Artificial Intelligence
[week #2]

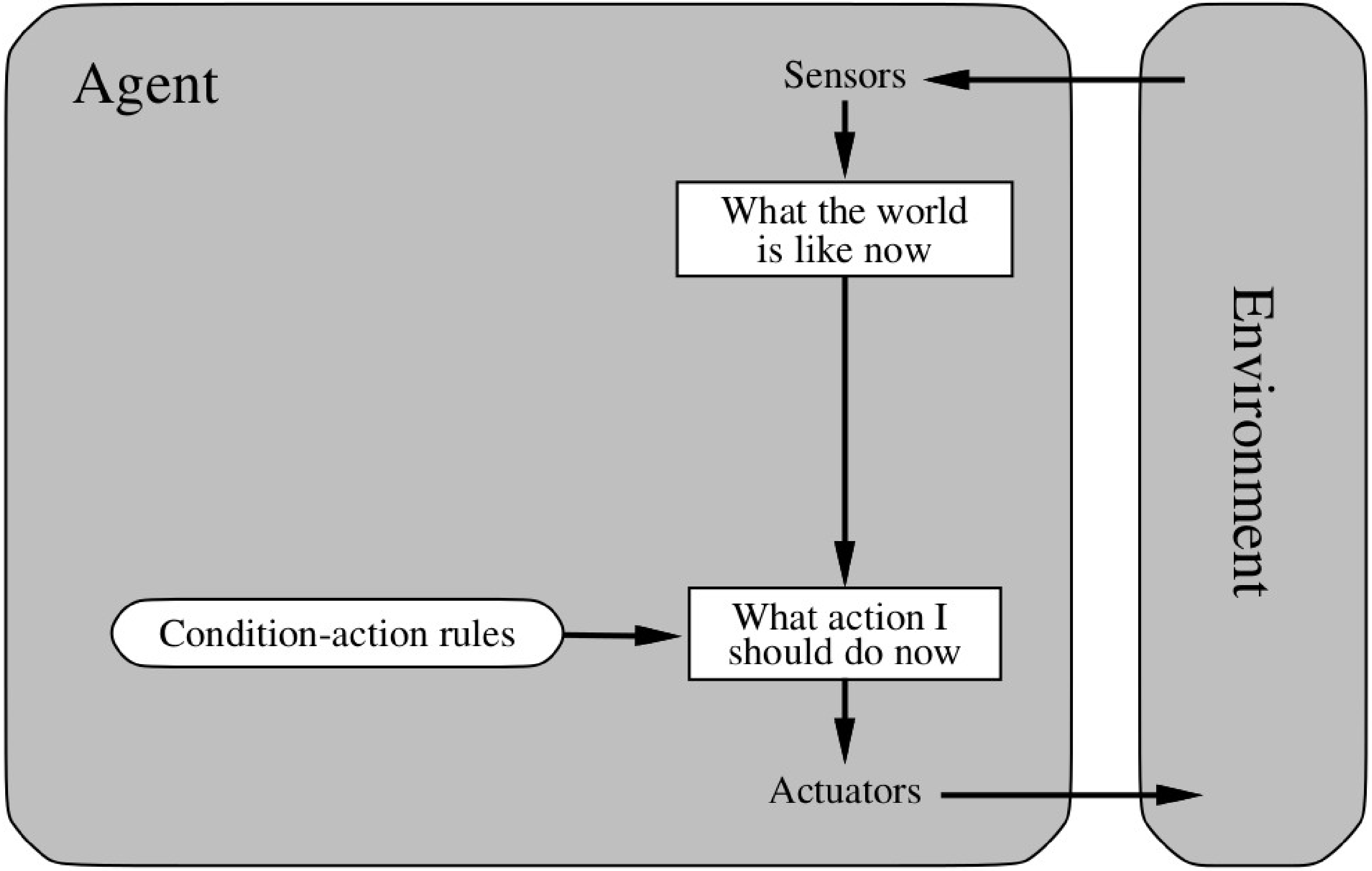
State Space Search

Hilmy. A. T
hilmi.tawakal@gmail.com

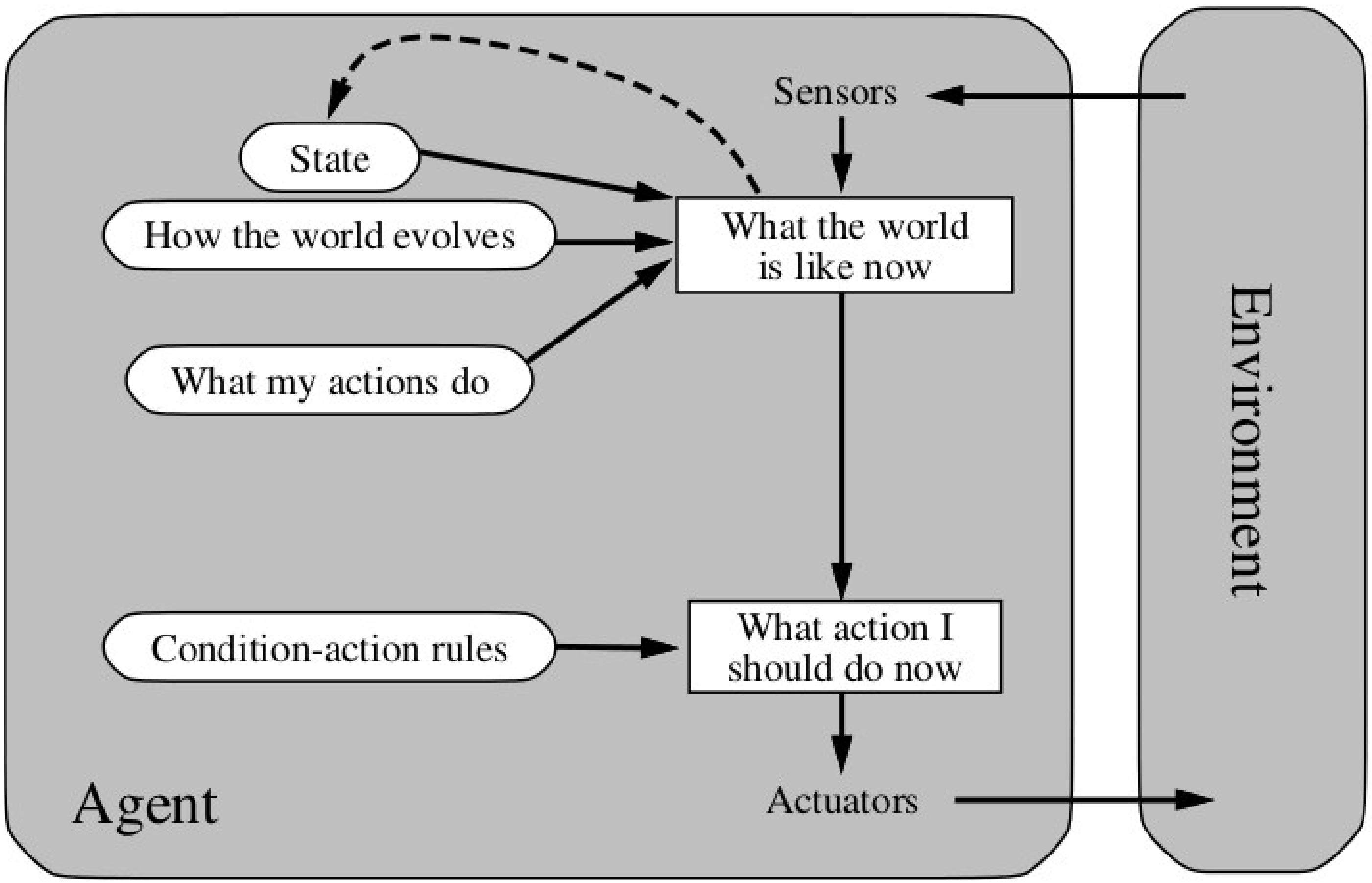
Agent types

- Simple reflex agents: based on last perception
- Model-based reflex agents: have internal representation about environment
- Goal-based agents: have information about goal, and choose action to achieve goal
- Utility-based agents: utility function → quantitative score about environment (performance measure)
- Learning agents: learn to improve performance

