Schedule

- Module 2 Review
- 1 paper presentation
- Begin lecture on search/planning

IMPORTANT

Reno Kriz's talk on "Takeaways from the SCALE 2024 Workshop on Event-Centric Video Retrieval"

Tuesday, October 8, 2024 · 1:30 - 2:30 PM

Information Technology/Engineering : 325B

Join Online Event

Abstract:
Information dissemination for current events has traditionally consisted of professionally collected and produced materials, leading to large collections of well-written news articles and high-quality videos. As a result, most prior work in event analysis and retrieval has focused on leveraging this traditional news content, particularly in English. However, much of the event-centric content today is generated by non-professionals, such as on-the-scene witnesses to events who hastily capture videos and upload them to the internet without further editing; these are challenging to find due to quality variance, as well as a lack of text or speech overlays providing clear descriptions of what is occurring. To address this gap, SCALE 2024, a 10-week research workshop hosted at the Human Language Technology Center of Excellence (HLTCOE), focused on multilingual event-centric video retrieval, or the task of finding videos about specific current events. Around 50 researchers and students participated in this workshop and were split up into five sub-teams. The Infrastructure team focused on developing MultiVENT 2.0, a challenging new video retrieval dataset consisting of 20x more videos than prior work and targeted queries about specific world events across six languages. The other teams worked on improving models from specific modalities, specifically Vision, Optical Character Recognition (OCR), Audio, and Text. Overall, we came away with three primary findings: extracting specific text from a video allows us to take better advantage of powerful methods from the text information retrieval community; LLM summarization of initial text outputs from videos is helpful, especially for noisy text coming from OCR; and no one modality is sufficient, with fusing outputs from all modalities resulting in significantly higher performance.

--

Reno Kriz is a research scientist at the Johns Hopkins University Human Language Technology Center of Excellence (HLTCOE). His primary research interests involve leverage large pre-trained models for a variety of natural language understanding tasks, including those crossing into other modalities, e.g., vision and speech understanding. These multimodal interests have recently involved the 2024 Summer Camp for Language Exploration (SCALE) on event-centric video retrieval and understanding. He received his PhD from the University of Pennsylvania where he worked with Chris Callison-Burch and Marianna Apidianaki on text simplification and natural language generation. Prior to that, he received BA degrees in Computer Science, Mathematics, and Economics from Vassar College.

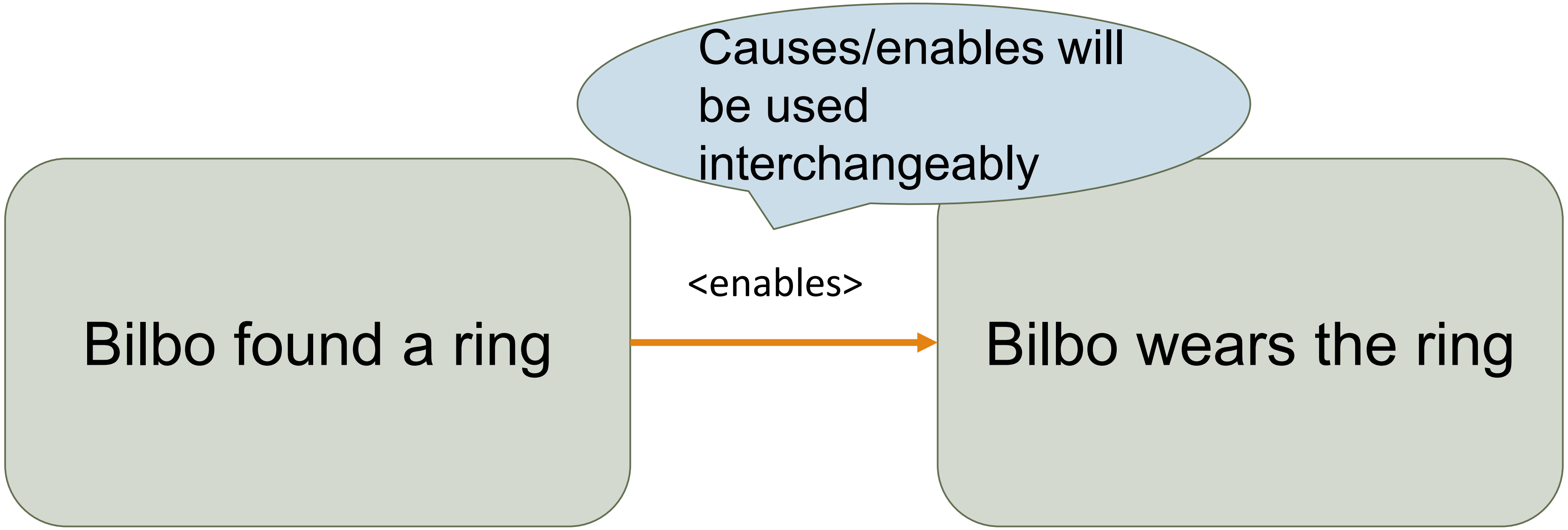
Module 2 Review

10/8/2024

CMSC 491/691 - INTERACTIVE FICTION AND TEXT GENERATION

DR. LARA J. MARTIN

Review: Causal Links



Review: Script

"A standard event sequence" that

- Lays out different paths/options
- Consists of causal chains
- <u>Can be used to leave out tedious details the reader is expected to know</u>

○ Can be considered a literary trope or a common social scenario

Schank, R. C., & Abelson, R. P. (1977). In *Scripts, plans, goals, and understanding: An enquiry into human knowledge structures* (The Artificial Intelligence Series, pp. 36–68). Erlbaum.

Review: Principle of Minimal Departure

"This law—to which I shall refer as the principle of minimal departure—states that we reconstrue the central world of a textual universe in the same way we reconstrue the alternate possible worlds of nonfactual statement: as conforming as far as possible to our representation of [the actual world]"

In other words:

The story world is expected to be like the real world, unless otherwise specified

Ryan, M.-L. (1991). Chapter 3: Reconstructing the Textual Universe: The Principle of Minimal Departure. In *Possible Worlds, Artificial Intelligence, and Narrative Theory* (pp. 48–60). Indiana Univ. Press.

Review: Linking Events

PROBABILISTIC

Occur frequently together (not necessarily because they had to)

Example:

I pour dog food in my dog's bowl.

I pet my dog.

CAUSAL

Occur because of one another

Example:

I pour dog food in my dog's bowl.

My dog eats dog food.

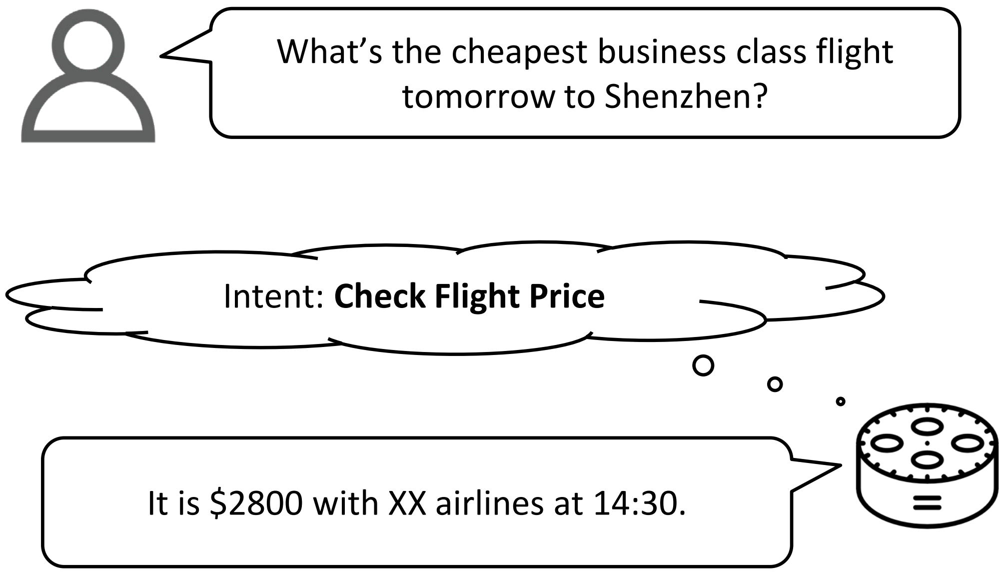
Review: What are procedures?

