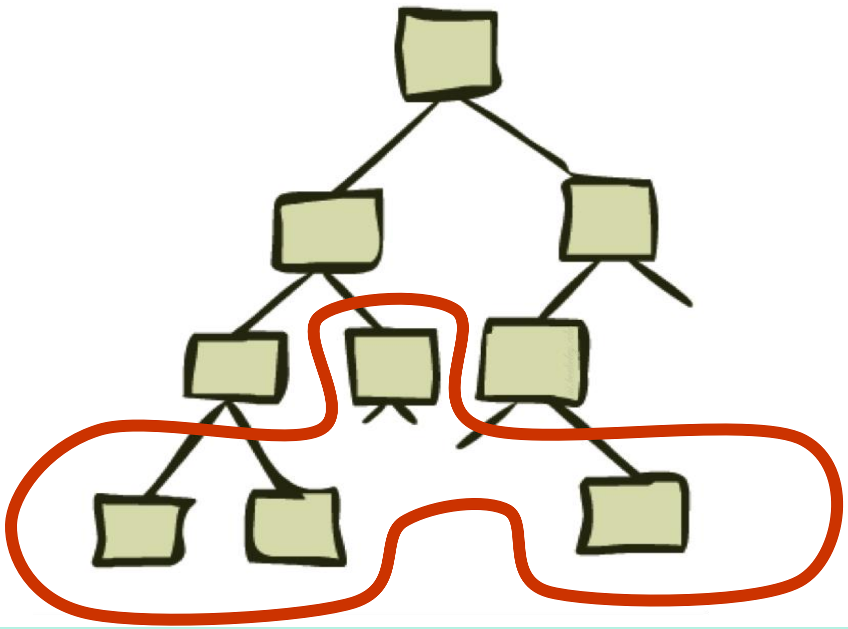
Advanced Topics in AI

Tree search and Graph search



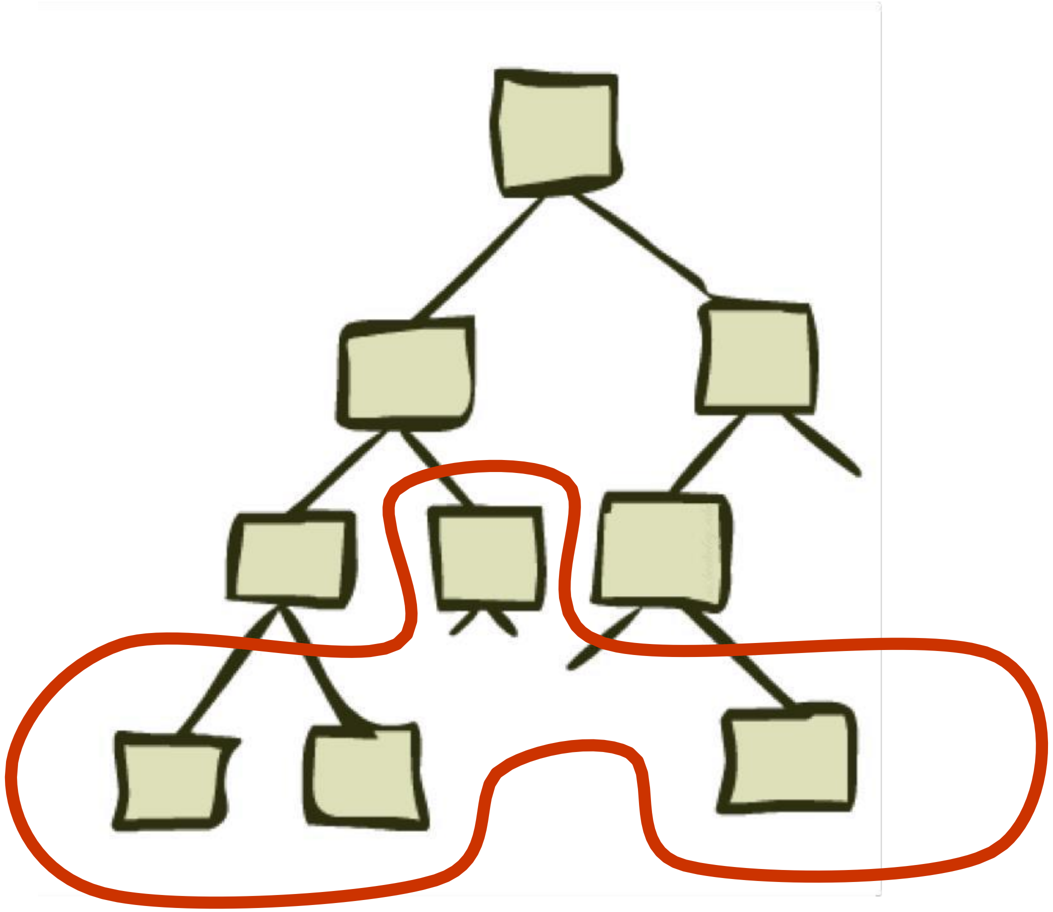
Instructor: Prof. Dr. techn. Wolfgang Nejdl

