

Introducing Uncertainty

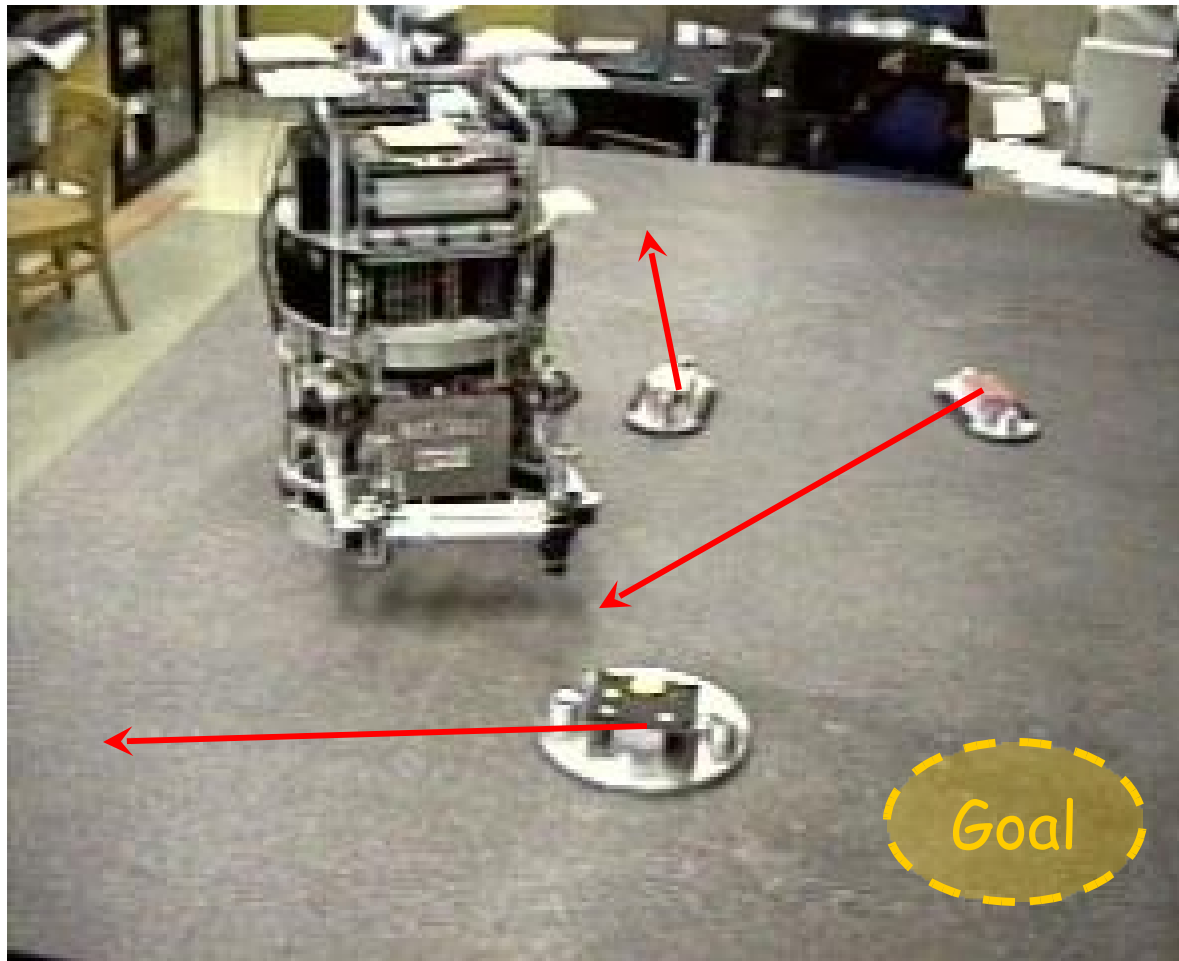
(It is not the world that is imperfect, it is our knowledge of it)



R&N: Chap. 3, Sect 3.6 + Chap. 13

- So far, we have assumed that:
 - World states are perfectly observable,
→ the current state is exactly known
 - Action representations are perfect,
→ states are exactly predicted
- We will now investigate how an agent can cope with **imperfect information**
- We will also study how **limited resources** (mainly time) affect reasoning
- Occasionally, we will consider cases where the world is **dynamic**

Introductory Example



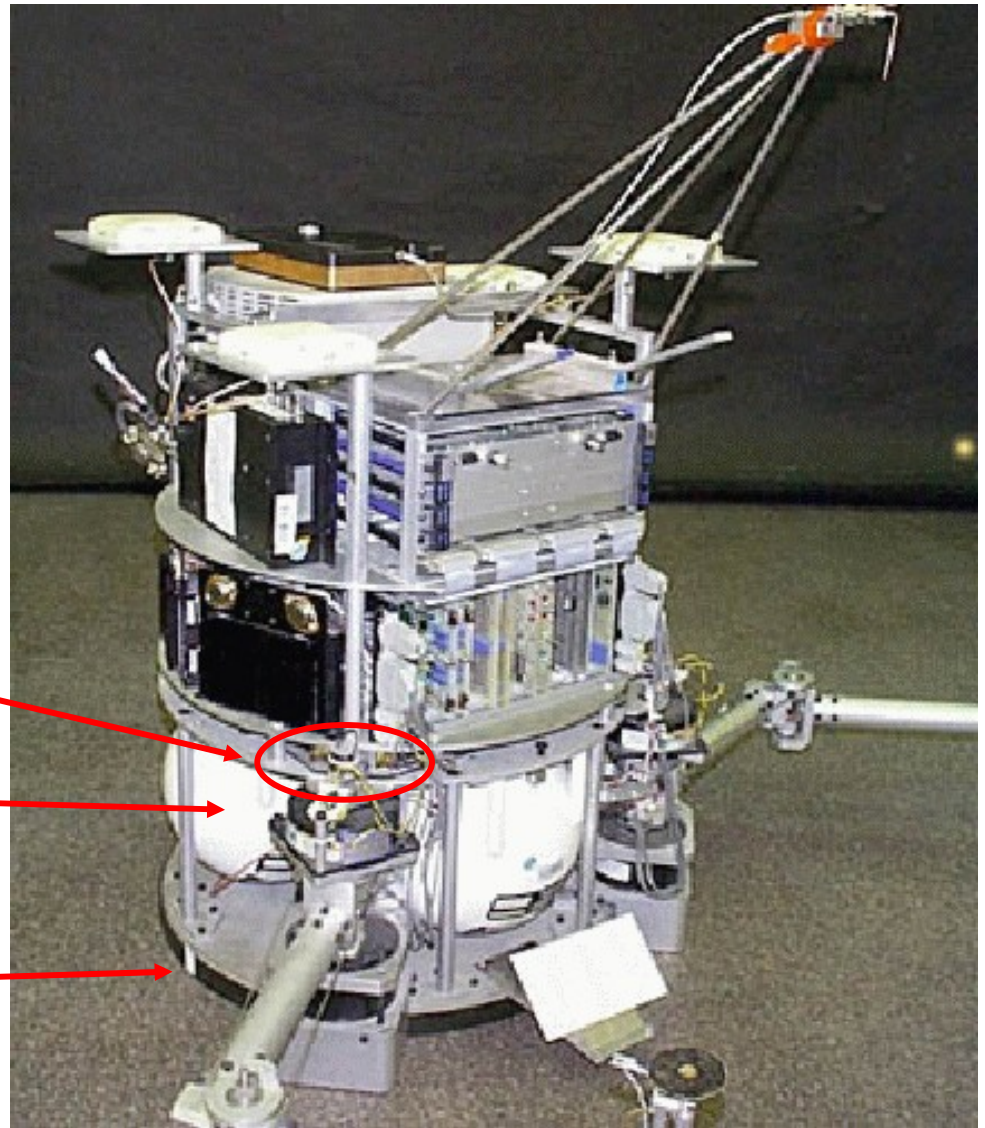
A robot with imperfect sensing must reach a goal location among moving obstacles (dynamic world)