- A procedure is "a series of **actions** conducted in a certain **order** or manner," as defined by Oxford
- A more refined definition: "a series of **steps** happening to achieve some **goal**[1]"
  - Why?
- Examples of procedures: instructions (recipes, manuals, navigation info, how-to guide), algorithm, scientific processes, etc.
  - We focus on **instructions**, which is human-centered and task-oriented
- Examples of non-procedures: news articles, novels, descriptions, etc.
  - Those are often narrative: events do not have a specific goal
  - The umbrella term is **script**[2]

 [1] (Momouchi, 1980) [2] (Schank, 1977)

Review: Intent Detection

- Task-oriented dialog systems needs to match an **utterance** to an **intent**, before making informed responses
- Sentence classification task
  - Given an utterance, and some candidate intents
  - Choose the correct intent
  - Evaluated by accuracy

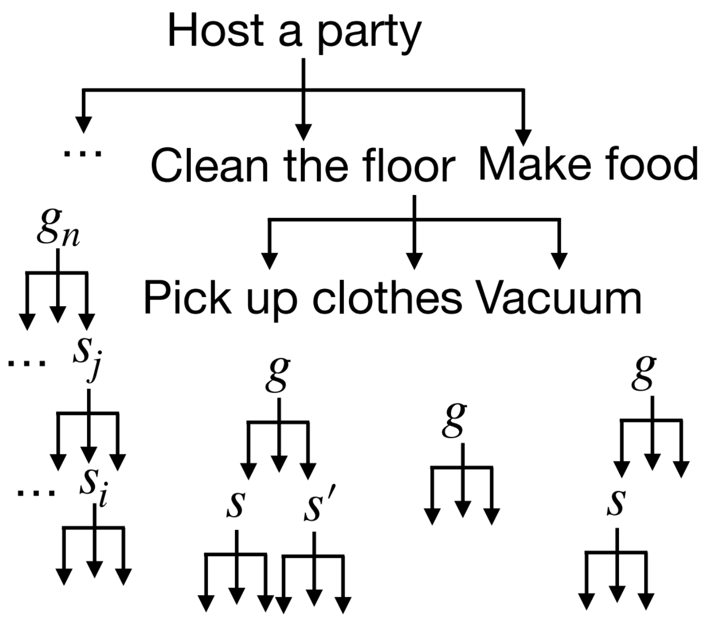


Example from Snips (Coucke et al., 2018)
Utterance: "Find the schedule at Star Theatres."
Candidate intents: Add to Playlist, Rate Book, Book Restaurant,
Get Weather, Play Music, Search Creative Work, **Search Screening Event**

(Zhang et al., 2020): Intent Detection with WikiHow

Review: Procedures are Hierarchical

- An event can simultaneously be a **goal** of one procedure, and a **step** in another
- A procedural hierarchy… So what?
  - Can "explain in more details" by expansion
  - Can shed light on event **granularity** (why?)
- How do you build such hierarchy?
  - To "host a party", I need to "clean the floor"; to "clean the floor", I need to do what?



(Zhou et al., 2022): Show Me More Details: Discovering Hierarchies of Procedures from Semi-structured Web Data

Review: Plan-and-Write

Carrie had just learned how to ride a bike. She didn't have a bike of her own. Carrie would sneak rides on her sister's bike. She got nervous on a hill and crashed into a wall. The bike frame bent and Carrie got a deep gash on her leg.

Carrie→bike→sneak→nervous→leg

Yao, L., Peng, N., Weischedel, R., Knight, K., Zhao, D., & Yan, R. (2019). Plan-And-Write: Towards Better Automatic Storytelling. *AAAI Conference on Artificial Intelligence (AAAI)*, *33*(1), 7378–7385. https://aaai.org/ojs/index.php/AAAI/article/view/4726

Review: Guided Open Story Generation Using Probabilistic Graphical Models

- Use discourse representation structure (DRS) parser to get semantic relationships

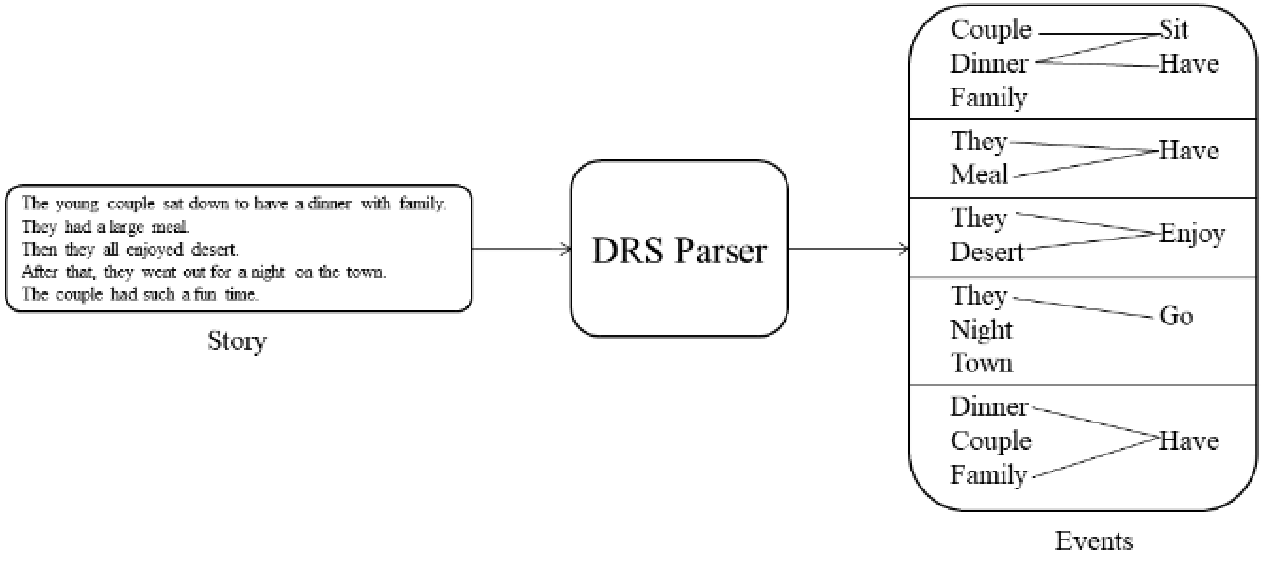


Figure 1: Event Representation using DRS Parser. Note that the words without edges are removed while forming the graphs.

Gandhi, S., & Harrison, B. (2019). Guided open story generation using probabilistic graphical models. *International Conference on the Foundations of Digital Games (FDG)*, 1–7. https://doi.org/10.1145/3337722.3341871

Review: Example of a Probabilistic Event Representation

From sentence, extract event representation:

(subject, verb, direct object, modifier, preposition)

Original sentence: yoda uses the force to take apart the platform

Events:

yoda use force Ø Ø
yoda take_apart platform Ø Ø

Generalized Events:

<PERSON>0 fit-54.3 power.n.01 Ø Ø

<PERSON>0 destroy-44 surface.n.01 Ø Ø

Martin, Lara J., et al. "Event representations for automated story generation with deep neural nets." *Thirty-Second AAAI Conference on Artificial Intelligence*. 2018.