Leibniz University Hannover

[These slides were created by Dan Klein and Pieter Abbeel for CS188 Intro to AI at UC Berkeley. All CS188 materials are available at http://ai.berkeley.edu.]

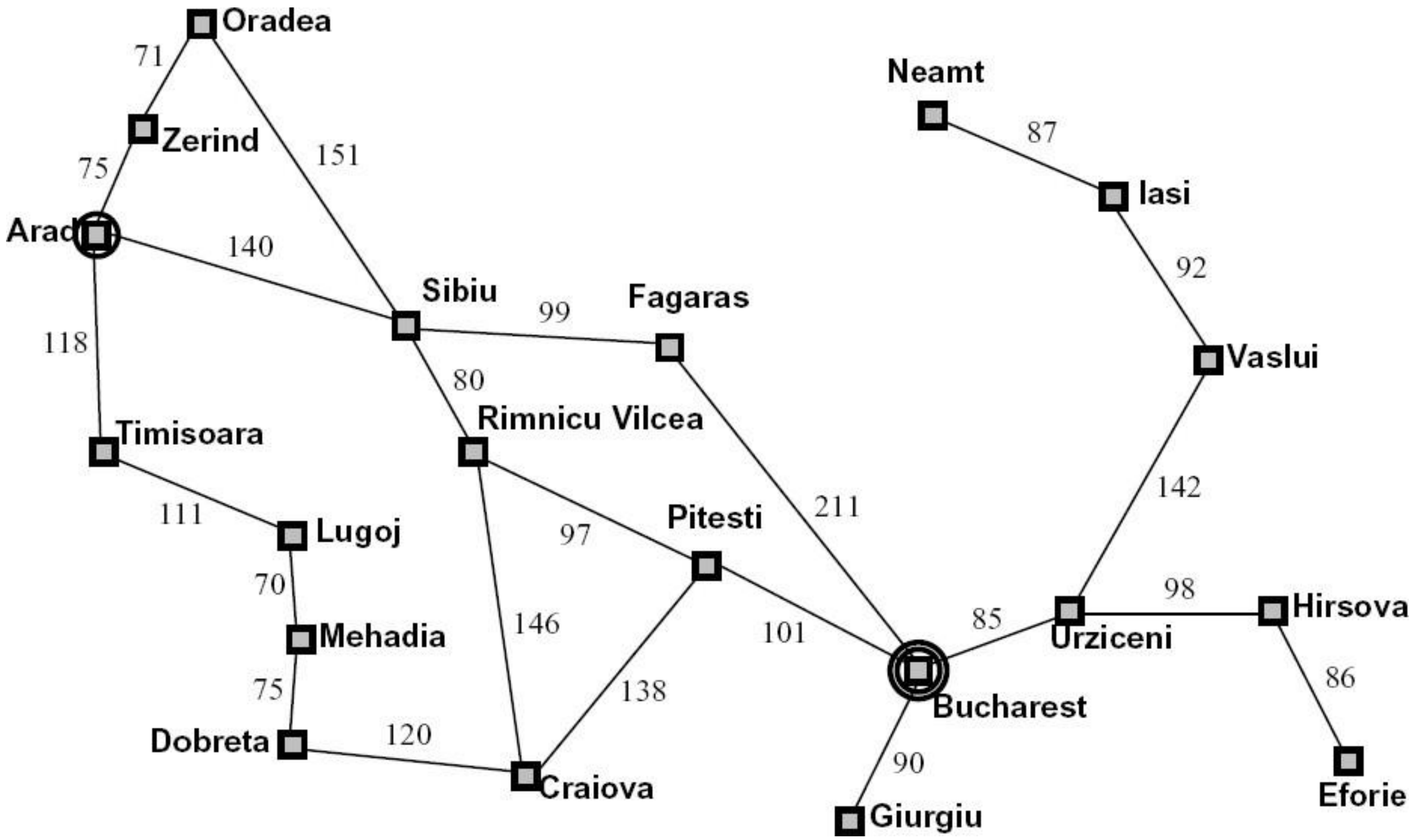
Co-financed by the Connecting Europe Facility of the European Union

