Other NP-Complete Problems

More terminology for Boolean expressions:

- A *literal* is a variable or the negation of a variable.
- A *clause* is a single literal or the disjunction (OR) of literals..
- A Boolean expression is in *conjunctive normal form* if it is a single clause or the conjunction (AND) of clauses. For example,
  $(\sim x \lor \sim y \lor z) \land (x \lor \sim y \lor \sim z)$

CNF-SAT is the language of satisfiable conjunctive normal form expressions.

Theorem: CNF-SAT is NP-Complete.
Proof: We will show that SAT reduces (in polynomial time) to CNF-SAT. In other words we will start with a Boolean expression s and produce expression s' so that s is in SAT if and only if s' is in CNF-SAT.

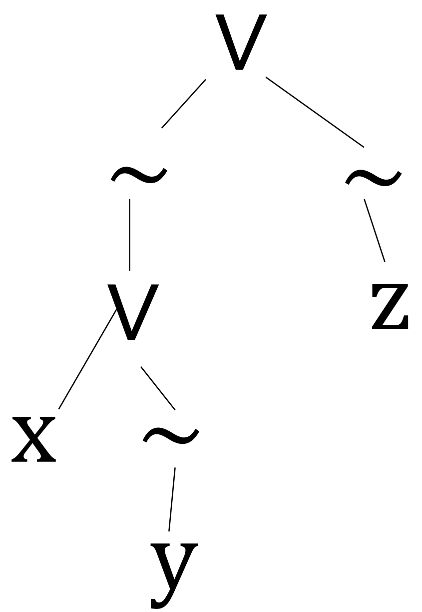
If we had a truth table for s it would be easy to make s'. For example, suppose we know that the only times s is F is when x=T, y=T, z=F and when x=F,y=T,z=T. We can build clauses that negate these instances:

$$s' = (\sim x \lor \sim y \lor z) \land (x \lor \sim y \lor \sim z)$$

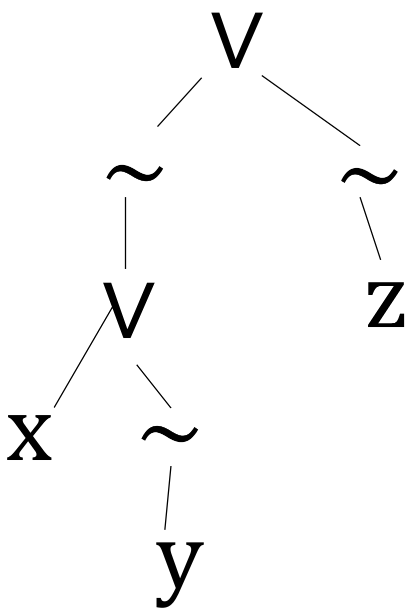
Unfortunately, building a truth table for s takes exponential time.

Rather than building a truth table, given s we will build a CNF expression s' that has additional variables (and so is not equivalent to s) but is satisfiable if and only if s is satisfiable.

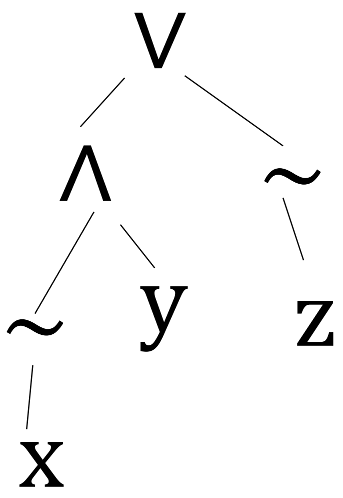
Step 1: Parse s into a parse tree.
For example, if s is ~(x V ~y) V ~z the parse tree is



Step 2: Walk down the tree using DeMorgan's laws to push negations to variables.

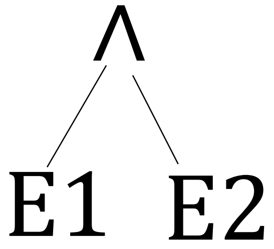


becomes



Step 3. Start at the leaves and walk up, replacing each node with a CNF expression that is satisfiable if and only if the subtree rooted at the node is satisfiable.

