COMP3121-Ass3-Q1

z5302513, Kihwan Baek

July 2021

1. Assumption

For every l^{th} and r^{th} operation where $1 \leq l \leq r \leq n$, we assume that T(l,r) and F(l,r) are the number of each ways to place brackets to make the expression starting from at the l^{th} symbol and ending at r^{th} symbol evaluate to true (T) and false (F).

2. Base cases

The base case is that $\mathrm{T}(i,i)$ is 1 if symbol i is true, and 0 if symbol i is false. The reverse applies to $\mathrm{F}(i,i)$.

3. Subproblem and recursions

We 'split' the expression around an operator m. So, everything to the left and right of the operator can be divided as two smaller expressions.

$$T(l,r) = \sum_{m=l}^{r-1} \mathrm{TSplit}(l,m,r)$$
$$F(l,r) = \sum_{m=l}^{r-1} \mathrm{FSplit}(l,m,r)$$

And also, we divide the splitting cases as the operator is changed.

$\mathrm{TSplit}(l,m,r) =$

1) $\mathrm{T}(l,m)\times\mathrm{T}(m+1,r)$ if $m =$'AND',
2) $\mathrm{T}(l,m)\times\mathrm{F}(m+1,r) + \mathrm{F}(l,m)\times\mathrm{T}(m+1,r) + \mathrm{F}(l,m)\times\mathrm{F}(m+1,r)$ if $m =$'NAND',
3) $\mathrm{T}(l,m)\times\mathrm{F}(m+1,r) + \mathrm{T}(l,m)\times\mathrm{T}(m+1,r) + \mathrm{F}(l,m)\times\mathrm{T}(m+1,r)$ if $m =$'OR',
4) $\mathrm{F}(l,m)\times\mathrm{F}(m+1,r)$ if $m =$'NOR'.

$\mathrm{FSplit}(l,m,r) =$

1) $\mathrm{T}(l,m)\times\mathrm{F}(m+1,r) + \mathrm{F}(l,m)\times\mathrm{F}(m+1,r) + \mathrm{F}(l,m)\times\mathrm{T}(m+1,r)$ if $m =$'AND',

2) $\mathrm{T}(l,m)\times\mathrm{T}(m+1,r)$ if m ='NAND',
3) $\mathrm{F}(l,m)\times\mathrm{F}(m+1,r)$ if m ='OR',
4) $\mathrm{T}(l,m)\times\mathrm{F}(m{+}1,r)+\mathrm{F}(l,m)\times\mathrm{T}(m{+}1,r)+\mathrm{T}(l,m)\times\mathrm{T}(m{+}1,r)$ if m ='NOR'.

4. How to obtain the final solution

As evaluating the subproblem and combine them based on recursion, we can get the final solution as getting the symbols.

5. Time complexity

If we assume there are n amount of 'true' and 'false', there will be 'n-1' splitting points (TSplit or FSplit). And also, the points are evaluated by $\mathrm{O}(n^2)$ different ranges that l and r could cover because when we put the left bracket at position k ($1 \leq k \leq n$), we have n-k+1 ways to put the right bracket. So the total time complexity for evaluating different ranges is $\mathrm{O}(n^2)$. Finally, it takes O(n-1)*$\mathrm{O}(n^2)$ = $\mathrm{O}(n^3)$ time.