Robot created at
Stanford's ARL Lab
to study issues
in robot control and
planning in no-gravity
space environment

air thrusters

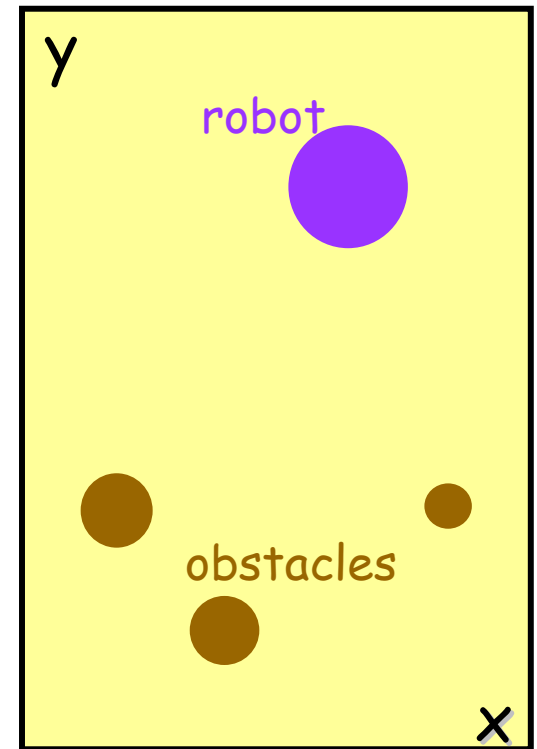
gas tank

air bearing



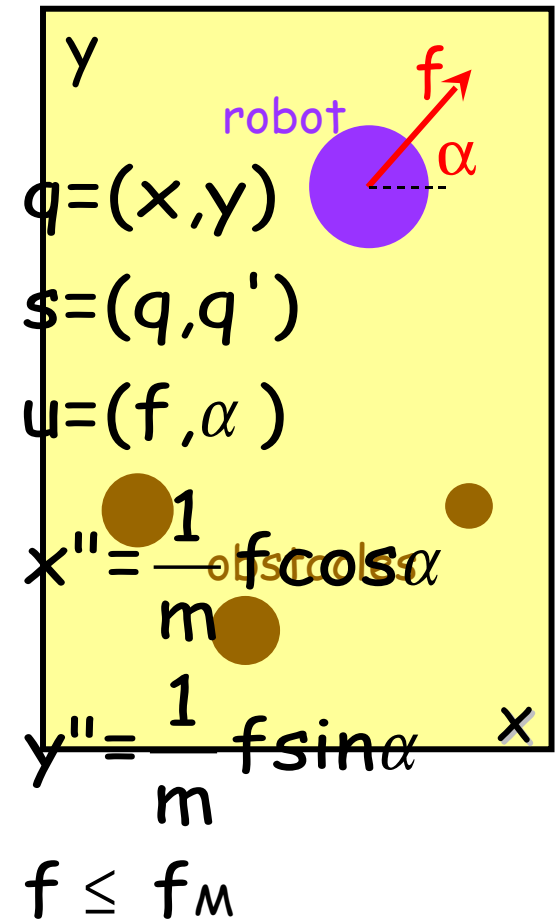
Model, Sensing, and Control

- The robot and the obstacles are represented as disks moving in the plane
- The position and velocity of each disc are measured by an overhead camera every $1/30$ sec



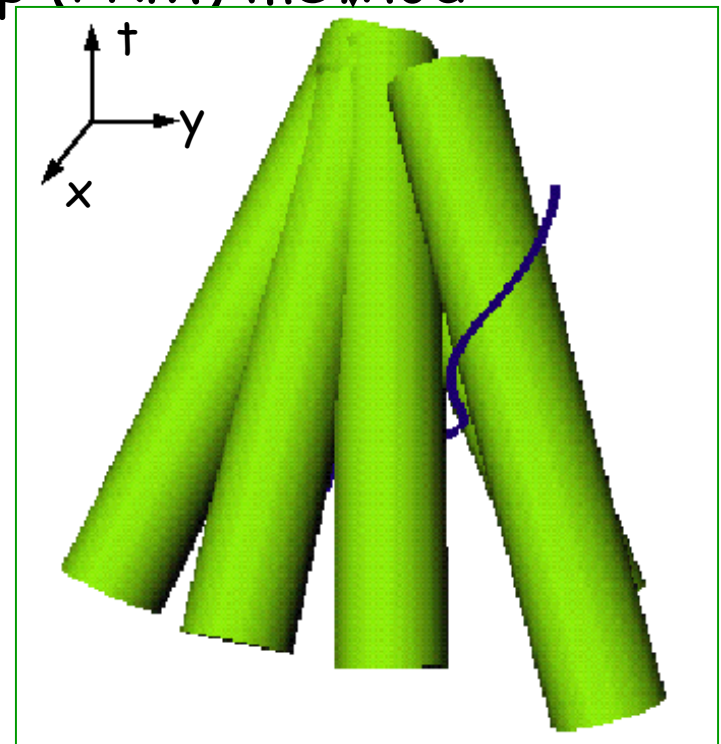
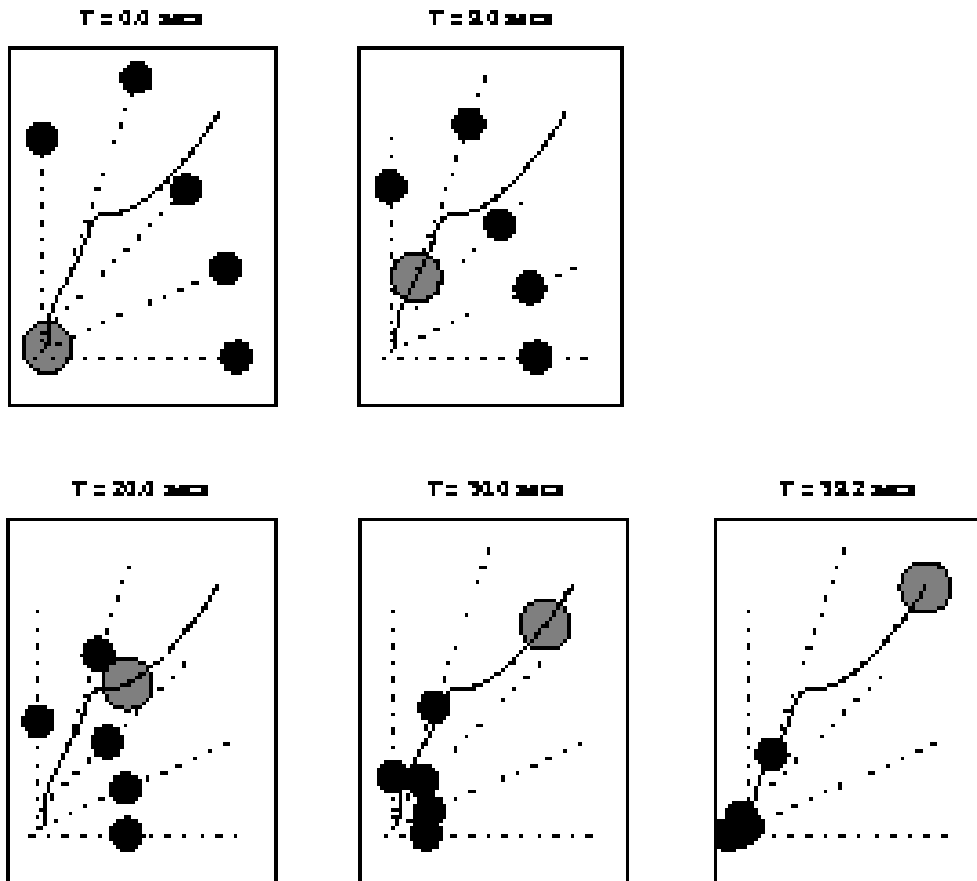
Model, Sensing, and Control