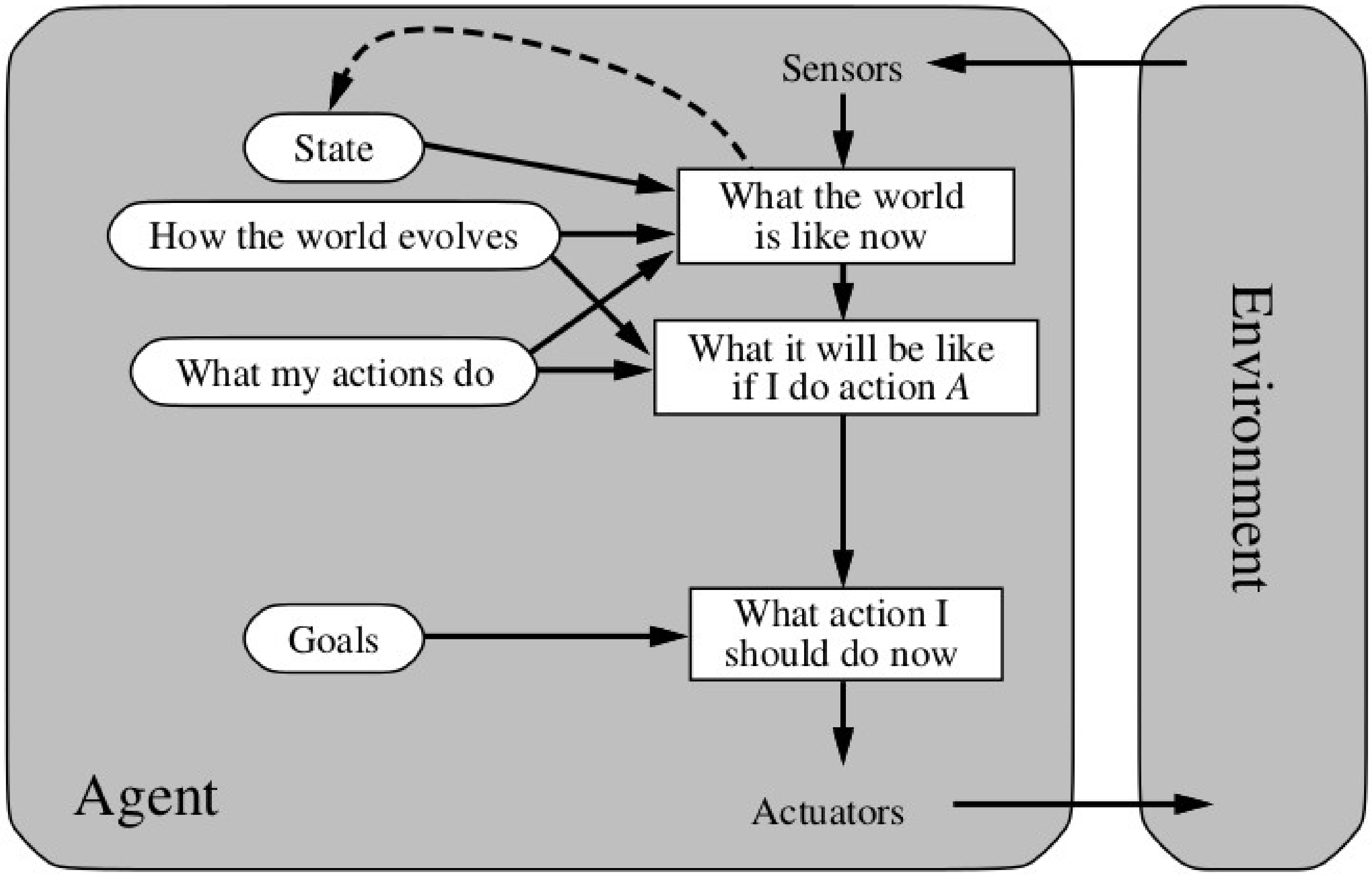
Simple reflex agents



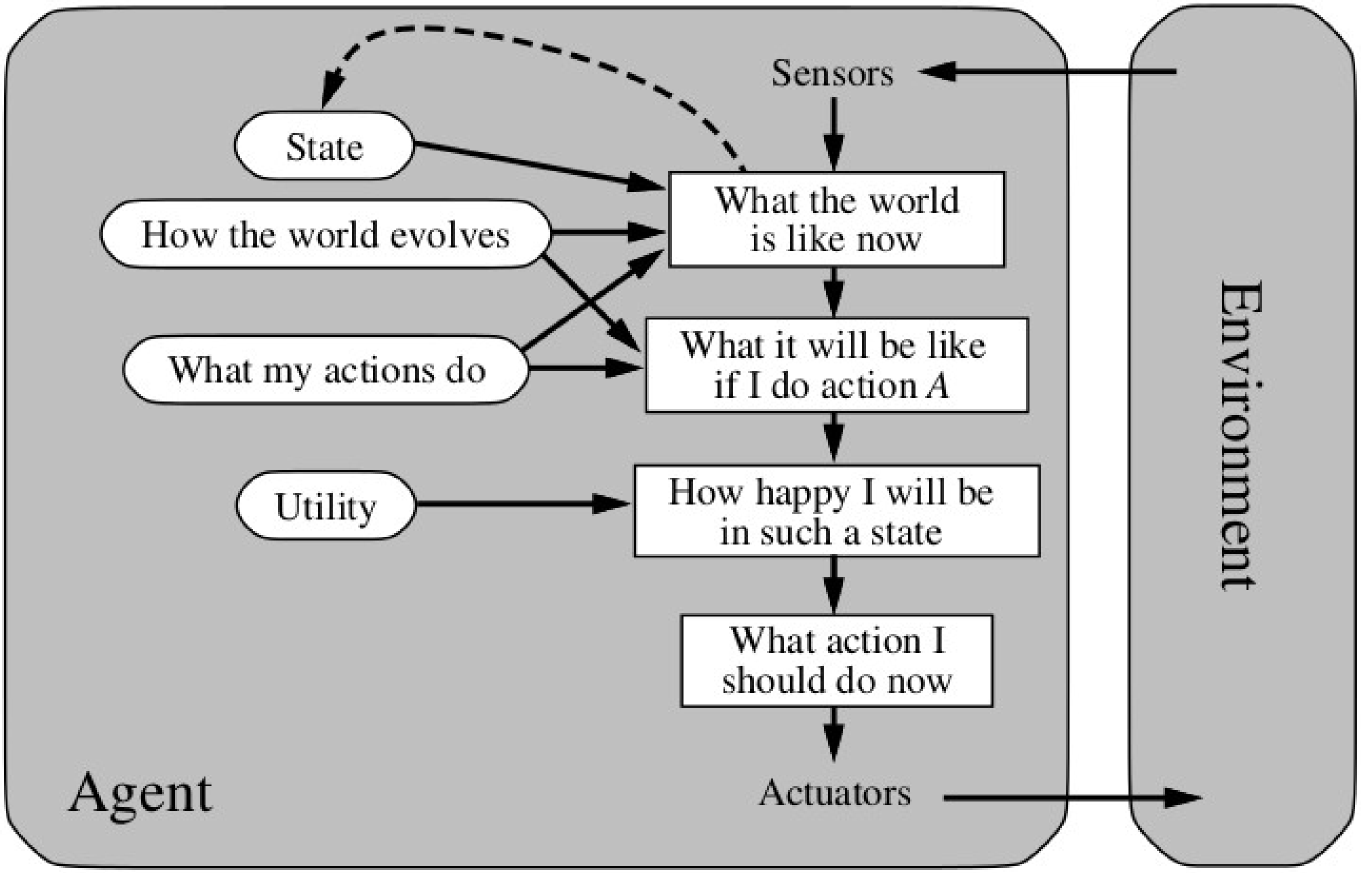
Model-based agents



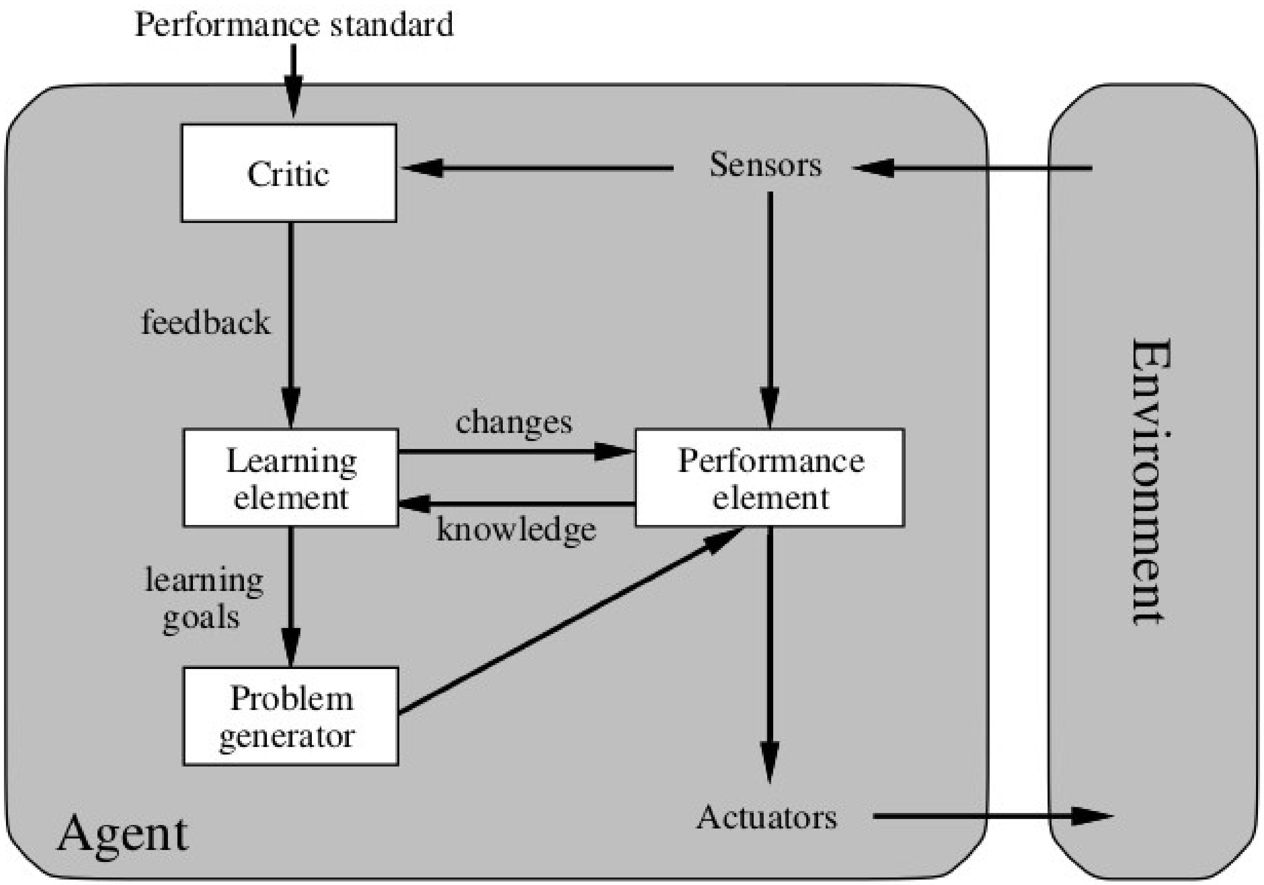
Goal-based agents



Utility-based agents



Learning agents



Problem solving agents

- Goal-based agents
- Have goal → evaluate actions and choose the best
- Create solutions → series of actions → complete goal