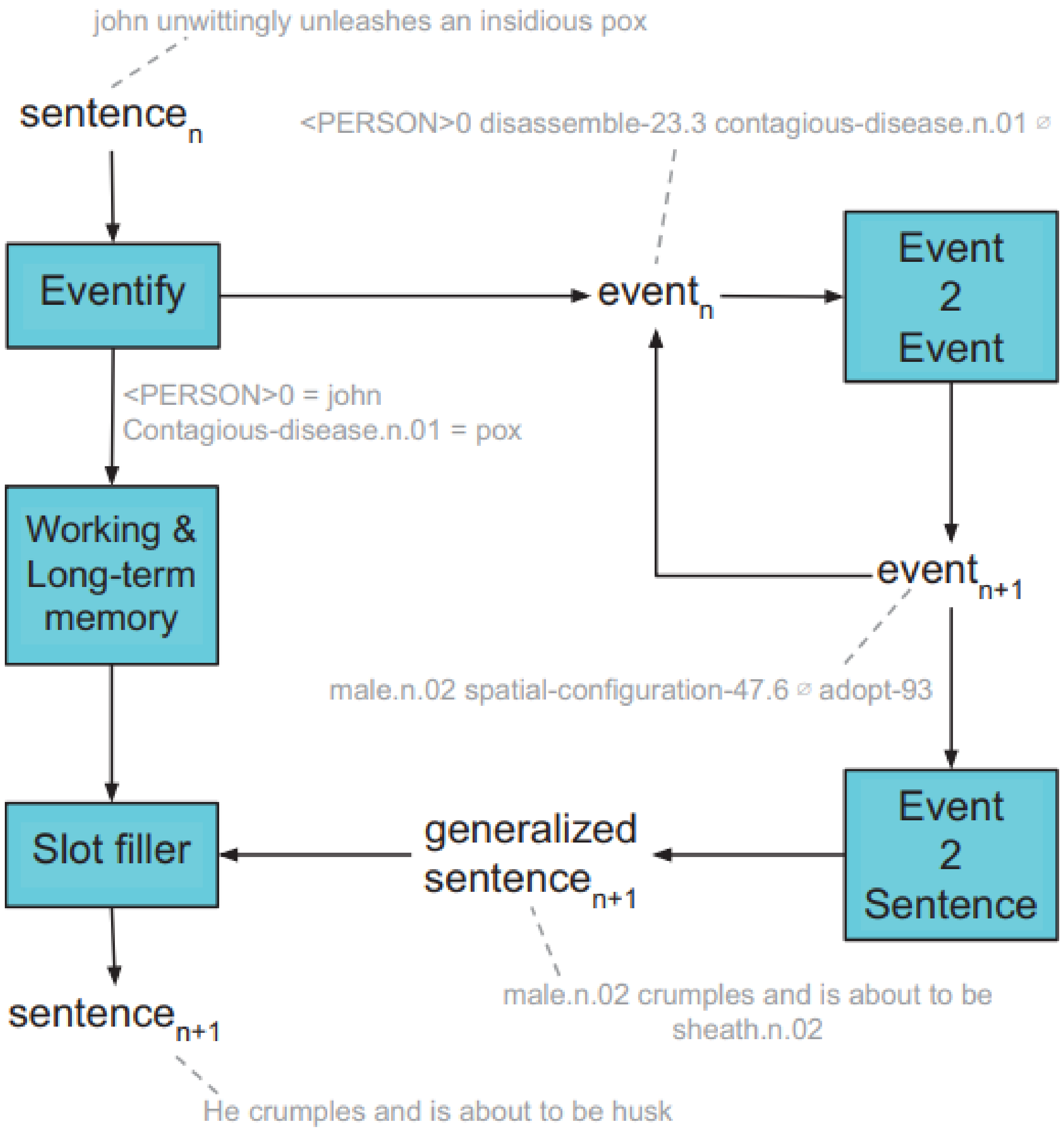
Review: Story Realization

Extract events from stories

Generate the plot using a seq2seq network

Use an ensemble of methods to find the best sentence given an event

Get a confidence score from each model, and accept the sentence if it's above a threshold



Ammanabrolu, P., Tien, E., Cheung, W., Luo, Z., Ma, W., Martin, L. J., & Riedl, M. O. (2020). Story Realization: Expanding Plot Events into Sentences. *AAAI Conference on Artificial Intelligence (AAAI)*, *34*(5), 7375–7382. https://ojs.aaai.org//index.php/AAAI/article/view/6232

Review: Story Cloze Test

Gina was worried the cookie dough in the tube would be gross.

She was very happy to find she was wrong.

The cookies from the tube were as good as from scratch.

Gina intended to only eat 2 cookies and save the rest.

A. Gina liked the cookies so much she ate them all in one sitting. ✓

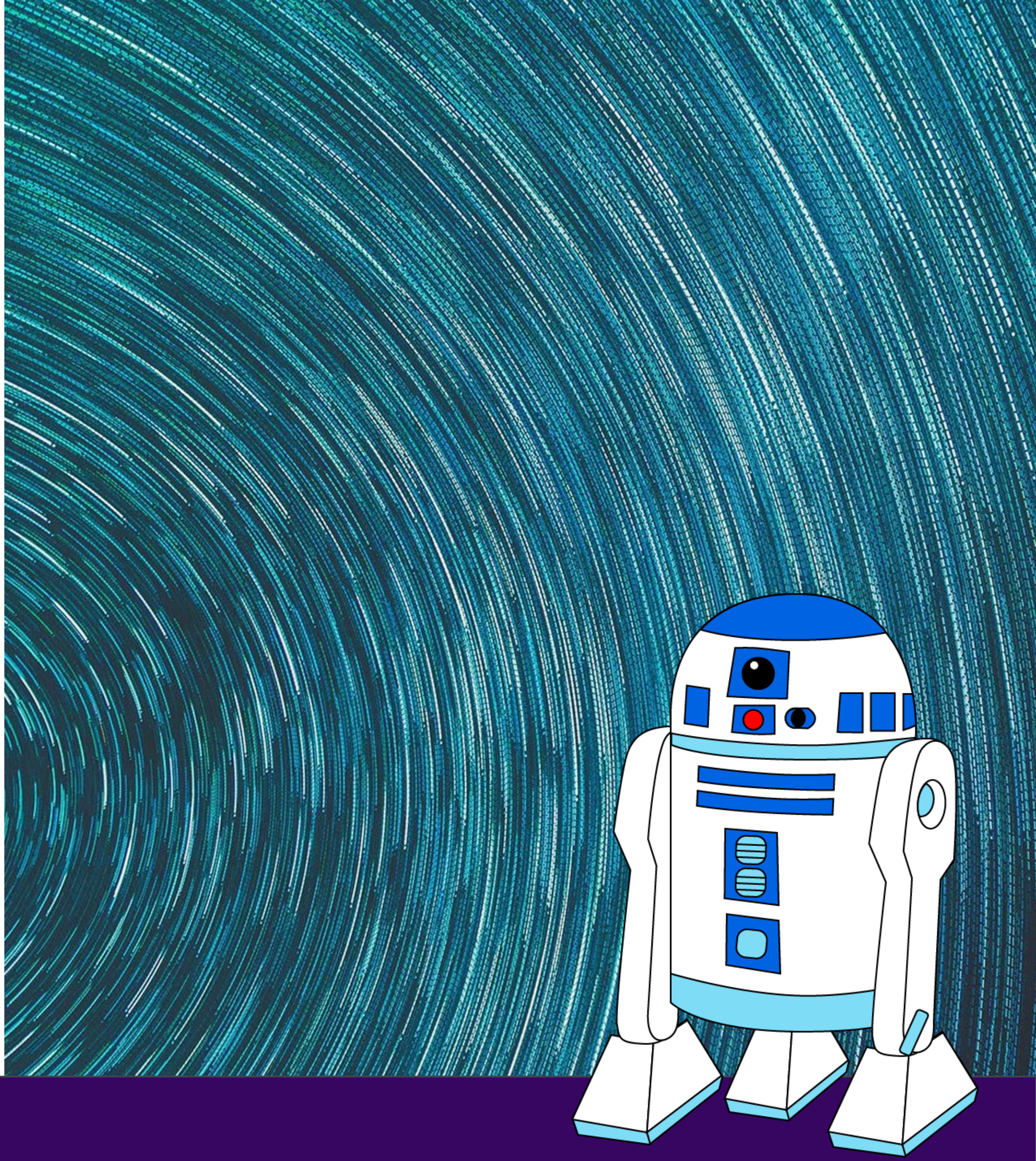
B. Gina gave the cookies away at her church.

Mostafazadeh, N., Chambers, N., He, X., Parikh, D., Batra, D., Vanderwende, L., Kohli, P., & Allen, J. (2016). A Corpus and Cloze Evaluation for Deeper Understanding of Commonsense Stories. *Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*, 839–849. http://www.aclweb.org/anthology/N16-1098

CMSC 491/691: Interactive Fiction and Text Generation

Search and Planning

AIMA Chapters 3 and 7



Learning Objectives

Remember how to setup a search problem

Review basic types of tree search algorithms

Define & implement a search problem (for Action Castle)

Problem-Solving Agents

A problem-solving agent must **plan**.

The computational process that it undertakes is called **search**.

It will consider a **sequence of actions** that form a **path** to a **goal state**.

Such a sequence is called a **solution.**

1. take pole
2. go out
3. go south
4. catch fish with pole
5. go north
6. pick rose
7. go north
8. go up
9. get branch
10. go down
11. go east
12. give the troll the fish
13. go east
14. hit guard with branch
15. get key
16. go east
17. get candle
18. go west
19. go down
20. light lamp
21. go down
22. light candle
23. read runes
24. get crown
25. go up
26. go up
27. go up
28. unlock door
29. go up
30. give rose to the princess
31. propose to the princess
32. down
33. down
34. east
35. east
36. wear crown
37. sit on throne

Review of Search Problems

AIMA 3.1-3.3

Formal Definition of a Search Problem

1. **States:** a set **S**
2. An **initial state** $s_i \in S$
3. **Actions:** a set **A**

   ∀ **s Actions(s)** = the set of actions that can be executed in **s**.

4. **Transition Model:** **∀ s∀ a∈Actions(s)**

   Result(s, a) → $\mathbf{s_r}$

   $\mathbf{s_r}$ is called a successor of **s**

   $\{s_i\} \cup$ **Successors**(s_i)* = state space

5. **Path cost** (Performance Measure):
   Must be additive, e.g. sum of distances, number of actions executed, …

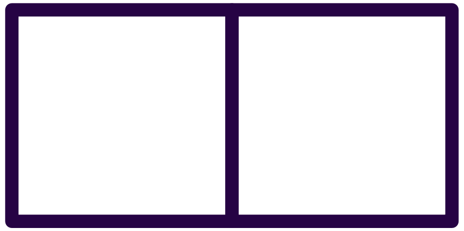
   c(x,a,y) is the step cost, assumed ≥ 0

   - (where action **a** goes from state **x** to state **y**)

6. **Goal test: Goal(s)**

   s is a goal state if **Goal(s)** is true.
   Can be implicit, e.g. **checkmate(s)**

Vacuum World



States: A state of the world says which objects are in which cells.

