Syntax tree in Compiler Design

Construction of Syntax Tree

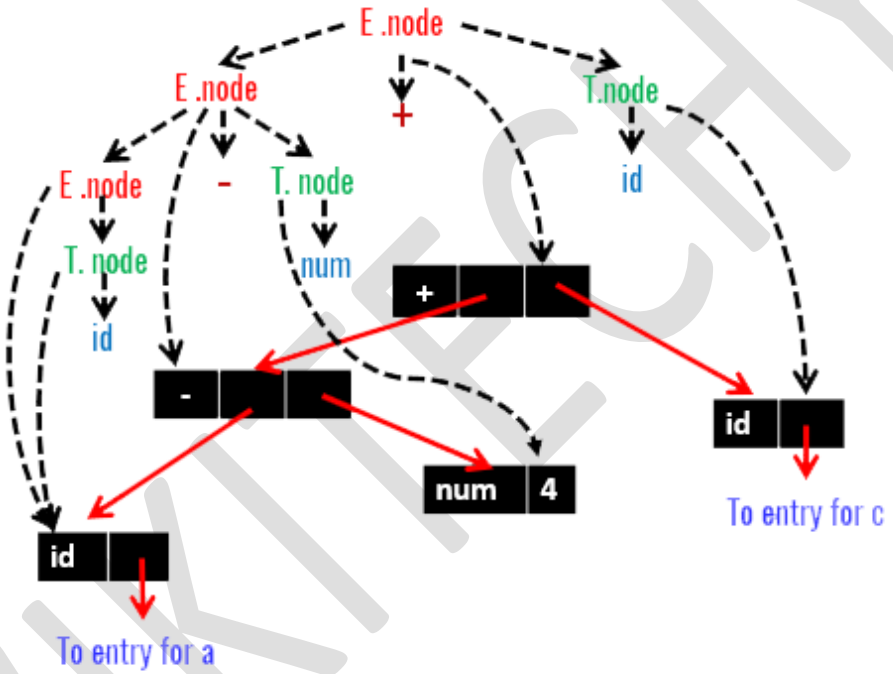
- Syntax directed definitions are very useful for construction of syntax trees. Each node in a syntax tree represents a construct. The children of the node represent the meaningful components of the construct.
- A syntax-tree node representing an expression E1 + E2 has label + and two children representing the subexpressions E1 and E2
- The nodes of a syntax tree are implemented by objects with a suitable number of fields. Each object will have an op field that is the label of the node.
- The objects will have additional fields as follows:
    - If the node is a leaf, an additional field holds the lexical value for the leaf. A constructor function Leaf (op, val) creates a leaf object. Alternatively, if nodes are viewed as records, then Leaf returns a pointer to a new record for a leaf.
    - If the node is an interior node, there are as many additional fields as the node has children in the syntax tree. A constructor function Node takes two or more arguments: Node(op, c1, c2, . . . , ck) creates an object with first field op and k additional fields for the k children c1, . . . , ck.

Example

- The L-attributed definition performs the same translation as the S-attributed definition.

 © Copyright 2016. All Rights Reserved.

```
p1 = new Leaf ( id, entry-a );
p2 = new Leaf ( num, 4 );
p3 = new Node ( '-', p1, p2 );
p4 = new Leaf ( id, entry-c );
p5 = new Node ( '+', p3, p4 );
```



Syntax tree for a - 4 + c

For More Details Click Here:

https://www.wikitechy.com/tutorials/compiler-design/syntax-tree-in-compiler-design

 © Copyright 2016. All Rights Reserved.