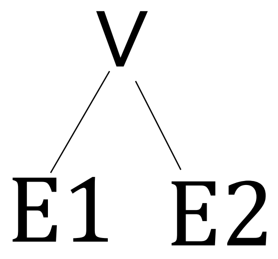
Case 3A: Suppose the tree is



and we have already replaced E1 with CNF expression F1 and E2 with F2. We replace the ∧-node with F1∧F2.

Case 3B:

Suppose the tree is



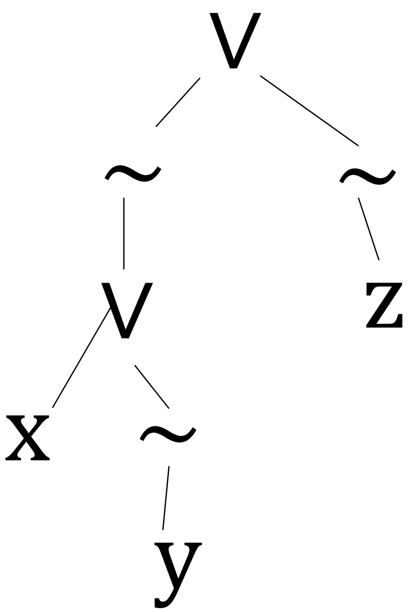
and we have already replaced E1 with CNF expression $F1=g_1 \wedge g_2 \wedge g_3 \wedge \ldots \wedge g_K$ (the g_i are the clauses of F1) and E2 with $F2=h_1 \wedge h_2 \wedge h_3 \wedge \ldots \wedge h_L$. Let y be a new variable not used in s or any of the F-expressions. We replace the ∨-node with

$F = (y \vee g_1) \wedge (y \vee g_2) \wedge \ldots \wedge (y \vee g_K) \wedge (\sim y \vee h_1) \wedge (\sim y \vee h_1) \wedge \ldots \wedge (\sim y \vee h_L)$

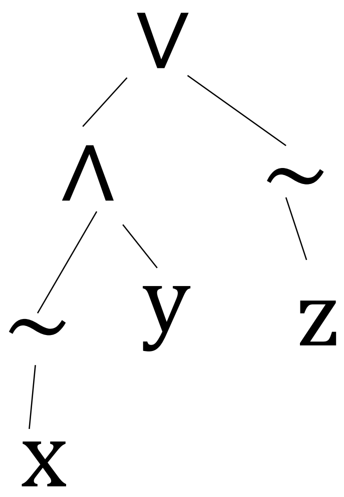
If y=T this requires $h_1 \wedge h_2 \wedge h_3 \wedge \ldots \wedge h_L$ to be T, so F2 must be T. Similarly, if y=F then F1 must be T. F is satisfiable if and only if $F1 \vee F2$ is satisfiable.

By the time we get to the root of the tree this has produced a CNF expression s' that is satisfiable if and only if s is satisfiable. If the length of s is n then s has no more than n literals, each with length no more than n, so $|s'| <= n^2$.