- What problem? → what solution?

Problem solving agents

- Goal formulation
- Problem formulation → action and state
- Search solution
- Execution

Example: trip to bogor

- Goal formulation: at Bogor
- Problem Formulation:
  - Action: Drive from town to town
  - State: towns between depok and bogor
- Search solution: series of towns from depok to bogor

Problem Formulation

- Initial state: beingAt(depok)
- Possible Actions: drive(depok,citayam)
- Successor function S: define state X, actions and state

  X = beingAt(depok)

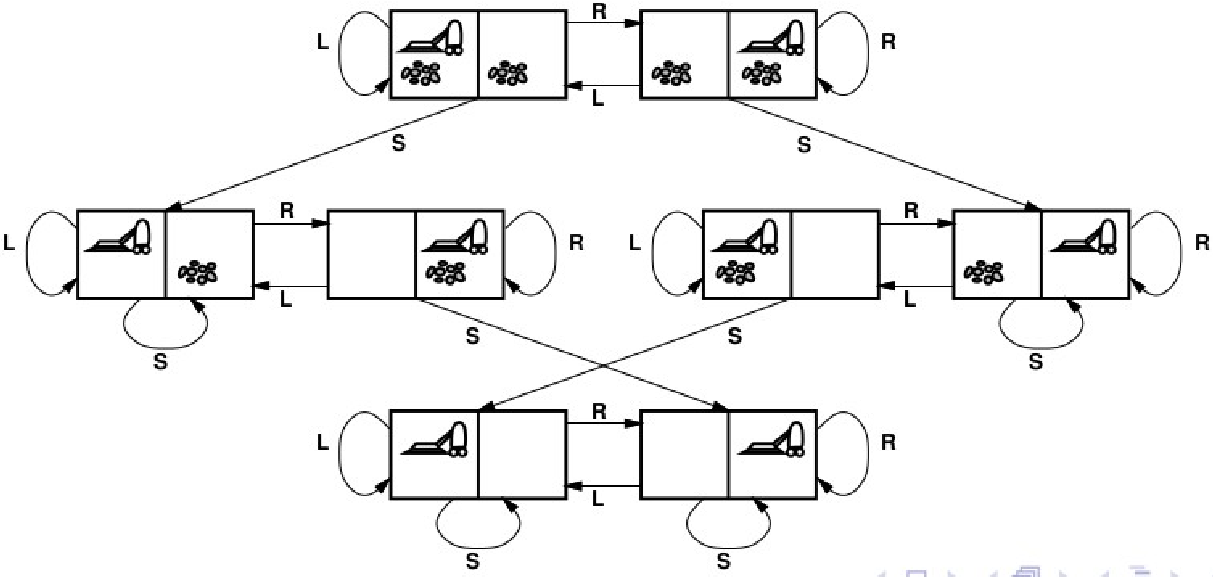
  S(X)={<drive(depok,citayam),beingAt(citayam)>, . . . }
- Initial state and Successor → state space → all state from initial state → graph → Path: series of state (connected by actions)

