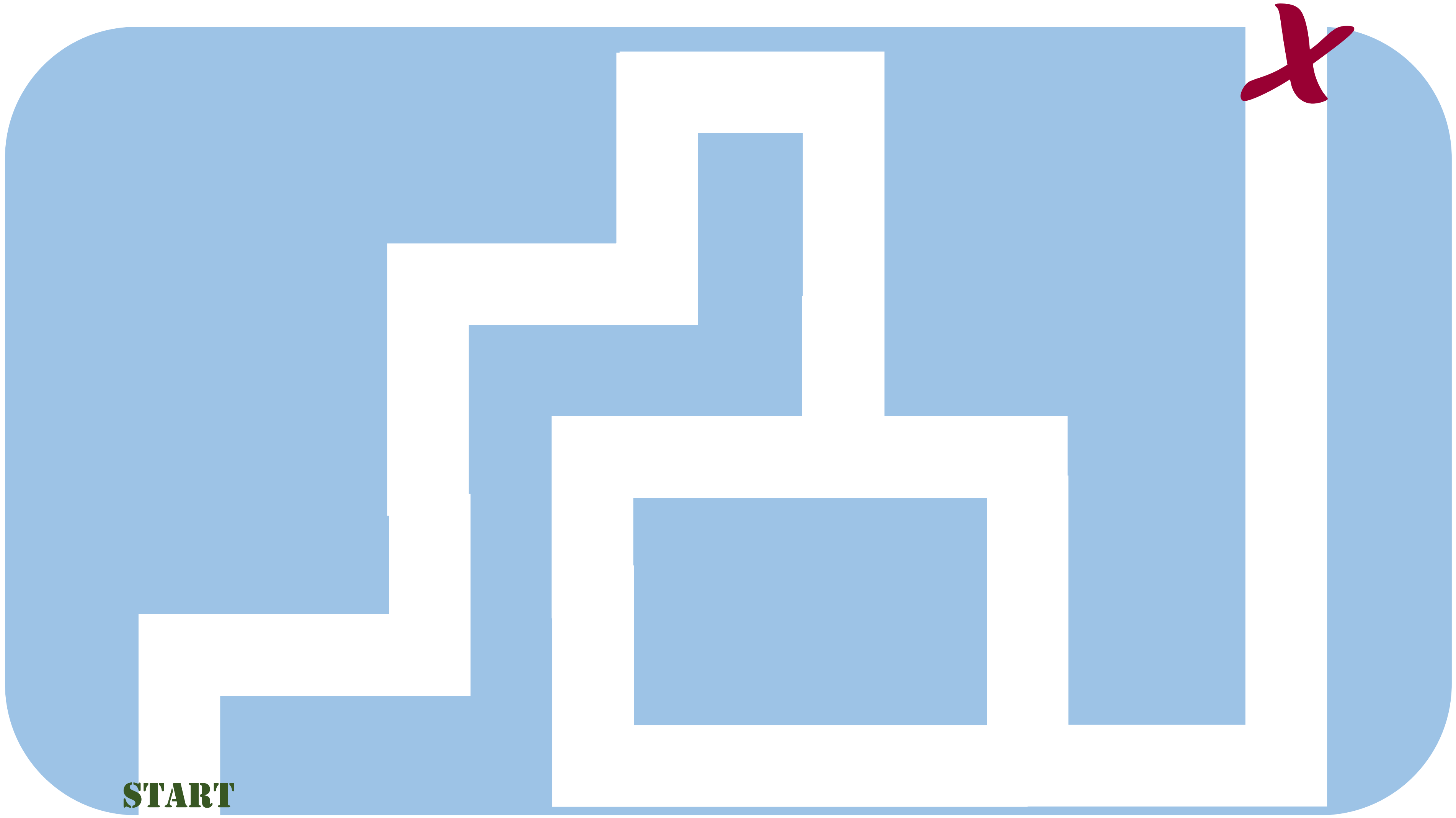
Navigation of AI Agents

הטכניון
היחידה ללימודי המשך
ולימודי חוץ ע״ש עזריאלי

TECHNION
Azrieli Continuing Education and
External Studies Division

- Basic search concepts
- Uninformed Search
    - Depth First Search
    - Breadth First Search
- Informed Search
    - GBFS
    - A*

Search Problem...



