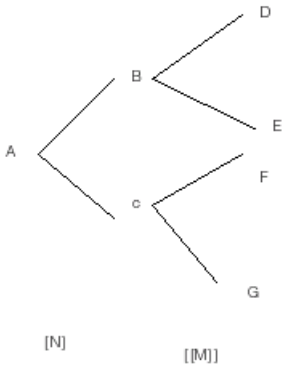
Syntax tree

I'm not sure about the description of formal syntax tree, but as far as my application is concerned the following is what I came up with.

For example, let's say we have a sample tree like thus:



Here, we want to say that [N] is the class of any vertical section of a tree that has root as a node, and [[M]] to be the class of any vertical section of a tree that is not [N]. In this case:

[N]=(A)(B,C)=A B,C.

[[M]]=(B,C)((D,E),(F,G))=B,C (D,E),(F,G).

[N][[M]]=(A)(B,C)((D,E),(F,G))=A B,C (D,E),(F,G). Let's consider setting AA=A.

Notice that we have achieved the feat of converting a recursive information(tree) into AB form that can easily be parsed. The AB form is a form that has two containers A,B and they are easy to parse because you can parse A,B sequentially. The whole tree in our example is of the class [N][[M]]. There could be others like [N] or [N][[M]][M]], etc.

The last way of representing a tree(other than as a diagram or in terms of [N] and [[M]]) is code itself. So let's do that for our example:

```
A {
   B {
      D,E
   }
   C {
      F,G
   }
}
```

Since we are dividing actual code into two parts in each step of the overall parsing action, we can view that they are product of each other, so that they may be arranged in a table. If my understanding of the terminology is correct(which I admit is quite rusty at the moment), then lexical steps are products with parsing steps.