And - Tree Based Search

Search Process

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

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} prunes leaves that are irrelevant for our search via a branch and bound operation, using a *beta* value. After this, F_{leaf} evaluates all the leaves, and calculates a number that will correspond to the state. 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} uses an additional helper function, $F_{penalty}$, which evaluates a penalty score of an assignment, based on the soft and hard constraints. For partial assignments, $F_{penalty}{}^*$ is used, which uses $Eval^*$ and $Constr^*$ instead. $F_{penalty}$ is used by both F_{leaf} and F_{bound}.

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

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

Using this we can define F_{leaf} :

$F_{leaf} : \{pr_1, ..., pr_n\} \rightarrow \mathbb{R}$
$F_{leaf} = (F_{penalty}(\{pr_1, ..., pr_n\}))$

F_{leaf} applies $F_{penalty}$ in order to calculate a numeric value for the search control.

F_{bound} is used by the search control to keep the tree size within reason. It uses β pruning to remove leaves that fail to beat the best found solution so far. We can define F_{bound} using F_{bound}. Once again we can use $F_{bound}{}^*$ to evaluate partial assignments using $F_{penalty}{}^*$.

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

$$\mathrm{F}_{\text{bound}} = \left\{ \begin{array}{l} \beta = \infty\text{: if } pr_i \in s_0 \\ \beta = F_{penalty}\text{: else} \end{array} \right\}$$

β_{best} is the smallest β value that F_{bound} or F_{bound} * has evaluated to.
if $\beta_{pr_i} \leq \beta_{best}$ then $\beta_{best} = \beta_{pr_i}$
if $\beta_{pr_i} > \beta_{best}$ then $pr_i = null$, pruning the leaf from the tree.

As there is is only one Div relation, F_{trans} is not used.
There is no backtracking in this search control.

A state is marked as *solved* when:

$\forall X \in Prob, X_i \cup \{\$\} = \emptyset$ where X_i is some X in $Prob$.
It is the state where each course and lab has a slot assigned to it. The state will take on the form $(pr, solved)$.

The search control operation operates in the following order:

1. Apply F_{leaf} to the tree.
2. 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. Div is applied to all unsolved branches $(pr, ?)$. It is at this point that all lower leaves are checked for $(pr, solved)$.
3. Apply F_{bound} to the tree, pruning the leaves that are out of bounds if needed.

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

$s_0 = pr =< X_1, ..., X_n >$ such that $\forall X_i \in pr, X_i = \$$

The goal state is G_{and} is reached when all branches are marked with $(pr, solved)$.