Tree Search



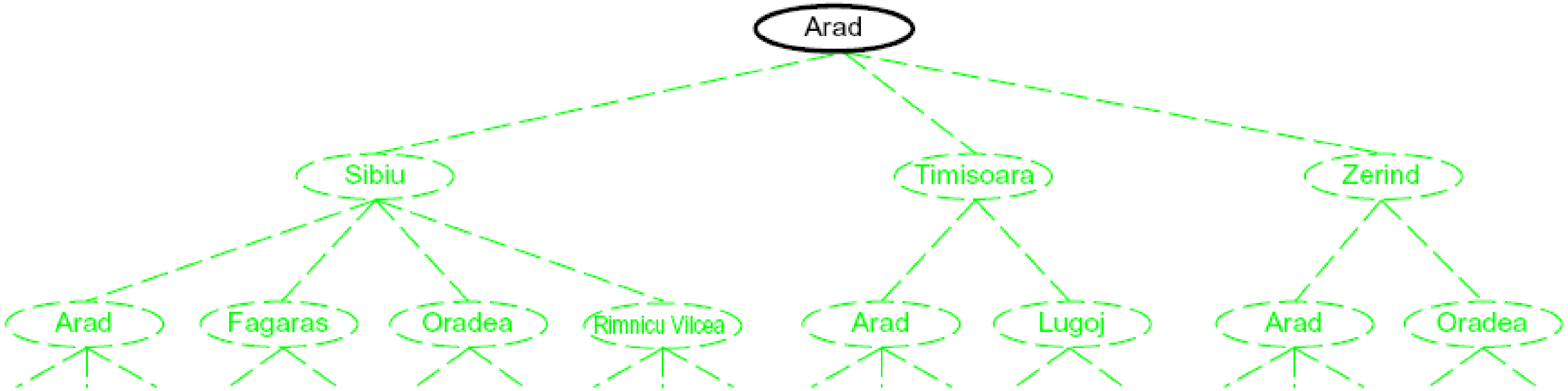
Co-financed by the Connecting Europe Facility of the European Union

Search Example: Romania



Co-financed by the Connecting Europe Facility of the European Union

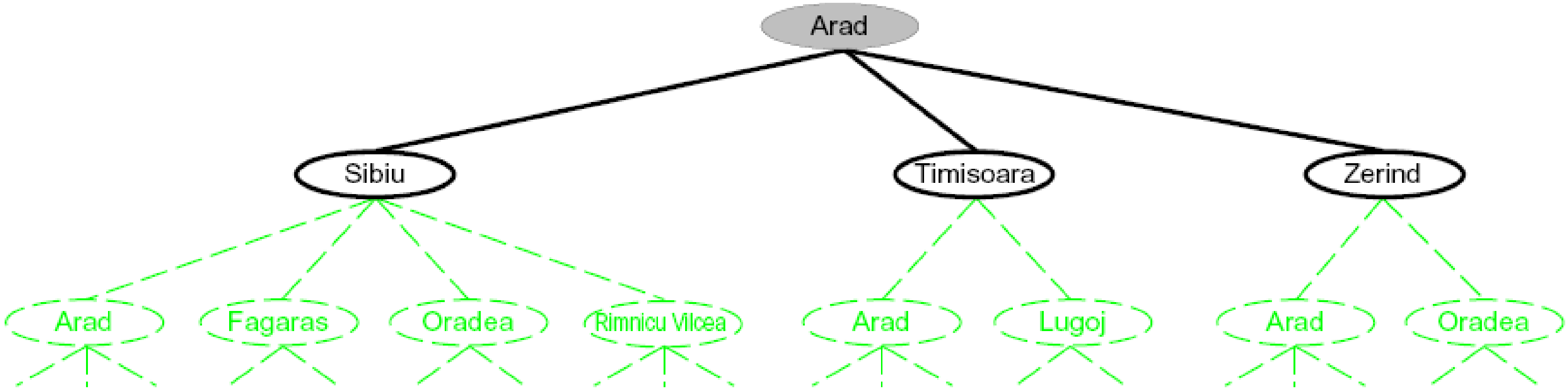
Searching with a Search Tree



- Search:
  - Expand out potential plans (tree nodes)
  - Maintain a fringe of partial plans under consideration
  - Try to expand as few tree nodes as possible