AI Agent

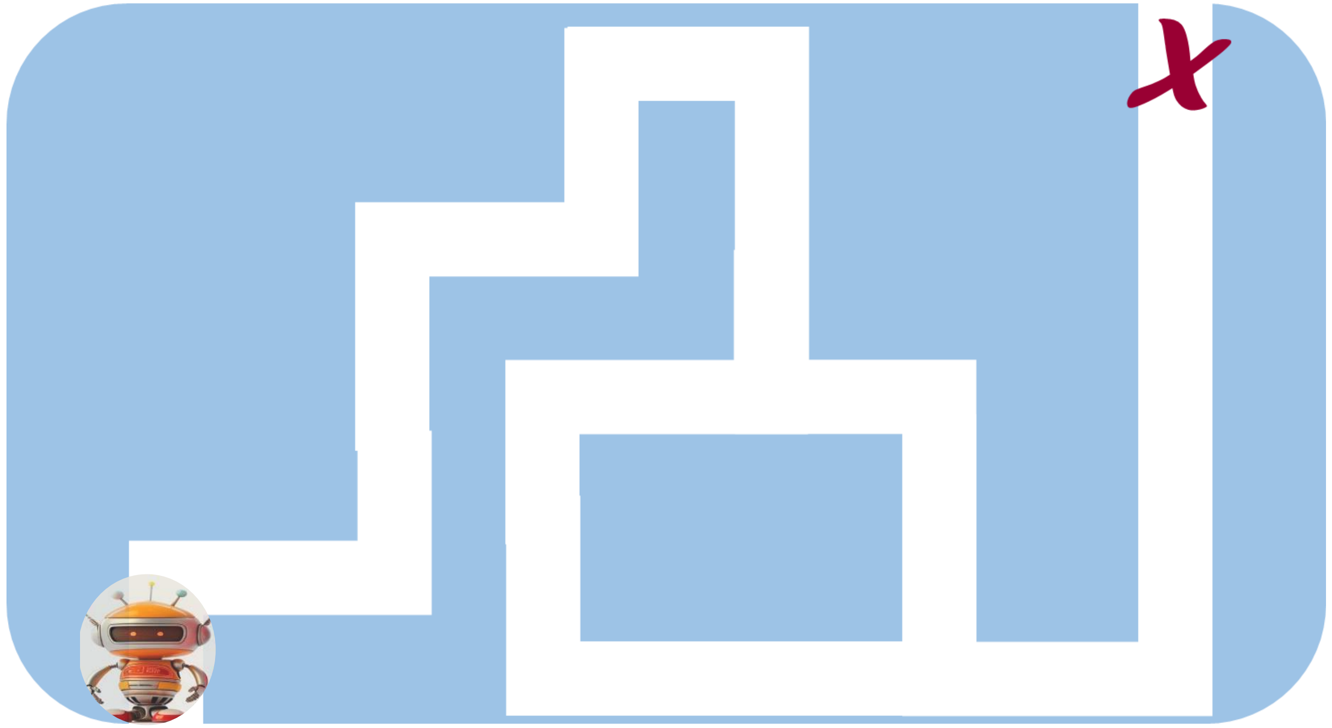
An AI agent is a system that perceives its environment, processes information, and takes actions to achieve specific goals.

It typically consists of **sensors** (to gather data), **decision-making algorithms**, and **actuators** (to perform actions).

Examples include virtual assistants, recommendation systems, and autonomous robots.

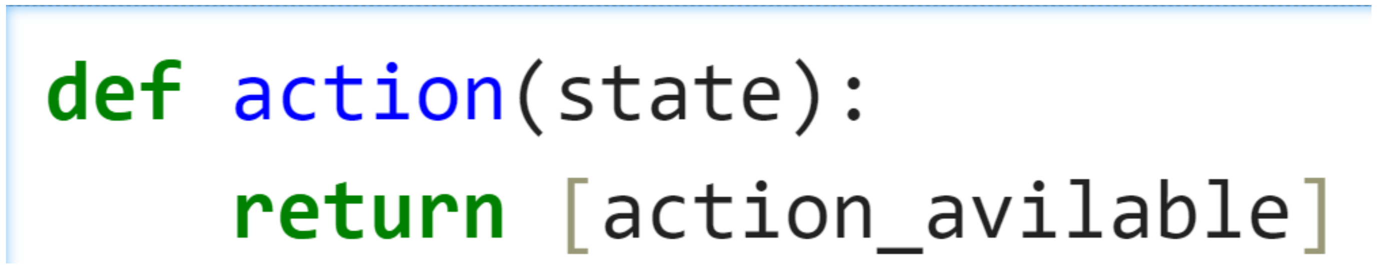
State:

A representation of the **current condition** or situation of the **agent** within the **environment**. It includes all relevant information to understand the environment at that moment.