In a simple two cell version,

- the agent can be in one cell at a time
- each cell can have dirt or not

2 positions for agent * 2^2 possibilities for dirt = 8 states.

With *n* cells, there are $n*2^n$ states.

Vacuum World

States: A state of the world says which objects are in which cells.

In a simple two cell version,

- the agent can be in one cell at a time
- each cell can have dirt or not

2 positions for agent * 2^2 possibilities for dirt = 8 states.

With *n* cells, there are $n*2^n$ states.

Vacuum World

States: A state of the world says which objects are in which cells.

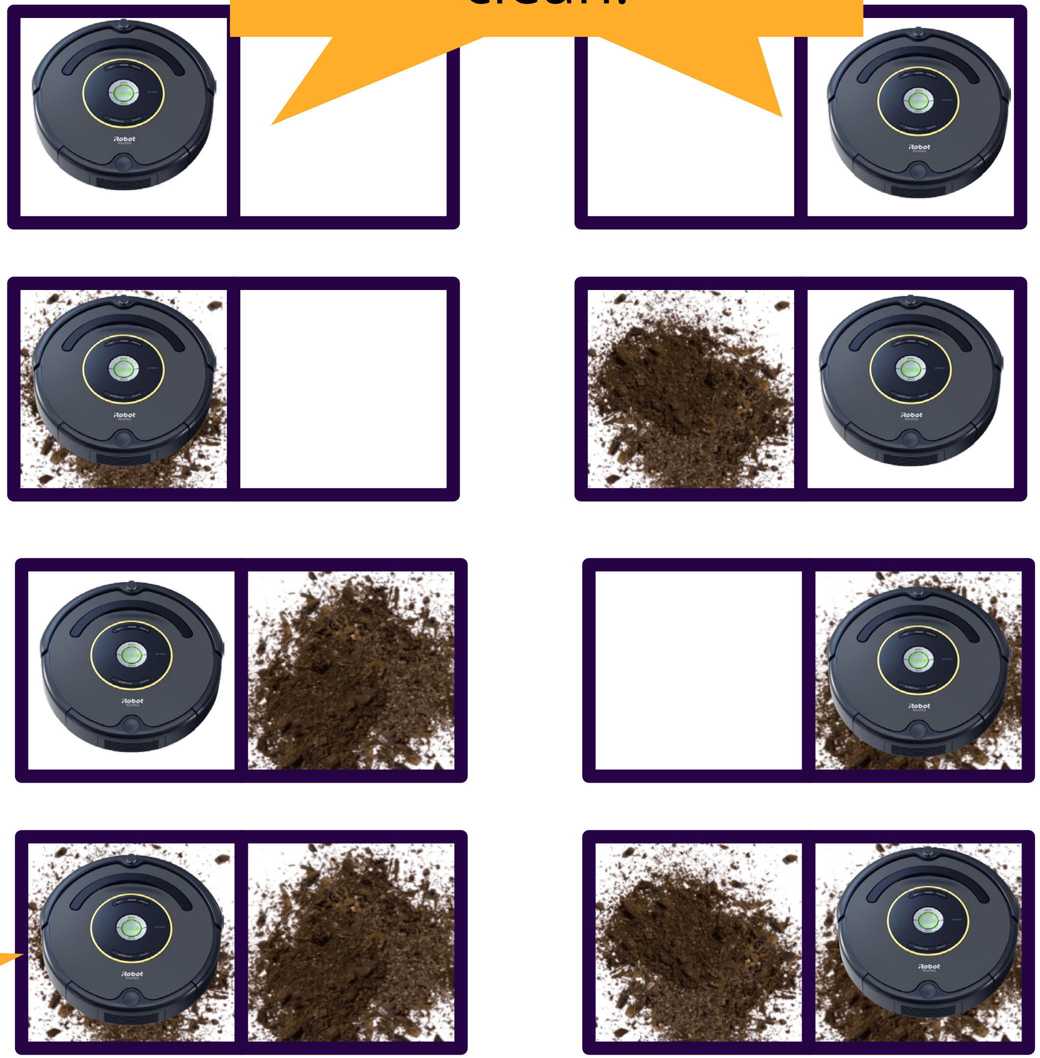
In a simple two cell version,

- the agent can be in one cell at a time
- each cell can have dirt or not

2 positions for agent * 2^2 possibilities for dirt = 8 states.

With *n* cells, there are $n*2^n$ states.

Goal states: States where everything is clean.



One state is designated as the **initial state**

Vacuum World

Actions:

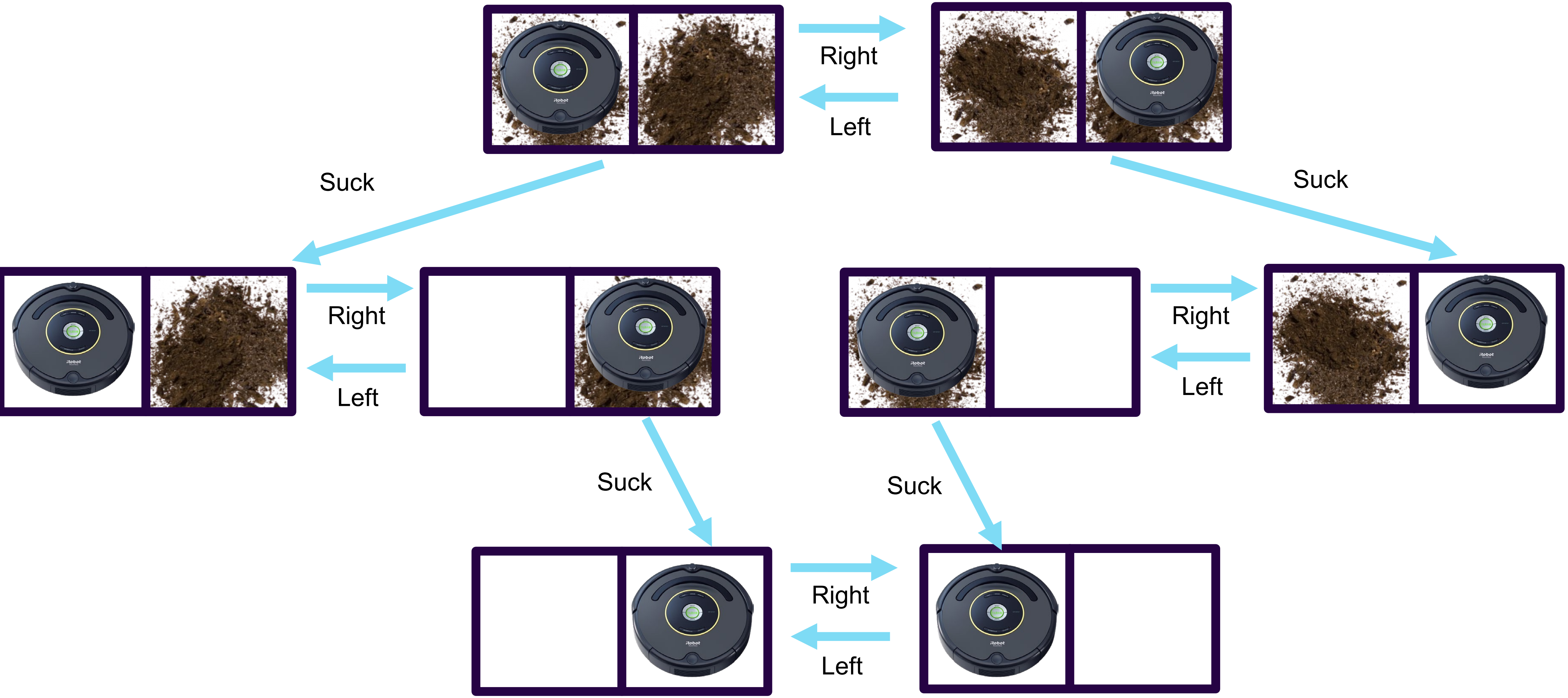
- *Suck*
- *Move Left*
- *Move Right*
- *(Move Up)*
- *(Move Down)*

Transition:

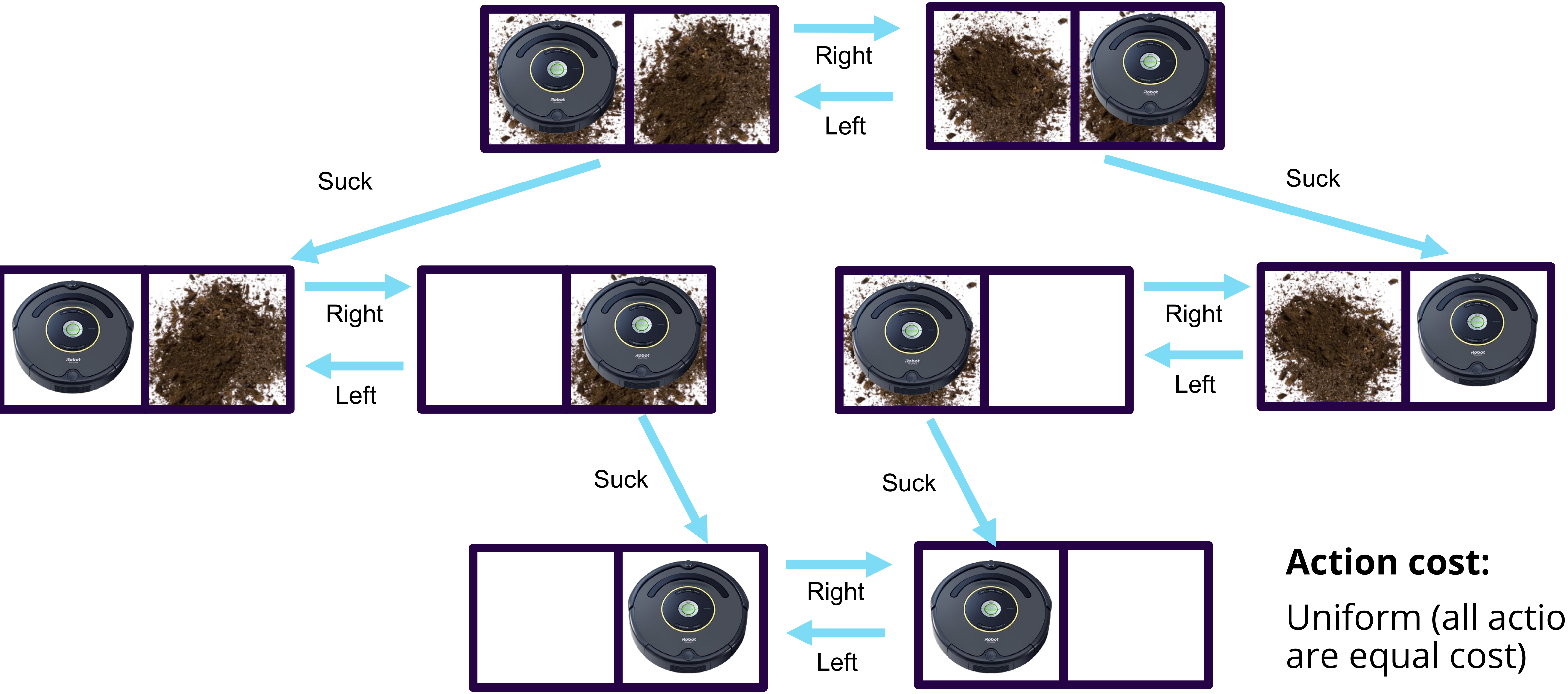
Suck – removes dirt

Move – moves in that direction, unless agent hits a wall, in which case it stays put.

Vacuum World



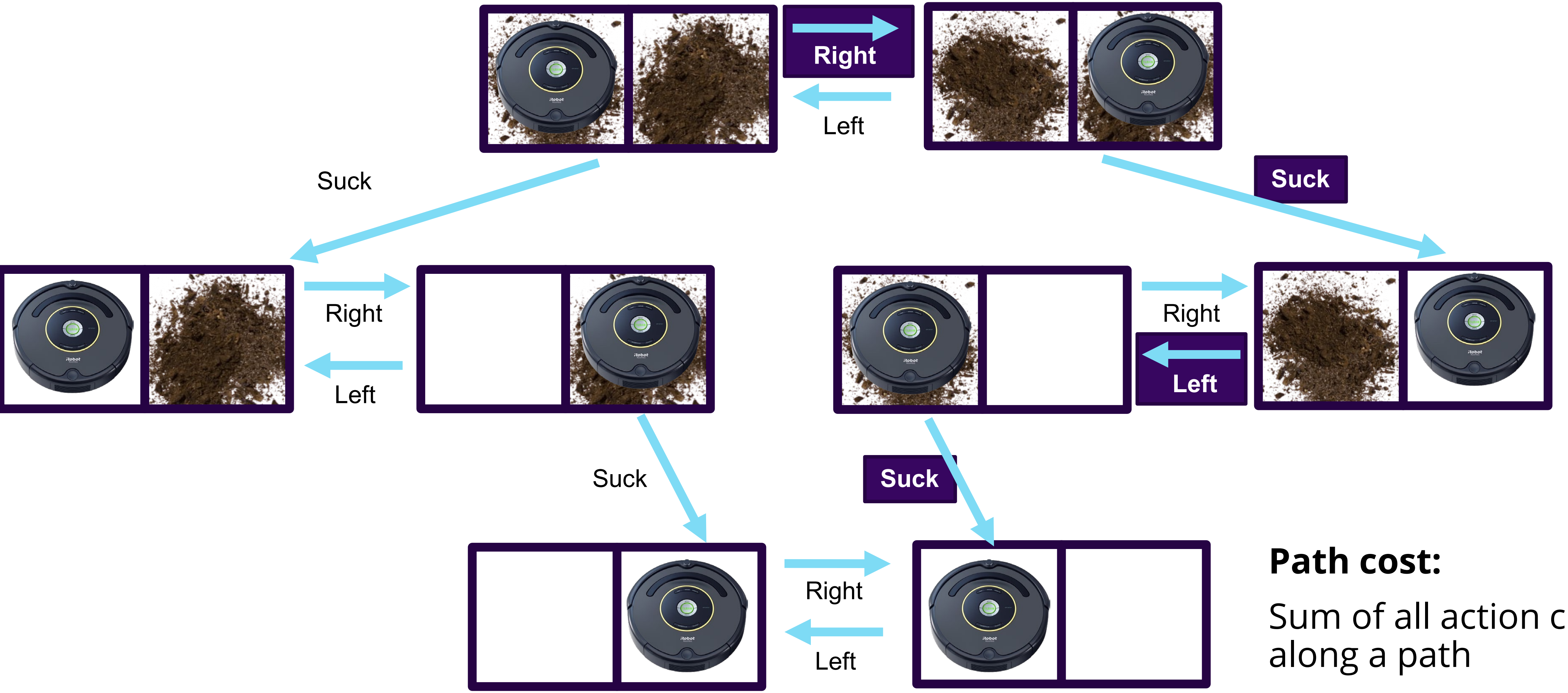
Vacuum World



Action cost:

Uniform (all actions are equal cost)