Co-financed by the Connecting Europe Facility of the European Union

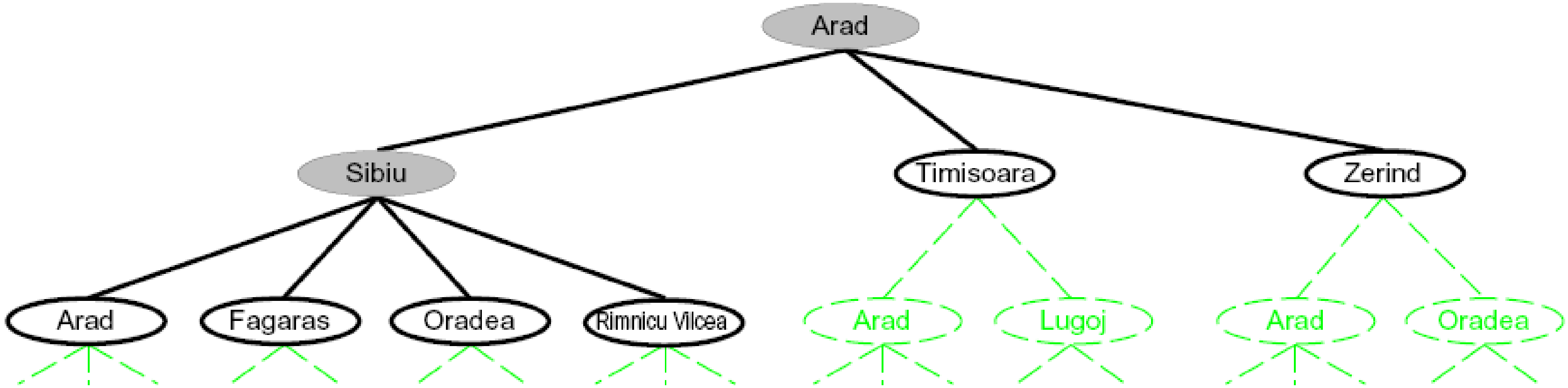
Searching with a Search Tree



- Search:
  - Expand out potential plans (tree nodes)
  - Maintain a fringe of partial plans under consideration
  - Try to expand as few tree nodes as possible

Co-financed by the Connecting Europe Facility of the European Union

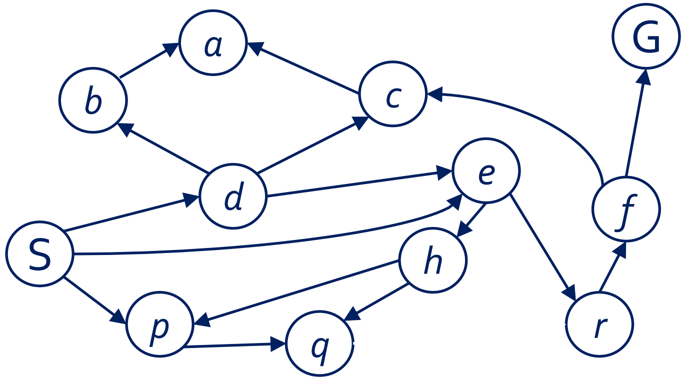
Searching with a Search Tree



- Search:
  - Expand out potential plans (tree nodes)
  - Maintain a fringe of partial plans under consideration
  - Try to expand as few tree nodes as possible

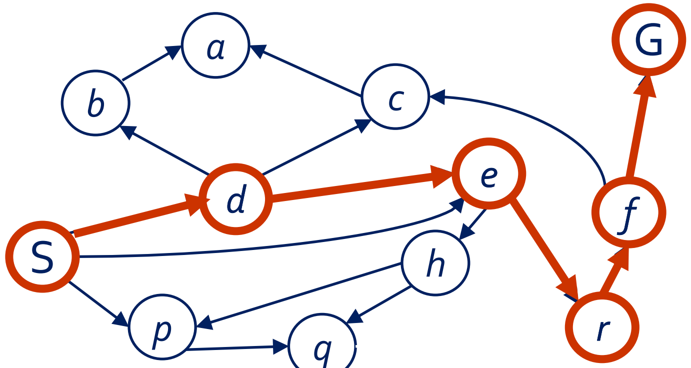
Co-financed by the Connecting Europe Facility of the European Union