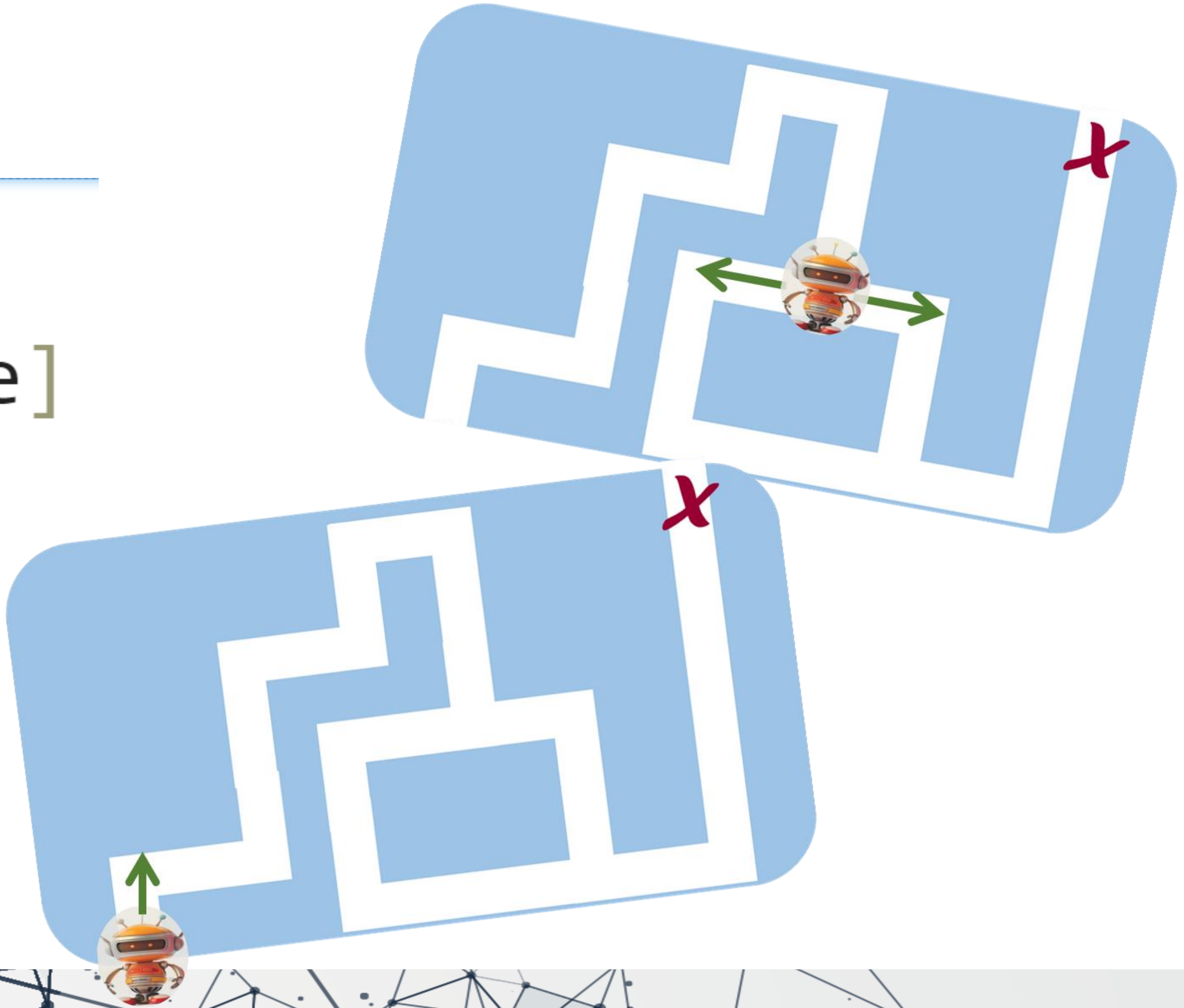


Action:

A step or move the agent can take to affect the environment or transition from one state to another.



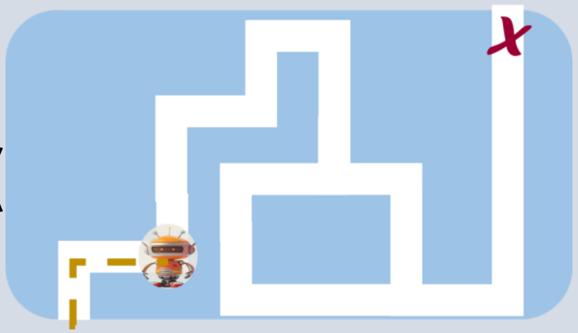
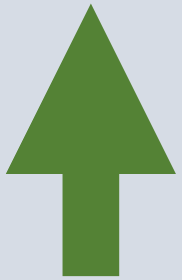
```
def action(state):
    return [action_avilable]
```

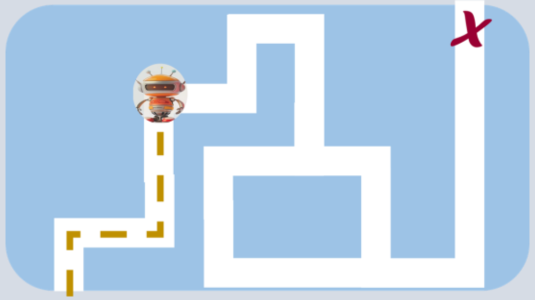


Transitional Model :

A mechanism that determines the result of an action taken in a given state, producing the next state. It describes how an action applied in the current state leads to a new state.