Traversing State space

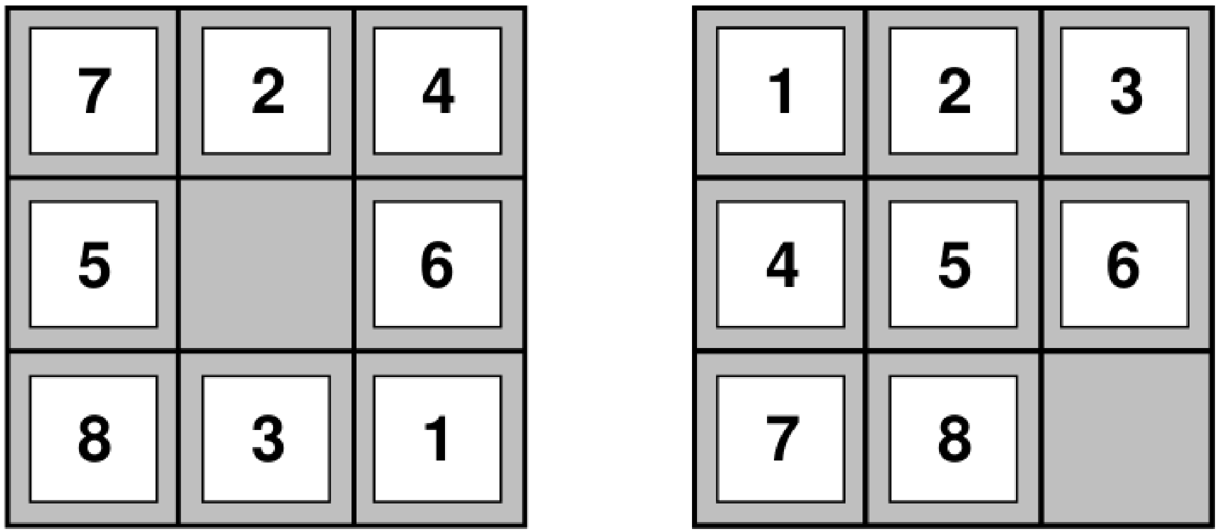
- Goal test: current state is the goal?
  - explicit: goal state (beingAt(bogor))
  - implicit: goal descriptions (check mate)
- Path cost function: function to calculate numeric value of each path → performance measure
- Solution: path from initial state to goal
- Optimal solution: solution with lowest path cost function

Example: vacuum cleaner

- State: agent location
- Possible action:doLeft(),doRight(),doSuck()
- Goal test: all rooms clean?
- Path cost: number of step in path



Example:8-puzzle

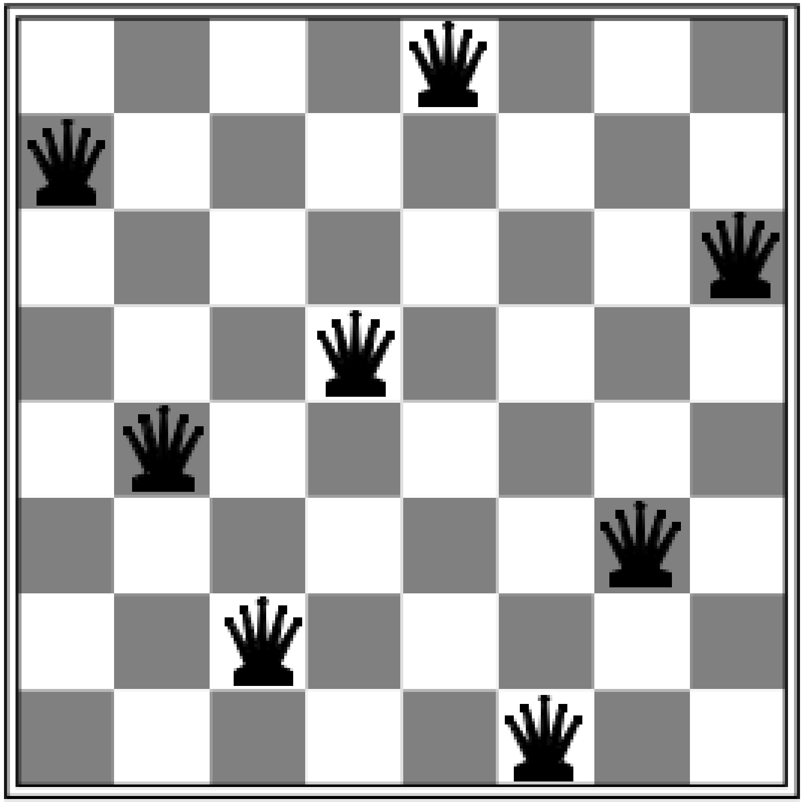


Start State

Goal State

- State?
- Possible action?
- Goal Test?
- Path cost?

Example: 8-Queens Problem



- State: chess with n queens
- Initial State: empty board
- Possible action: put queen on the board
- Goal test: 8 queens in the board, no overtaking

Example: 8-Queens Problem

- With above problem definition:

  $64 \times 63 \times 62 \times \ldots \times 57 = 1.8\text{x}10^{14}$ path!

- Too complex, impossible to solve
- Alternative:

  - state: 8 queens in board, one per column at first n columns

  - possible action:put queen on empty leftmost column

  - now state space 2057 !!

