CSCI203

Week 4 – Lecture B

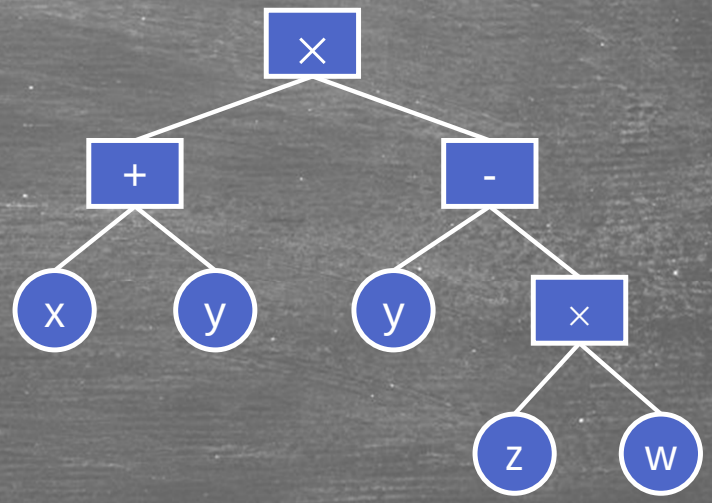
Fun With Binary Trees

- Expression trees.
- Building an expression tree:
  - with help from a stack.
- All trees are binary trees.

Binary Expression Trees

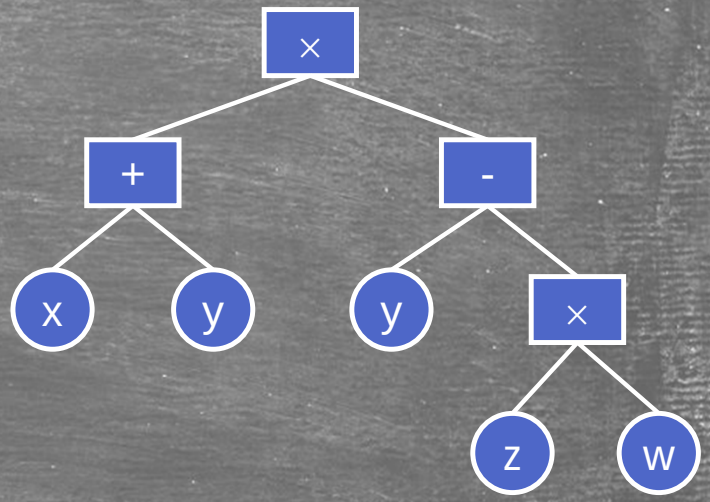
Expression trees

- Consider an algebraic expression:
  - (x+y) × (y-(z × w))
- We can represent this expression as a binary tree where:
  - The internal nodes are the operations;
  - The leaves are the variables.
- Thus:



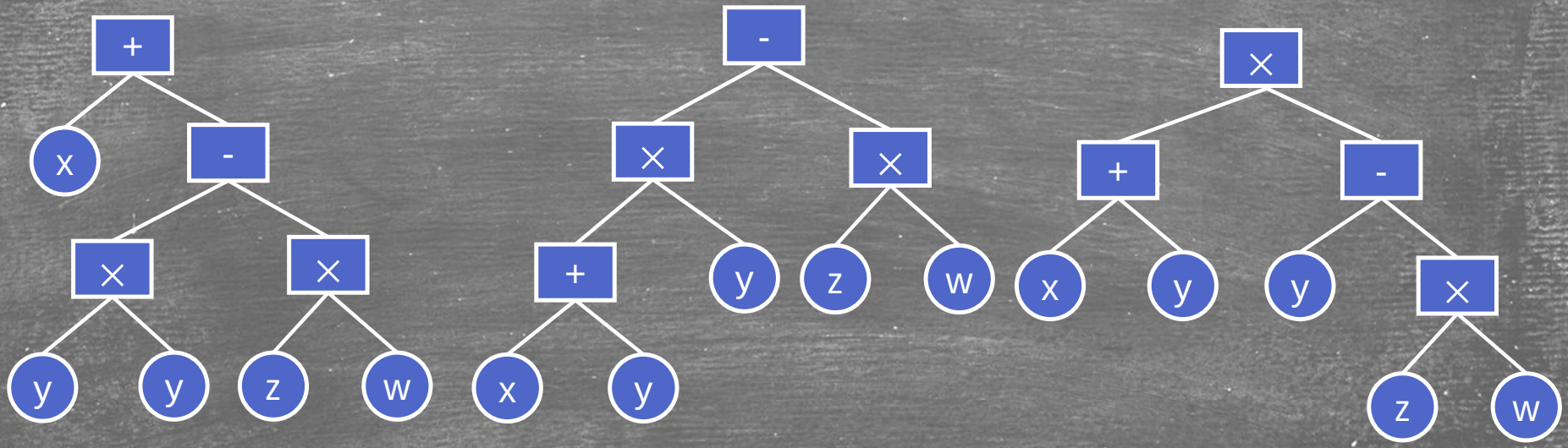
Traversal of an Expression Tree

- We can traverse an expression tree:
- Pre-order:
  - ×+xy-y×zw
- In-order
  - x+y×y-z×w
- Post-order
  - xy+yzw×-×
- Note that, although the in-order traversal almost produces the expression we started with, we have no parentheses.
  - The in-order traversal can represent many trees.



x+y×y-z×w (In-Order Traversal)

- This expression can represent...