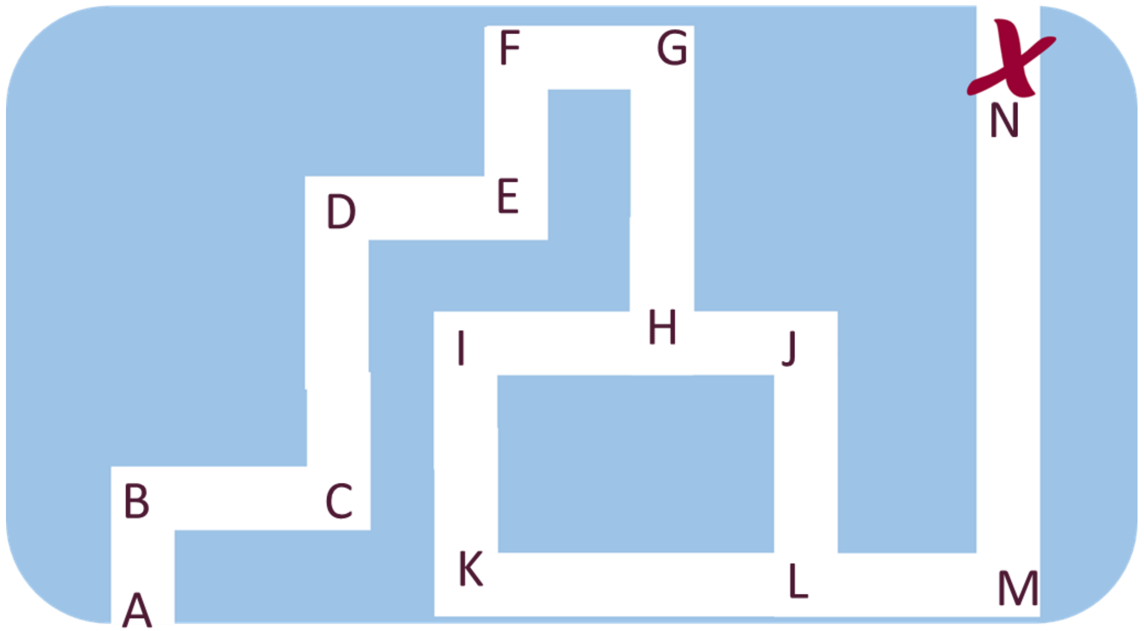
```
def result(state, action):
    return new_state
```

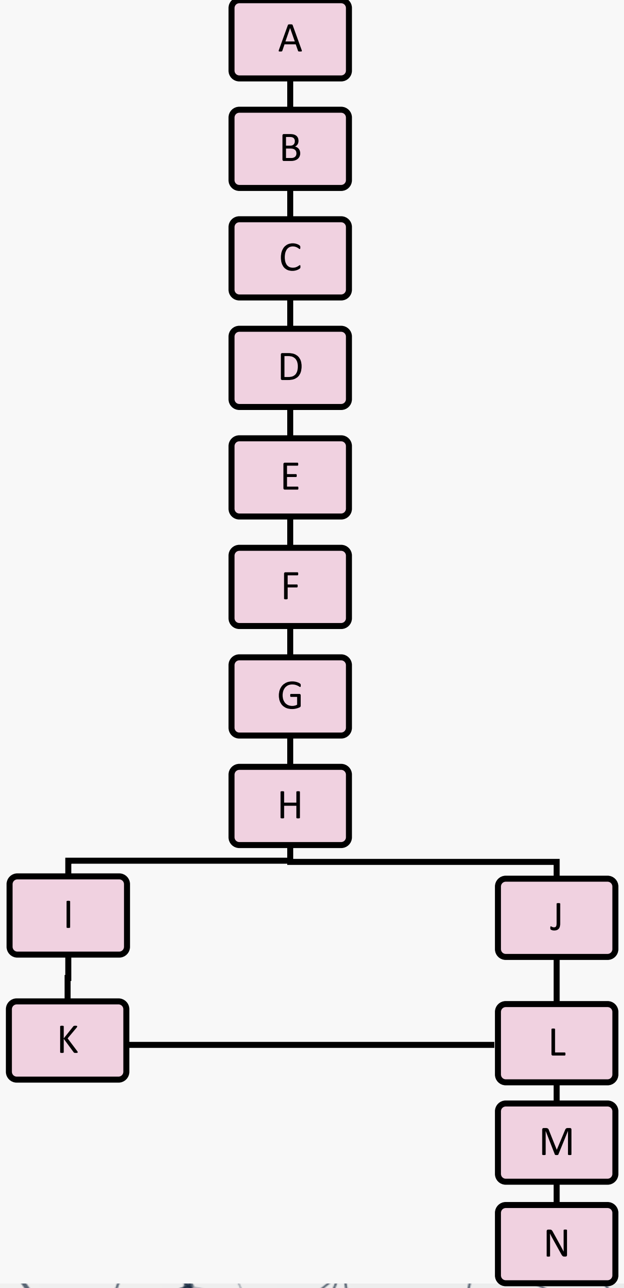
def result ( , ):