Find Solution From Search Tree

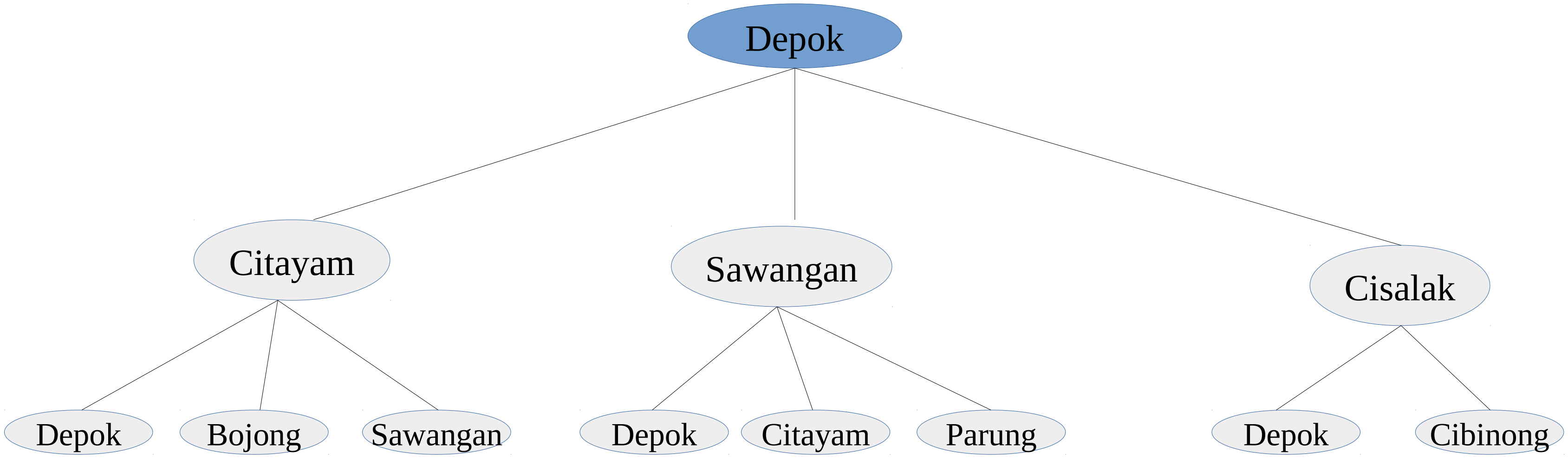
- After problem definition → find solution with search algorithm
- Search tree → representation of state space
- Search tree: collection nodes

  → node: representation of state in path

  → have child, parent, depth, path cost
- Root node → initial state
- Node expansion: implement successor function to node→ produce new child
- Fringle: node that not expand yet

