COMP2411 Tutorial 13

1. Complete the program for computing the list of subformulas of a formula, discussed in class.

2. Write a program for computing the number of nodes of a tree in the nested list representation.

3. Write a program for computing the number of leaves of a tree in the nested list representation.

4. Recall that the program \texttt{concat(X,Y,Z)} discussed is class needs to perform a recursive pass through a given list \texttt{X} in order to concatenate it with a given list \texttt{Y}. A \textit{difference list} is a way of representing a list that allows the concatenation operation to be performed more efficiently. The difference list representation of the list \([a_1, \ldots, a_n]\) is the term \(l([a_1, \ldots, a_n|X], X)\) where \(X\) is a variable. (A fresh variable must be used for each difference list.) The empty list is represented by the term \(l(X,X)\). Two difference lists may be concatenated by the following code:

\[
dconcat(l(L,XL),l(M,XM),l(L,XM)) :- XL = M.
\]

Draw an SLD resolution tree to show how the difference list representations of \([1,2,3,4,5]\) and \([6,7,8]\) are concatenated by this code, and explain how it works in general.

5. Write a definite clause grammar for the set of sequences of the form \(a^n b^m c^{n+m}\).

6. Extend the grammar for arithmetic expressions discussed in the lecture to handle all integer expressions, not just digits, so that

\[
[\text{‘(', 1, 4, 5, ‘+', 3, 5, ‘)}]\]

is parsed and the program returns the term \texttt{plus(145,35)}. Can you find a way to do this that ensures that backtracking is not required to parse the integers?