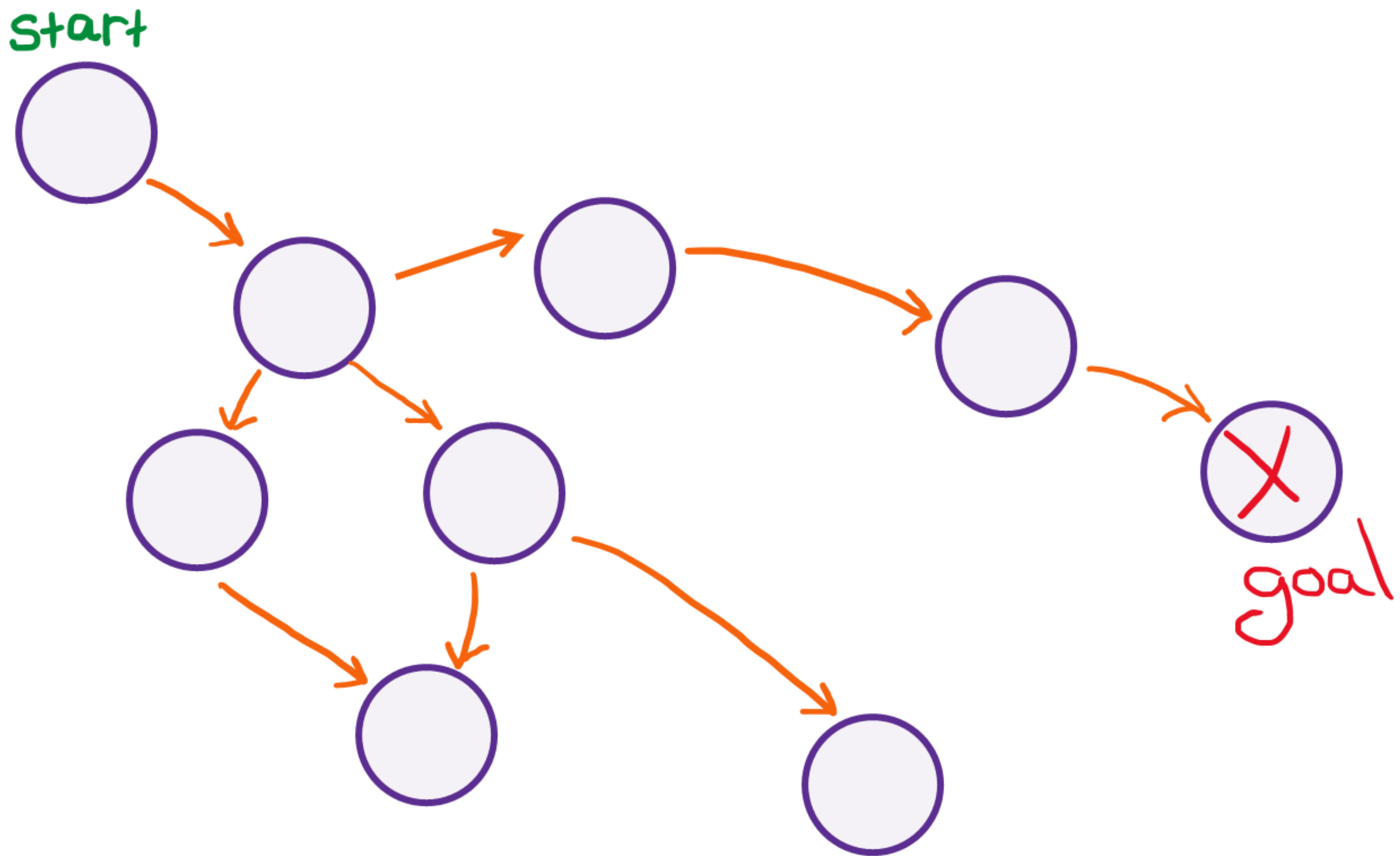
return 

State Space:

The set of all possible states the agent can occupy. It represents the entire space of states the agent can explore or transition between.







