Adrien Protzel

1. Parse trees visualize the true order and precedence of the given syntax grammar, supporting the understanding of semantics.

1. A simple example is of writing out a math equation: 1 + 2 \* 3. Which can be written in left or right associativity. 1+(2\*3)=7, (1+2)\*3=9  
     
    + \*

/ \ /\

1 \* + 3

/ \ / \

2 3 1 2

These two parse trees have the same equation but the order of which the numbers will be processed is different because of precedence. The first tree will focus on the lowest numbers (2, 3) and its operand \* before adding the 1 which will equal 7. The second tree prioritized the addition before multiplication which then equals 9. This is why a universal order is important like PEMDAS to have a certain order of operations and then make the parse trees execute in a correct precedence order.

1. Both trees are very similar, syntax trees seeming to be written more cleanly with short clear leaf ends and only two branches per node. While the parse trees will have three branches, two showing the next node and one being the operator itself. The final nodes are dangling leaves. The parse tree writes itself out as more line my line versus syntax which is operator then the values associated with it.

Parse trees are better at showing the extant order of syntax execution, which is helpful for finding logic issues and resolving them. While the syntax tree is easier to read by humans which can help structure an overall design.