Perception and Action
Organisms in the real world have to do two basic things in order to survive:
• They have to gather information about their environment (perception) and
• based on this information, they have to manipulate their environment (including themselves) in a way that is advantageous to them (action).

The action in turn may cause a change in the organism’s perception, which can lead to a different type of action.
We call this the perception-action cycle.

Perception and Action
Complex organisms do not just perceive and act, but they also have an internal state that changes based on the success of previous perception-action cycles. This is the mechanism of learning.
-Examples?
-With or without internal state? What is the difference?

A Trip (From Real World) to Grid-Space World
• Grid-space world is an extremely simple model of our own world.
• We will first consider a very simple robot that lives in grid-space world and has no internal state.

Some Rules in Grid-Space World
• A robot in grid-space world can sense whether neighboring cells are empty or not.
• The cells can be empty or contain objects or agents.
• The agents are confined to the floor and can move from cell to cell.

Perception and Action
The robot can move to a free adjacent cell in its column or row. Consequently, there are four possible actions that it can take:
• north: moves the robot one cell up
• east: moves the robot one cell to the right
• south: moves the robot one cell down
• west: moves the robot one cell to the left

Immediate Perception-Action
Now that we specified the robot’s capabilities, its environment, and its task, we need to give “life” to the robot.
In other words, we have to specify a function that maps sensory inputs to movement actions so that the robot will carry out its task.
Since we do not want the robot to remember or learn anything, one such function would be sufficient.
However, it is useful to decompose it in the following way (next slide):
Immediate Perception-Action

The functional decomposition has two advantages:

- **Multiple action functions** can be added that receive the same feature vector as their input,
- It is possible to **add an internal state** to the system to implement memory and learning.

The Robot's Perception

For our robot, we define four different features \( x_1, \ldots, x_4 \) that are important to it. Each feature has value 1 if and only if at least one of the shaded cells is not free:

The Robot's Action

To execute action, we define an ordered set of rules:

- If \( x_1 = 0 \) and \( x_2 = 0 \) and \( x_3 = 0 \) and \( x_4 = 0 \) move north
- If \( x_1 = 1 \) and \( x_2 = 0 \) move east
- If \( x_2 = 1 \) and \( x_3 = 0 \) move south
- If \( x_3 = 1 \) and \( x_4 = 0 \) move west
- If \( x_4 = 1 \) and \( x_1 = 0 \) move north

Task Completion

The robot is supposed to find a cell next to a boundary or object and then follow that boundary forever.

Perception and Action
Production Systems

- **Production systems** are a standardized way to represent action functions.
- A production system consists of an ordered list of production rules (productions).
- Each rule is written in the form condition $\rightarrow$ action.
- A production system is therefore written like:
  
  $c_1 \rightarrow a_1$
  $c_2 \rightarrow a_2$
  ...
  $c_m \rightarrow a_m$
  
- The action of the first rule whose condition evaluates to 1 is executed.

Using Boolean notation, the production system for our boundary-following robot looks like this:

$x_4 \overline{x_1} \rightarrow \text{north}$
$x_3 \overline{x_4} \rightarrow \text{west}$
$x_2 \overline{x_3} \rightarrow \text{south}$
$x_1 \overline{x_2} \rightarrow \text{east}$
$1 \rightarrow \text{north}$