Node

- current state
- previous state
- action

(taken/available)
- cost

Search Algorithems

Frontier

the set of nodes that are **next in line to be explored**.

- The frontier consists of nodes that have been discovered but not yet visited or processed.
- The frontier represents the "boundary" between the explored and the unexplored.

Explored Set

a collection of nodes (or states) that have **already been visited** and processed during a graph or tree traversal. It helps prevent revisiting the same nodes, avoiding infinite loops in cyclic graphs.

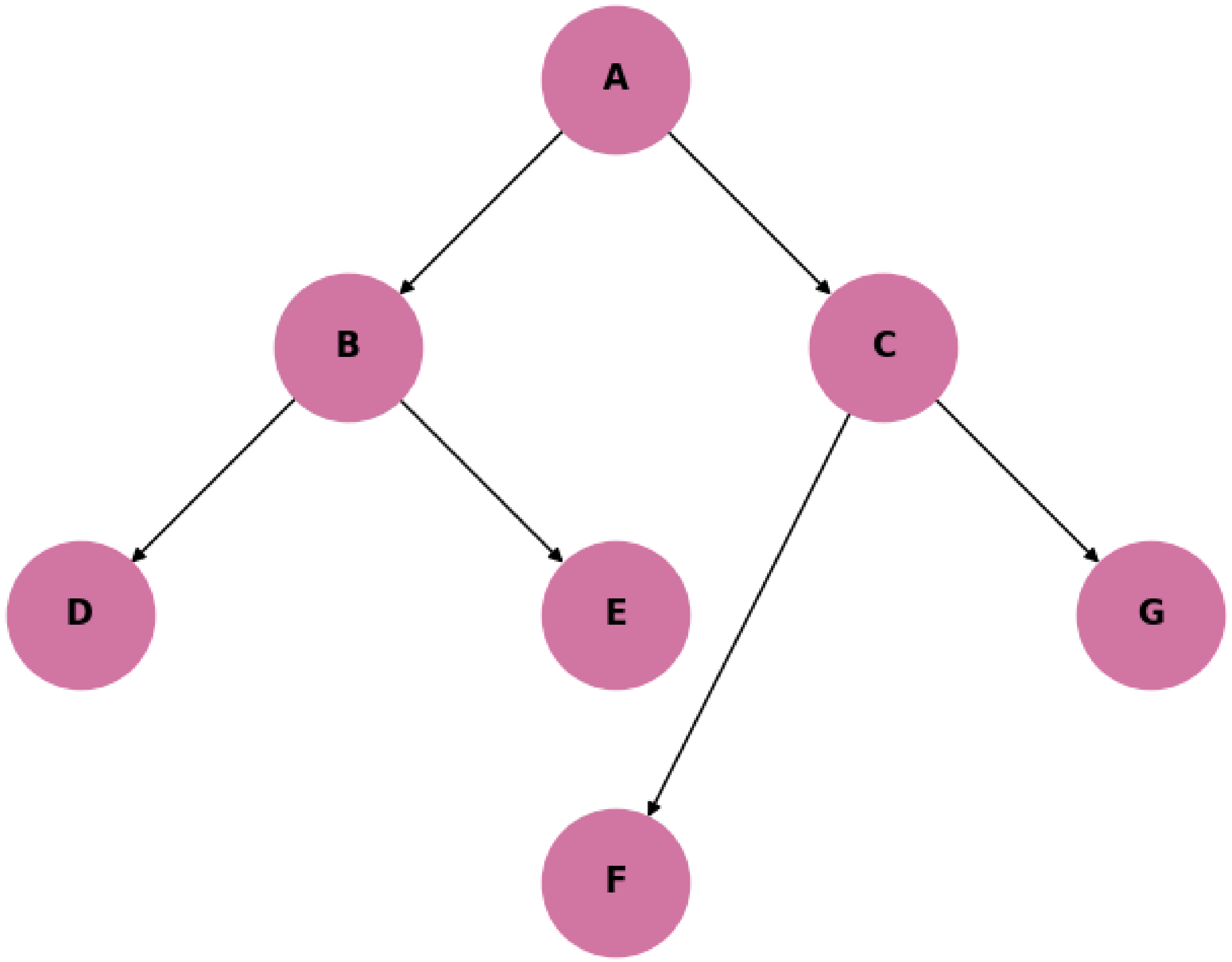
DFS

Depth First Search - is an algorithm that searches or explores a graph or tree by going as deep as possible along each branch, then backtracking when it reaches a dead end. It's useful for tasks like pathfinding, finding connected components, and solving mazes.