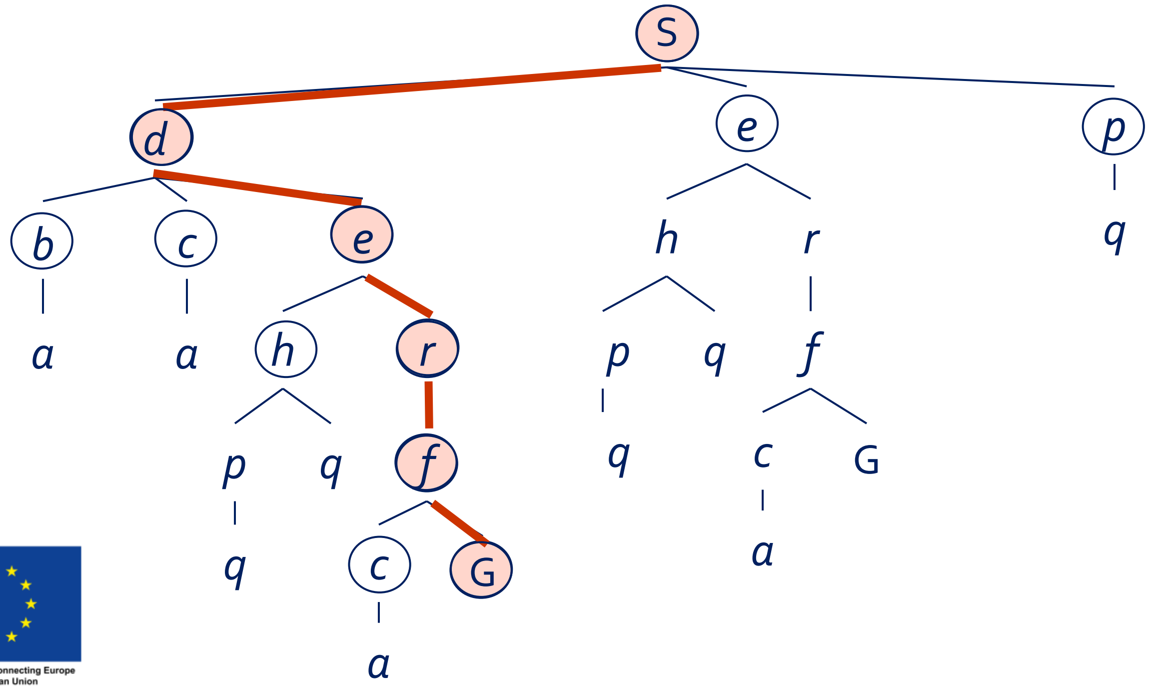
Example: Tree Search



Co-financed by the Connecting Europe Facility of the European Union

Example: Tree Search





~~s~~

~~s → d~~

s → e

s → p

s → d → b

s → d → c

~~s → d → e~~

s → d → e → h

~~s → d → e → r~~

~~s → d → e → r → f~~

s → d → e → r → f → c

~~s → d → e → r → f → G~~

Co-financed by the Connecting Europe Facility of the European Union

Graph Search

- Idea: never expand a state twice
- How to implement:
  - Tree search + set of expanded states ("closed set")
  - Expand the search tree node-by-node, but…
  - Before expanding a node, check to make sure its state has never been expanded before
  - If not new, skip it, if new add to closed set
- Important: store the closed set as a set, not a list
- Can graph search wreck completeness? Why/why not?
- How about optimality?

Co-financed by the Connecting Europe Facility of the European Union

Tree Search Pseudo-Code

```
function TREE-SEARCH (problem, fringe) return a solution, or a failure
    fringe ← INSERT(MAKE-NODE(INITIAL-STATE[problem]), fringe)
    loop do
        if fringe is empty then return failure
        node ← REMOVE-FRONT(fringe)
        if GOAL-TEST(problem, STATE[node]) then return node
        for child-node in EXPAND(STATE[node], problem), do
            fringe ← INSERT(child-node, fringe)
        end
    end
```

- Important ideas:
  - Fringe
  - Expansion
  - Exploration strategy
- **Main question**: which fringe nodes to explore?

