State Space Search

To solve a particular problem, we need to do 3 things:

1. **Define** the problem precisely.
2. **Isolate and represent** the task knowledge that is necessary to solve the problem.
3. **Choose and apply** the best problem-solving technique(s) to the particular problem.

Problem Representation

- The key to effective problem solving is good representation. A good representation scheme has the following properties:
  - **Make important objects and relations explicit.**
  - **Suppress unimportant and irrelevant detail.**
  - **Expose natural constraints.**
  - **Concise – efficiently describe a given scenario.**
  - **Complete – everything necessary can be described.**

States and Operators

- One way is to think in terms of states. A state is a description of a system at some given point in time.

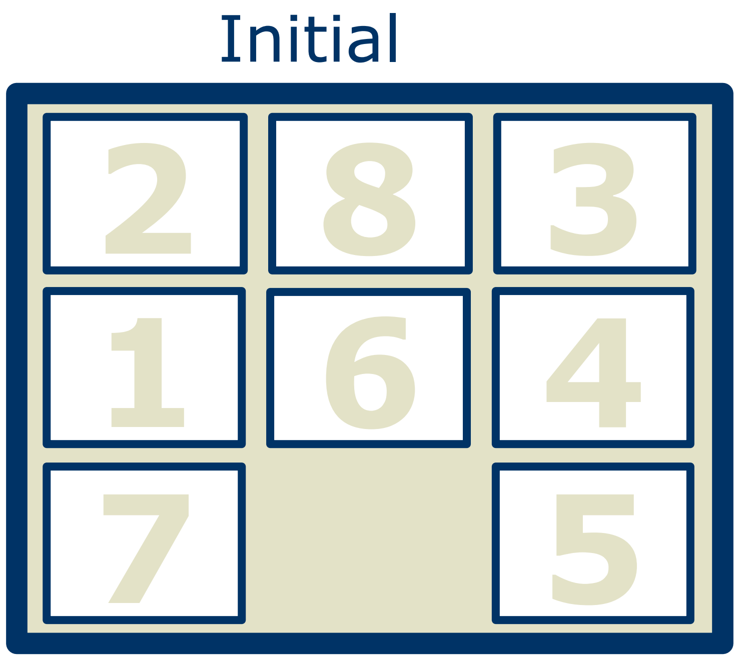
- Many problems can be formally defined by the following components:

    - An **start** state

    - A **goal** state

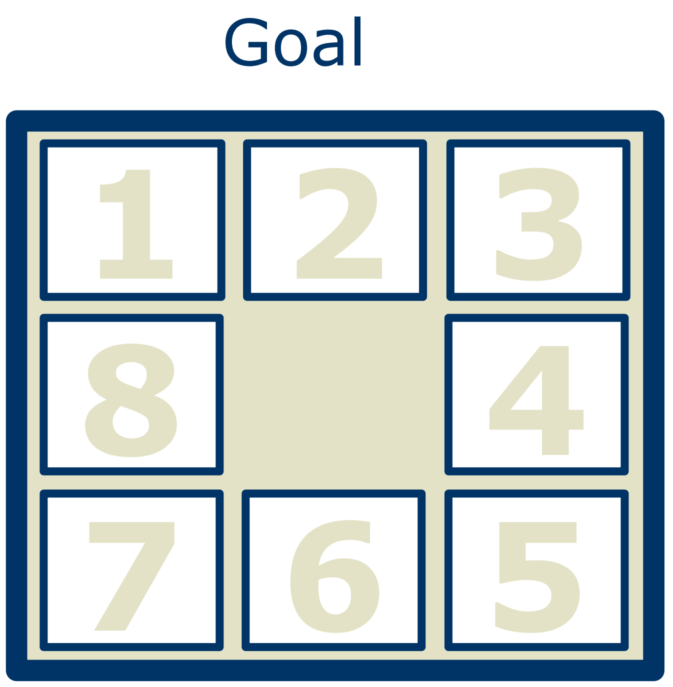
    - An **operator** set - each operator can transform one state into another.

The 8-Puzzle



Q: How many operators?

A: 32? That is, move 1 left, move 1 right, move 1 up, move 1 down, move 2 left, …



The reality…

We only need 4 operators for moving the blank:

if the blank is not at top edge **then** move it up

if the blank is not at bottom edge **then** move it down

if the blank is not at left edge **then** move it left

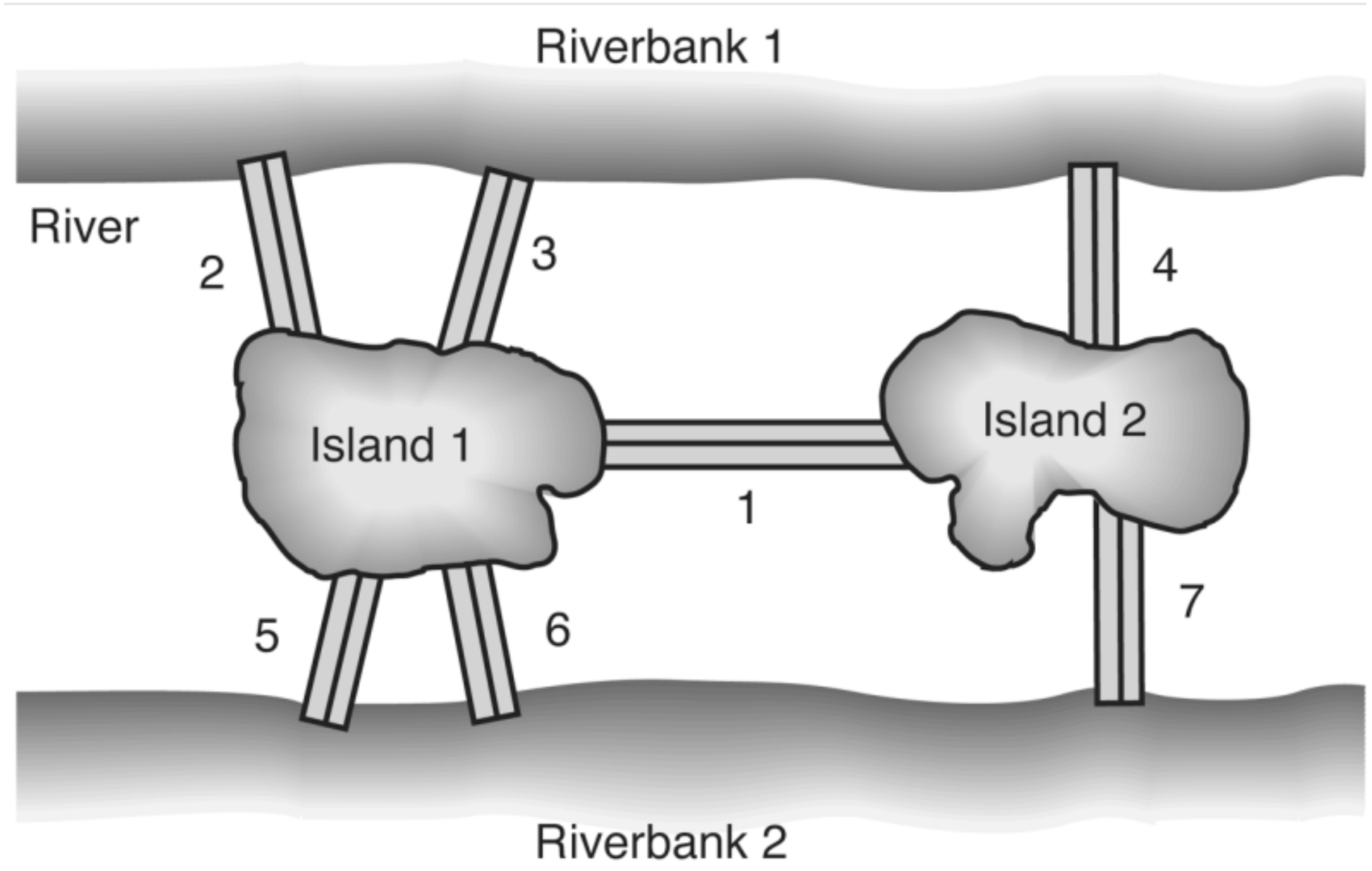
if the blank is not at right edge **then** move it right

This state space has 9! = 362,880 states.

How many reachable states?

Euler & the Bridges of Konigsberg

- Can we walk through the city of Konigsberg and cross each of its seven bridges exactly once?



The Farmer Jones River Problem

Farmer Jones owns a small boat (Bounty) goat, a pet wolf (don't ask!), and a prize cabbage.

To get to the annual cabbage show he must cross from the south bank of the river to the north bank, but Bounty will only allow him to travel with one other object (it's a big cabbage!).

If left unattended the wolf will eat the goat and the goat will eat the cabbage.

How can farmer Jones take his goat, wolf and cabbage to the show?

Representing the Problem

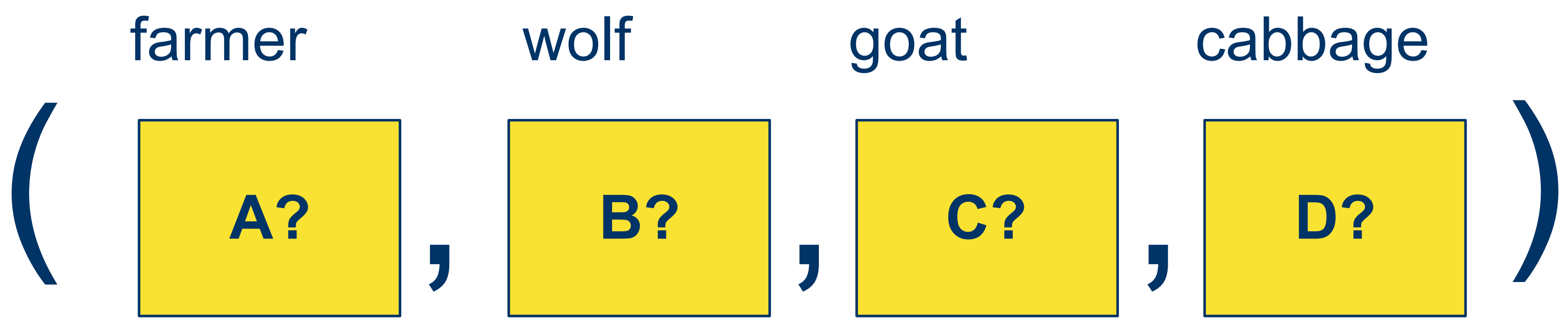
- Described in English the river problem is confusing enough to be difficult to solve. It emphasises irrelevant details (the wolf being a family pet, the name of the boat etc.) and disguises very relevant information (the positions of the objects, etc.).

- Problem representation is very important…
  - **Make important objects and relations explicit.**
  - **Suppress unimportant and irrelevant detail.**
  - **Expose natural constraints.**
  - **Concise – efficiently describe a given scenario.**
  - **Complete – everything necessary can be described.**

- We can come up with a much better representation.

For Example…

- Consider the Farmer Jones problem again. In this problem information can be represented in the form of the following 4-tuple.



- Each variable position stores the location of a particular object.
  - For instance, **the initial state** where the farmer, the goat, the wolf and the cabbage are on the south bank is given by **(S,S,S,S)**.
  - The **goal state** is represented by **(N,N,N,N)**.

Example – Defining Operators

- **There are 4 transforming operators:**
  - Farmer-Takes-Self

    $(A?,B?,C?,D?) \longrightarrow (\text{Opposite}(A?), B?,C?,D?)$
  - Farmer-Takes-Wolf

    $(A?,B?,C?,D?) \longrightarrow (\text{Opposite}(A?), \text{Opposite}(B?),C?,D?)$
  - Farmer-Takes-Goat

    $(A?,B?,C?,D?) \longrightarrow (\text{Opposite}(A?), B?, \text{Opposite}(C?),D?)$
  - Farmer-Takes-Cabbage

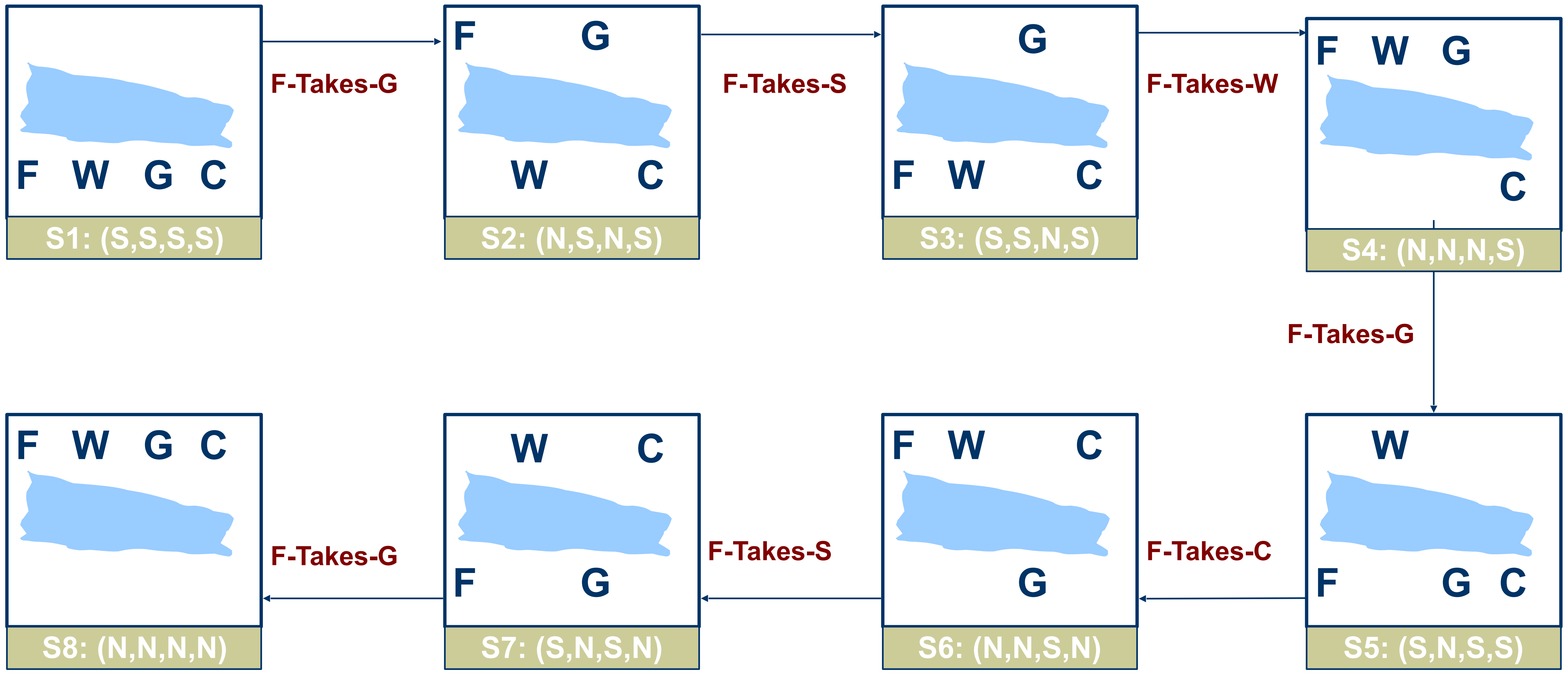
    $(A?,B?,C?,D?) \longrightarrow (\text{Opposite}(A?), B?,C?, \text{Opposite}(D?))$

- **Finally, a state is safe as long as the wolf and the goat or the cabbage are not left unattended – this is the precondition to the above operators.**

How do we represent the constraints?