LIFO (Last In, First Out)

is a method of organizing and accessing data where the last item added is the first one removed.

it is typically implemented using a **stack** data structure.


A
B
C
D
E
G
F

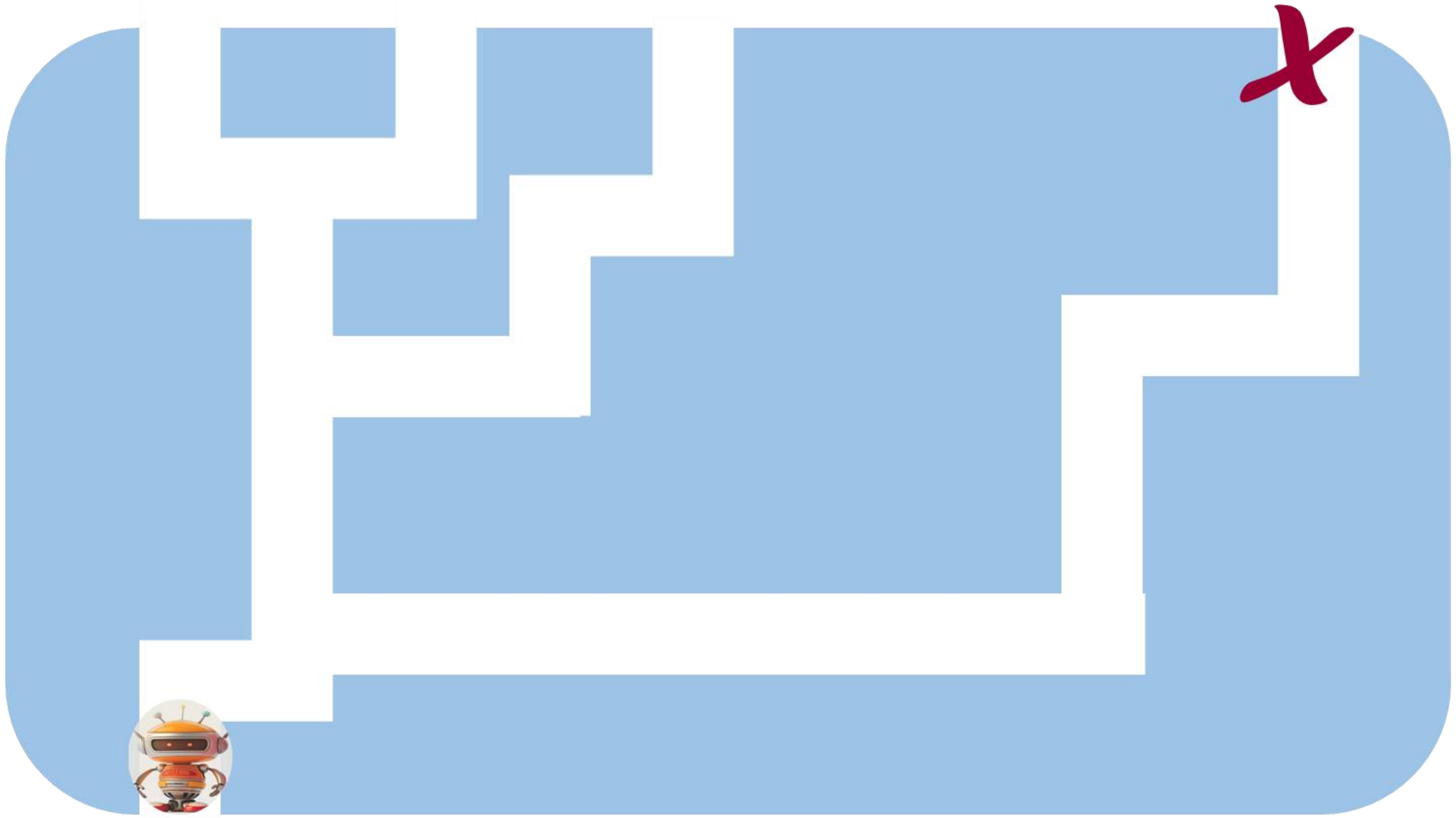
BFS

Breadth First Search is an algorithm that searches or explores a graph or tree level by level, starting from the root (or a given node) and visiting all its neighbors before moving to the next level. It's useful for tasks like finding the shortest path, exploring all reachable nodes, or solving problems where proximity matters.