- The robot and the obstacles are represented as disks moving in the plane
- The position and velocity of each disc are measured by an overhead camera within 1/30 sec
- The robot controls the magnitude f and the orientation α of the total pushing force exerted by the robot



Motion Planning

The robot plans its trajectories in **configuration×time space** using a probabilistic roadmap (PRM) method



Obstacle map to cylinders in configuration×time space 7

But executing this trajectory is likely to fail ...

- The measured velocities of the obstacles are inaccurate
 - Tiny particles of dust on the table affect trajectories and contribute further to deviation
 - Obstacles are likely to deviate from their expected trajectories
 - Planning takes time, and during this time, obstacles keep moving
 - The computed robot trajectory is not properly synchronized with those of the obstacles
- The robot may hit an obstacle before reaching its goal

But executing this trajectory is likely to fail ...

- The measured velocities of the obstacles are inaccurate
- Tiny particles of dust on the table affect trajectories and contribute further to deviation
 - Obstacles are likely to deviate from their expected trajectories
- Planning takes time, and during this time, obstacles are moving

The computed path to the destination is not guaranteed

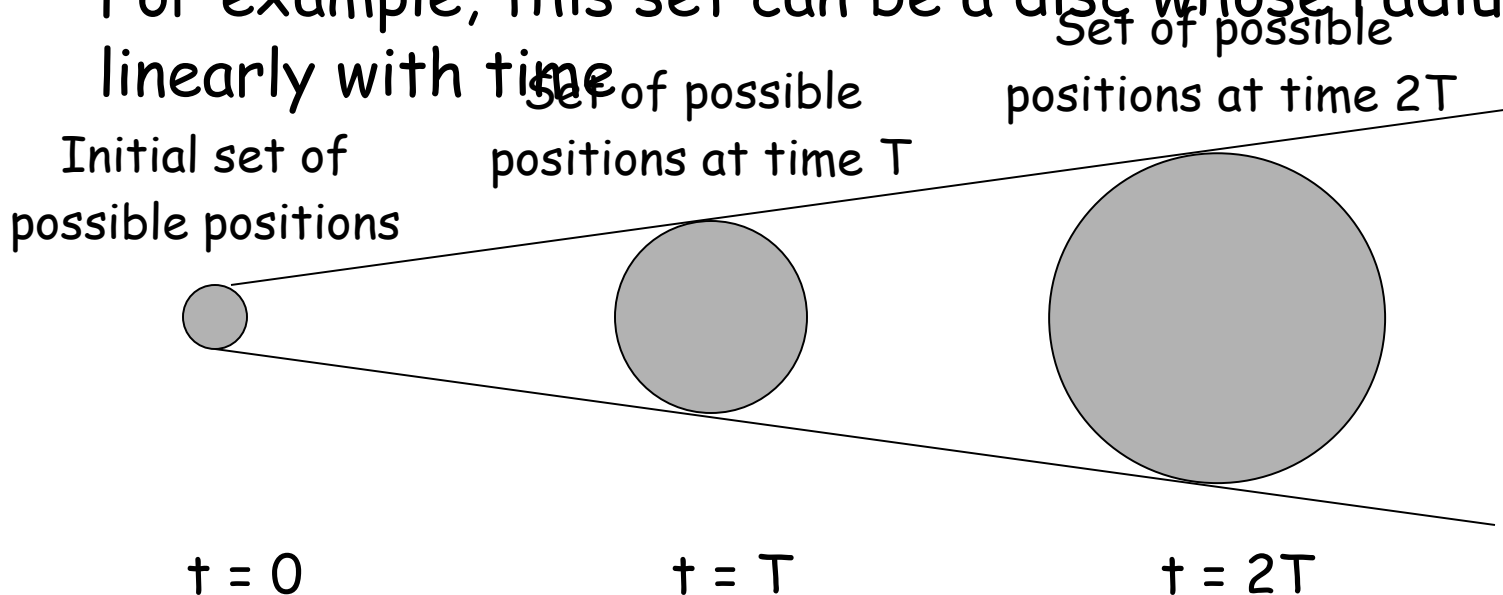


Planning must take both uncertainty in world state and time constraints into account

→ The robot may hit an obstacle before reaching its goal

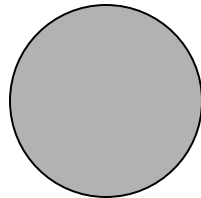
Dealing with Uncertainty

- The robot can handle uncertainty in an obstacle position by representing the set of all positions of the obstacle that the robot think possible at each time (belief state)
- For example, this set can be a disc whose radius grows linearly with time



Dealing with Uncertainty

- The robot can handle uncertainty in an obstacle position by representing the set of all positions of the obstacle that the robot think possible at each time (belief state)
- For example, this set can be a disc whose radius grows linearly with time



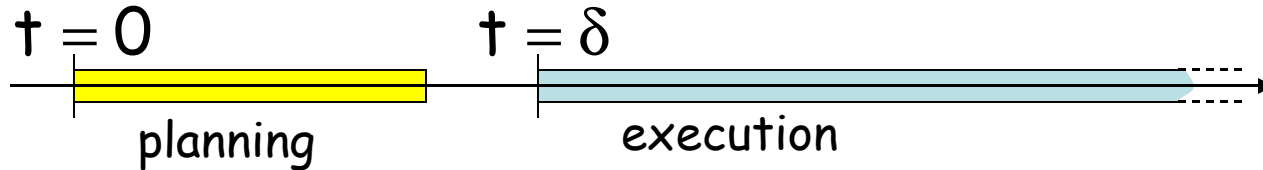
$t = T$

The robot must plan to be outside this disc at time $t = T$

Dealing with Uncertainty

- The robot can handle uncertainty in an obstacle position by representing the set of all positions of the obstacle that the robot think possible at each time (belief state)
- For example, this set can be a disc whose radius grows linearly with time
- The forbidden regions in configuration \times time space are **cones**, instead of cylinders
- The trajectory planning method remains essentially unchanged

Dealing with Planning Time



- Let $t=0$ the time when planning starts. A time limit δ is given to the planner
- The planner computes the states that will be possible at $t = \delta$ and use them as the possible initial states
- It returns a trajectory at some $t < \delta$, whose execution will start at $t = \delta$
- Since the PRM planner isn't absolutely guaranteed to find a solution within δ , it computes two trajectories using the same roadmap: one to the goal, the other to any position where the robot will be safe for at least an additional δ . Since there are usually many such positions the second trajectory is at least one order

Are we done?