- ...as well as many other trees
- We can modify the in-order traversal to insert parentheses:

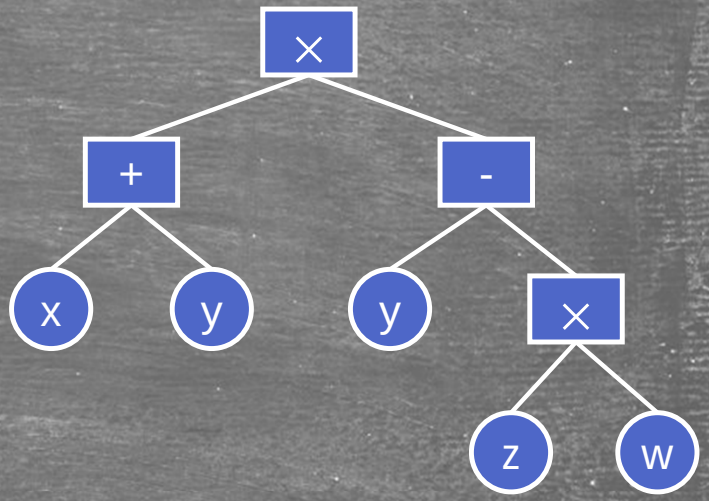
Fixing the Parentheses

```
procedure in_order_parentheses (node: ^tree_node)
    print “(“
    in_order_parentheses(node.left_child)
    print node.contents
    in_order_parentheses(node.right_child)
    print “)”
end
```

- Note: this may result in more parentheses than we started with, but the expression will be correct and match the original one.

xy+yzw×-× (Post-Order Traversal)

- The post-order traversal is equivalent to a unique expression tree;
  - The one we started with.
- We can use this fact to construct the expression tree from the post-order traversal.
- We will do this with the aid of a stack:



Stacking an expression

- The rules we follow are:
  1. Start with an empty stack
  2. Take each symbol of the expression, left to right:
     a) If the symbol is a letter:
        i. push it onto the stack
     b) If the symbol is an operator:
        I. Pop the top two elements of the stack as R and L in order
        II. Create a tree with the operator as root and L and R as the left and right children respectively.
- When the last symbol has been processed, the complete expression tree is the top of the stack.

An Example

- Consider the expression xy+yzw*-*:
- Moving from left to right:
- Symbol:
- x y + y z w * - *
- Stack:

- Note: we can do something similar by processing the pre-order traversal from end to start.

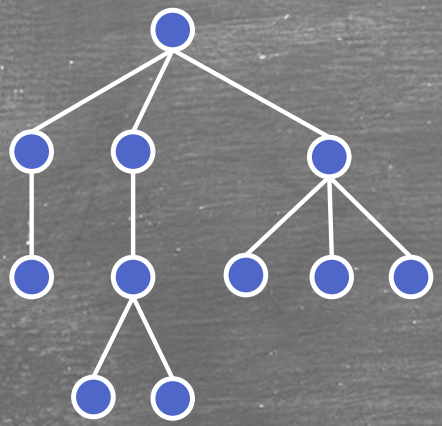
K-ary Trees

We have branches everywhere

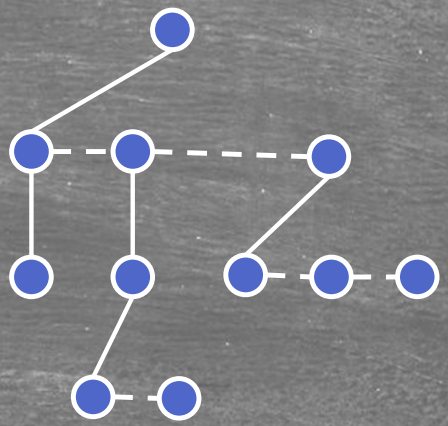
K-ary Trees

- How do we represent a tree that can have more than two child nodes?
- What if we allow trees of arbitrary degree?
- Must we use dynamic memory?
  - ```
    type K_node = record
        value: stuff
        children: array of ^K_node
    ```
  - Where children is an array of variable length.
    ```

K-ary Trees

- Consider another possible representation:
  - ```
    type K_node = record
        value: stuff
        child: ^K_node
        next:  ^K_node
    ```
  - Here:
    - **child** points to the first child of the current node
    - **next** points to the next sibling of the current node
  - We can see this best with an example:
    ```

- This K-ary tree:



is equivalent to this binary tree:



Finite State Machines

(Nothing to do with trees)

Finite State Machines

- This is a useful programming tool for processing streams of data.
- We define the process in terms of:
  1. A set of states that the program can be in.
  2. A set of possible inputs
  3. What action must be taken when we see a particular input and we are in a particular state.
     This includes the new state to change to.
- We can represent this as a table.

An Example: Getting Words

- In this example we will process characters one-at-a-time and assemble them into words.
- STATES:
  - We have two states:
    1. Waiting for a word to start
    2. Inside a word
- INPUTS:
  - We have three possible inputs:
    1. Letter (A..Z, a..z)
    2. Whitespace (space, tab, newline)
    3. Other (punctuation, etc.)

An Example: Getting Words

- We can represent the finite-state machine with a table:

State	Input	Process	New state
No word	Letter	Start a new word	In word
No word	Whitespace	Do nothing	No word
No word	Other	Do nothing	No word
In word	Letter	Add the letter to the word	In word
In word	Whitespace	Finish the current word	No word
In word	Other	Do nothing	In word

- Note: this table is simplified.
  - We should treat newline and end-of-file as separate inputs.