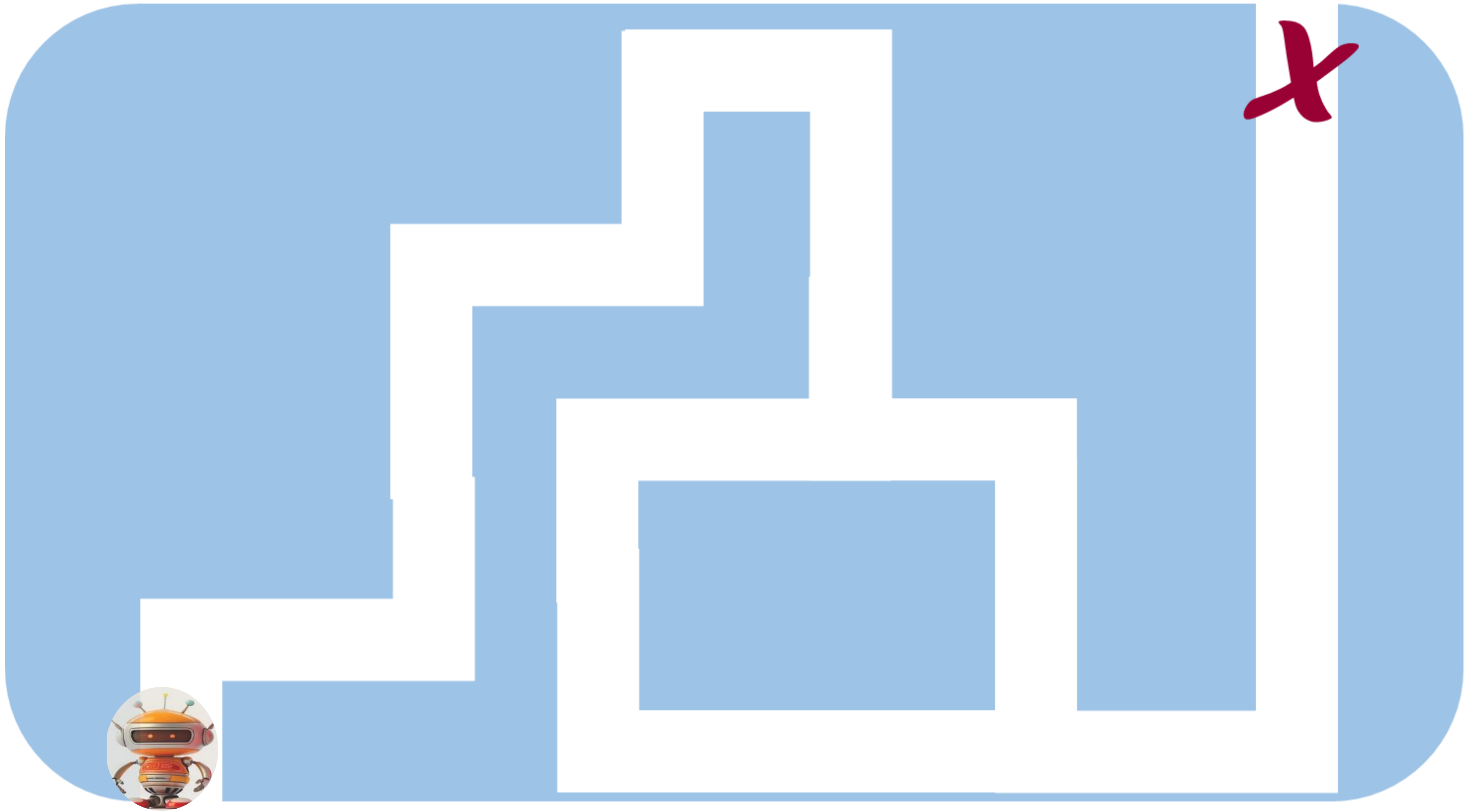
FIFO (First In, First Out)

is a method of organizing and accessing data where the **first item added is the first one removed**. The earliest added item is processed first. It is typically implemented using a **queue** data structure.

TECHNION
Azrieli Continuing Education and External Studies Division



DFS?



EFS?

Greedy Best-First Search (GBFS):

expands the node which appears closest to the goal, based only on a **heuristic function** h(n).

•**Key Idea:** It chooses the path that looks best **in the short term** (greedy).

•**Pro:** Faster than other searches when the heuristic is good.

•**Con:** Does not guarantee the shortest path (non-optimal).

A*

expands nodes based on both the cost so far and the estimated cost to the goal. f(n)=g(n)+h(n)

Key Idea: Combines the benefits of Dijkstra's (cost so far) and GBFS (heuristic).

Pro: Guarantees the shortest path if $h(n)$h(n) is admissible (never overestimates).

Con: Can be slower than GBFS due to considering all costs.

- **DFS (Depth First Search):** Explores deeply along each path before backtracking.
- **BFS (Breadth First Search):** Explores level by level (breadth-first).
- **GBFS (Greedy Best-First Search):** Chooses nodes based only on a heuristic (estimation of goal proximity).
- ***A *:*** Combines path cost and heuristic to find the most optimal path.