Example: In an earlier example we parsed s = ~(x ∨ ~y) ∨ ~z as

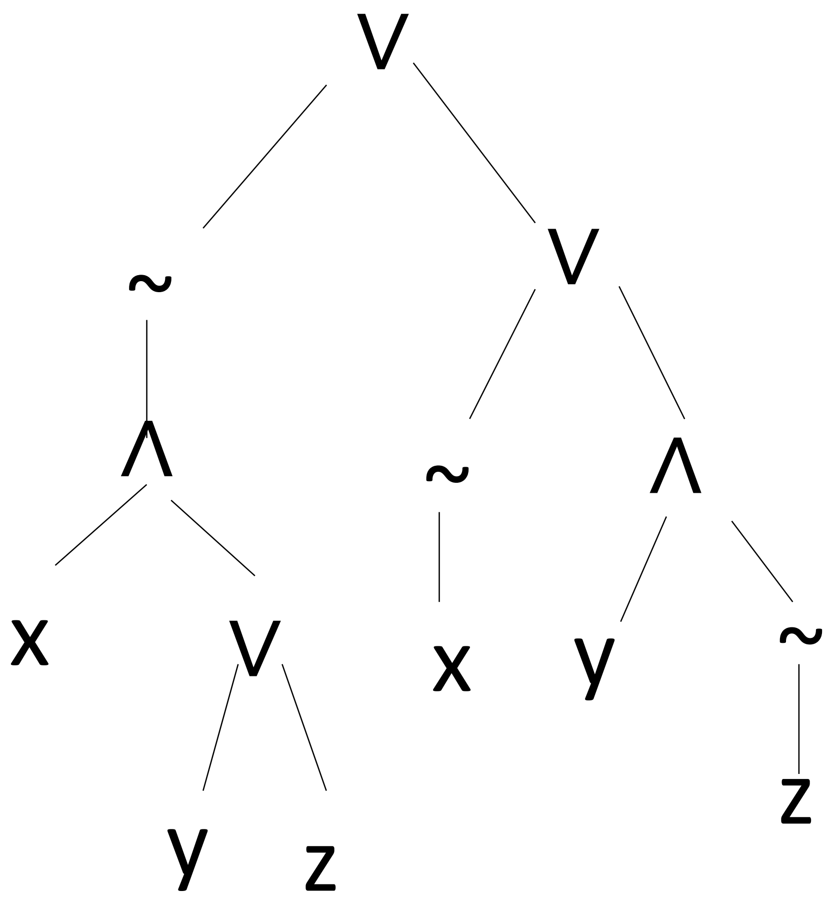


and converted that to

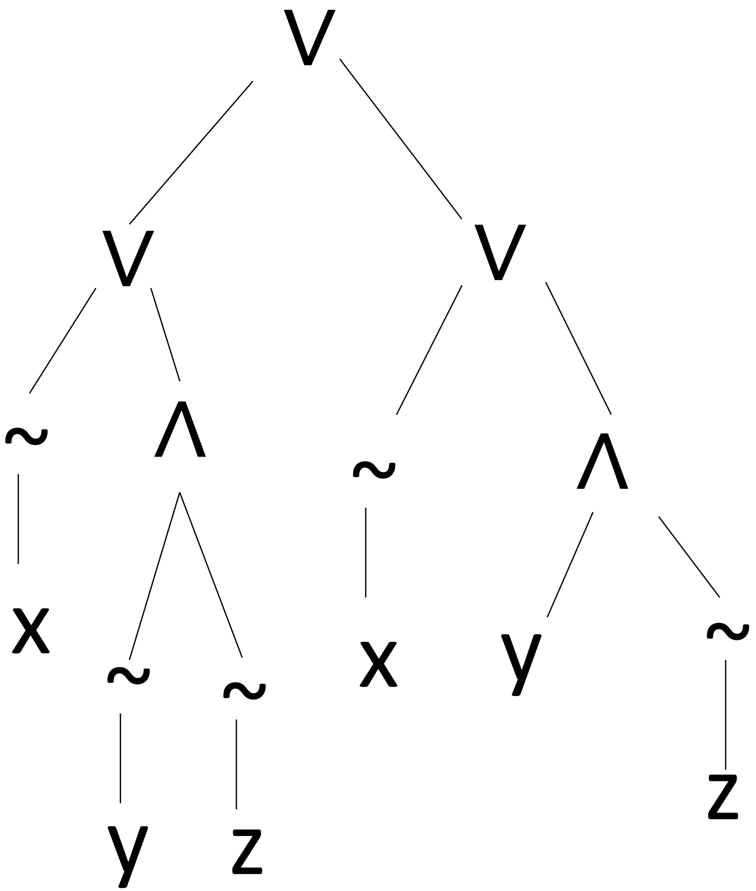


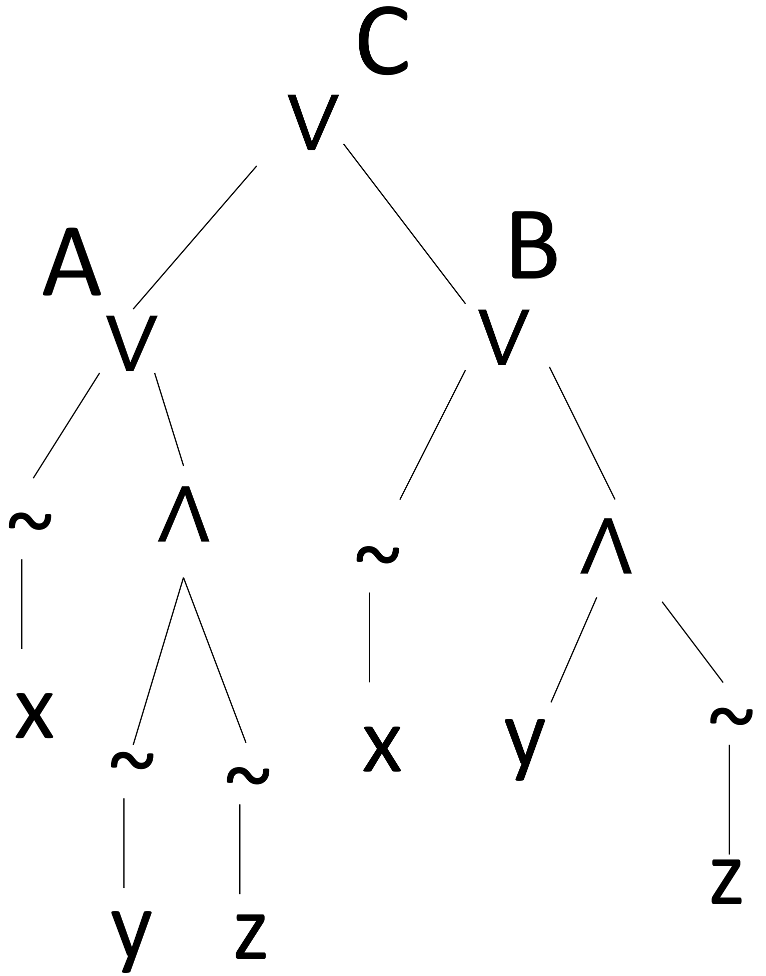
The corresponding CNF expression is (w∨~x)∧(w∨y) ∧(~w∨~z)

Example: Start with ~(x∧(y∨z))∨~x∨(y∧~z). This parses into



which converts to





Node A becomes (w∨~x)∧(~w∨~y)∧(~w∨~z)

B becomes (t∨~x)∧(~t∨y)∧(~t∨~z)

C becomes

(u∨w∨~x)∧(u∨~w∨~y)∧(u∨~w∨~z)∧(~u∨t∨~x)∧(~u∨~t∨y)∧(~u∨~t∨~z)

3CNF is the language of conjunctive normal form expressions where each clause has exactly 3 literals. For example, one expression in 3CNF is $(x \vee \sim y \vee z) \wedge (x \vee y \vee \sim z)$

3CNF-SAT (also called 3SAT) is the language of satisfiable 3CNF expressions.

Theorem: 3CNF-SAT is NP-Complete
Proof: We will reduce CNF-SAT to 3CNF-SAT by converting CNF expressions to 3CNF expressions.

Let $e = e_1 \wedge e_2 \wedge e_3 \wedge .... \wedge e_k$ be an expressioin in CNF. Each e_i must be a disjunction of literals.