Co-financed by the Connecting Europe Facility of the European Union

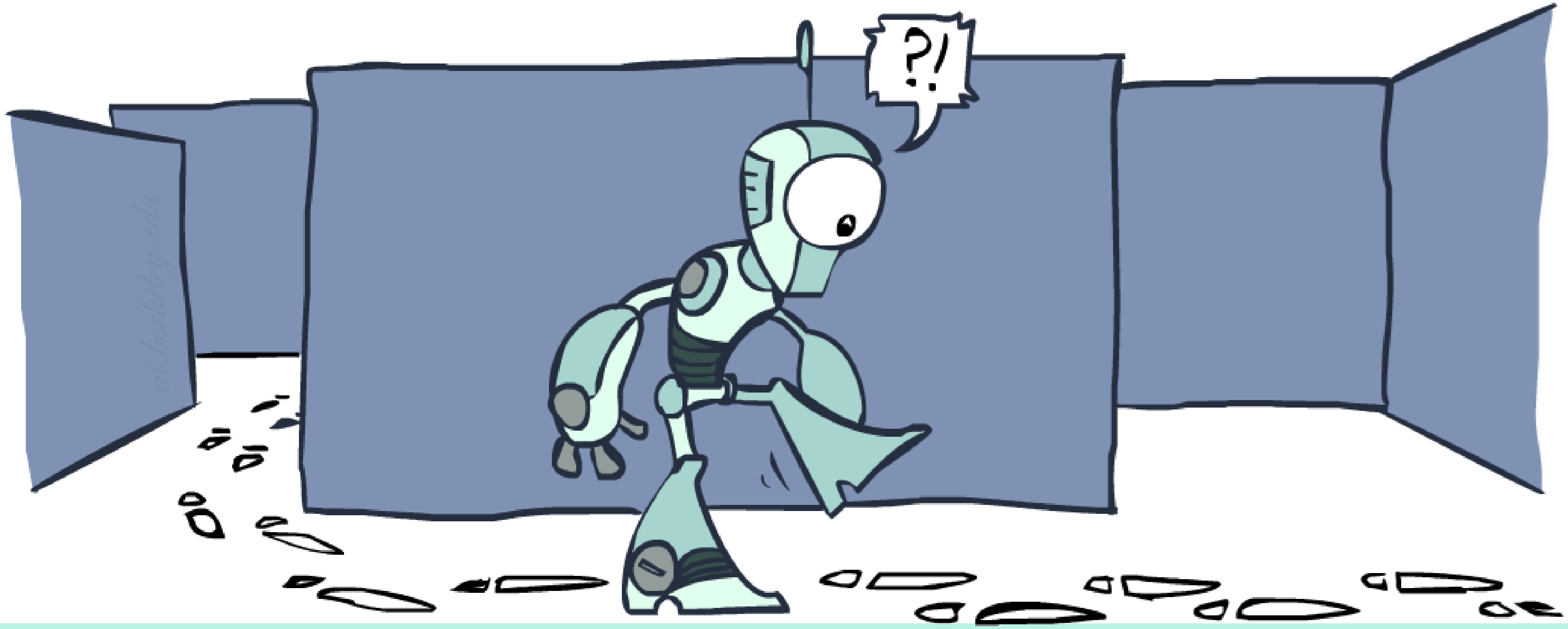
Graph Search Pseudo-Code

function GRAPH-SEARCH (*problem*, *fringe*) **return** a solution, or a failure
    closed ← an empty set
    fringe ← INSERT(MAKE-NODE(INITIAL-STATE[*problem*]), *fringe*)
    loop do
        if *fringe* is empty **then return** failure
        node ← REMOVE-FRONT(*fringe*)
        if GOAL-TEST(*problem*, STATE[*node*]) **then return** *node*
        if STATE[*node*] is not in *closed* **then**
            add STATE[*node*] to *closed*
            for *child-node* in EXPAND(STATE[*node*], *problem*), **do**
                fringe ← INSERT(*child-node*, *fringe*)
            end
    end

Co-financed by the Connecting Europe Facility of the European Union

Advanced Topics in AI

Next: DFS and BFS



Instructor: Prof. Dr. techn. Wolfgang Nejdl

Leibniz University Hannover

Co-financed by the Connecting Europe Facility of the European Union