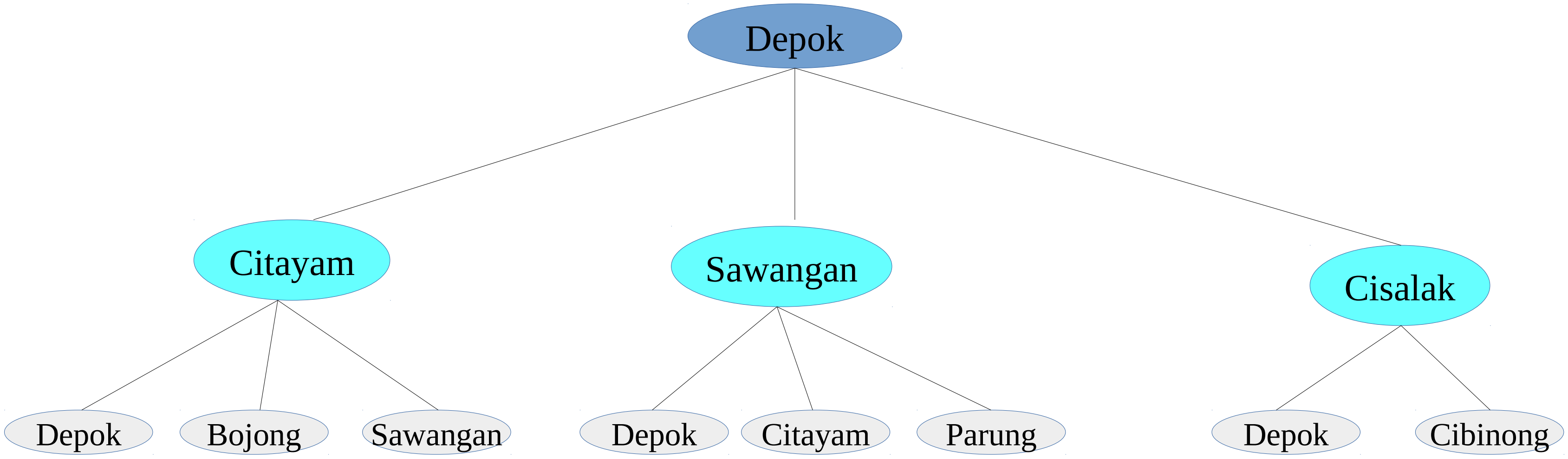
Example

- Root node (depok) as current node



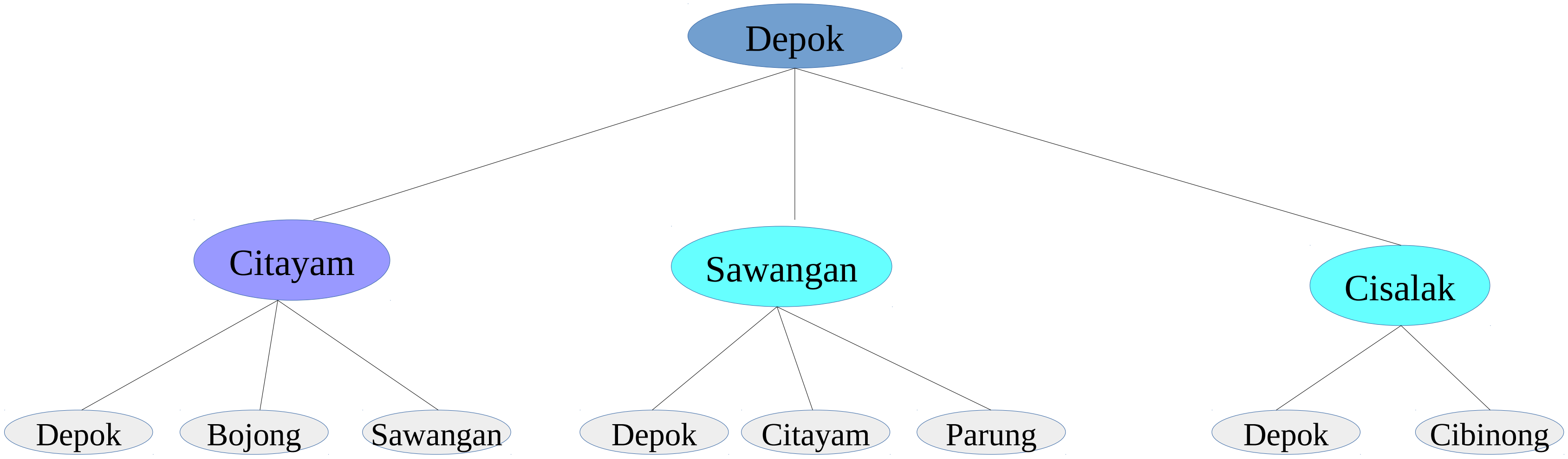
Example

- Root node (depok) as current node
- Expand node



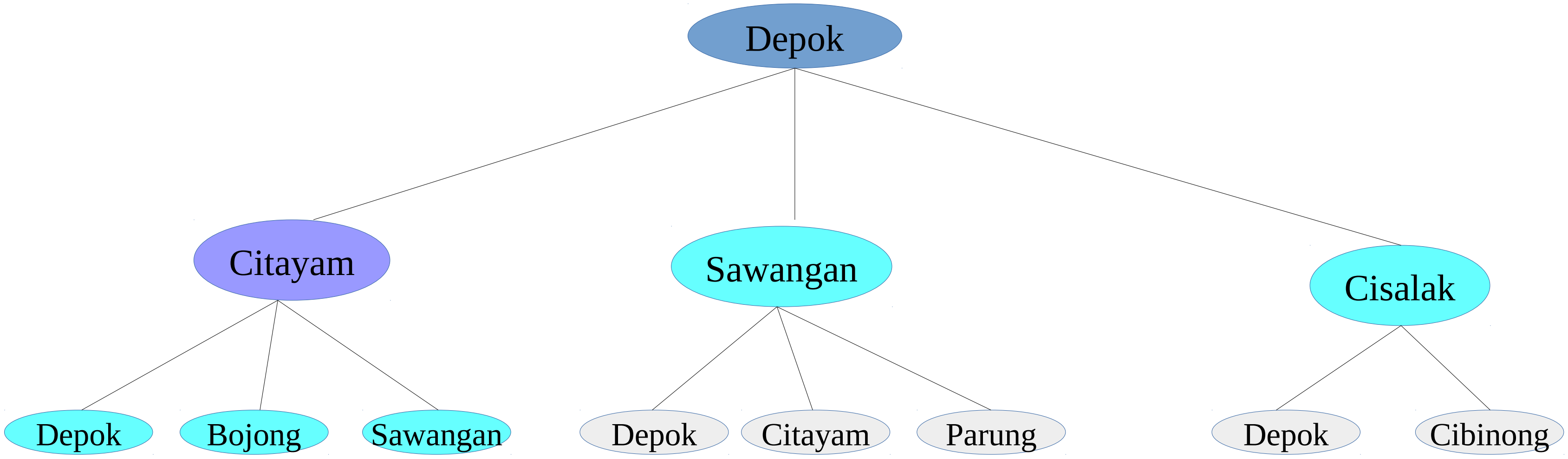
Example

- Root node (depok) as current node
- Expand node
- Choose one node as current node