a) Suppose e_i has only one literal, x. Let r and s be new variables. Replace e_i by $f_i=(x \vee r \vee s) \wedge (x \vee \sim r \vee \sim s) \wedge (x \vee r \vee \sim s) \wedge (x \vee \sim r \vee \sim s)$ f_i can be satisfied if and only if x is satisfied.
b) Suppose e_i has only two literals, such as $x \vee y$ Let r be a new variable and replace e_i by $f_i=(x \vee y \vee r) \wedge (x \vee y \vee \sim r)$

c) Suppose ei has 4 literals: ei = x1 V x2 V x3 V x4. Let r be a new variable. Then f_i=(x1 V x2 V r) Λ(x3 V x4 V ~r)

d) Suppose e_i has 5 literals: $e_i = x_1 \vee x_2 \vee x_3 \vee x_4 \vee x_5$. Let s_1 and s_2 be new variables. Then

$$f_i=(x_1 \vee x_2 \vee s_1) \wedge (x_3 \vee \sim s_1 \vee s_2) \wedge (x_4 \vee x_5 \vee \sim s_2)$$

s_1	s_2	f_i reduces to
T	T	$x_4 \vee x_5$
T	F	x_3
F	T	$(x_1 \vee x_2) \wedge (x_4 \vee x_5)$
F	F	$x_1 \vee x_2$

We can extend this pattern to any number of literals. If e_i has n literals then f_i has n-2 clauses each with 3 literals and uses n-2 new variables. $|f_i| <= 3*|e_i|$ so the length of the 3CNF expression this builds is a polynomial function of the length of the original CNF expression.