- Not quite !
- The uncertainty model may itself be incorrect, e.g.:
 - There may be more dust on the table than anticipated
 - Some obstacles have the ability to change trajectories
- But if we are too careful, we will end up with forbidden regions so big that no solution trajectory will exist any more
- So, it might be better to take some "risk"




The robot must **monitor** the execution of the planned trajectory and be prepared to **re-plan** a new trajectory

Are we done?

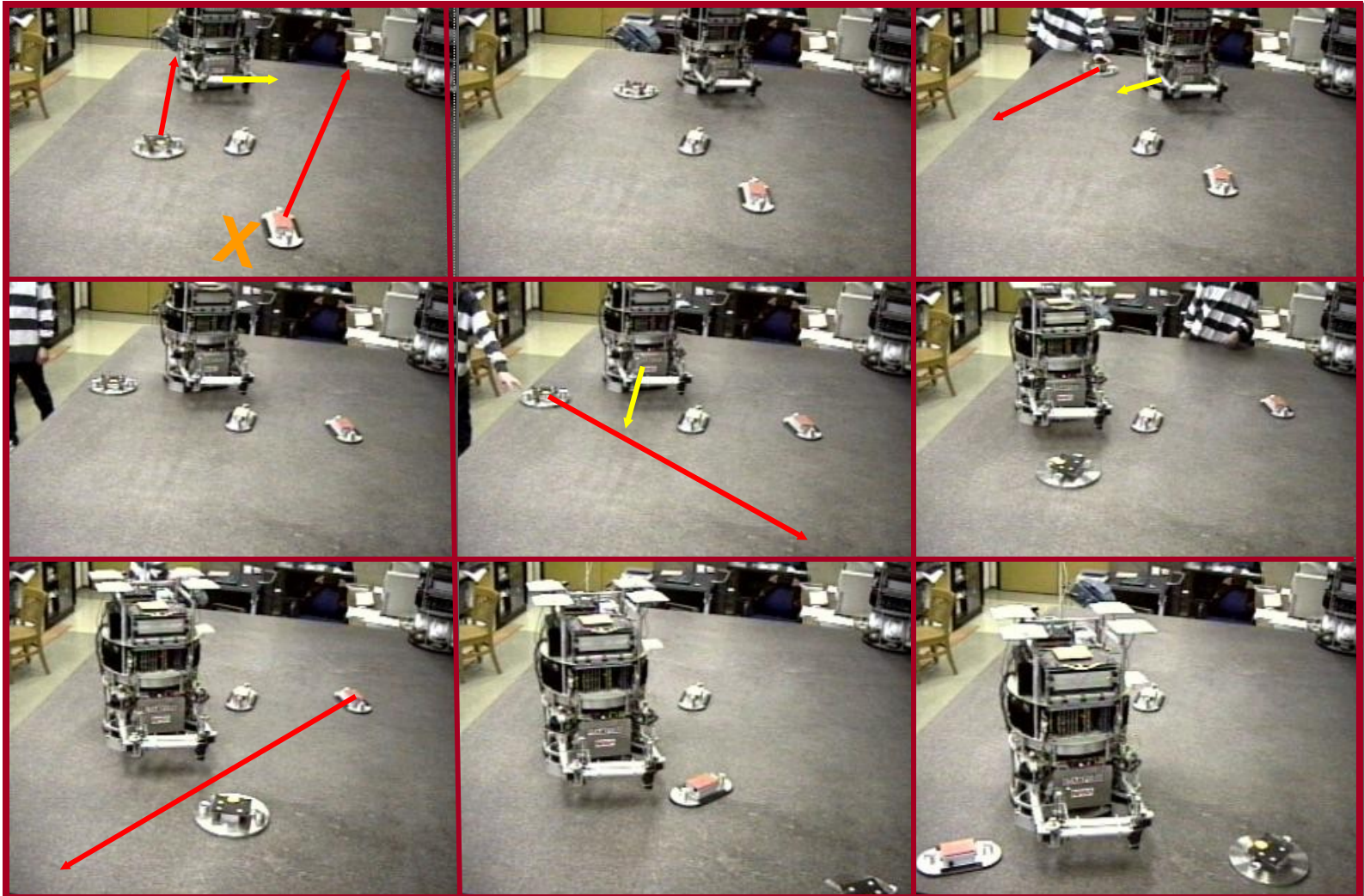
Execution monitoring consists of using the camera (at 30Hz) to verify that all obstacles are at positions allowed by the robot's uncertainty model

If an obstacle has an unexpected position, the planner is called back to compute a new trajectory.



The robot must **monitor** the execution of the planned trajectory and be prepared to **re-plan** a new trajectory

Experimental Run



Total duration : 40 sec

Is this guaranteed to work?

Of course not :

- Thrusters might get clogged
- The robot may run out of air or battery
- The granite table may suddenly break into pieces
- Etc ...

[Unbounded uncertainty]

Sources of Uncertainty

The Real World and its Representation

3x3 matrix filled
with 1, 2, ..., 8, and
'empty'

Agent's conceptualization
(→ representation language)

Real world

8-puzzle

The Real World and its Representation

Agent's conceptualization
(→ representation language)

Logic sentences using
propositions like
Block(A), On(A,B),
Handempty, ...
and connectives

Real world

Blocks world

The Real World and its Representation

Geometric models
and equations
of motion

Agent's conceptualization
(→ representation language)

Real world

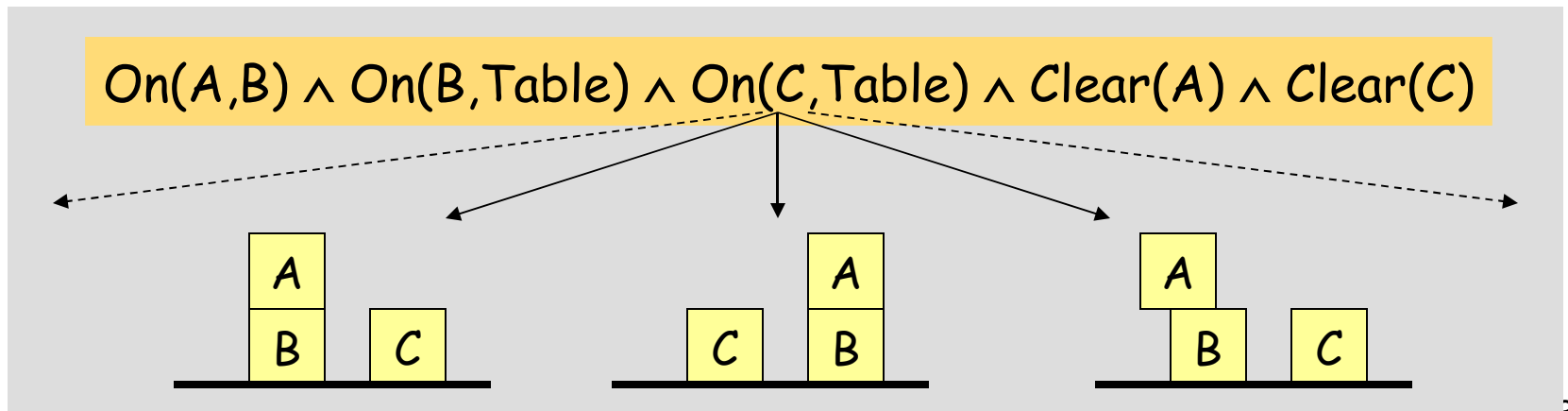
Air-bearing robot
navigating among
moving obstacles

Who provides the representation language?