Example – One Solution…

While there are other possibilities here is one 7 step solution to the river problem.



So Far ...

Problem...

(N N N N) {Start State}

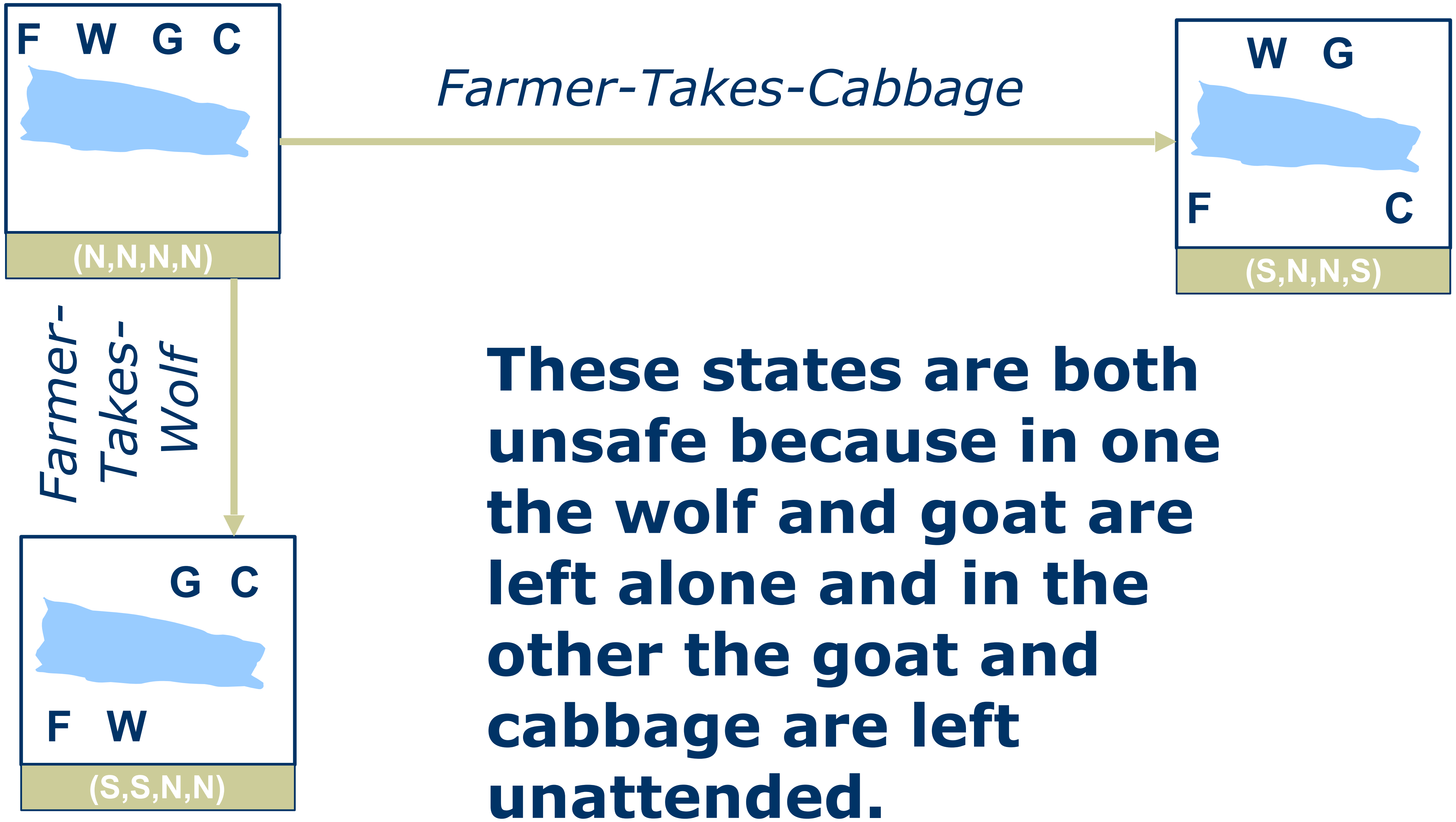
?

(S S S S) {Goal State}

One Solution

$$\begin{tikzcd}
(N\ N\ N\ N) \arrow[d, "\textit{Farmer takes goat}"] \\
(S\ N\ S\ N) \arrow[d, "\textit{Farmer takes self}"] \\
(N\ N\ S\ N) \arrow[d, "\textit{Farmer takes wolf}"] \\
(S\ S\ S\ N) \arrow[d, "\textit{Farmer takes goat}"] \\
(N\ S\ N\ N) \arrow[d, "\textit{Farmer takes cabbage}"] \\
(S\ S\ N\ S) \arrow[d, "\textit{Farmer takes self}"] \\
(N\ S\ N\ S) \arrow[d, "\textit{Farmer takes goat}"] \\
(S\ S\ S\ S)
\end{tikzcd}$$

Safe & Unsafe States



These states are both unsafe because in one the wolf and goat are left alone and in the other the goat and cabbage are left unattended.

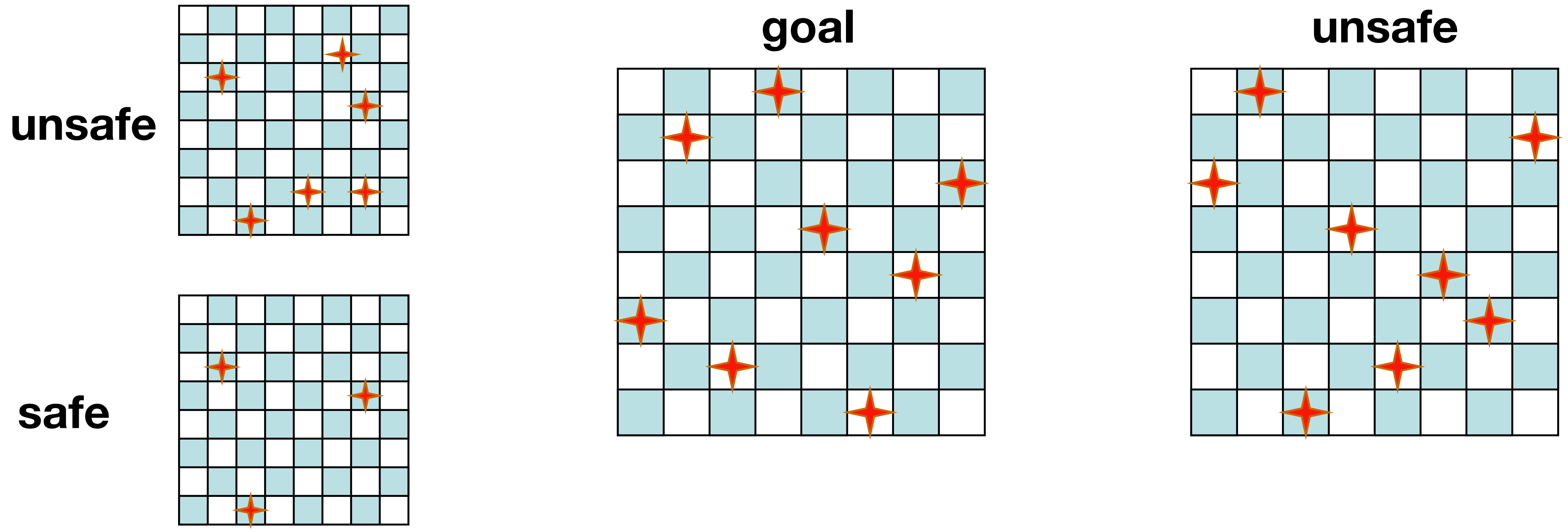
River Problem Search Tree

- Draw the first level of the search tree from (N,N,N,N)
- Expand the SNSN node
- Expand the NNSN node

What is the Actual State Space

- The set of all possible states?
- The set of reachable states (from the start space)?
- The set of safe (or valid) states?

- **8-Queens problem:**
  - ☐ 64 x 63 x 62 x…x 57 = 3x10^4 possible states
  - ☐ Queens only added in safe squares ~ 2000 states
  - ☐ ~ 90 goal states

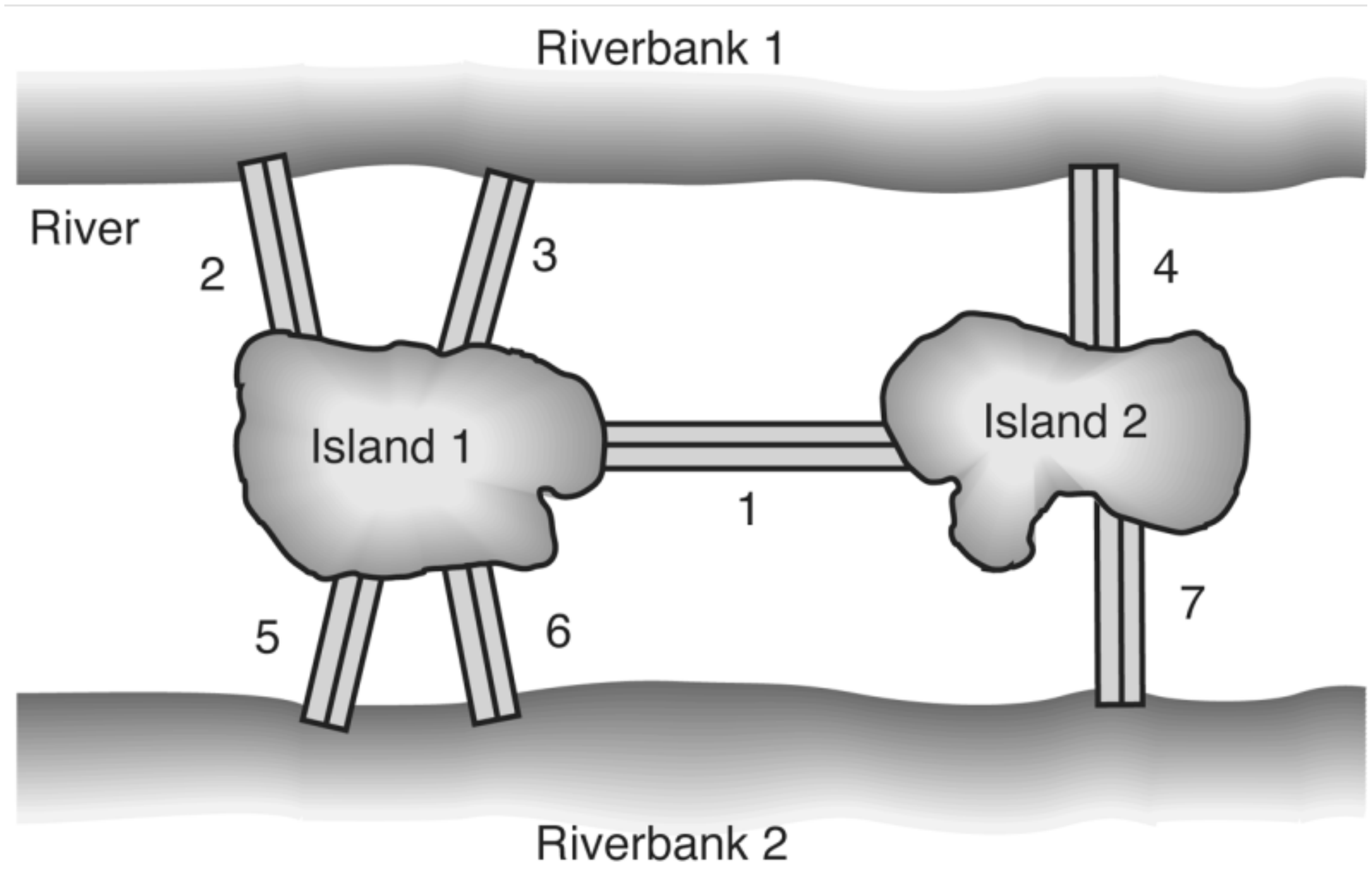


What is the solution?

