfies all clauses in Γ

and depth first search (DFS)

e.g. a solution

(2) If Γ is not satisfiable,
every branch is an assignment
that makes some clause of Γ false
(every t.a falsifies it) (thus Γ is false under that t.a.).

In this case (2), we can construct
a resolution refutation of Γ
on the tree, proceeding bottom up,
beginning with clauses made false
at the leaves.

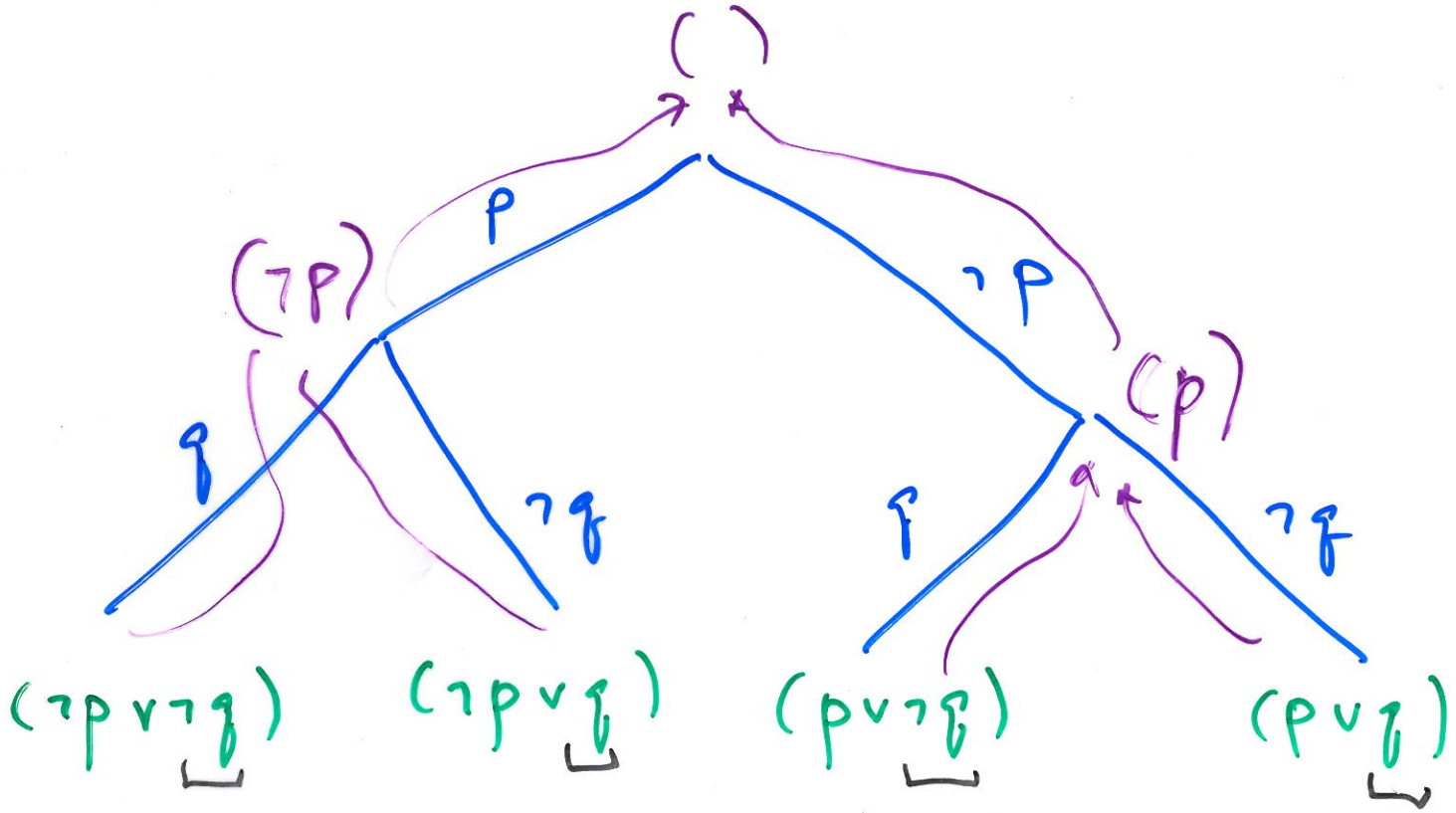
To build the refutation:

(1) label each leaf with a clause it makes false

(2) if node v has children $(p \vee A)$ and $(\neg p \vee B)$, label v with $(A \vee B)$

(3) if node v "branches on q", but the clause on one of its children does not contain q or $\neg q$, label v with that clause.