- The agent's designer
- As of today, no practical techniques exist allowing an agent to autonomously abstract features of the real world into useful concepts and develop its own representation language using these concepts
- Inductive learning techniques are steps in this direction, but much more is needed
- The issues discussed in the following slides arise whether the representation language is provided by the agent's designer or developed over time by the agent ²²

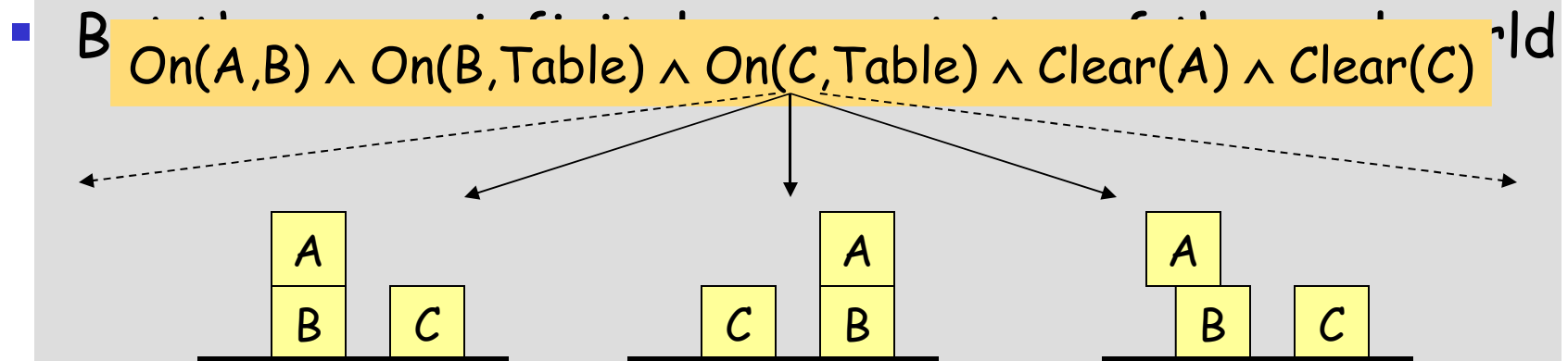
First Source of Uncertainty: The Representation Language

- There are many more states of the real world than can be expressed in the representation language
- So, any state represented in the language may correspond to many different states of the real world, which the agent can't represent distinguishably



First Source of Uncertainty: The Representation Language

- 6 propositions $On(x,y)$, where $x, y = A, B, C$ and $x \neq y$
- 3 propositions $On(x,Table)$, where $x = A, B, C$
- 3 propositions $Clear(x)$, where $x = A, B, C$
- At most 2^{12} states can be distinguished in the language [in fact much fewer, because of state constraints such as $On(x,y) \rightarrow \neg On(y,x)$]



→ An action representation may be incorrect ...

Stack(C, A)

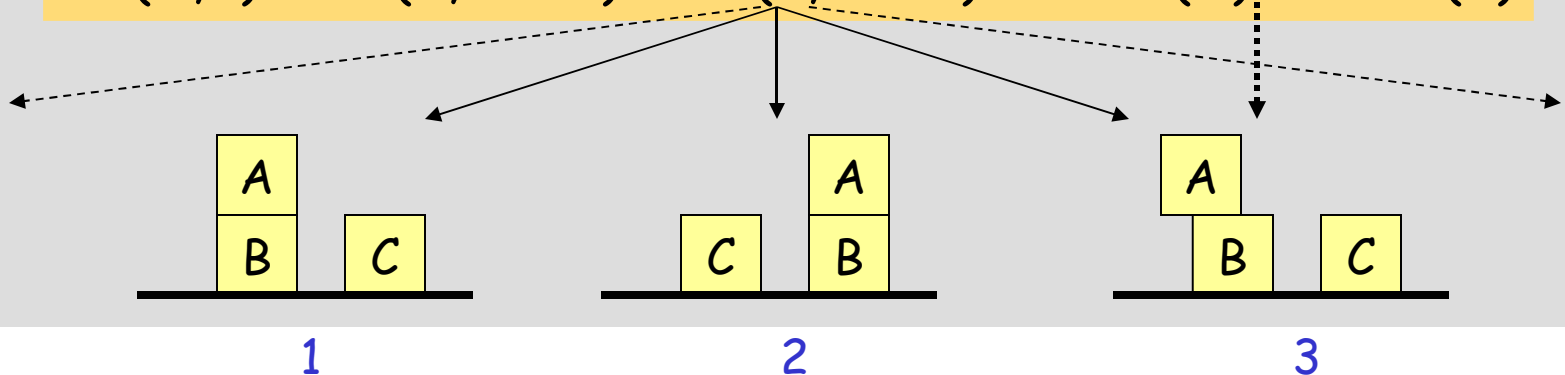
$P = \text{Holding}(C) \wedge \text{Block}(C) \wedge \text{Block}(A) \wedge \text{Clear}(A)$

$D = \text{Clear}(A), \text{Holding}(C)$

$A = \text{On}(C, A), \text{Clear}(C), \text{Handempty}$

is likely not to have the described effects in case 3 because the precondition is "incomplete"

$\text{On}(A, B) \wedge \text{On}(B, \text{Table}) \wedge \text{On}(C, \text{Table}) \wedge \text{Clear}(A) \wedge \text{Clear}(C)$



