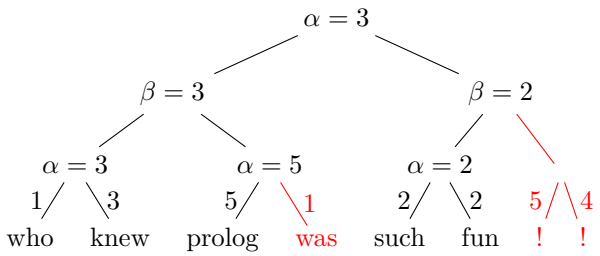
Declaratieve Talen

Prolog 1

1 Alpha-Beta Pruning



Figuur 1: The tree used in the exercise. Pruned branches are higlighted in red.

Assume that we have two players, both playing optimally, one player tries to maximise his score, while the other tries to minimise it. The maximising player plays first. This process can be represented by a minmax tree.

The goal of this exercise is to implement the alpha-beta pruning algorithm. The algorithm maintains two values while exploring the minmax tree: the α-value is the largest score that the maximising player is sure he can achieve, while the β value is the smallest score that the minimising player is sure he can achieve. If, while exploring some node, β becomes less than or equal to α, then further exploration of this node is aborted by the algorithm, as the other player will never pick this node, since he already has a better choice availble. For more information, illustrations and pseudo-code, please consult the relevant Wikipedia article.

For this exercise, represent trees with the following Prolog terms:

- `leaf(S,V)` representing a leaf node with score `S` and value `V`.
- `max(L,R)` for a max-node with subtrees `L` and `R`. `min(L,R)` for a min-node with subtrees `L` and `R`.

Now, implement a predicate `alpha_beta/5` that prunes the tree, by replacing pruned branches with `nil` as soon as $\beta \leq \alpha$. For the tree shown in Figure 1, the output is as follows:

```
% alpha_beta(Tree,Alpha,Beta,Score,NewTree)
?- alpha_beta(max(
               min(
                max(leaf(1,who),leaf(3,knew)),
                max(leaf(5,prolog),leaf(1,was))),
               min(
                max(leaf(2,such),leaf(2,fun)),
                max(leaf(5,!),leaf(5,!)))),
               -10,10,S,T).
S = 3,
T = max(min(max(leaf(1,who),leaf(3,knew)),max(leaf(5,prolog),nil)),
    min(max(leaf(2,such),leaf(2,fun)),nil)).
```

Extra Extend the internal `node` to have more than two subtrees, by using a list instead of two explicit left and right trees.