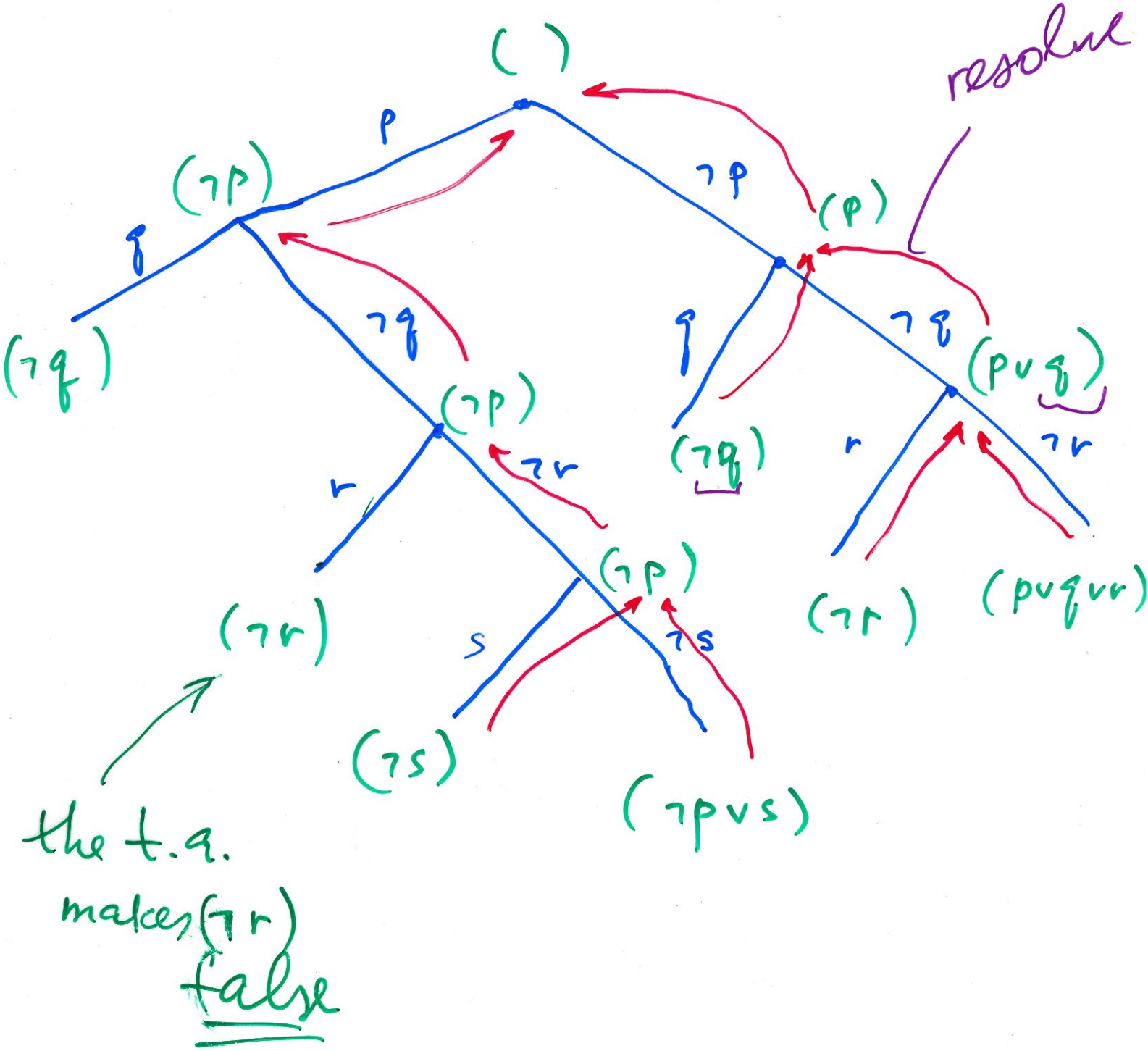
Ex. $\Gamma = (p \vee q) \wedge (\neg p \vee \neg q) \wedge (\neg p \vee q) \wedge (p \vee \neg q)$



Note: we may terminate any branch "early", as soon as it makes a branch false, and the method # still works

see next example

Ex. $\Gamma = (p \vee q \vee r) \wedge (\neg p \vee s) \wedge (\neg q) \wedge (\neg r) \wedge (\neg s)$



Backtracking for SAT

BT-SAT (Γ, τ)

// Γ is a set of clauses (a f-la in CNF)

// τ is a partial truth assign

// initial call is with τ empty

if $\tau(C) =$ false for some $C \in \Gamma$

return "UnSAT" // the current t.a τ is not good => backtrack 1 step

else

$L \leftarrow$ a literal from Γ for which $\tau(L)$ is undefined

if BT-SAT (Γ, $\tau \cup \{L\}$ returns "SAT",

return "SAT"

else

return BT-SAT (Γ, $\tau \cup \{\neg L\}$)

end if

end end if

- Most industrial SAT solvers use this process - plus many smart refinements

- Good SAT solvers are millions of times faster than naive ones

- There are highly competitive annual SAT solver competitions, with entries from universities and industry

There are very efficient SAT solvers;

SAT competitions every year

Lawrence Ryan, former SFU student, created a very fast SAT solver