... or may describe several alternative effects

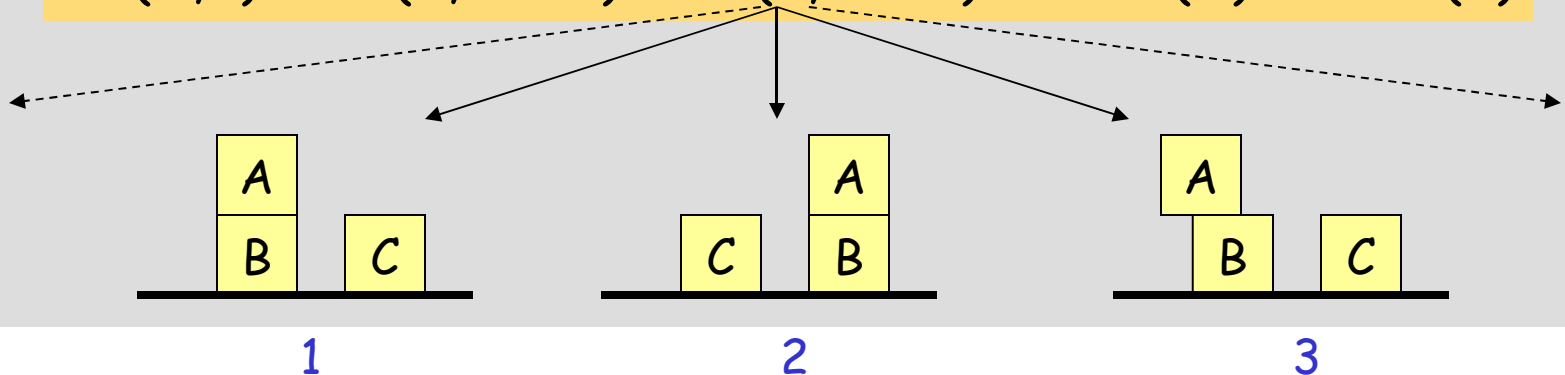
Stack(C, A)

$P = \text{Holding}(C) \wedge \text{Block}(C) \wedge \text{Block}(A) \wedge \text{Clear}(A)$
[If $\text{On}(A, x) \wedge (x \neq \text{Table})$]

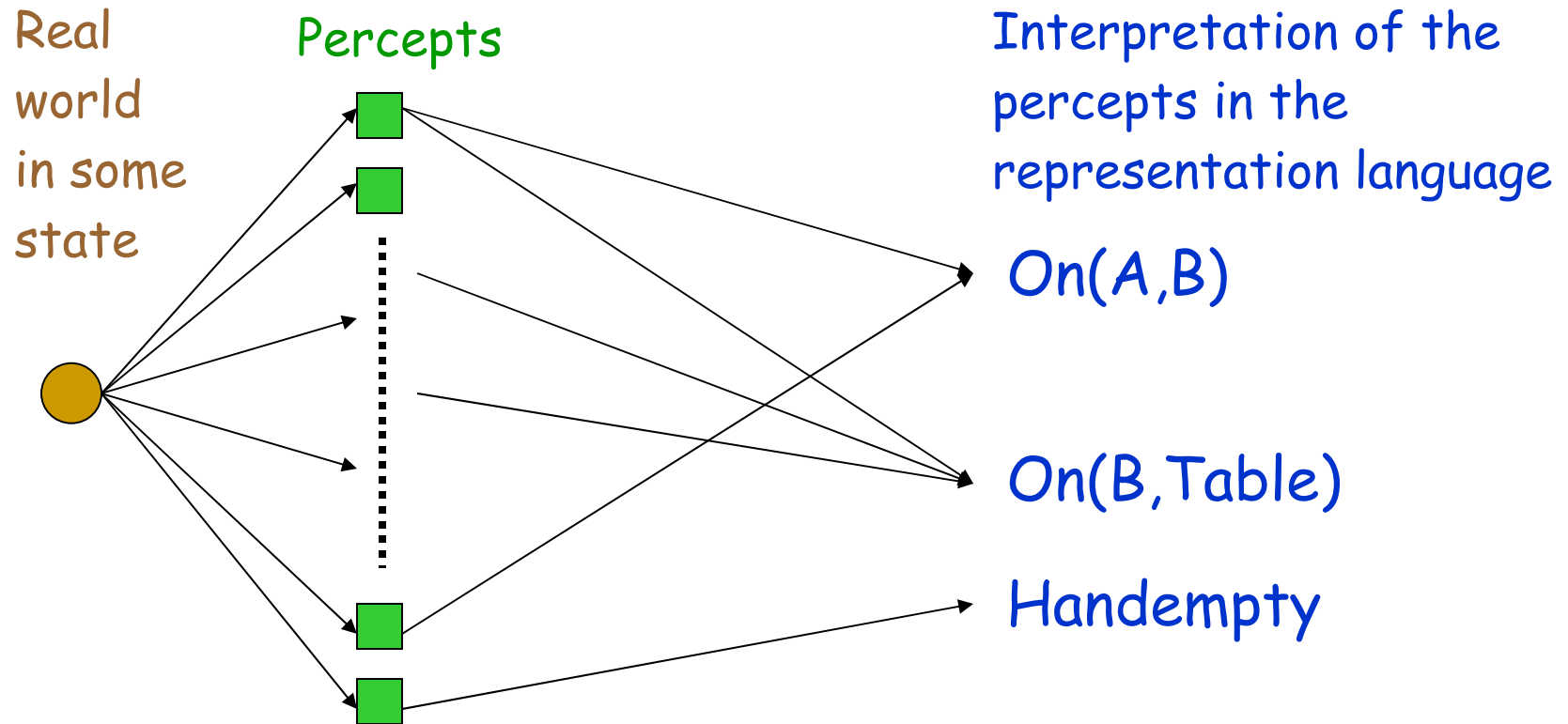
OR
 $E_1 \left\{ \begin{array}{l} D = \text{Clear}(A), \text{Holding}(C) \\ A = \text{On}(C, A), \text{Clear}(C), \text{Handempty} \end{array} \right.$

$E_2 \left\{ \begin{array}{l} D = \text{Holding}(C), \text{On}(A, x) \\ A = \text{On}(C, \text{Table}), \text{Clear}(C), \text{Handempty}, \text{On}(A, \text{Table}), \end{array} \right.$

$\text{On}(A, B) \wedge \text{On}(B, \text{Table}) \wedge \text{On}(C, \text{Table}) \wedge \text{Clear}(A) \wedge \text{Clear}(C)$



Observation of the Real World

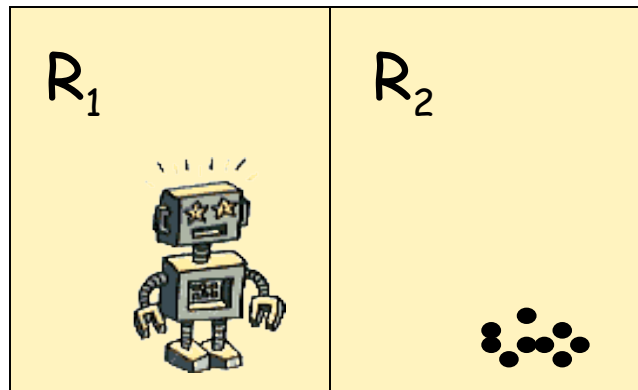


Percepts can be user's inputs, sensory data (e.g., image pixels), information received from other agents, ...