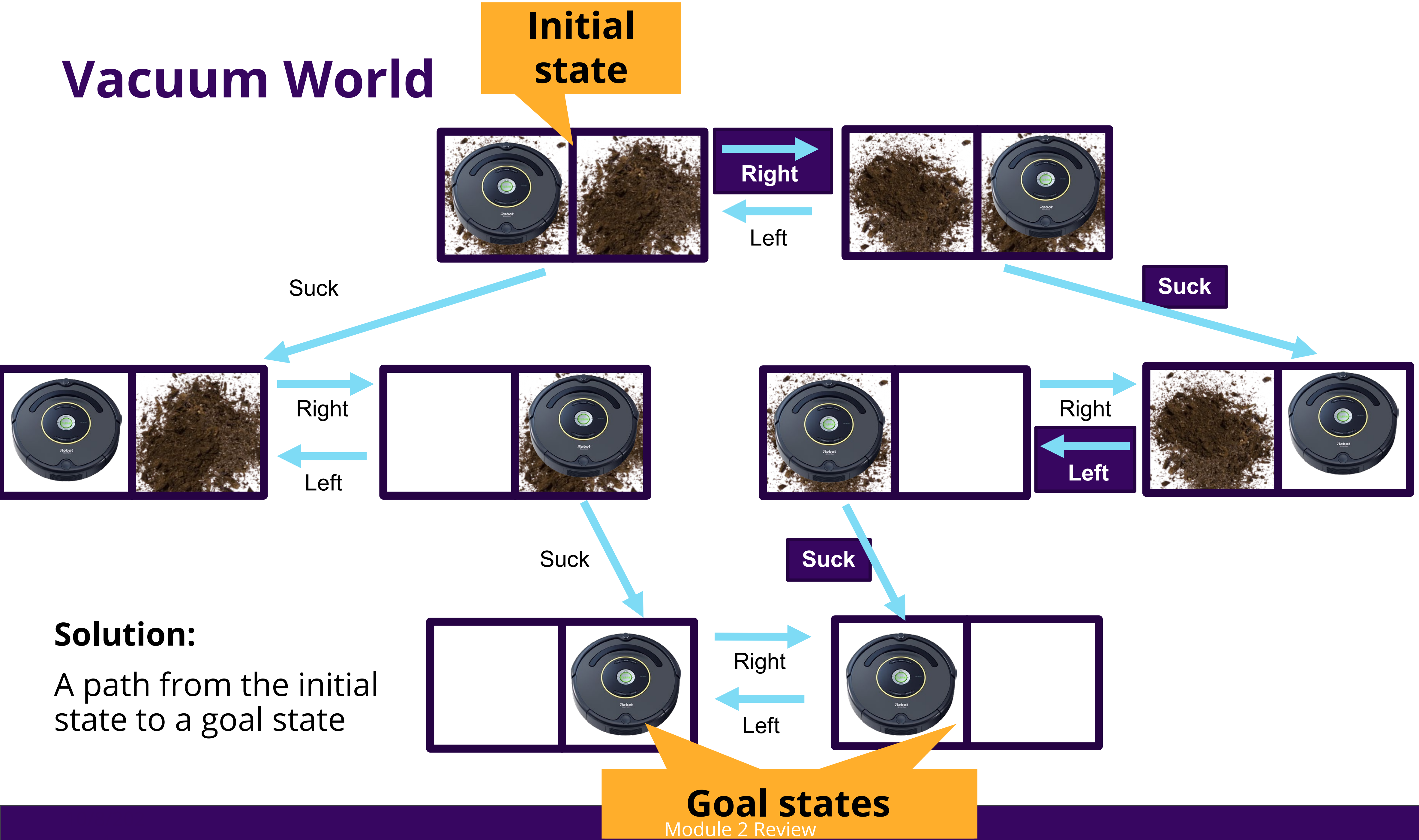
Vacuum World



Path cost:

Sum of all action costs along a path

Vacuum World



Solution:

A path from the initial state to a goal state

Search Algorithms

Useful Concepts

State space: the set of all states reachable from the initial state by *any* sequence of actions

- *When several operators can apply to each state, this gets large very quickly*
- *Might be a proper subset of the set of configurations*

Path: a sequence of actions leading from one state s_j to another state s_k

Solution: a path from the initial state s_i to a state s_f that satisfies the goal test

Search tree: a way of representing the paths that a search algorithm has explored. The root is the initial state, leaves of the tree are successor states.

Frontier: those states that are available for *expanding* (for applying legal actions to)

Solutions and *Optimal* Solutions

A *solution* is a sequence of actions from the initial state to a goal state.

Optimal Solution: **A solution is optimal if no solution has a lower path cost.**

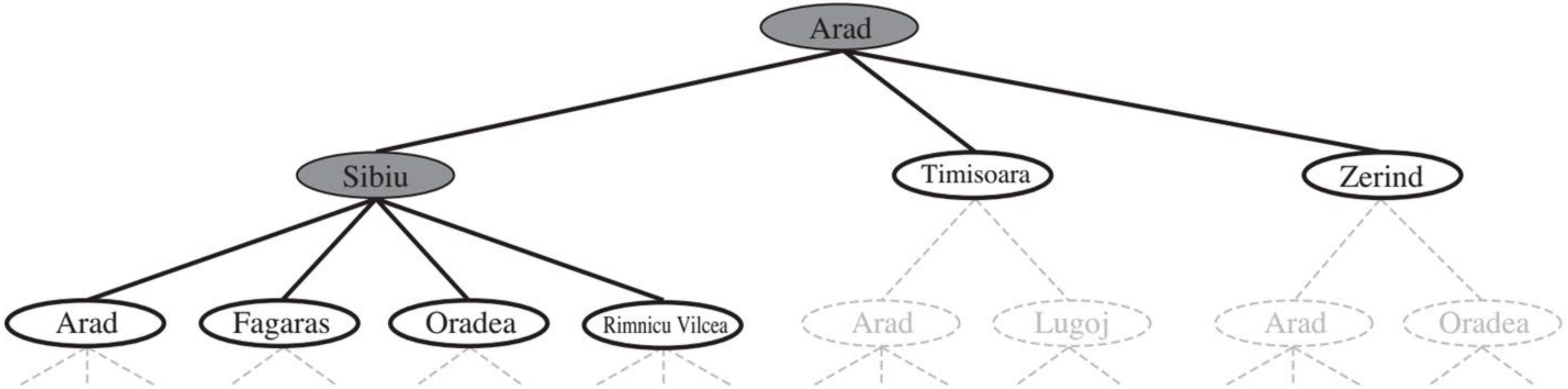
Basic search algorithms: *Tree Search*

Generalized algorithm to solve search problems

Enumerate in some order all possible paths from the initial state

- Here: search through ***explicit tree generation***
  - ROOT= initial state.
  - Nodes in search tree generated through ***transition model***
  - Tree search treats different paths to the same node as distinct

Generalized tree search



function TREE-SEARCH(*problem, strategy*) return a solution or failure

Initialize frontier to the *initial state* of the *problem*

do

if the frontier is empty then return *failure*

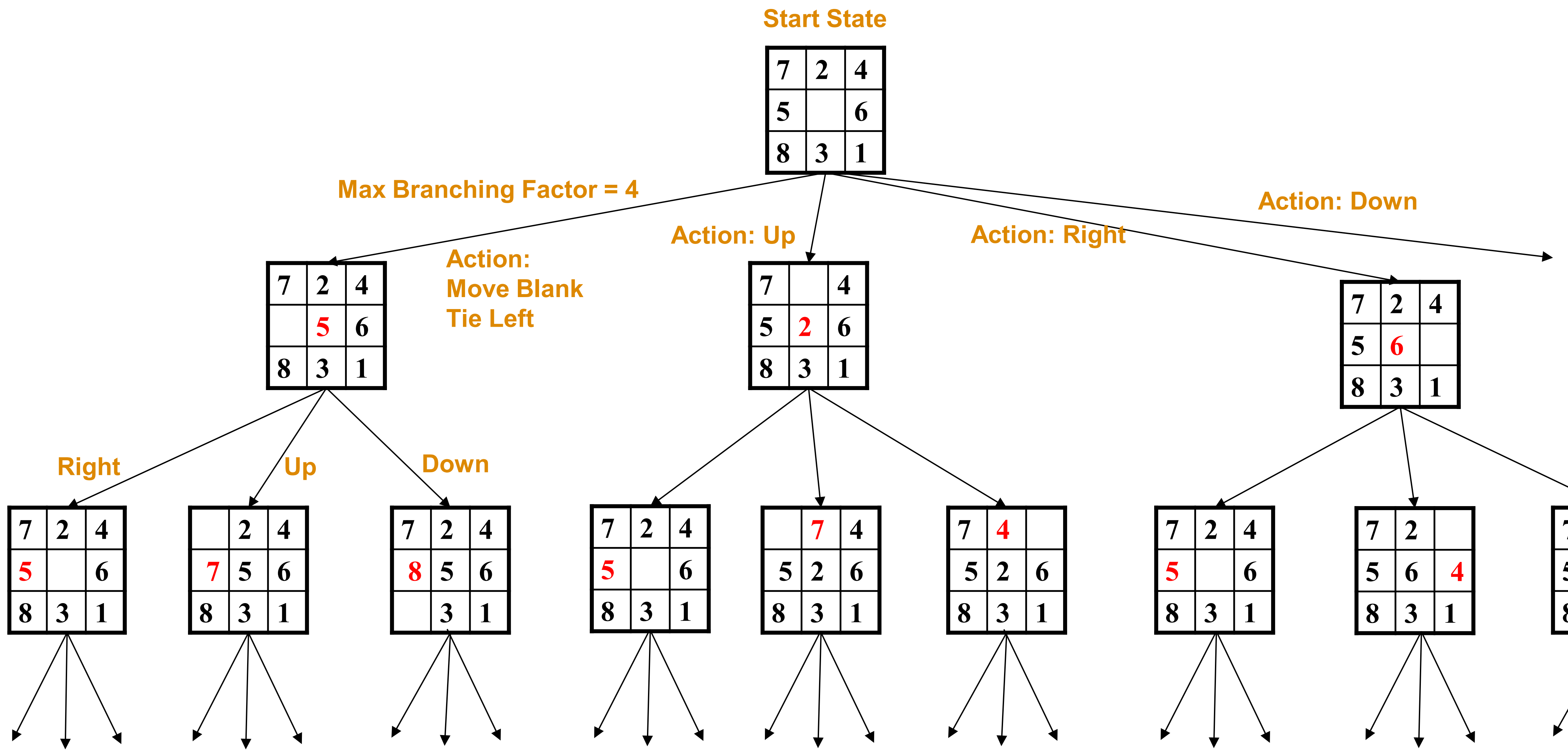
choose leaf node for expansion according to strategy & remove from frontier

