And - Tree Based Search

Search Process

Having defined the search model, we are now ready to define the Search Process and Search Instance.

The And_{tree} will begin with a single node, $s_0 = (pr, ?)$, and its expansion will be defined by the recursive relation Erw_{and}. Each iteration will use Div to expand the tree and create new nodes. After each Div, F_{bound} evaluates all solved leaves via a branch and bound operation, using a *beta* value. The search control uses this β value to prune once a solution has been found. After this, F_{leaf} evaluates all the leaves, and calculates a number, assessing each leaf. The search control will prioritize applying Div to the lowest value leaves first. The search control will choose the left most leaf in the case that F_{leaf} provides a tie between multiple leaves.

F_{leaf} evaluates a penalty score of an assignment, based on the soft and hard constraints. For partial assignments, $F_{leaf}{}^*$ is used, which uses $Eval^*$ and $Constr^*$ instead.

F_{leaf} : $\{pr_1, ..., pr_i, ..., pr_n\} \rightarrow \mathbb{R}$ where $1 \leq i \leq n$

$$\mathrm{F_{leaf}} = \left\{ \begin{array}{l} \infty\text{: if } Constr(pr_i) = \text{false} \\ Eval(pr_i)\text{: else} \end{array} \right\}$$

F_{bound} is used by the search control to keep the tree size within reason. It sets the β values of all complete solutions, when the state is $(pr, solved)$. We can define F_{bound} as follows: Once again we can use $F_{bound}{}^*$ to evaluate partial assignments.

F_{bound} : $\{pr_1, ..., pr_i, ..., pr_n\} \rightarrow pr_i$ where $1 \leq i \leq n$

$$\mathrm{F_{bound}} = \left\{ \begin{array}{l} \beta = \beta\text{: if } (pr_i, ?) \\ \beta = Eval(pr_i, ..., pr_n\text{: else} \end{array} \right\}$$

β_{best} is the smallest β value that F_{bound}.
if $\beta_{pr_i} \leq \beta_{best}$ then $\beta_{best} = \beta_{pr_i}$

As there is is only one Div relation, F_{trans} is not used.

There is no backtracking in this search control.

The search control operation operates in the following order:

1. Apply Div to the tree, prioritizing the branch with the lowest F_{leaf} value. In case of a tie, the left most branch is used.
2. Apply F_{bound}.
3. Apply F_{leaf}. If $F_{leaf} \geq \beta$, then prune the leaf.
4. We check states to see if they are solved: We mark a state as $(pr, solved)$ if $\forall X \in pr, X \neq \$$ and $F_{leaf}(pr_1, ..., pr_n) \neq \infty$
5. We choose the smallest F_{leaf} value of the leaves that are marked $(pr, ?)$ and apply Div again, starting the process over again.

Search Instance:
As before the initial search state is s_0:

$s_0 = pr =< X_1, ..., X_n >$ such that $\forall X_i \in pr, X_i = \$$ or s_0 is some partial assignment given as an input to the search.
We also set $\beta = \infty$.

The goal state is G_{and} is reached when all leaf nodes are marked with (pr, yes).

The optimal solution is the leaf node with the lowest F_{leaf} value.