AND OR Graphs:.

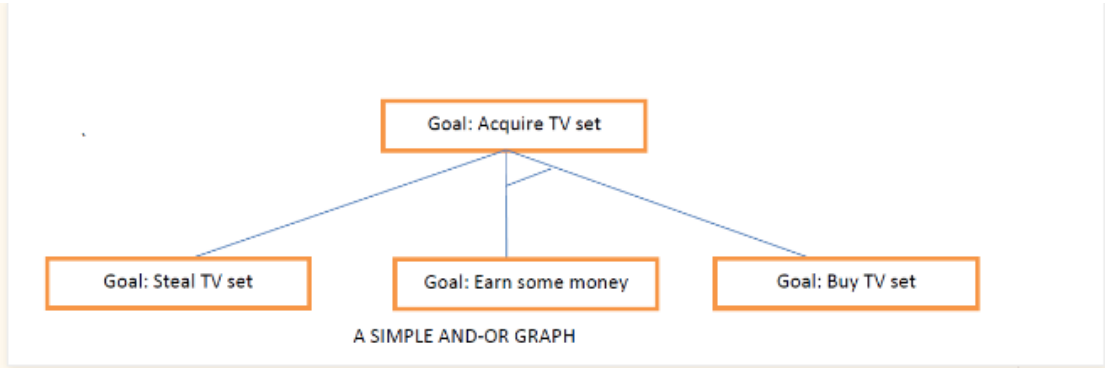
PROBLEM REDUCTION:

So far we have considered search strategies for OR graphs through which we want to find a single path to a goal. Such structure represent the fact that we know how to get from anode to a goal state if we can discover how to get from that node to a goal state along any one of the branches leaving it.

AND-OR GRAPHS

The AND-OR GRAPH (or tree) is useful for representing the solution of problems that can solved by decomposing them into a set of smaller problems, all of which must then be solved. This decomposition, or reduction, generates arcs that we call AND arcs. One AND arc may point to any number of successor nodes, all of which must be solved in order for the arc to point to a solution. Just as in an OR graph, several arcs may emerge from a single node, indicating a variety of ways in which the original problem might be solved. This is why the structure is called not simply an AND-graph but rather an AND-OR graph (which also happens to be an AND-OR tree)

EXAMPLE FOR AND-OR GRAPH



A SIMPLE AND-OR GRAPH

ALGORITHM:

Let G be a graph with only starting node INIT.

Repeat the followings until INIT is labeled SOLVED or h(INIT) > FUTILITY

a) Select an unexpanded node from the most promising path from INIT (call it NODE)

b) Generate successors of NODE. If there are none, set h(NODE) = FUTILITY (i.e., NODE is unsolvable); otherwise for each SUCCESSOR that is not an ancestor of NODE do the following:

i. Add SUCCESSSOR to G.

ii. If SUCCESSOR is a terminal node, label it SOLVED and set h(SUCCESSOR) = 0.

iii. If SUCCESSPR is not a terminal node, compute its h

c) Propagate the newly discovered information up the graph by doing the following: let S be set of SOLVED nodes or nodes whose h values have been changed and need to have values propagated back to their parents. Initialize S to Node. Until S is empty repeat the followings:

i. Remove a node from S and call it CURRENT.

ii. Compute the cost of each of the arcs emerging from CURRENT. Assign minimum cost of its successors as its h.

iii. Mark the best path out of CURRENT by marking the arc that had the minimum cost in step ii

iv. Mark CURRENT as SOLVED if all of the nodes connected to it through new labeled arc have been labeled SOLVED

v. If CURRENT has been labeled SOLVED or its cost was just changed, propagate its new cost back up through the graph. So add all of the ancestors of CURRENT to S.