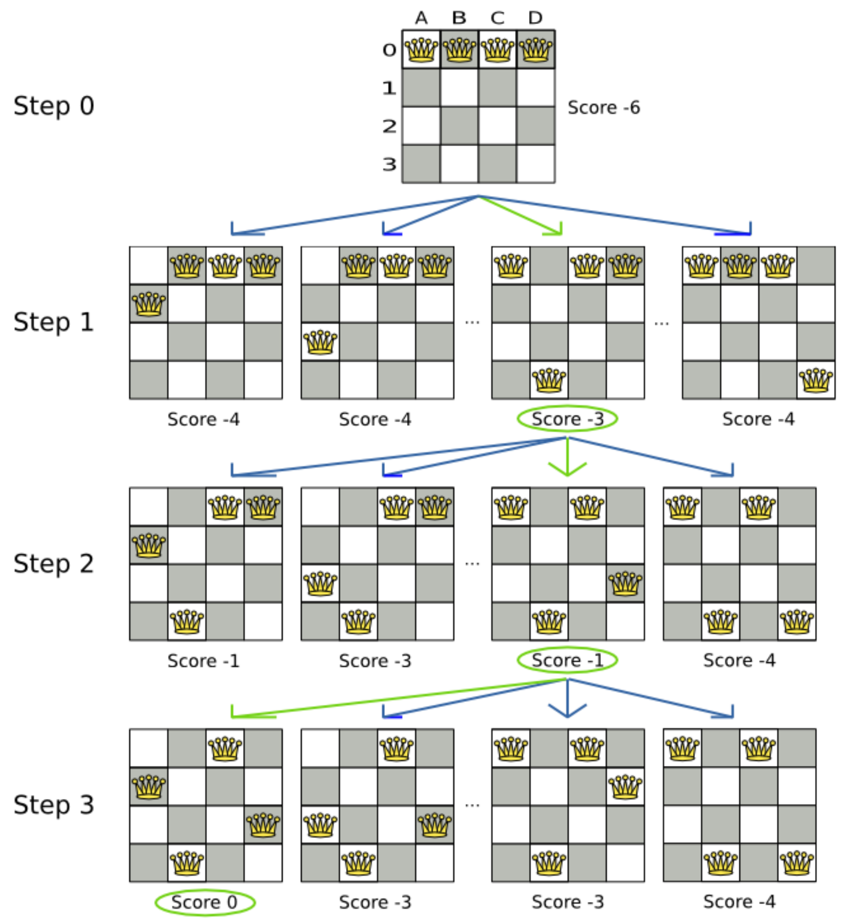


Example

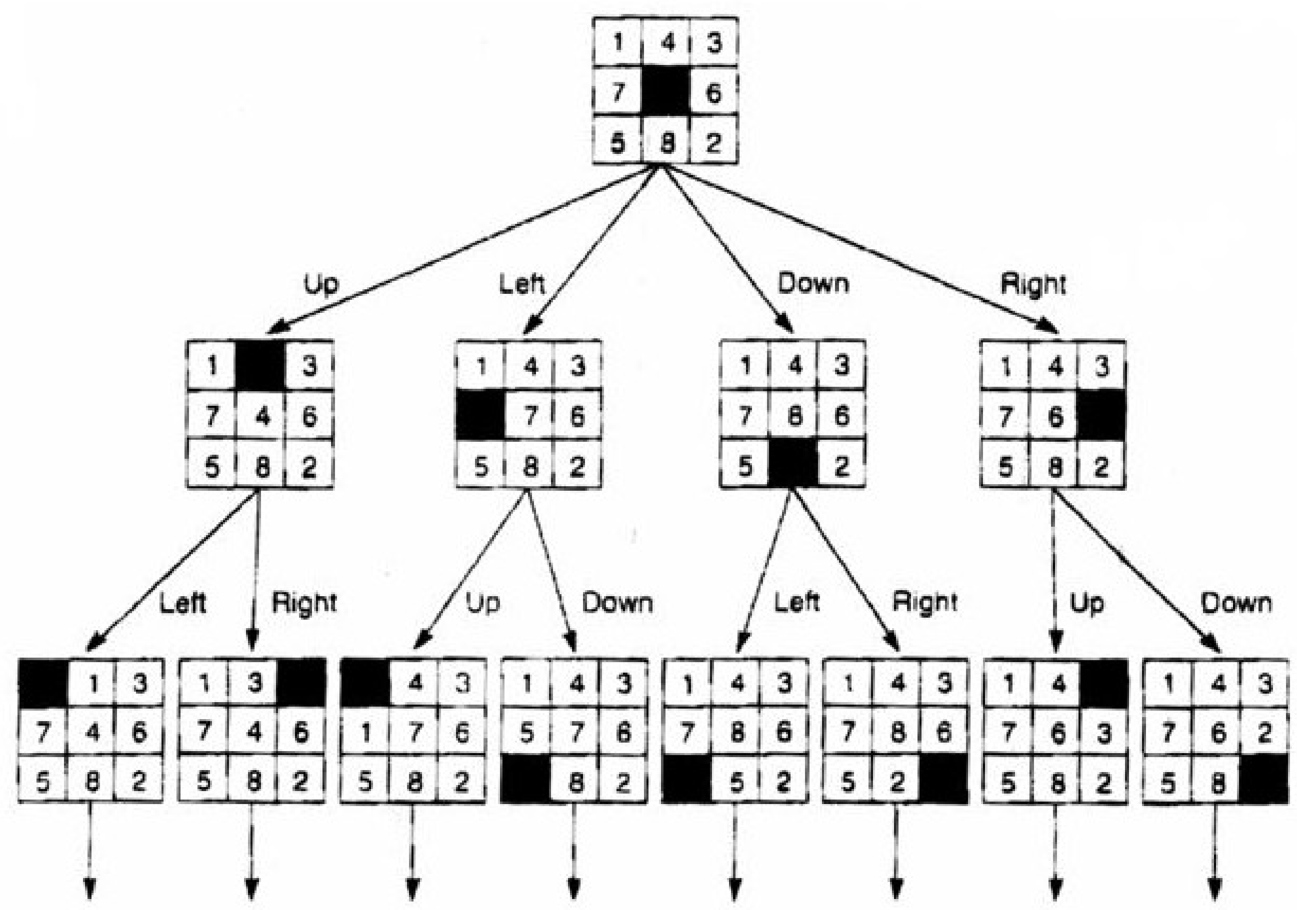
- Root node (depok) as current node
- Expand node
- Choose one node as current node
- Do previous step



Example queens problem



Example 8-puzzle



Search strategy

- Different in node expansion
- Strategy evaluation:

  - completeness
  - time complexity
  - space complexity
  - optimality

- Time and space complexity, measured by:

  - b → branching factor
  - d → depth of optimal solution
  - m → maximum depth

Search strategy

- Breadth-first search
- Uniform-cost search
- Depth-first search
- Depth-limited search
- Iterative-deepening search