- Planning Problems:
  - The path from the start state to the goal state **IS** the solution
  - The plan
  - e.g. Jug problem, shortest path, 8-puzzle
- Other Problems:
  - The goal state is the solution
  - e.g. 8-Queens, Travelling Salesperson Problem

The Bridges of Konigsberg

- What is an appropriate State Space representation?
  - Represent the problem as a network - nodes & edges
  - Starting from Riverbank 1 start drawing a search tree

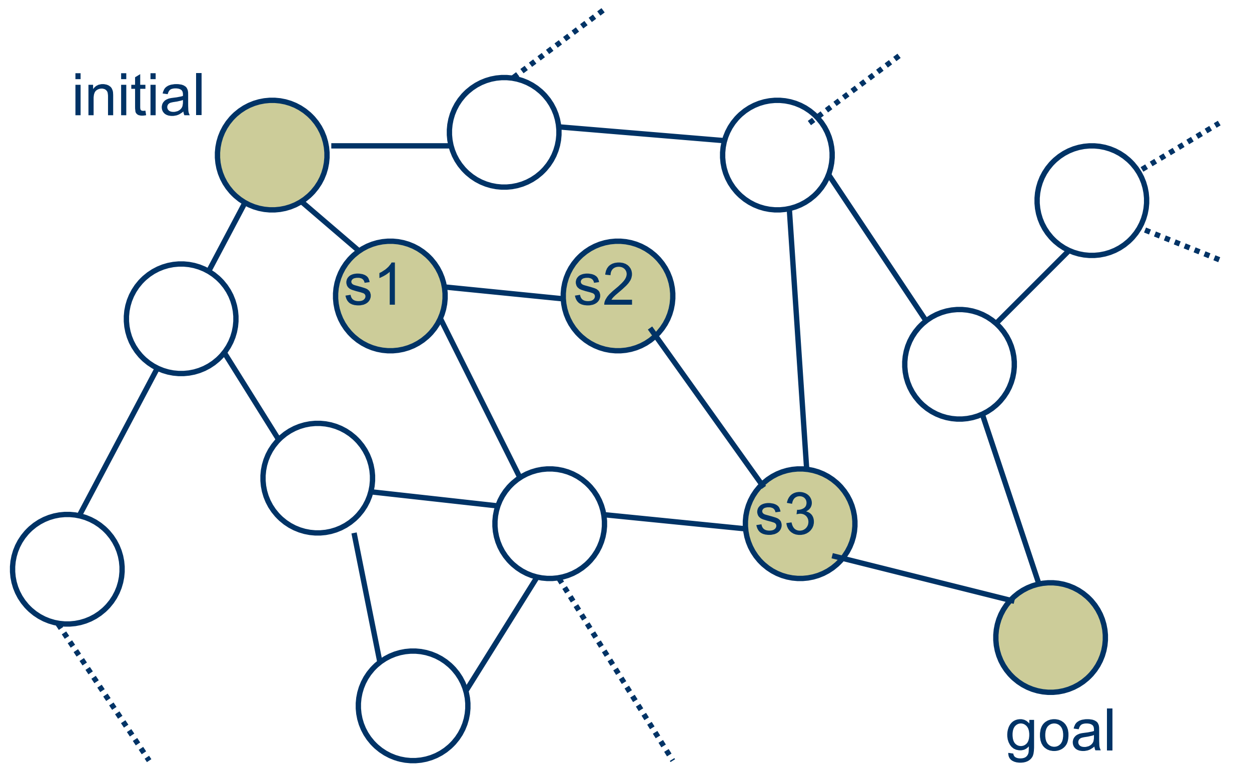


State Space Search

- Many problems can be cast as **search problems** where the objective is to find a **path** from some **start state** to some **goal state**.

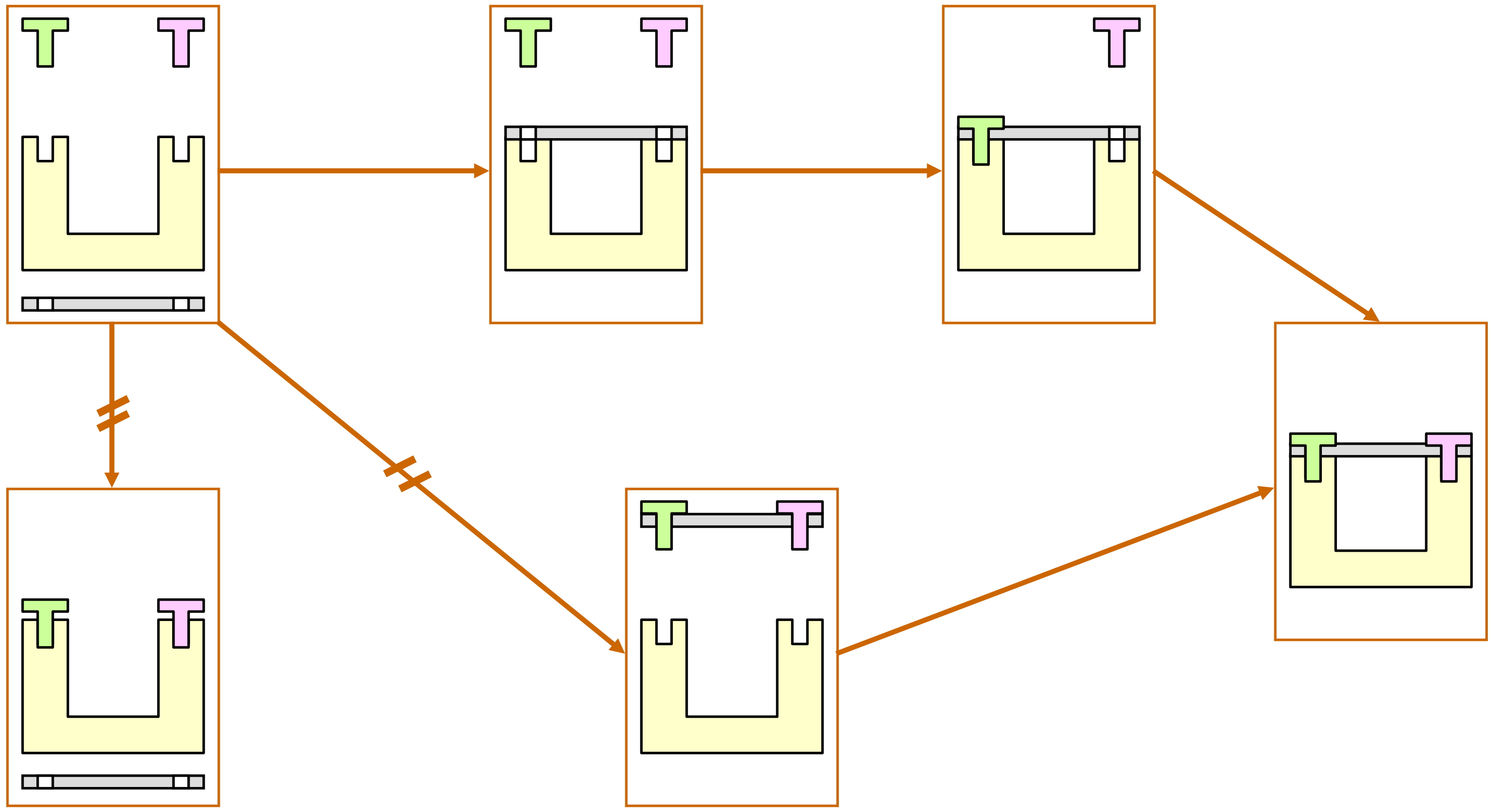
- The resulting path corresponds to the **problem solution** and is made up of a sequence of **actions**.

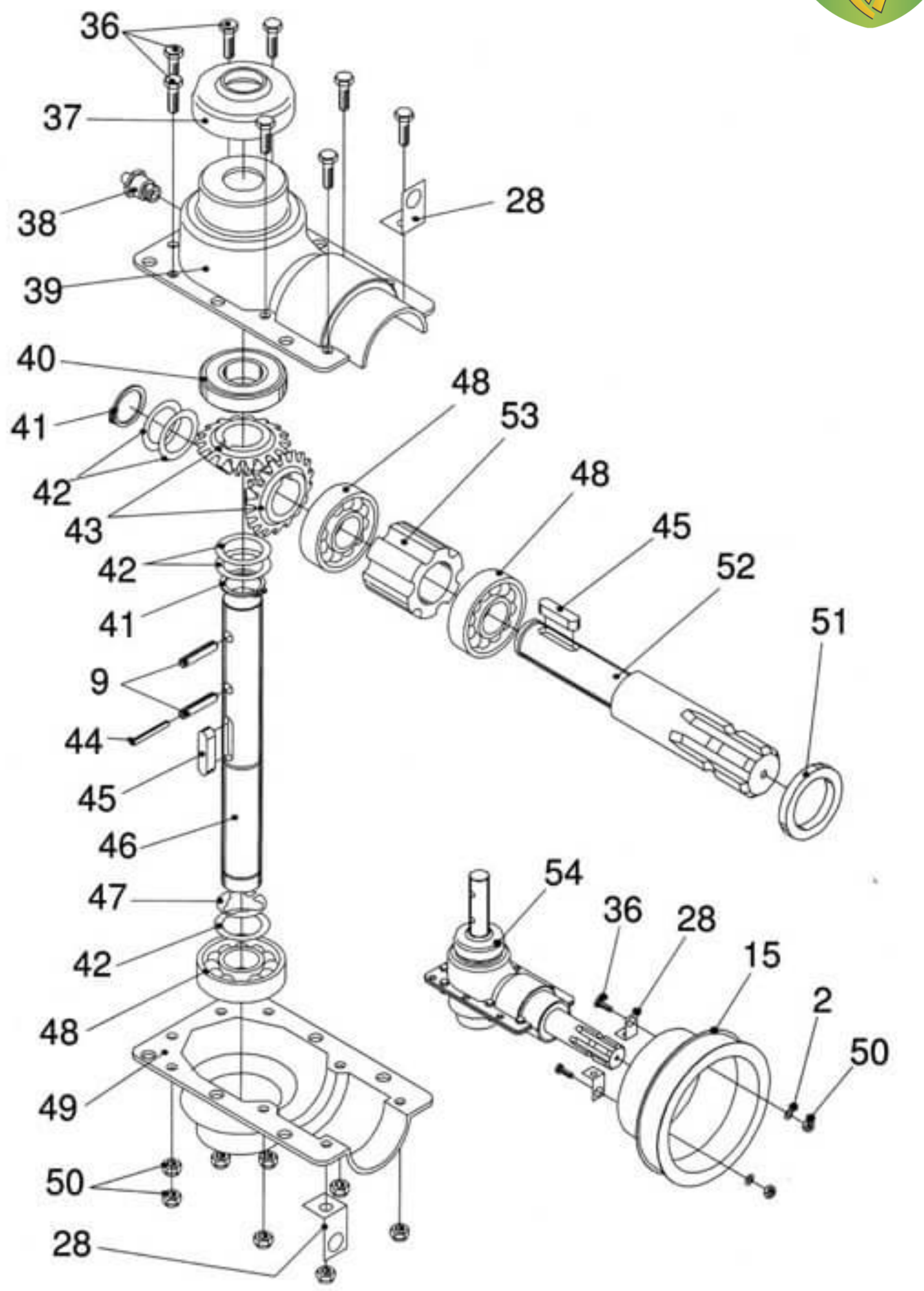
- The construction of the path involves the search for an appropriate sequence of actions and is governed by a **search strategy.**



Real world planning problems…

- Logistics
  - Military
  - Commerce / Trading
- Manufacturing





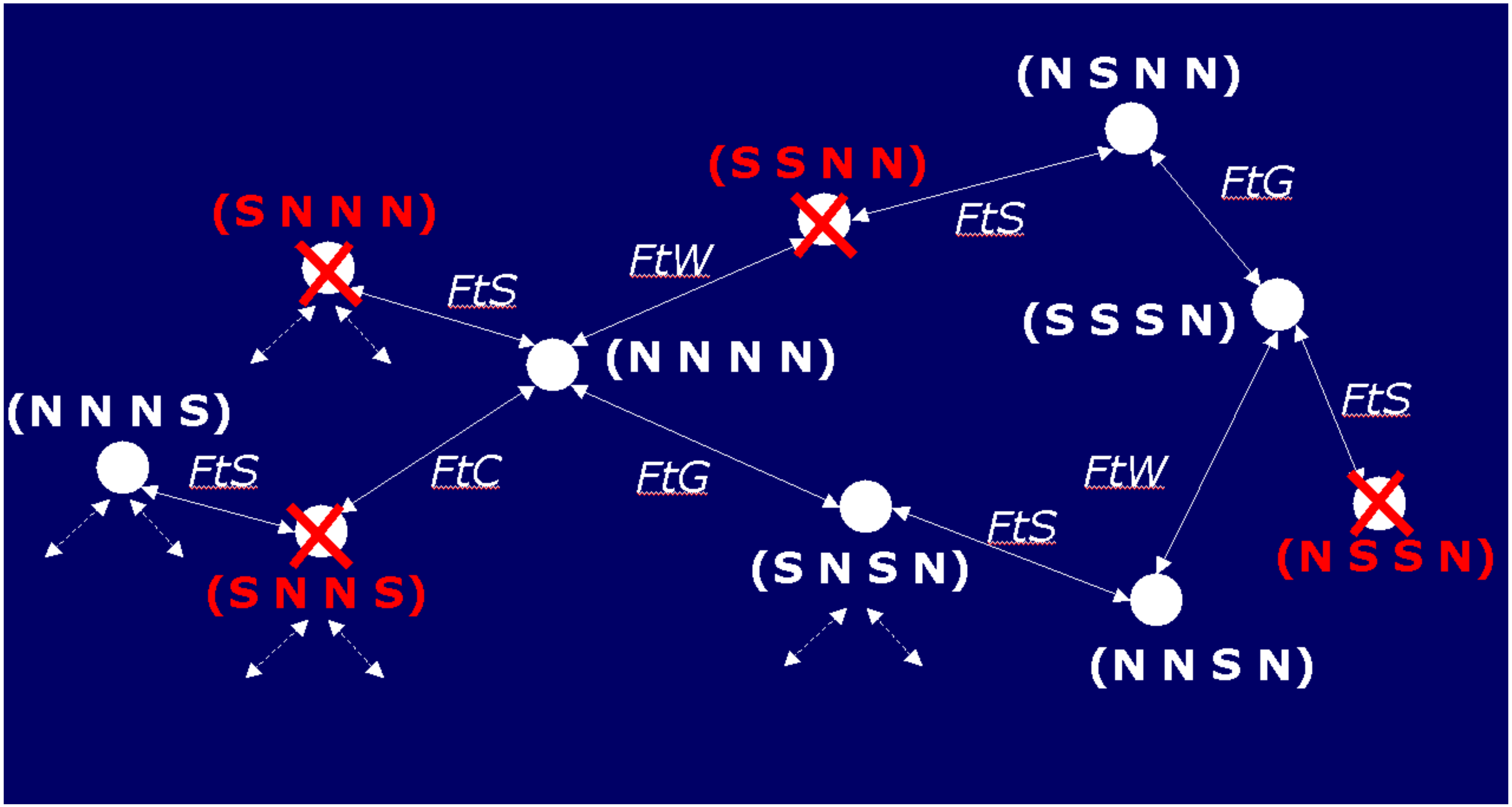
State Space *vs.* Search Tree

- **The state space is the set of all valid states that can exist in our "problem universe".**
  - Each node in the state space corresponds to an individual state.
  - Nodes are directionally connected according by state transforming operators.