Second source of Uncertainty: Imperfect Observation of the World

Observation of the world can be:

- **Partial**, e.g., a vision sensor can't see through obstacles (lack of percepts)

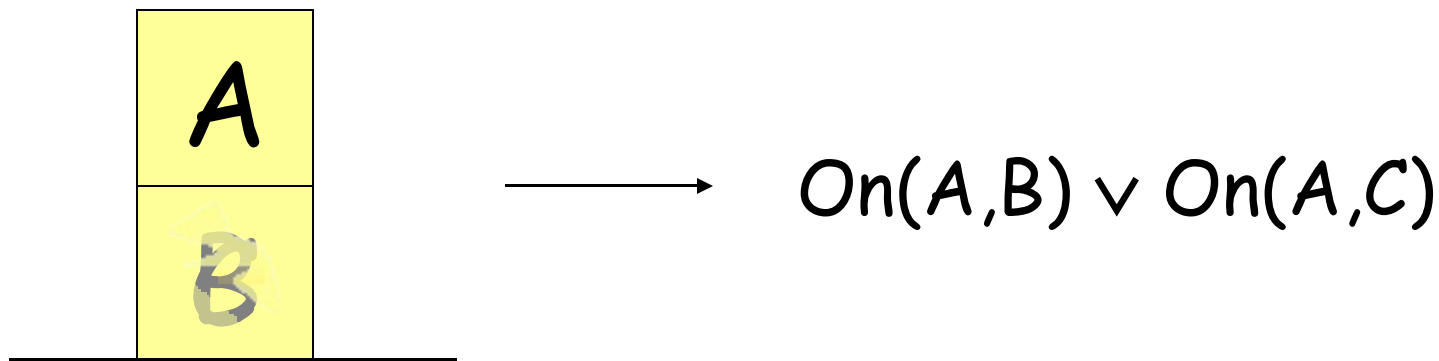


The robot may not know whether
there is dust in room R₂

Second source of Uncertainty: Imperfect Observation of the World

Observation of the world can be:

- Partial, e.g., a vision sensor can't see through obstacles
- **Ambiguous**, e.g., percepts have multiple possible interpretations



Second source of Uncertainty: Imperfect Observation of the World

Observation of the world can be:

- Partial, e.g., a vision sensor can't see through obstacles
- Ambiguous, e.g., percepts have multiple possible interpretations
- **Incorrect**

Third Source of Uncertainty: Ignorance, Laziness, Efficiency

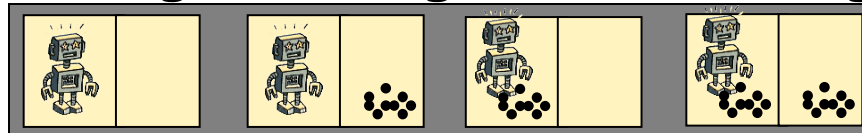
- An action may have a long list of preconditions, e.g.:
Drive-Car:
 $P = \text{Have}(\text{Keys}) \wedge \neg \text{Empty}(\text{Gas-Tank}) \wedge \text{Battery-Ok} \wedge$
 $\text{Ignition-Ok} \wedge \neg \text{Flat-Tires} \wedge \neg \text{Stolen}(\text{Car}) \dots$
- The agent's designer may ignore some preconditions ... or by laziness or for efficiency, may not want to include all of them in the action representation
- The result is a representation that is either incorrect - executing the action may not have the described effects - or that describes several alternative effects

Representation of Uncertainty

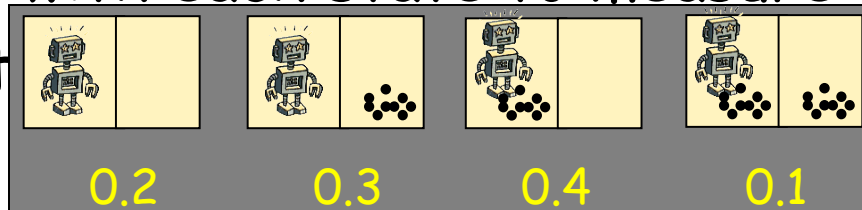
- Many models of uncertainty
- We will consider two important models:
 - **Non-deterministic model:**
Uncertainty is represented by a set of possible values, e.g., a set of possible worlds, a set of possible effects, ...
→ Today's lecture
 - **Probabilistic model:**
Uncertainty is represented by a probabilistic distribution over a set of possible values
→ The next lecture

Example: Belief State

- In the presence of non-deterministic sensory uncertainty, an agent **belief state** represents all the states of the world that it thinks are possible at a given time or at a given stage of reasoning

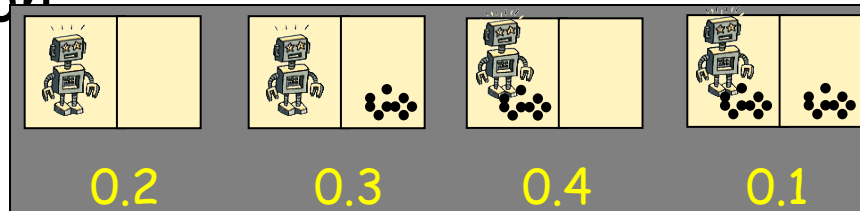


- In the probabilistic model of uncertainty, a probability is associated with each state to measure its likelihood to be the actual state



What do probabilities mean?

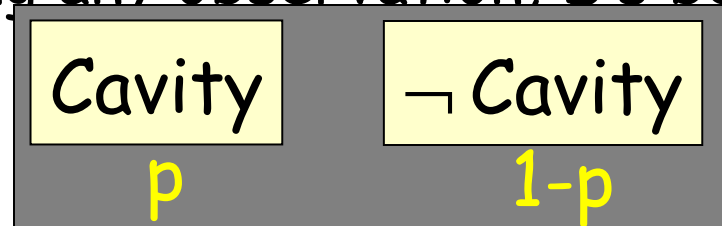
- Probabilities have a natural **frequency interpretation**
- The agent believes that if it was able to return many times to a situation where it has the same belief state, then the actual states in this situation would occur at a relative frequency defined by the probabilistic distribution



↑ This state would occur 20% of the times

Example

- Consider a world where a dentist agent D meets a new patient P
- D is interested in only one thing: whether P has a cavity, which D models using the proposition Cavity
- Before making any observation, D's belief state is:



- This means that D believes that a fraction p of patients have cavities

Where do probabilities come from?

- Frequencies observed in the past, e.g., by the agent, its designer, or others
- Symmetries, e.g.:
 - If I roll a dice, each of the 6 outcomes has probability $1/6$
- Subjectivism, e.g.:
 - If I drive on Highway 280 at 120mph, I will get a speeding ticket with probability 0.6
 - Principle of indifference: If there is no knowledge to consider one possibility more probable than another, give them the same probability

Non-Deterministic vs. Probabilistic

- If the world is adversarial and the agent uses probabilistic methods, it is likely to fail consistently
- If the world is non-adversarial and failure must be absolutely avoided, then non-deterministic techniques are likely to be more efficient computationally
- In other cases, probabilistic methods may be a better option, especially if there are several "goal" states providing different rewards and life does not end when one is reached

Uncertainty and Errors

- The uncertainty model may itself be incorrect
- Representing uncertainty can reduce the risk of errors, but does not eliminate it entirely !!
- Execution monitoring is required to detect errors and (hopefully) fix them [closed-loop execution]
 - What to monitor?
 - How to fix errors?

What to monitor?