if node contains goal state then return *solution*

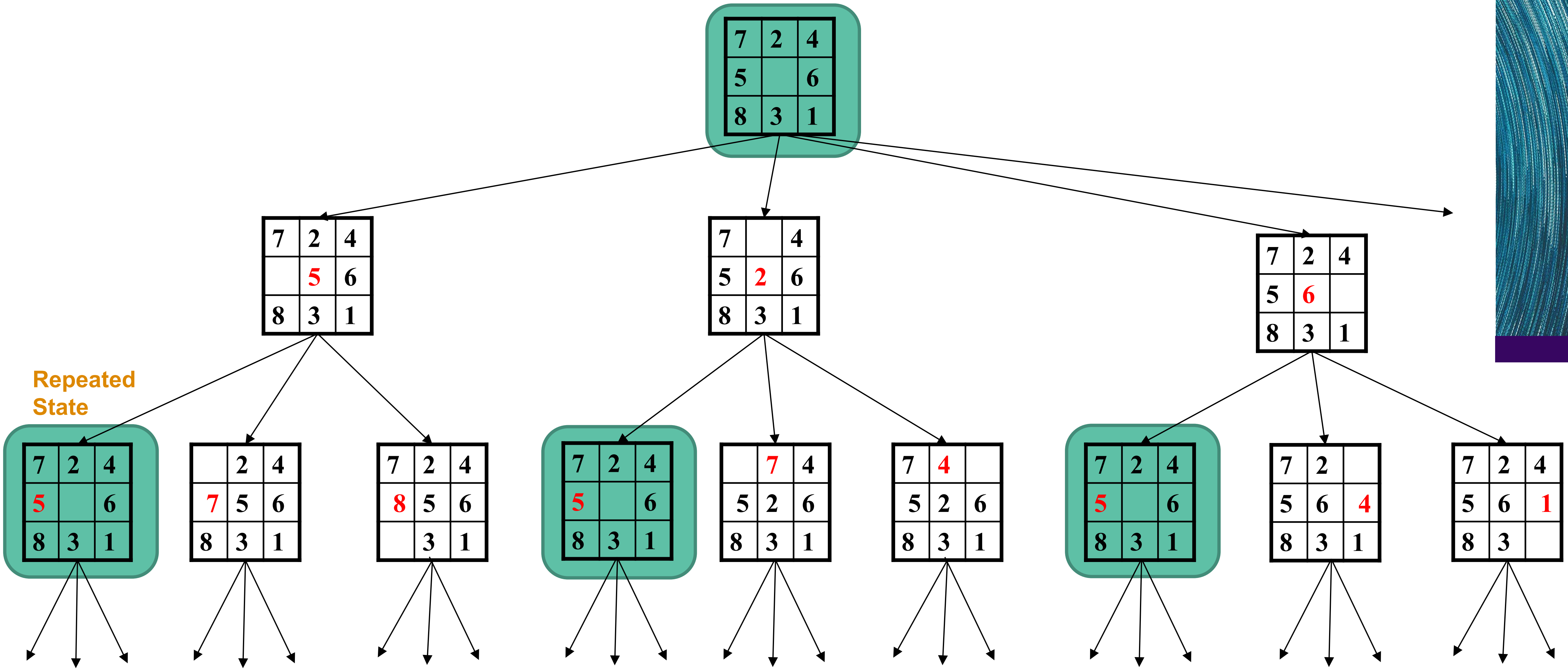
else expand the node and add resulting nodes to the frontier

The strategy determines search process!

8-Puzzle *Search Tree*



8-Puzzle *Search Tree*



Graph Search vs Tree Search

function TREE-SEARCH(*problem*) **returns** a solution, or failure
    initialize the frontier using the initial state of *problem*
    loop do
        if the frontier is empty **then return** failure
        choose a leaf nose and remove it from the frontier
        if the node contains a goal state **then return** the corresponding solution
        expand the chosen node, adding the resulting nodes to the frontier

function GRAPH-SEARCH(*problem*) returns a solution, or failure
    initialize the frontier using the initial state of *problem*
    initialize the explored set to be empty
    loop do
        if the frontier is empty **then return** failure
        choose a leaf node and remove it from the frontier
        if the node contains a goal state **then return** the corresponding solution
        add node to the explored set
        expand the chosen node, adding the resulting nodes to the frontier
            only if not in the frontier of explored set

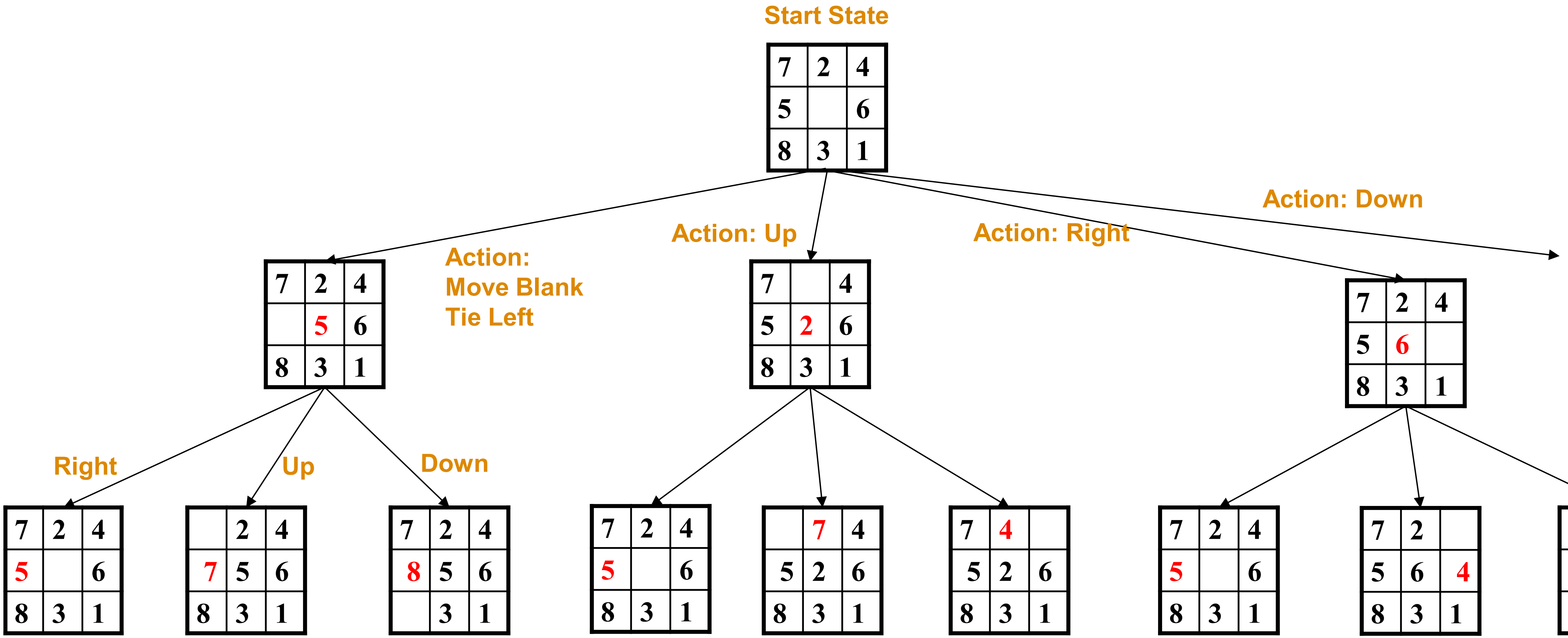
Search Strategies

Several classic search algorithms differ only by the order of how they expand their search trees

You can implement them by using different queue data structures

- **Depth-first search** = LIFO queue
- **Breadth-first search** = FIFO queue
- **Greedy best-first search** or **A* search** = Priority queue

8-Puzzle Breadth-first search



Search Algorithms

Dimensions for evaluation

- ***Completeness***- always find the solution?
- ***Optimality*** - finds a least cost solution (lowest path cost) first?
- ***Time complexity*** - # of nodes generated *(worst case)*
- ***Space complexity*** **- # of nodes simultaneously in memory** *(worst case)*

Time/space complexity variables

- *b*, *maximum branching factor* of search tree
- *d*, *depth* of the shallowest goal node
- *m*, *maximum length* of any path in the state space (potentially ∞)

Properties of breadth-first search

Complete? Yes (if *b* is finite)

Optimal? Yes, if cost = 1 per step (not optimal in general)

Time Complexity? $1+b+b^2+b^3+\ldots+b^d = O(b^d)$

Space Complexity? $O(b^d)$ (keeps every node in memory)

Time/space complexity variables

- *b*, *maximum branching factor* of search tree
- *d*, *depth* of the shallowest goal node
- *m*, *maximum length* of any path in the state space (potentially ∞)

BFS versus DFS

Time/space complexity variables

- *b*, *maximum branching factor* of search tree
- *d*, *depth* of the shallowest goal node
- *m*, *maximum length* of any path in the state space (potentially ∞)

Breadth-first

- ☑ Complete,
- ☑ Optimal
- ☒ *but* uses $O(b^d)$ space

Depth-first

- ☒ Not complete *unless m is bounded*
- ☒ Not optimal
- ☒ Uses $O(b^m)$ time; terrible if m >> d
- ☑ *but* only uses ***O(b*m) space***

Exponential Space (and time) Is Not Good…

- Exponential complexity uninformed search problems *cannot* be solved for any but the smallest instances.
- *(Memory* requirements are a bigger problem than *execution* time.)