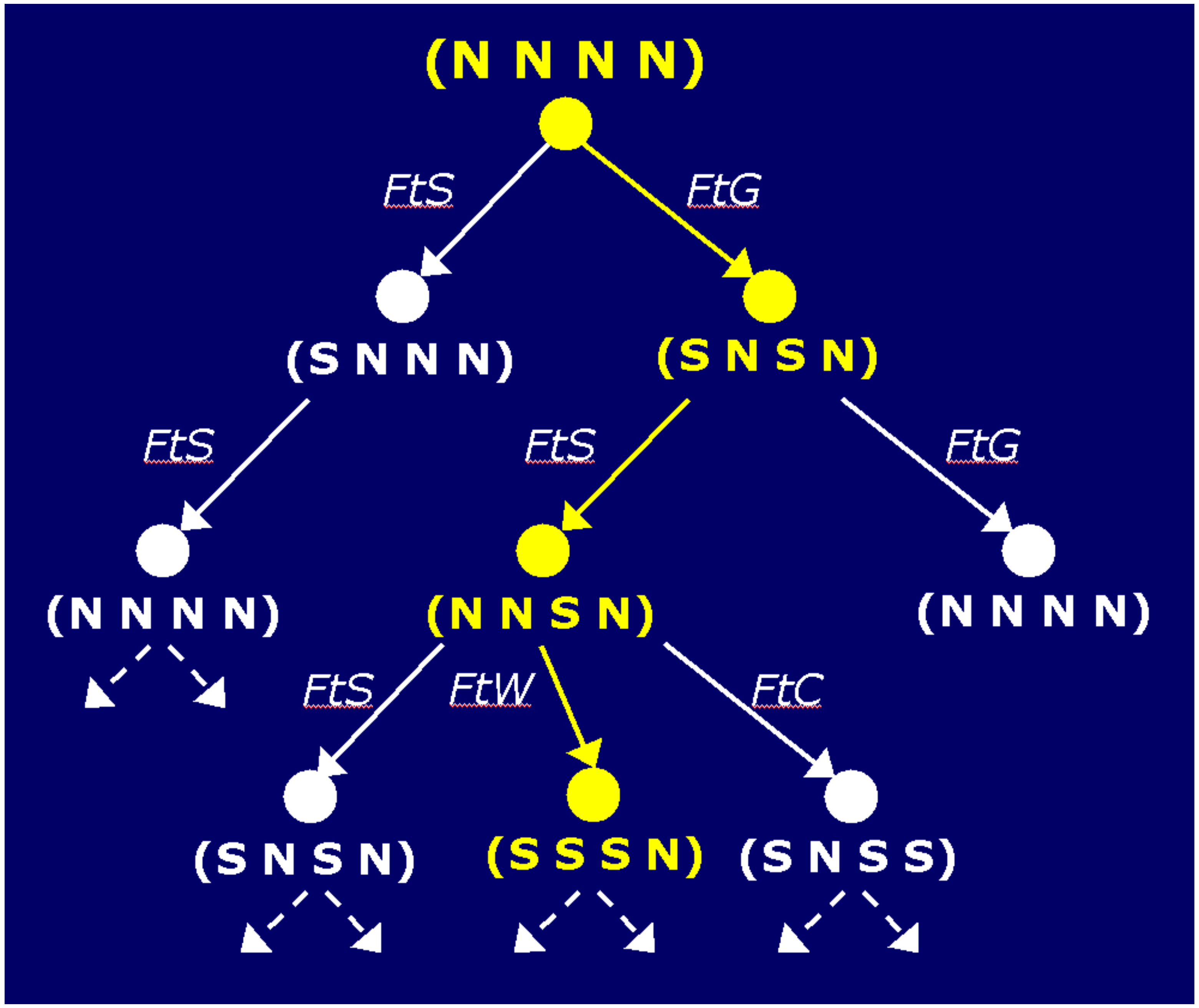
- **The search tree corresponds to the sequence of paths that can be reached from a given start state.**
  - A search tree is rooted at a particular start state.
  - Each node in a search tree corresponds to a sequence of states (path) from the start state (root) to a given state.

- **Search algorithms solve problems by constructing a search tree rooted at the start state and terminating at the goal state(s).**

“Farmer Jones” State Space



“Farmer Jones” Search Tree



What should you know at this stage

Can you answer these questions?

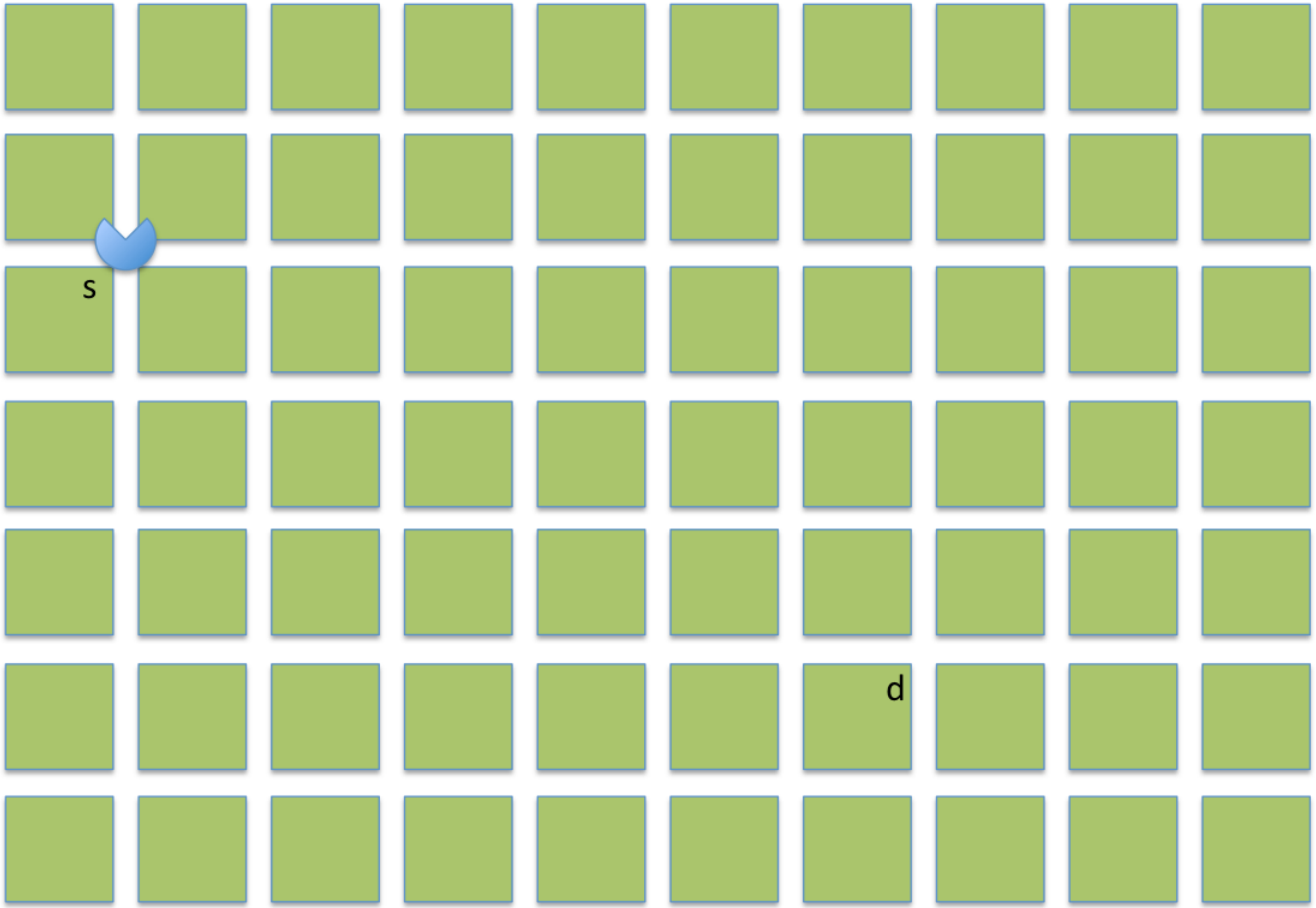
1. Explain with the aid of two examples what state-space search is.
2. For each example, propose a representation for the state space.
3. Discuss the size of the state-space in each case

Remember Pac-man?

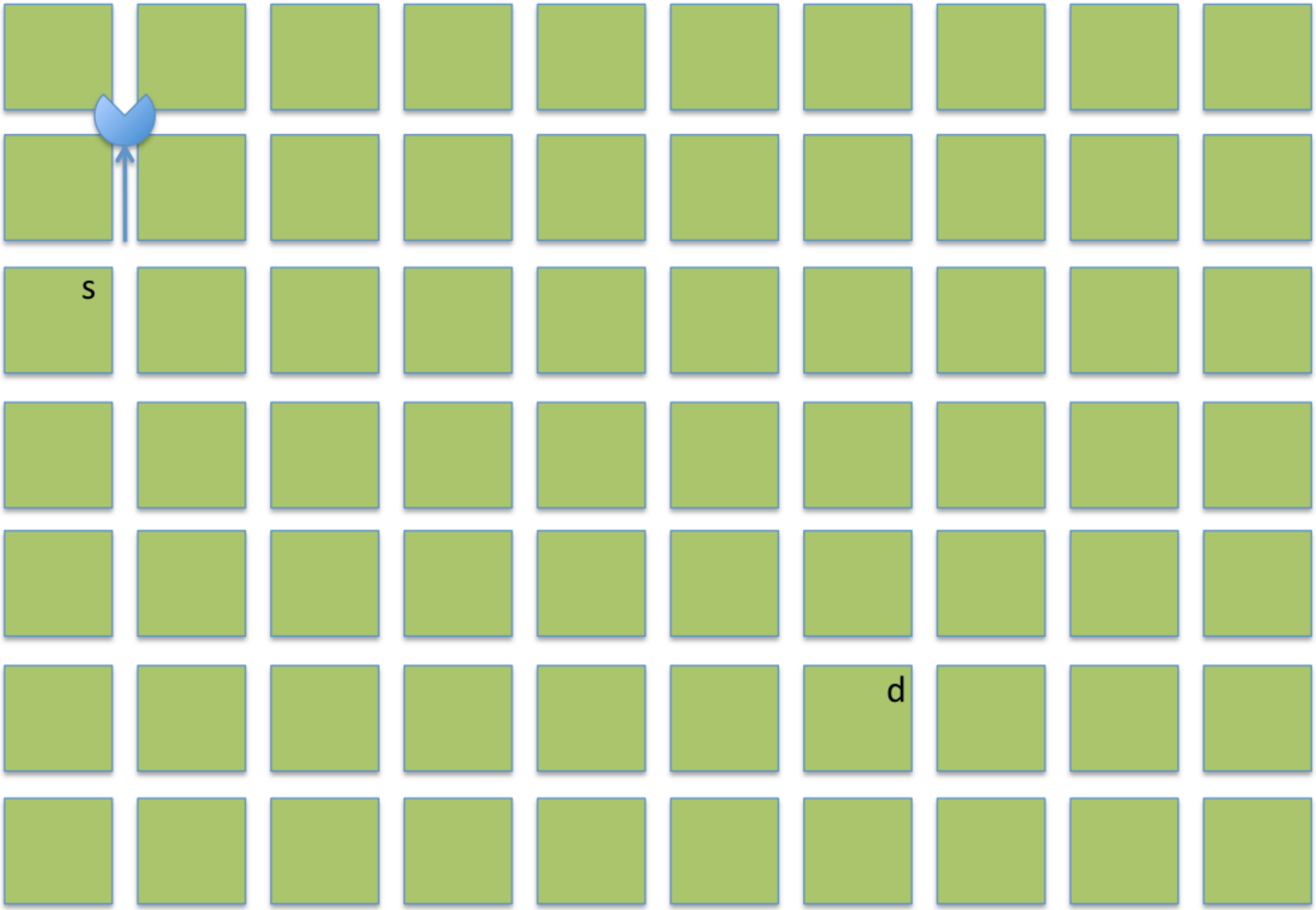
- For those of you who took COMP10110 two years ago, you might remember the following search problem:

Pac-man – a robot with a destination!

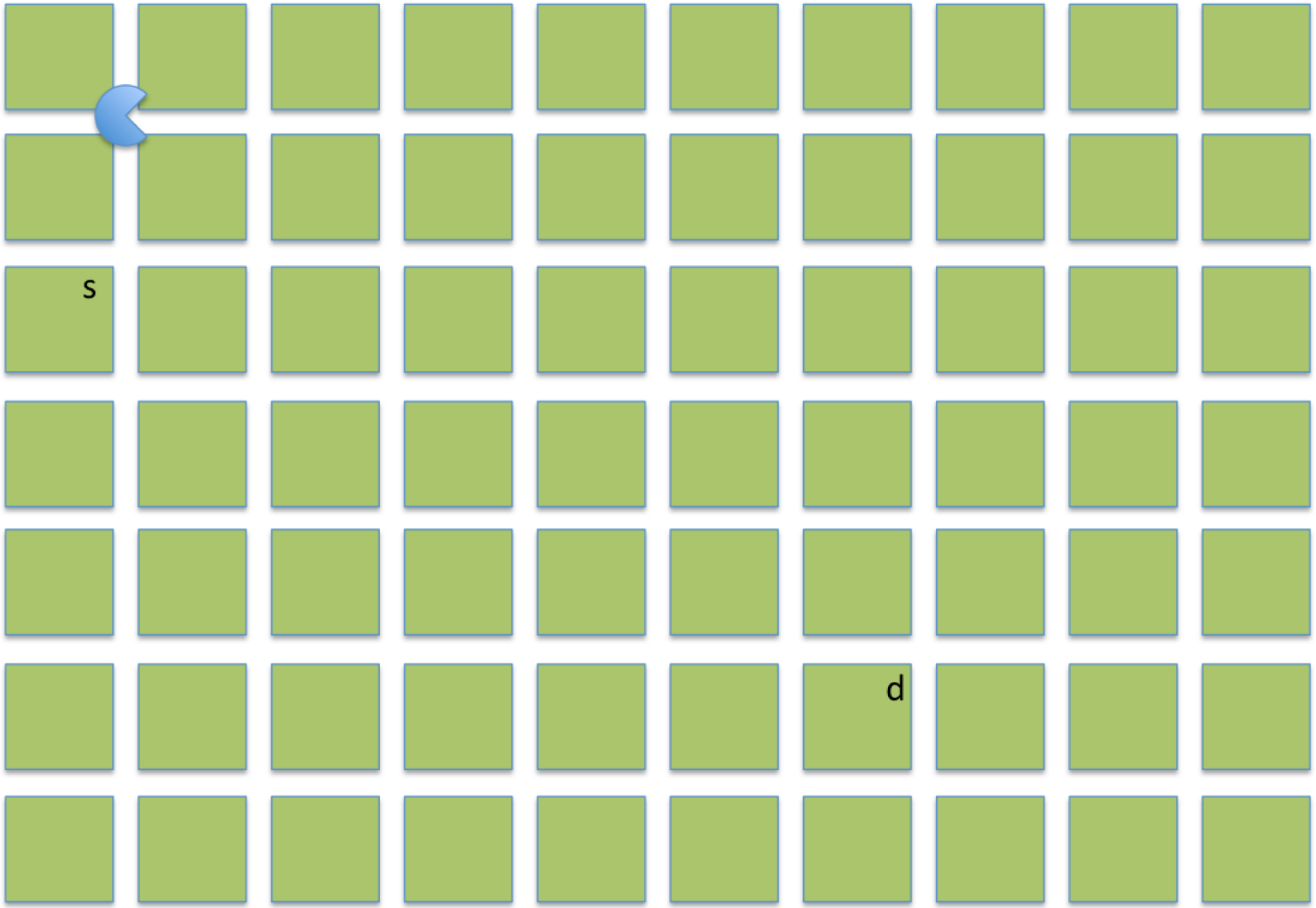
- Consider instead a very simple robot – let's call it Pac-man – navigating through a grid of streets that intersect at junctions.
- Pac-man understands just *three* basic operations
    1. `R` – right turn . . . rotate 90 degrees to the right.
    2. `L` – left turn . . . rotate 90 degrees to the left.
    3. `W` – walk . . . move straight-head in the direction you are facing to the next junction.
- The goal, or problem to solve, is to get Pac-man to navigate from a starting-point (`s`) to a destination (`d`)
- An algorithm to solve this problem will consist of a sequence of operations, `R`, `L` or `W` that successfully moves Pac-man from `s` to `d`.



