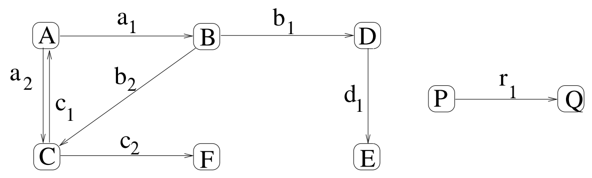
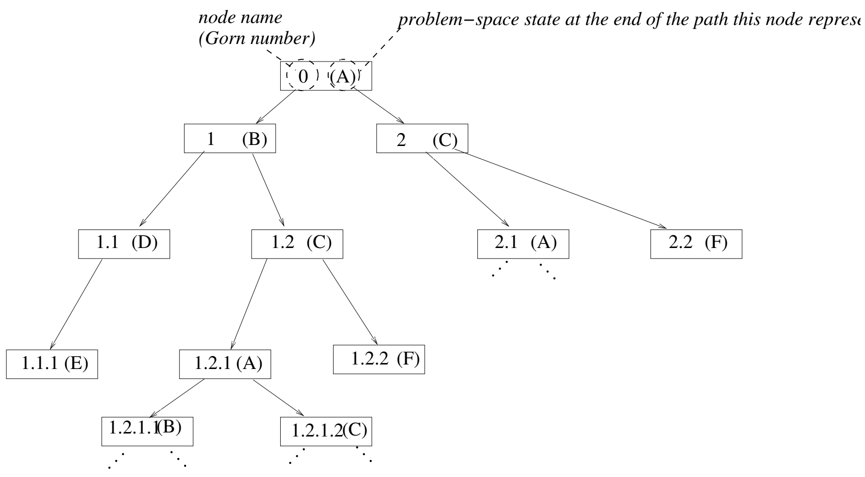
Computation, Information, and Intelligence (ENGRI/CS/INFO/COGST 172), Spring 2007
1/31/07: **Lecture aid — Path trees and depth-first search (DFS)**

Agenda: problem-solving as search; concepts regarding path trees, which explicitly represent all paths starting from the initial state within a given problem-space specification; depth-first search, which seeks a solution path within a (possibly infinite) path tree.
Follow-up to last time: As a self-test, see if you can figure out why there is a *two*-state legal, complete, well-defined specification for the course-requirements problem. (Hint: consider complex actions.)

I. Small problem-space specification A is the initial state, and P is the only goal state.



II. Corresponding infinite path tree



The precise formatting of information within nodes is not important. Here, we are using parentheses to reinforce the point that path-tree nodes are different than states in the corresponding problem space. In general, it is more convenient to omit the parentheses, as we did in Homework One. Also, we will generally omit action labels on path-tree edges ("arrows") for clarity.

III. Definitions regarding trees : Trees consist of ≥ 1 *nodes* (a.k.a. *vertices*, singular *vertex*) and *directed edges* ("arrows") (a.k.a. *edges*) between some pairs of nodes. The *root* node has no edges into it; for every other node, there is exactly one way to get from the root to it following edges in the proper direction.

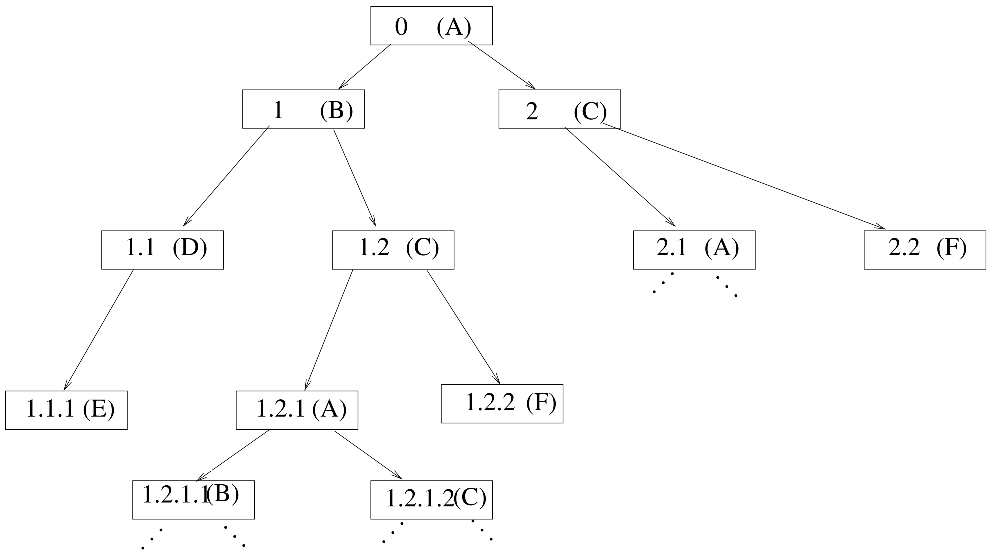
IV. Gorn numbering : the root's Gorn number is 0. The i^{th} child (i starting from 1) of the node with Gorn number j is $j.i$, except we omit the leading "0." prefix (note the decimal point) for convenience.
Note: treat the "dots" as digit separators. So, $2 > 1$ and $2.1 > 1.2$ and $3.12 > 3.3$ and $1.1.1 > 14.4$

(OVER)

V. Depth-first search

1. Mark node 0 visited (e.g., "v1").
2. Choose the **deepest (largest-Gorn-numbered)** visited, non-removed node n.
   (a) If n's label is a problem-space goal state, declare success and **stop**;
   (b) otherwise, if n's label repeats the label of a previously visited node or is childless, remove it and all its descendants;
   (c) otherwise, mark n's **leftmost (least-Gorn-numbered)** unvisited, non-removed child as visited.
3. If the tree still has nodes, repeat step 2.
4. If the entire tree has been removed, declare failure.

The same path tree as on the previous page



0 (A)
1 (B)
2 (C)
1.1 (D)
1.2 (C)
2.1 (A)
2.2 (F)
1.1.1 (E)
1.2.1 (A)
1.2.2 (F)
1.2.1.1 (B)
1.2.1.2 (C)