DEPTH	NODES	TIME	MEMORY
2	**110**	**0.11 milliseconds**	**107 kilobytes**
4	**11110**	**11 milliseconds**	**10.6 megabytes**
6	**10^6**	**1.1 seconds**	**1 gigabytes**
8	**10^8**	**2 minutes**	**103 gigabytes**
10	**10^{10}**	**3 hours**	**10 terabytes**
12	**10^{12}**	**13 days**	**1 petabytes**
14	**10^{14}**	**3.5 years**	**99 petabytles**

Assumes b=10, 1M nodes/sec, 1000 bytes/node

Action Castle

Art: Formulating a Search Problem

Decide:

Which properties matter & how to represent

- *Initial State, Goal State, Possible Intermediate States*

Which actions are possible & how to represent

- *Operator Set: Actions and Transition Model*

Which action is next

- *Path Cost Function*

Formulation greatly affects combinatorics of search space and therefore speed of search

Action Castle Map Navigation

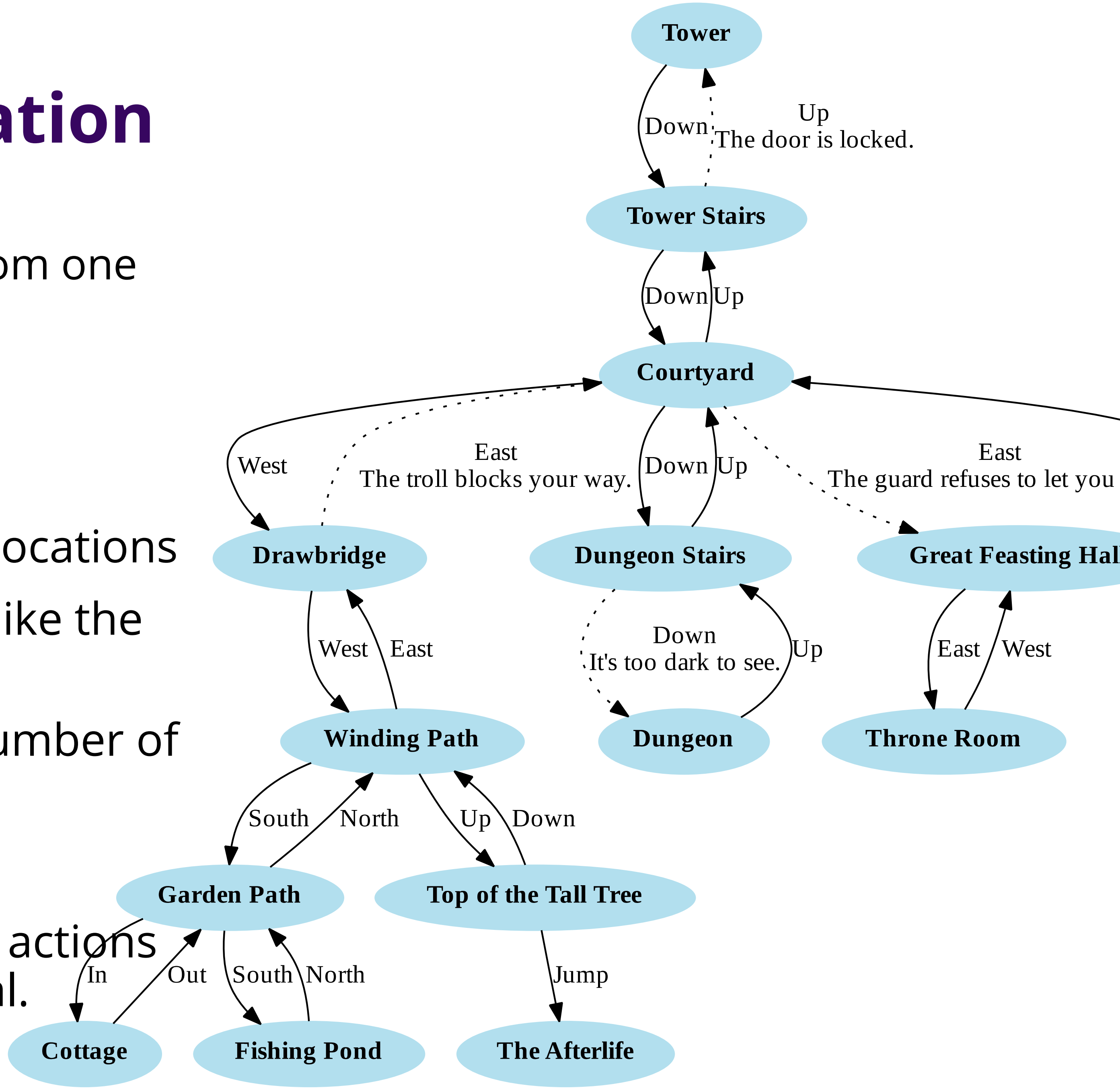
Let's consider the sub-task of navigating from one location to another.

Formulate the ***search problem***

- States: locations in the game
- Actions: move between connected locations
- Goal: move to a particular location like the **Throne Room**
- Performance measure: minimize number of moves to arrive at the goal

Find a ***solution***

- Algorithm that returns sequence of actions to get from the start sate to the goal.



```python
def BFS(game, goal_conditions):
  command_sequence = []
  if goal_test(game, goal_conditions): return co

  frontier = queue.Queue()
  frontier.put((game, command_sequence))

  visited = dict()
  visited[get_state(game)] = True

  while not frontier.empty():
    (current_game, command_sequence) = frontier.get()
    current_state = get_state(current_game)
    parser = Parser(current_game)
    available_actions = get_available_actions(current_game)

    for command in available_actions:
      # Clone the current game with its state
      new_game = copy.deepcopy(current_game)
      # Apply the command to it to get the resulting state
      parser = Parser(new_game)
      parser.parse_command(command)
      new_state = get_state(new_game)
      # Update the sequence of actions that we took to get to the resulting state
      new_command_sequence = copy.copy(command_sequence)
      new_command_sequence.append(command)
      if not new_state in visited:
        visited[new_state] = True
        if goal_test(new_game, goal_conditions):
        frontier.put((new_game, new_command_sequence))
  # Return None to indicate there is no solution.
  return None
```

The frontier tracks order of unexpanded search nodes. Here we're using a FIFO queue

The visited dictionary prevents us from revising states.

TODO: implement get_state()

get_available_actions() to return all commands that could be used here.

TODO: implement get_available_actions()

The parser can execute this command to get the resulting state.

Check to see if this state satisfies the goal test, if so, return the command sequence that got us here.

TODO: implement goal_test()

```python
def BFS(game, goal_conditions):
    command_sequence = []
    if goal_test(game, goal_conditions): return command_sequence

    frontier = queue.Queue()
    frontier.put((game, command_sequence))

    visited = dict()
    visited[get_state(game)] = True

    while not frontier.empty():
        (current_game, command_sequence) = frontier.get
        current_state = get_state(current_game)
        parser = Parser(current_game)
        available_actions = get_available_actions(current_game)

        for command in available_actions:
            # Clone the current game with its state
            new_game = copy.deepcopy(current_game)
            # Apply the command to it to get the resulting state
            parser = Parser(new_game)
            parser.parse_command(command)
            new_state = get_state(new_game)
            # Update the sequence of actions that we took to get to the resulting state
            new_command_sequence = copy.copy(command_sequence)
            new_command_sequence.append(command)
            if not new_state in visited:
                visited[new_state] = True
                if goal_test(new_game, goal_conditions):
                frontier.put((new_game, new_command_sequence))
    # Return None to indicate there is no solution.
    return None
```

Tip: We can store multiple objects on the frontier as a tuple.

Tip: To be used a key in the dictionary get_state() must return an immutable object

Tip: use deepcopy here

Tip: For BFS, apply the goal test before putting the new item on the frontier

Action Castle

Let's consider the full game.

Actions

Start State

Transitions

State Space

Goal test

Actions

Go

Move to a location

Get

Add an item to inventory

Special

Perform a special action with an item like "Catch fish with pole"

~~**Drop**~~

~~Leave an item in current location~~

State Info

Location of Player

Items in their inventory

Location of all items / NPCs

Blocks like

- **Troll guarding bridge,**
- **Locked door to tower,**
- **Guard barring entry to castle**

In-Class Activity

https://laramartin.net/interactive-fiction-class//in_class_activities/search/action-castle-search.html

https://bit.ly/4eSjws8