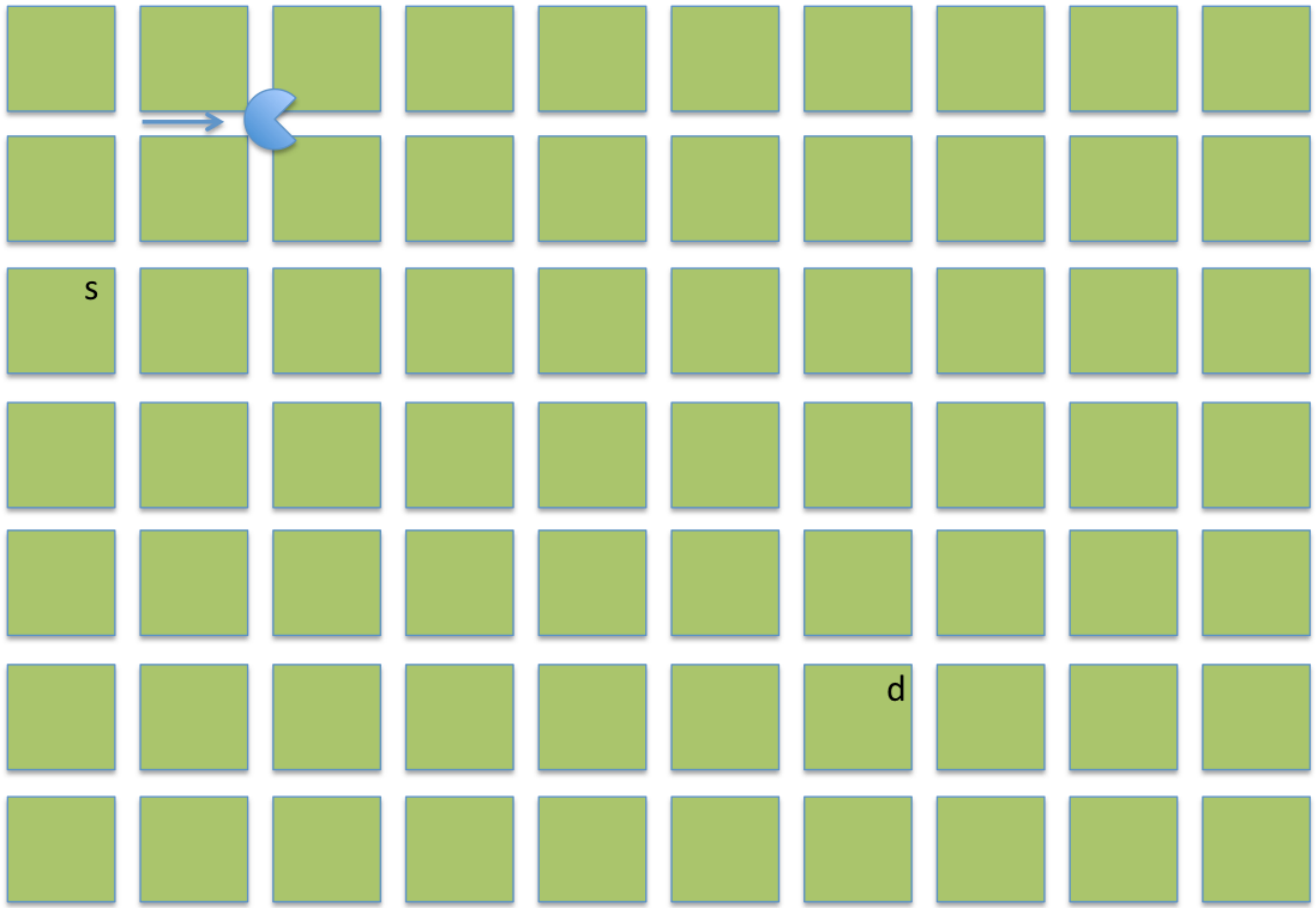
W
R
W
R
W
W
W
W
L
W
W
W
W
W



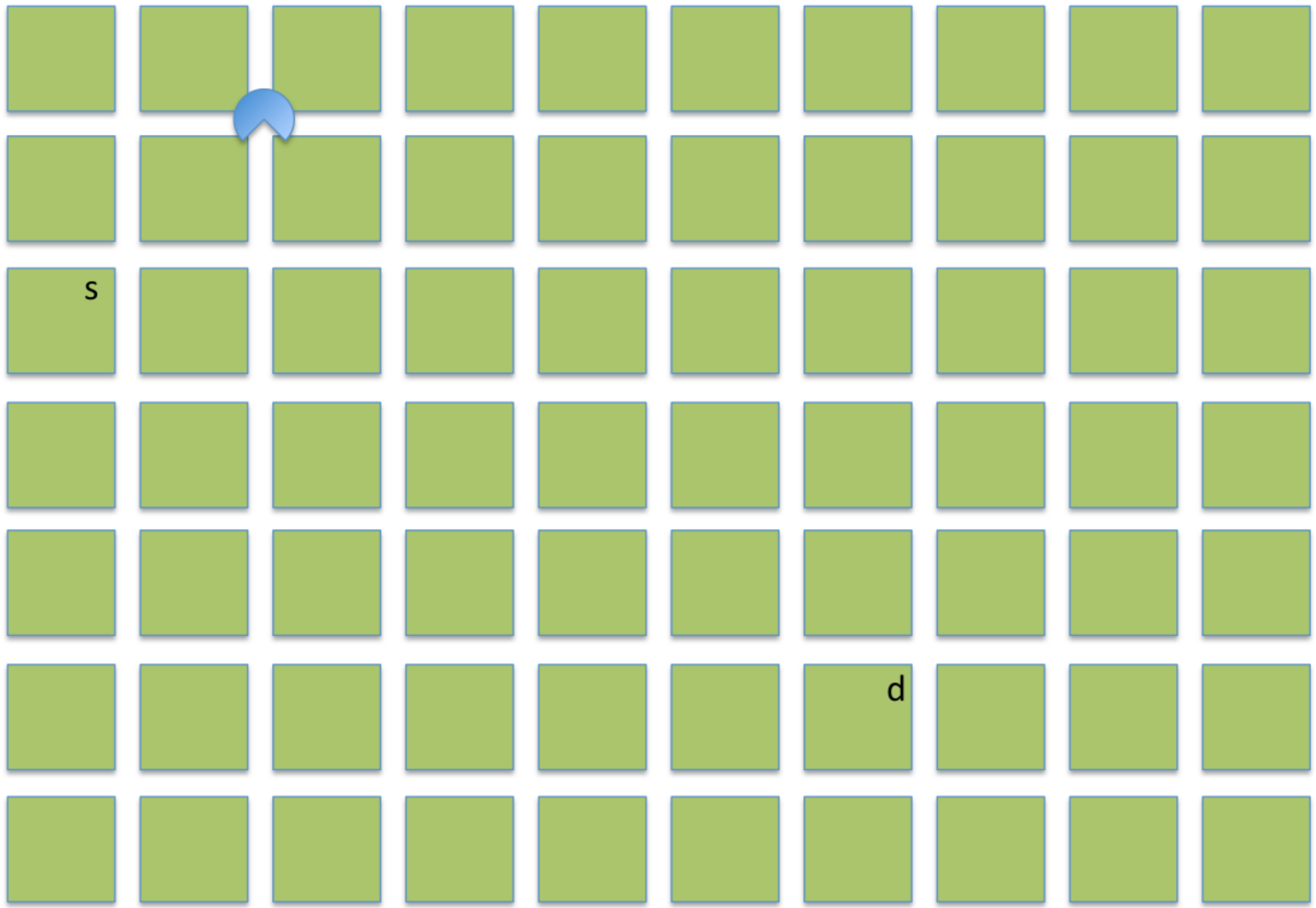
W
R
W
R
W
W
W
W
L
W
W
W
W
W



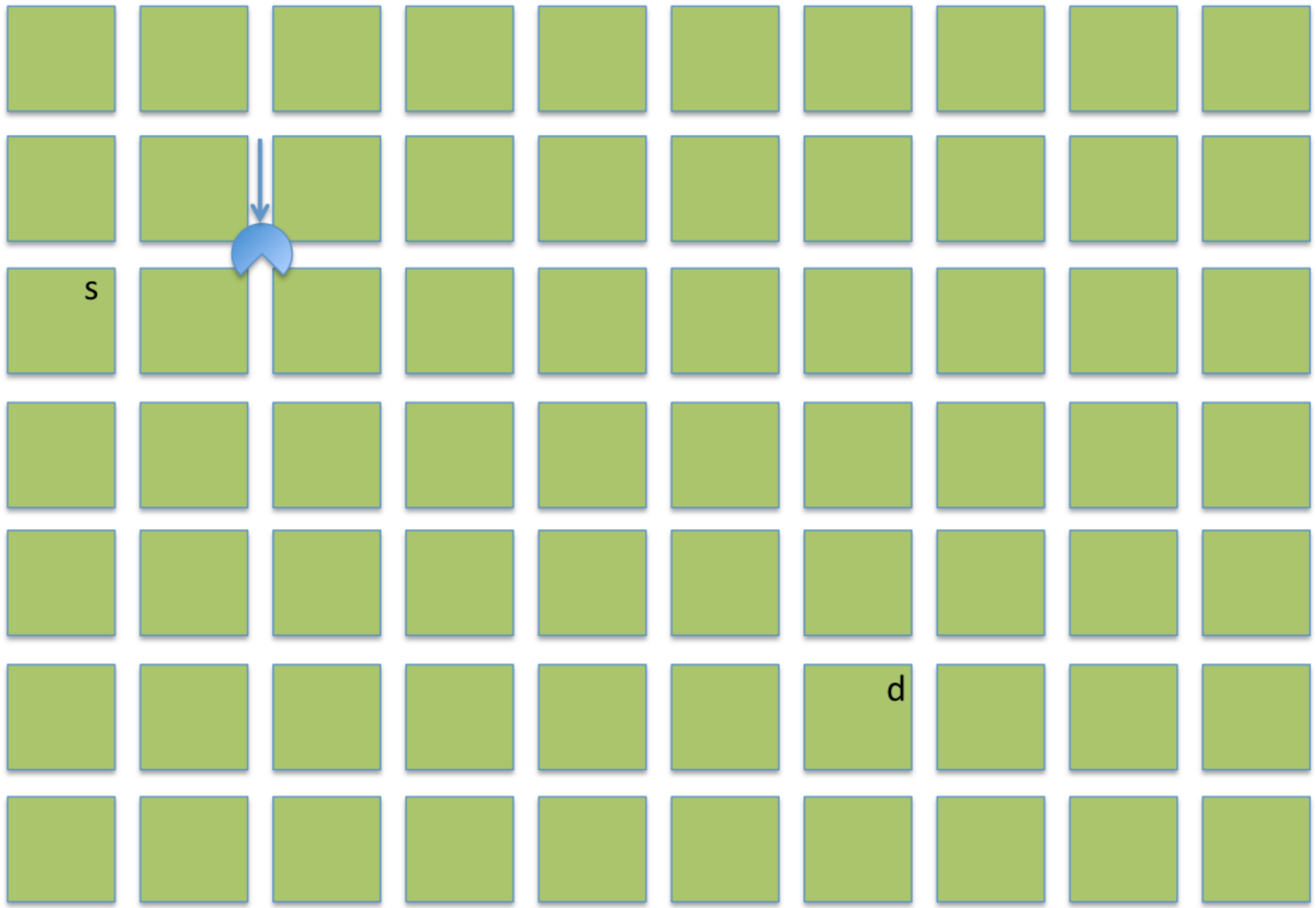
W
R
W
R
W
W
W
W
L
W
W
W
W
W



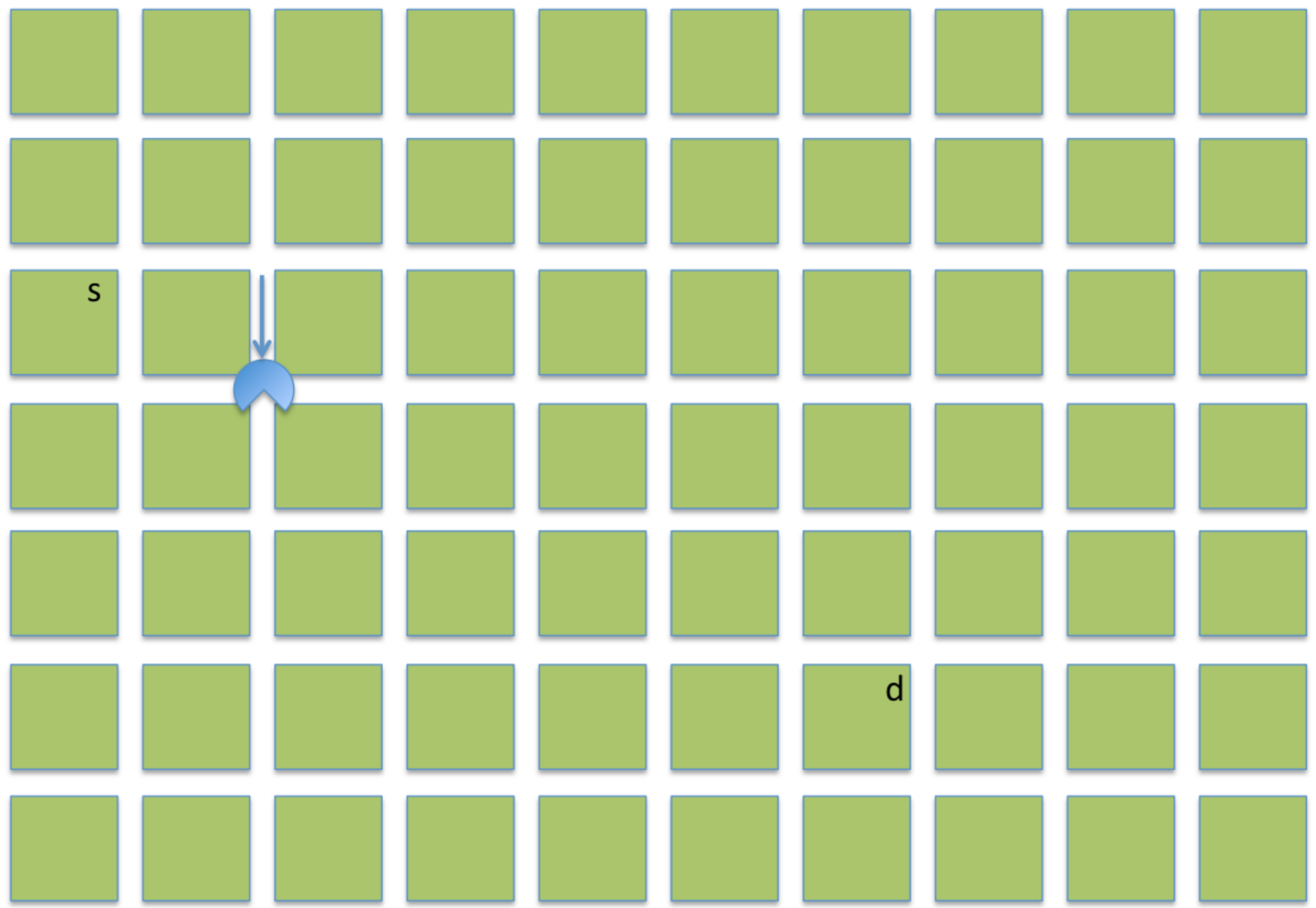
W
R
W
R
W
W
W
W
L
W
W
W
W
W



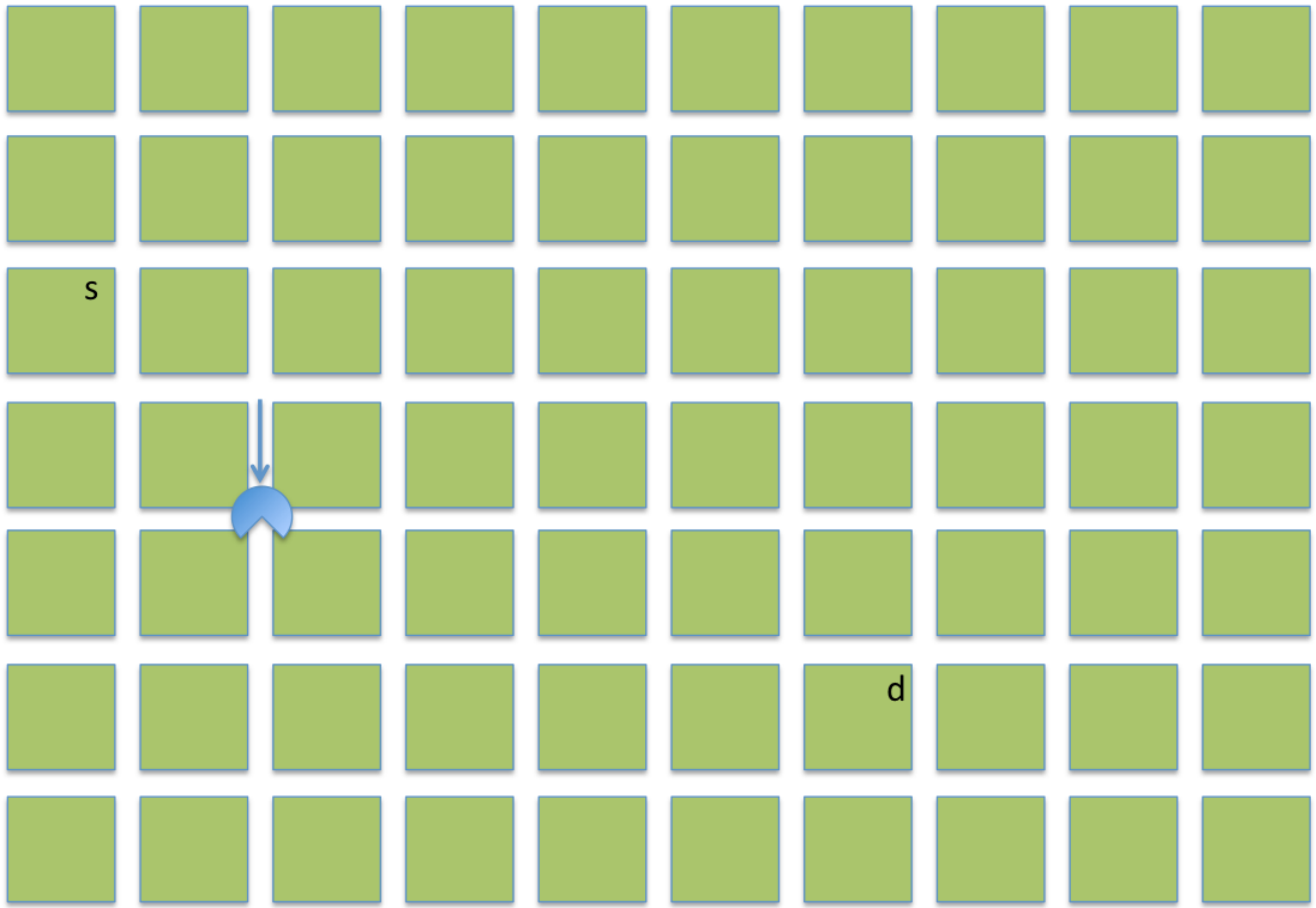
W
R
W
R
W
W
W
W
L
W
W
W
W
W



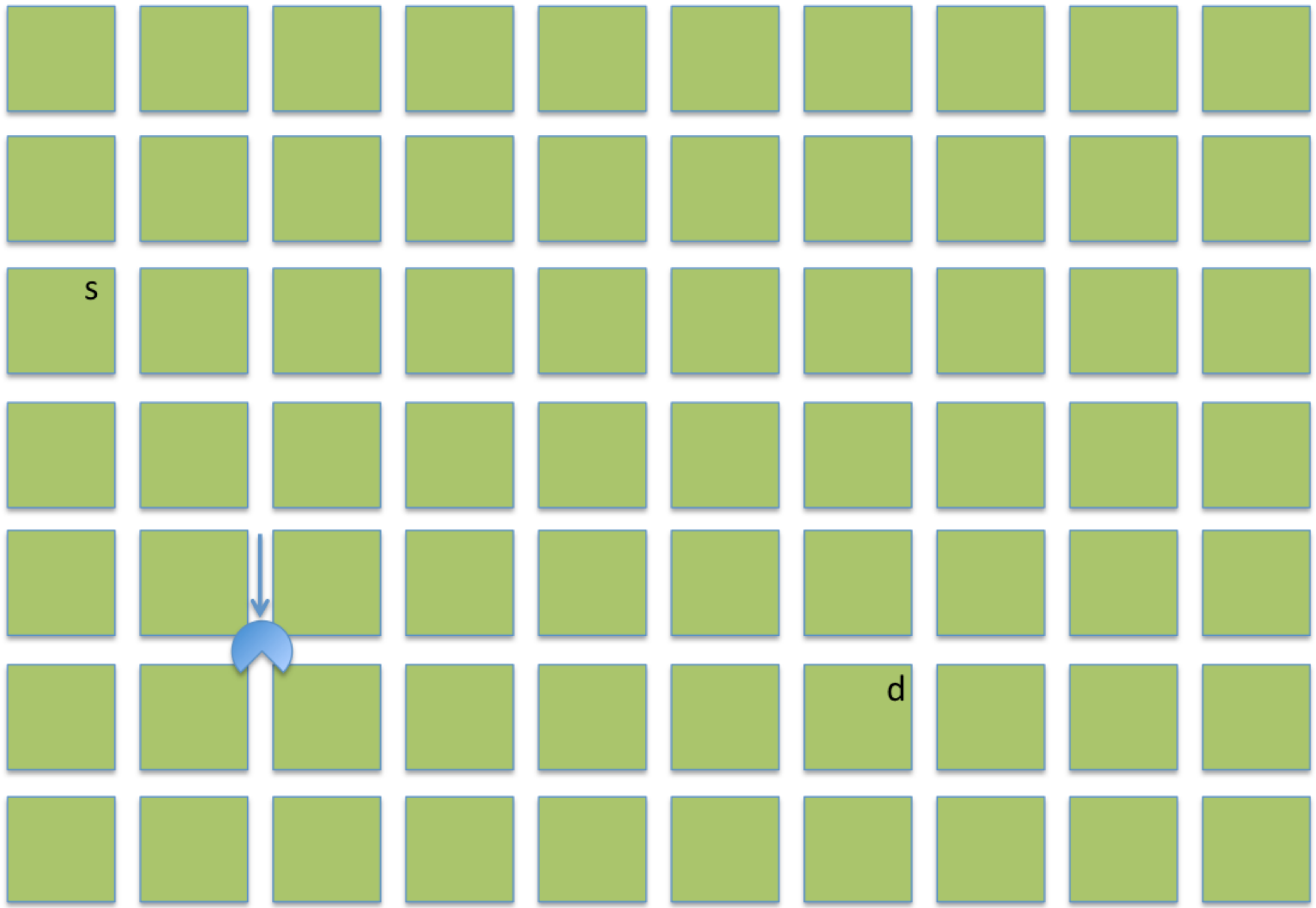
W
R
W
R
W
W
W
W
L
W
W
W
W
W



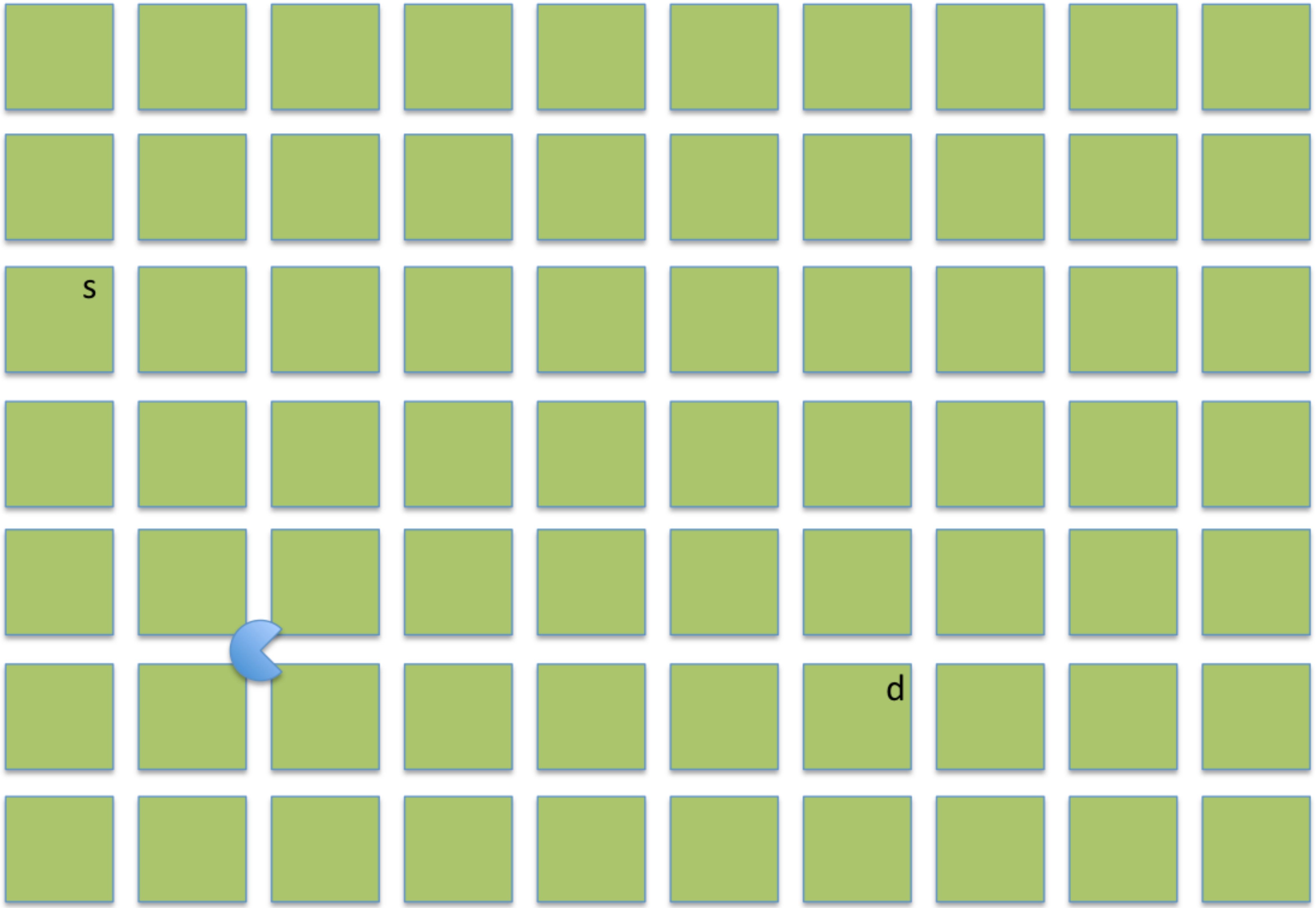
W
R
W
R
W
W
W
W
L
W
W
W
W
W



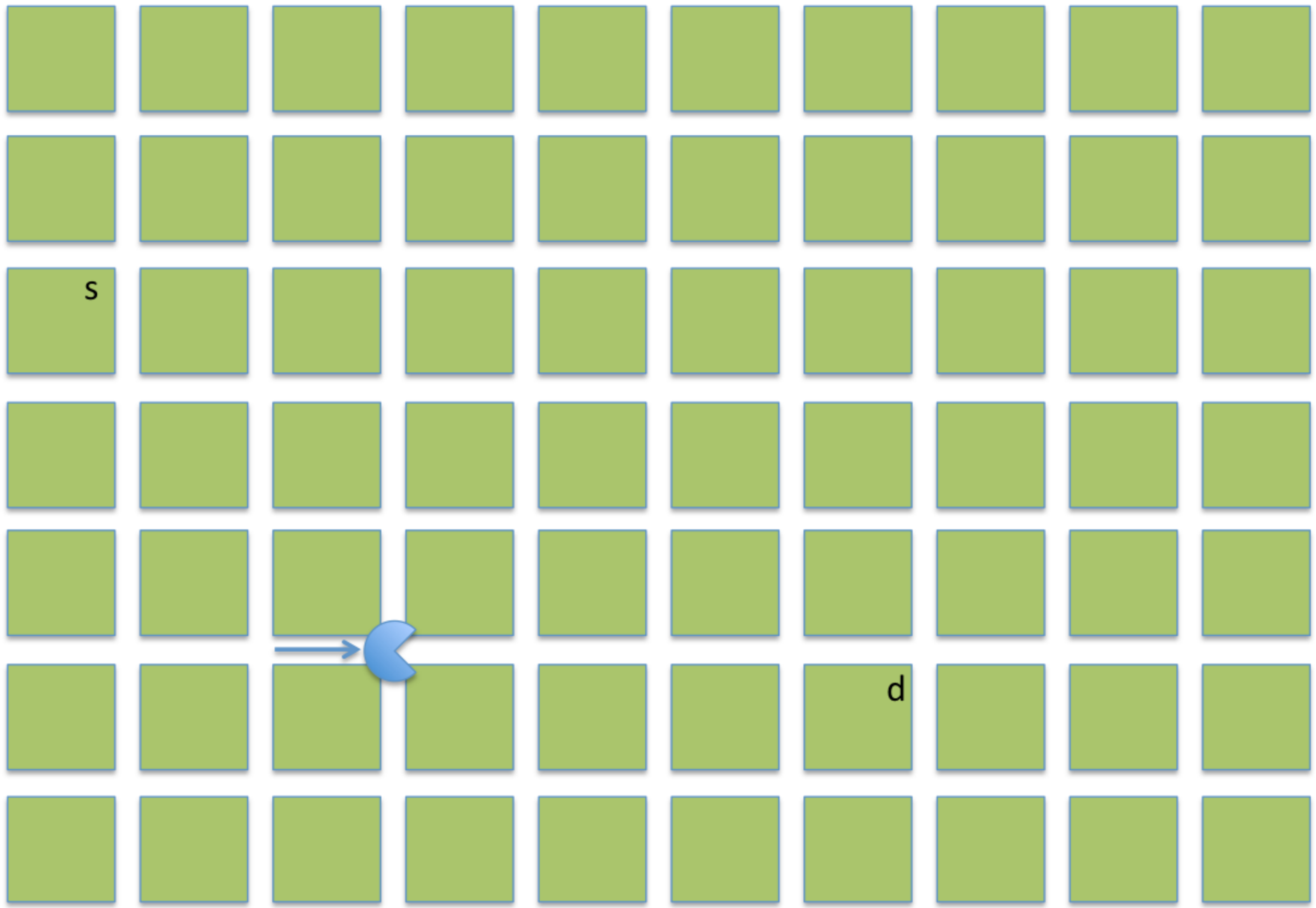
W
R
W
R
W
W
W
W
L
W
W
W
W
W



