Representations for lists

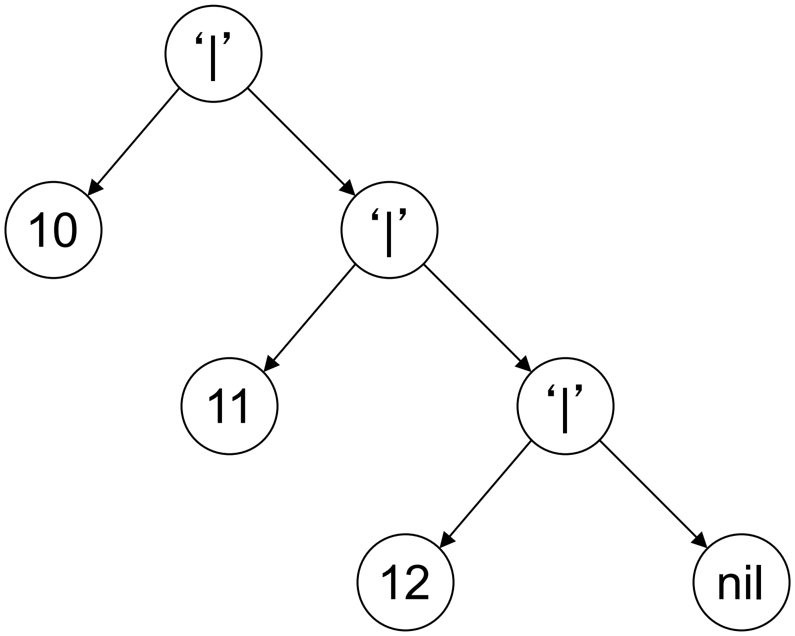
- The EBNF rule gives one textual representation
  - <List <Int>> ⇒
    10 | <List <Int>> ⇒
    10 | 11 | <List <Int>> ⇒
    10 | 11 | 12 | <List <Int>> ⇒
    10 | 11 | 12 | nil

We repeatedly replace the left-hand side of the rule by a possible value, until no more can be replaced

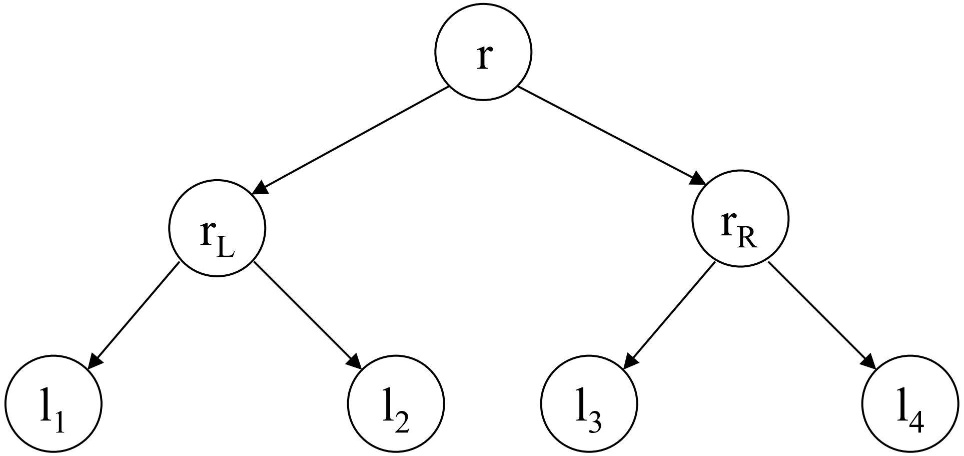
- Oz allows another textual representation
  - Bracket notation: [10 11 12]
  - In memory, [10 11 12] is identical to 10 | 11 | 12 | nil
  - Different textual representations of the same thing are called syntactic sugar

Graphical representation of a list

- Graphical representations are very useful for reasoning
  - Humans have very powerful visual reasoning abilities
- We start from the leftmost pair, namely 10 | <List <Int>>
  - We draw three nodes with arrows between them
  - We then replace the node <List <Int>> as before
- This is an example of a more general structure called a tree



Trees and binary trees



- A **tree** is either a leaf node (which is an empty tree) or a root node with arrows to a set of trees (called subtrees)
- A **binary tree** is a tree where all root nodes have exactly two subtrees (usually called left and right)