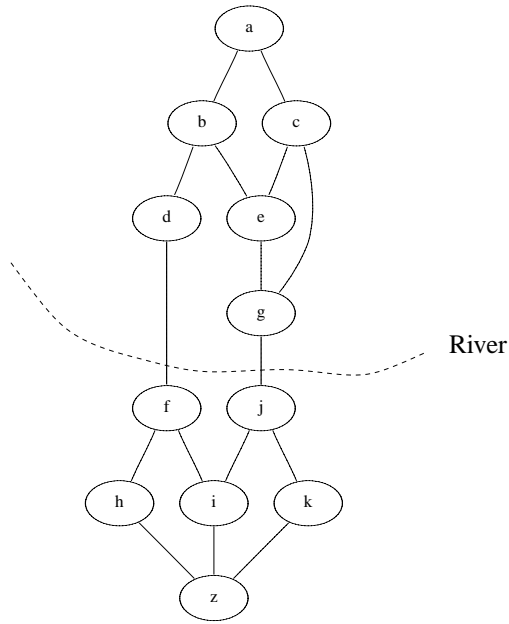


# Problem Reduction (And/Or Graphs, see Bratko)

## Decomposable problem:



1

**Idea:** any solution has to use a bridge

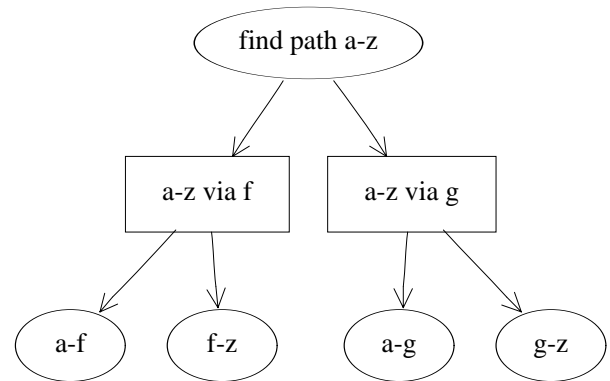
**I.e.:** for path a-z find  
 path a-z via f  
 or path a-z via g

## Decomposition:

for path a-z via f find  
 path a-f  
 and path f-z

analogously for g

**And/Or Graph:** or-nodes as ellipses, and-nodes as boxes



2

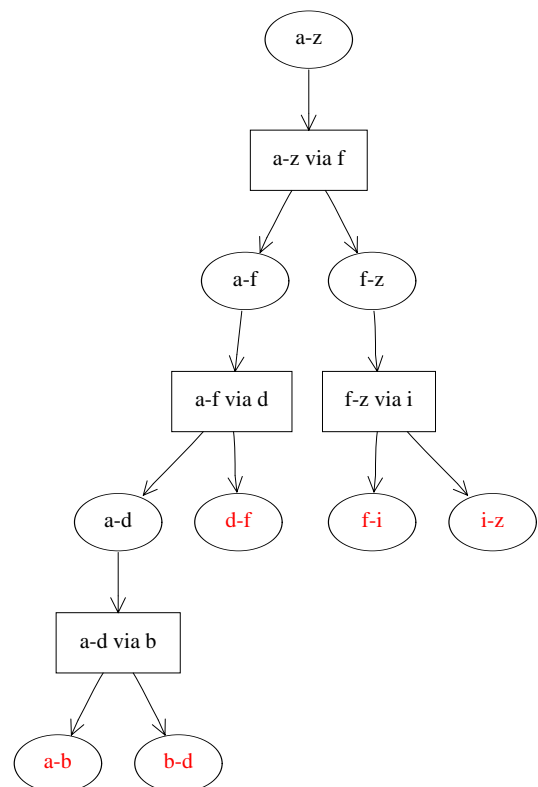
## And/Or Graphs (contd.)

**Note:** goal "nodes" are subproblems that are trivial or atomic, e.g. direct route from a-c

## Solution Tree:

- the problem is the root node of the solution tree  $\mathcal{T}$
- if  $P$  is an *or* node, exactly one successor (with its solution tree) is in  $\mathcal{T}$
- if  $P$  is an *and* node, all of its successors (with their solution trees) are in  $\mathcal{T}$

## Example



3

4

## Costs

**Costs:** for this, one has to consider

- arc costs
- node costs

**Calculation:**

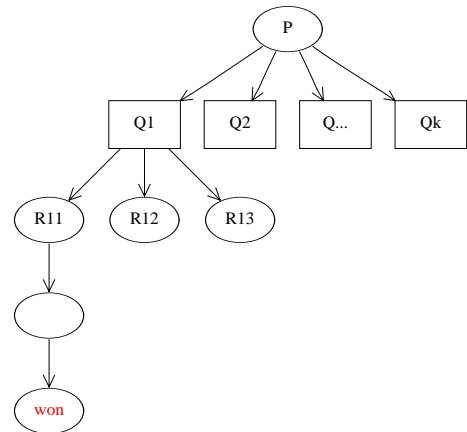
or nodes of form  $X-Z$

and nodes of form  $X-Z$  via  $Y$

- primitive if  $X-Z$  connected
- in that case, cost of node  $X-Z$  is relative distance between  $X$  and  $Z$
- cost of other nodes is 0

**End Games:** consider

- game with only *win/loss*
- 2 players  $a$  and  $b$
- playing alternatively
- solution: win for  $a$



**Interpretation:** game is won if solution tree exists, i.e. tree begins with an

**or node:** there is a choice for  $a$  leading to an

**and node:** such that all possible choices for  $b$  lead to

**or node:** and so on until

**Goal:** successful solution (win) is found

5

6

## Interpretation

**It means:**  $a$  has won (solution tree) if it is either in a *winning position* or it can always choose a move leading to a *losing position* of  $b$ ; i.e. a position such that all moves that  $b$  can choose lead to a winning position of  $a$  (i.e. again to a solution tree).

**Note:**  $a$  does not have to have a solution tree. Either  $b$  could have a solution tree (in which  $a$  loses) or neither of them have, so none of the players can force a win.

## Endgame Algorithm

**Endgame Algorithm:** for  $a$

1. consider final (0-step) winning positions for  $a$
2. compute 1-step losing positions for  $b$ , i.e. all positions for  $b$  from which *all* immediate successors lead to a 0-step winning position for  $a$
3. compute 2-step winning positions for  $a$ , i.e. all positions where  $a$  can choose *one* immediate successor to lead to a 1-step losing position for  $b$
4. compute 3-step losing positions for  $b$ , i.e. all positions for  $b$  where *all* successors lead to a less-than-3 (i.e. 2- or 0-) winning position for  $a$ .
5. and so on, until no more new positions are collected or maximum depth are exhausted

**Result:** if no maximum depth limit, the final outcome is a list of winning positions for  $a$  (with maximum depths), a list of losing positions for  $b$  (with maximum depths) and a list of tied positions

7

8