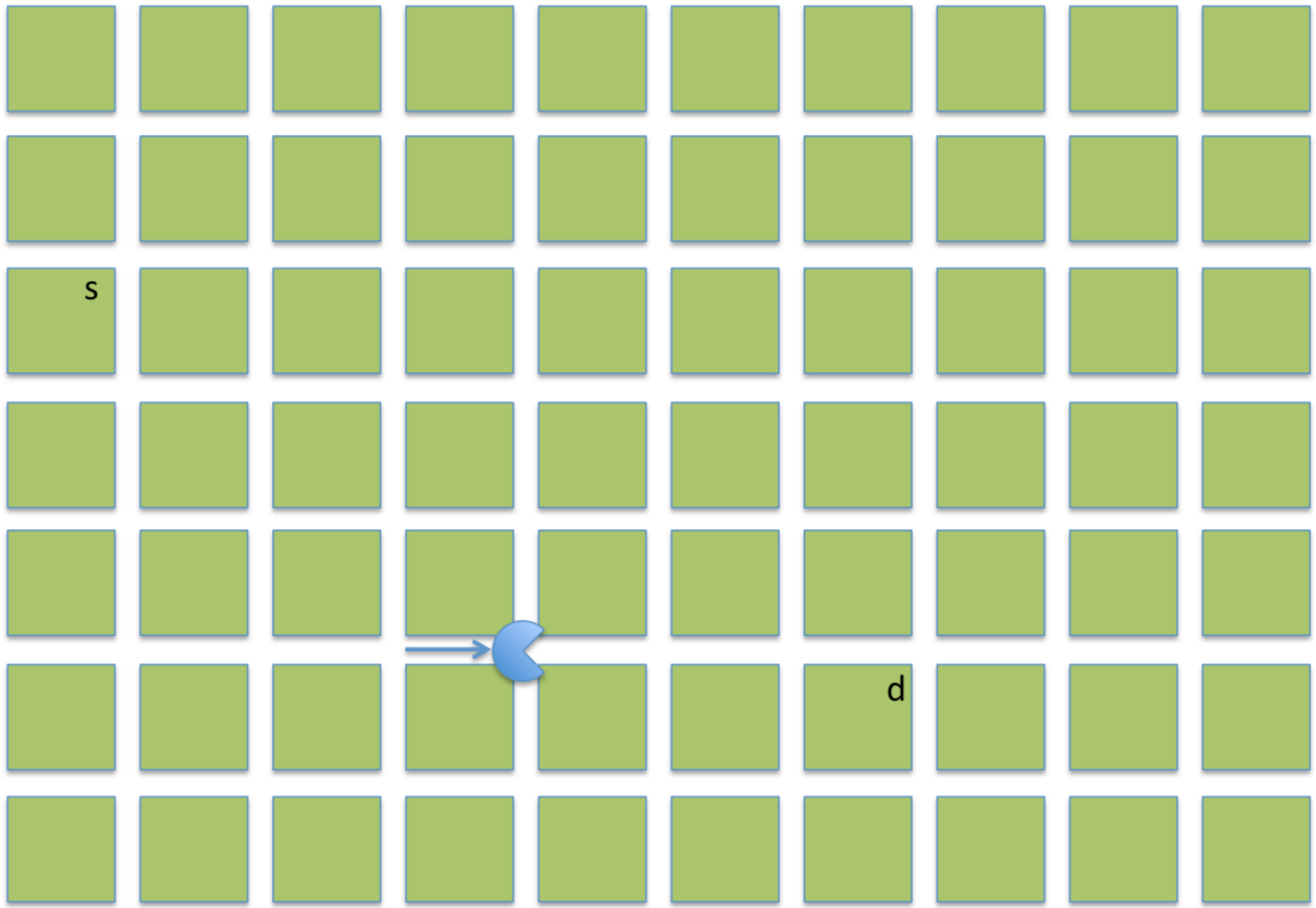
W
R
W
R
W
W
W
W
L
W
W
W
W
W



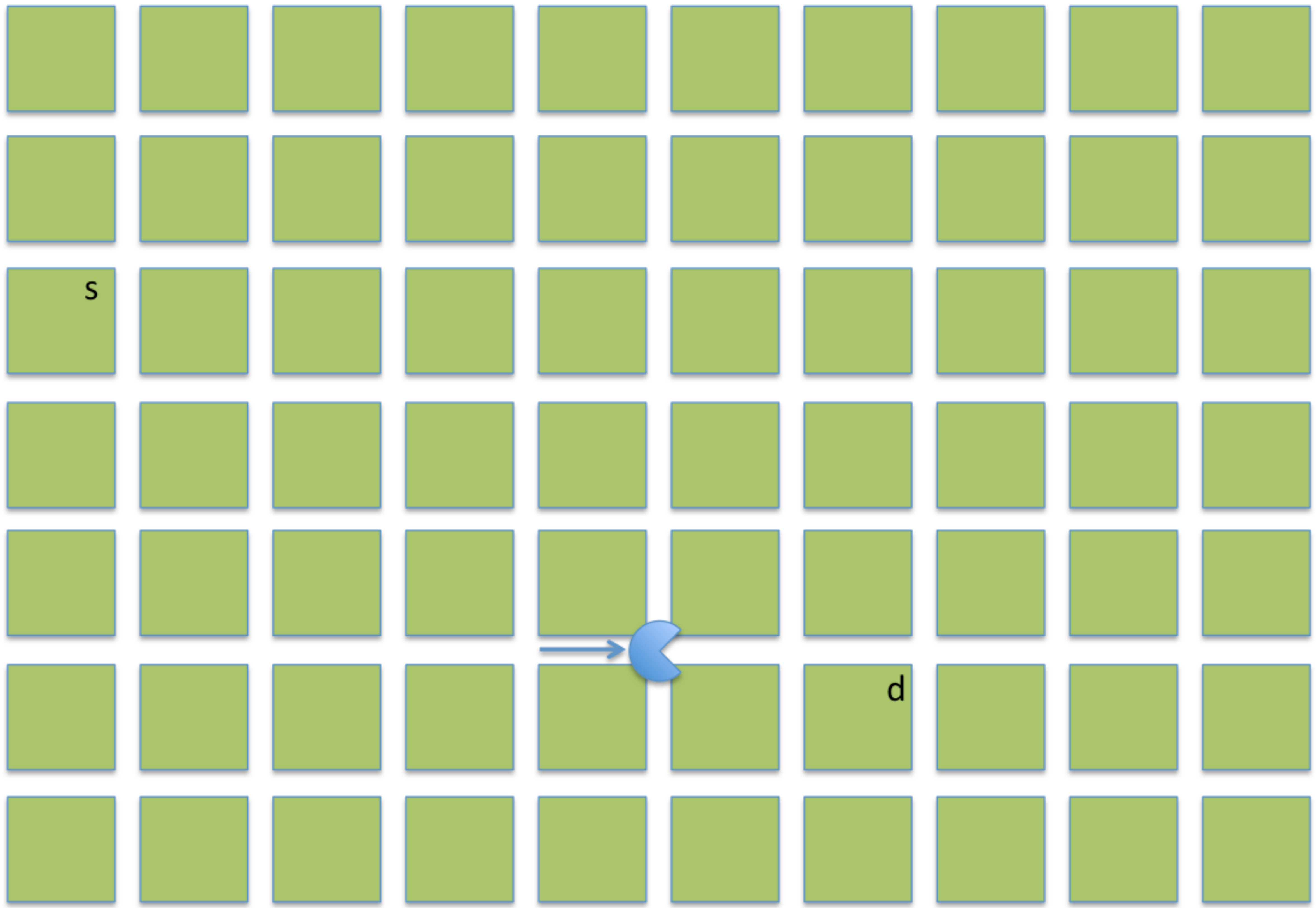
W
R
W
R
W
W
W
W
L
W
W
W
W
W



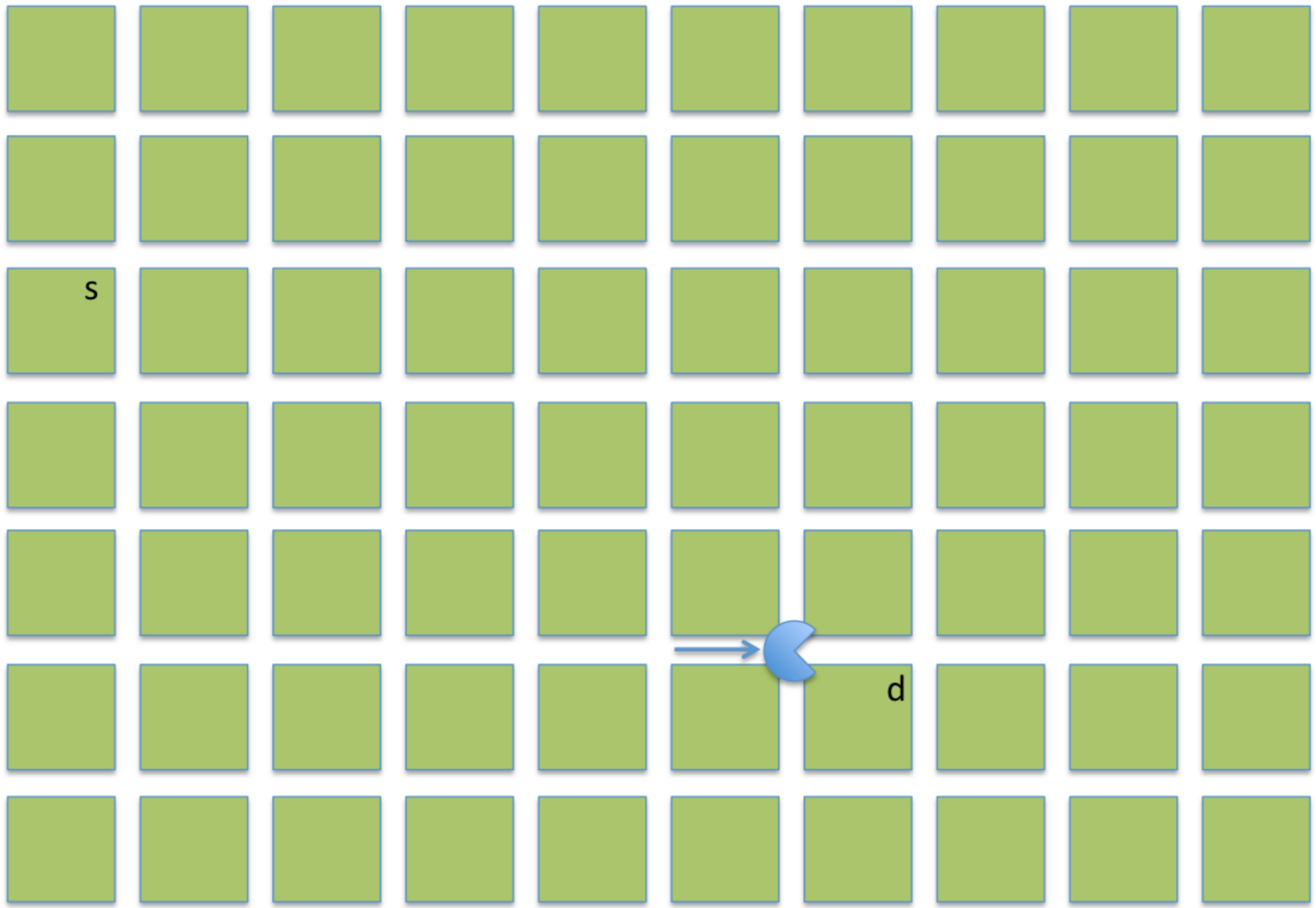
W
R
W
R
W
W
W
W
L
W
W
W
W
W



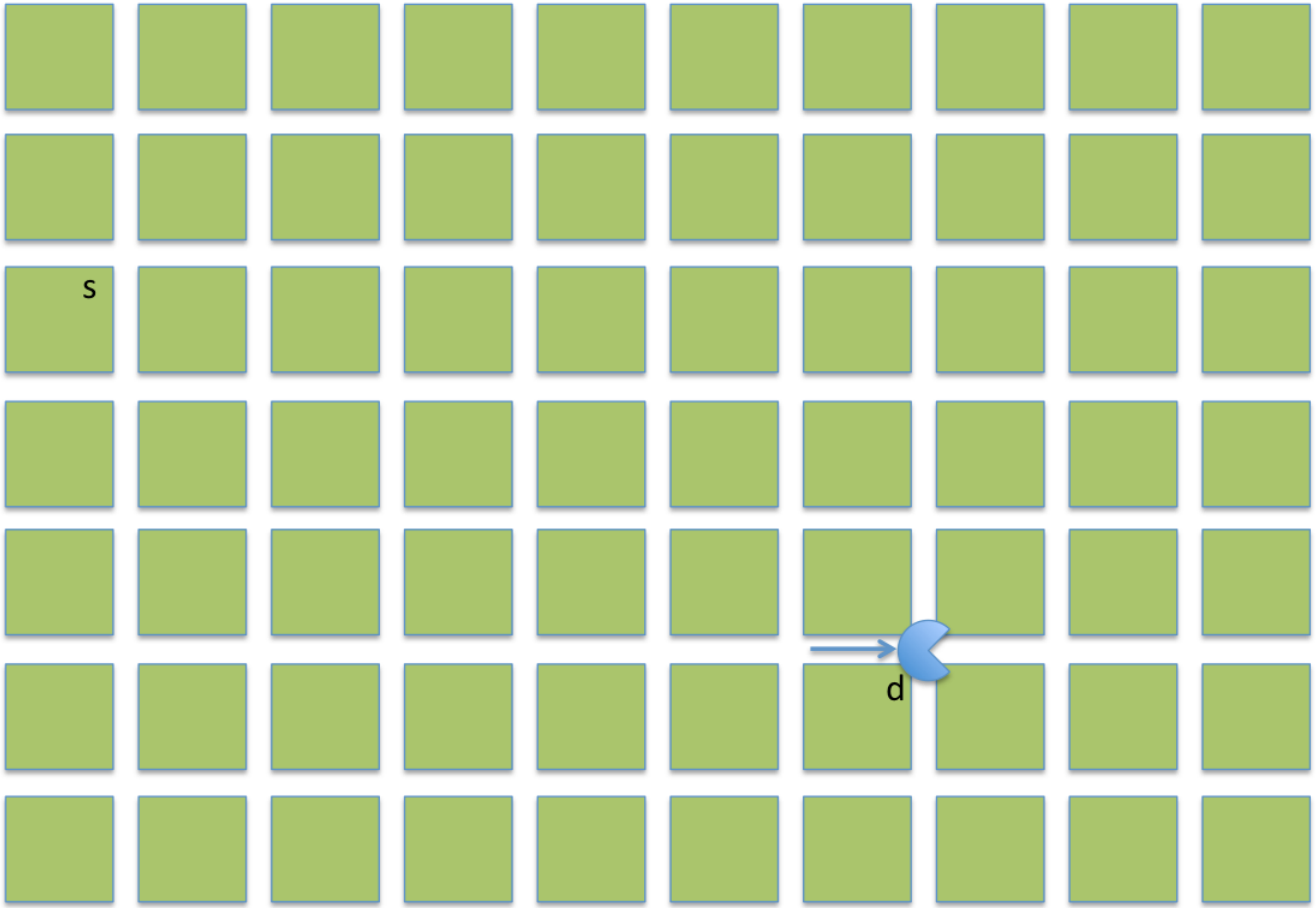
W
R
W
R
W
W
W
W
L
W
W
W
W
W



W
R
W
R
W
W
W
W
L
W
W
W
W
W



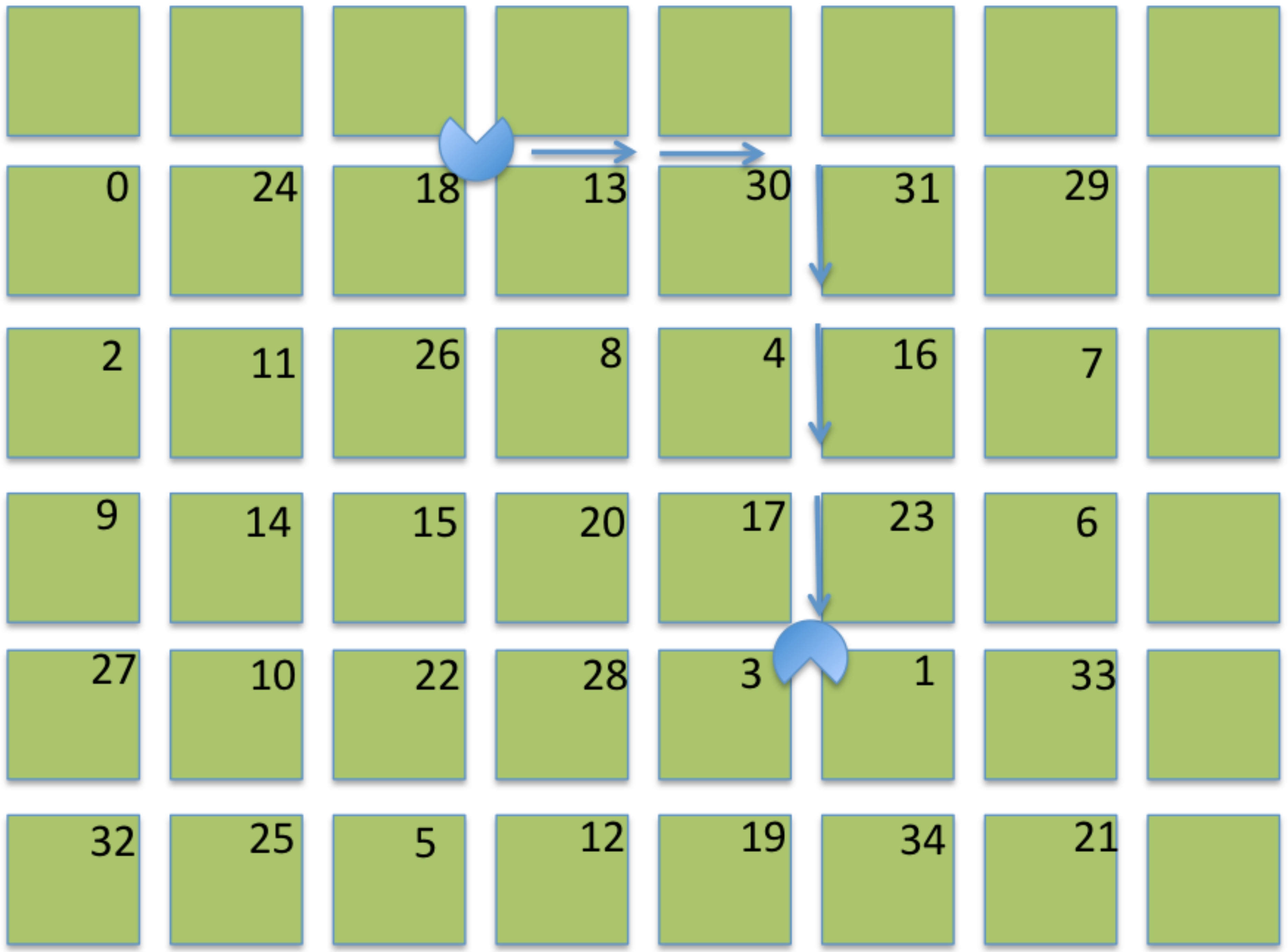
W
R
W
R
W
W
W
W
L
W
W
W
W
W



W
R
W
R
W
W
W
W
L
W
W
W
W
W

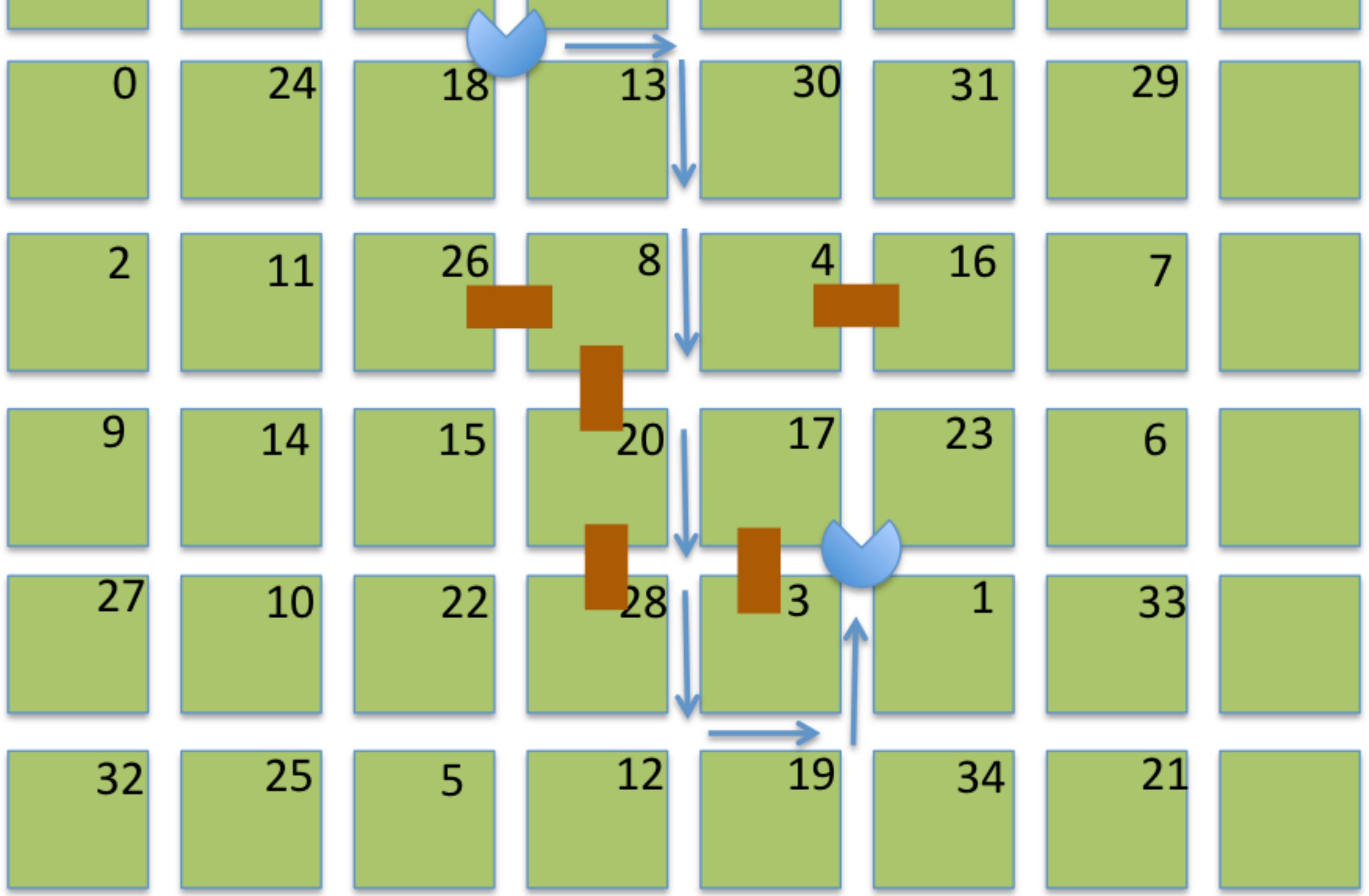
Pacman Route Finding

- There are a number of routes for Pacman to the destination – and knowing the grid co-ordinates makes it easy to find such a route.
- However, suppose there is no direct correspondence between the junction id and Pacman's location.
- Pacman only knows the junctions that he can move to from the current junction.
- Moreover, some of these junctions are blocked.



Source = 18
Destination = 3

Junctions are now labelled randomly from 0 to 34.
Pacman has a map that tells him what the neighbours of each junction are e.g
For example, he knows the (left, right,upper,lower) junctions at junction 8 are (26,4,13,20)
As another example, he knows the (left, right, upper,lower) junction at junction 0 are (-1, 24,-1,2), with -1 telling him in cannot walk in the corresponding direction.



Source = 18
Destination = 3

To make life a bit harder for Pacman, let's assume that some of the junctions are blocked

For example, in the above case, Pacman's map at junction 20 will tell him the available routes are (-1,17,8,28), since the left junction is blocked.

Pacman would like to find the shortest route to his destination – if one exists.

Pacman Route Finding

- Pacman has to search for the destination – chose paths to follows, keeping a track of where he has visited until he reaches his destination.