# CS456 UNLV Spring Semester 2018

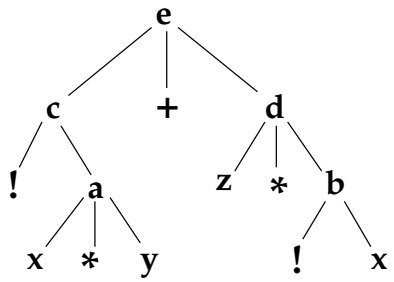
## Reduction of SAT to 3-CNF-SAT

The internet contains pages that explain how SAT can be reduced to 3-SAT. In this handout, I will give a brief explanation of a reduction, illustrated by a small example.

Suppose you are given a Boolean expression $E$. Our goal is to construct a Boolean Expression $E'$ in conjunctive normal form such that $E$ and $E'$ are equisatisfiable, and such that the length of $E'$ is at most a polynomial function of the length of $E$. We outline the construction below.

- First, eliminate all operators except conjunction, disjunction, and negation, *i.e.,* which we write as *, +, and !. For example, $u \Rightarrow v$ is replaced by the equisatisfiable formula $!u + v$, $u = v$ by $(u+!v) * (v+!u)$, and $u \neq v$ by $(u + v) * (!u+!v)$. Any other Boolean operator can similarly be eliminated.
- Then, draw a parse tree for $e$, where there is a leaf for each operator and each variable, and an internal node for each larger sub-expression of $e$. Then replace each internal node by a new variable. These new variables must be distinct, and none can be the same as a variable in the original formula.
- Then, for every internal node, write the Boolean formula which sets its variable equal to the expression given by its children. Replace each such equality by an equivalent conjunction of CNF clauses, and let $E'$ be the conjunction of all those clauses.

Here is an example. Let $E$ be the Boolean expression $!(x * y) + z*!x$. We need a variable for each subexpression of $E$. We introduce variables $a, b, c, d, e$, where $a$ represents $x * y$, $b$ represents $!x$, $c$ represents $!(x * y)$, $d$ represent $z*!x$, and $e$ represents $!(x * y) + z*!x$. We then draw the parse tree for $E$:




Given an assignment of the original variables $x$, $y$, and $z$, we place the assigned Boolean values (either true or false) at the leaves, and then evaluate bottom-up. The root is then true if and only if the assignment is satisfying.

The following Boolean expression states that the nodes of the tree are properly evaluated and that the root is true. The expression is satisfiable if and only if $E$ is satisfiable.

$$e * (e = c * d) * (c =!a) * (a = x * y) * (d = z * b) * (b =!x)$$

To obtain conjunction normal form, we replace each clause by an equisatisfiable CNF formula using the following rules:

1. Replace $u = !v$ by $(u + v) * (!u + !v)$.
2. Replace $u = v + w$ by $(u + !v) * (u + !w) * (!u + v + w)$.
3. Replace $u = v * w$ by $(!u + v) * (!u + w) * (u + !v + !w)$.

We then obtain the CNF formula $E'$:

$$e*(e+!c)*(e+!d)*(!e+c+d)*(c+a!*(!c+!a)*(!a+x)*(!a+y)*(a+!x+!y)*(!d+z)*(!d+b)*(d+!z+!b)*(b+x)*(!b+!x)$$

$E$ and $E'$ are equisatisfiable. Finally, we can then put $E'$ into 3-CNF form by "padding" each clause with redundant terms, as needed. For example, we can replace $e$ by $(e + e + e)$, $(!e + c)$ by $(!e + c + c)$, *etc.*