- **Action monitoring:**
 - Check preconditions before executing an action and effects after
 - Not very efficient (e.g., a precondition may have been false for a while)
- **Plan monitoring:**
 - Check the preconditions of the entire remaining plans
 - → **Triangle tables**

Key-in-Box Problem

Grasp-Key-in- R_2

$P = \text{In}(\text{Robot}, R_2) \wedge \text{In}(\text{Key}, R_2)$

$D = \emptyset$

$A = \text{Holding}(\text{Key})$

Lock-Door

$P = \text{Holding}(\text{Key})$

$D = \text{Unlocked}(\text{Door})$

$A = \text{Locked}(\text{Door})$

Move-Key-from- R_2 -into- R_1

$P = \text{In}(\text{Robot}, R_2) \wedge \text{Holding}(\text{Key}) \wedge \text{Unlocked}(\text{Door})$

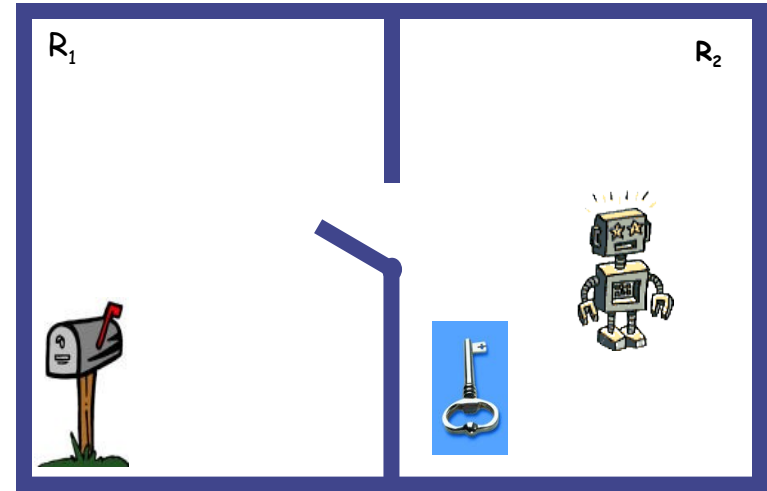
$D = \text{In}(\text{Robot}, R_2), \text{In}(\text{Key}, R_2)$

$A = \text{In}(\text{Robot}, R_1), \text{In}(\text{Key}, R_1)$

Put-Key-Into-Box

$P = \text{In}(\text{Robot}, R_1) \wedge \text{Holding}(\text{Key})$

$D = \text{Holding}(\text{Key}), \text{In}(\text{Key}, R_1)$



Triangle Table

Plan:

Grasp-Key-in- R_2 , Move-Key-from- R_2 -into- R_1 ,
 Lock-Door, Put-Key-Into-Box
 to achieve $\text{Locked}(\text{Door}) \wedge \text{In}(\text{Key}, \text{Box})$

In(Robot, R_2)				
In(Key, R_2)	Grasp-Key-in- R_2			
In(Robot, R_2)				
Unlocked(Door)	Holding(Key)	Move-Key-from- R_2 -into- R_1		
	Holding(Key)		Lock-Door	
	Holding(Key)	In(Robot, R_1)		Put-Key-Into-Box
			Locked(Door)	In(Key, Box)

Triangle Table

Propositions from the initial state that are necessary to the applicability of actions in the plan and the achievement of the goal

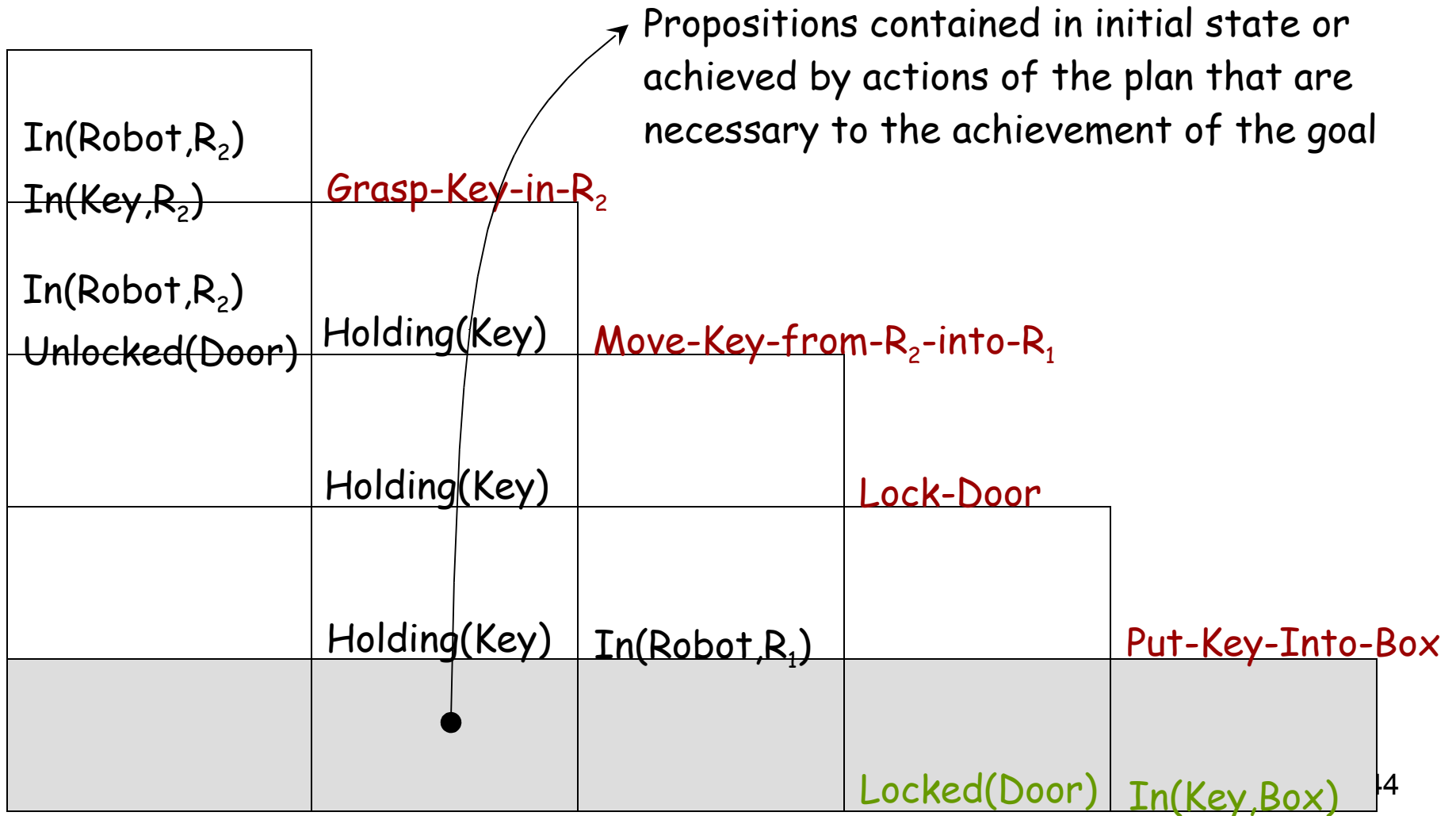
In(Robot, R ₂)				
In(Key, R ₂)		<i>Grasp-Key-in-R₂</i>		
In(Robot, R ₂)				
Unlocked(Door)	Holding(Key)	<i>Move-Key-from-R₂-into-R₁</i>		
	Holding(Key)		<i>Lock-Door</i>	
	Holding(Key)	In(Robot, R ₁)		<i>Put-Key-Into-Box</i>
			<i>Locked(Door)</i>	<i>In(Key, Box)</i>

Triangle Table

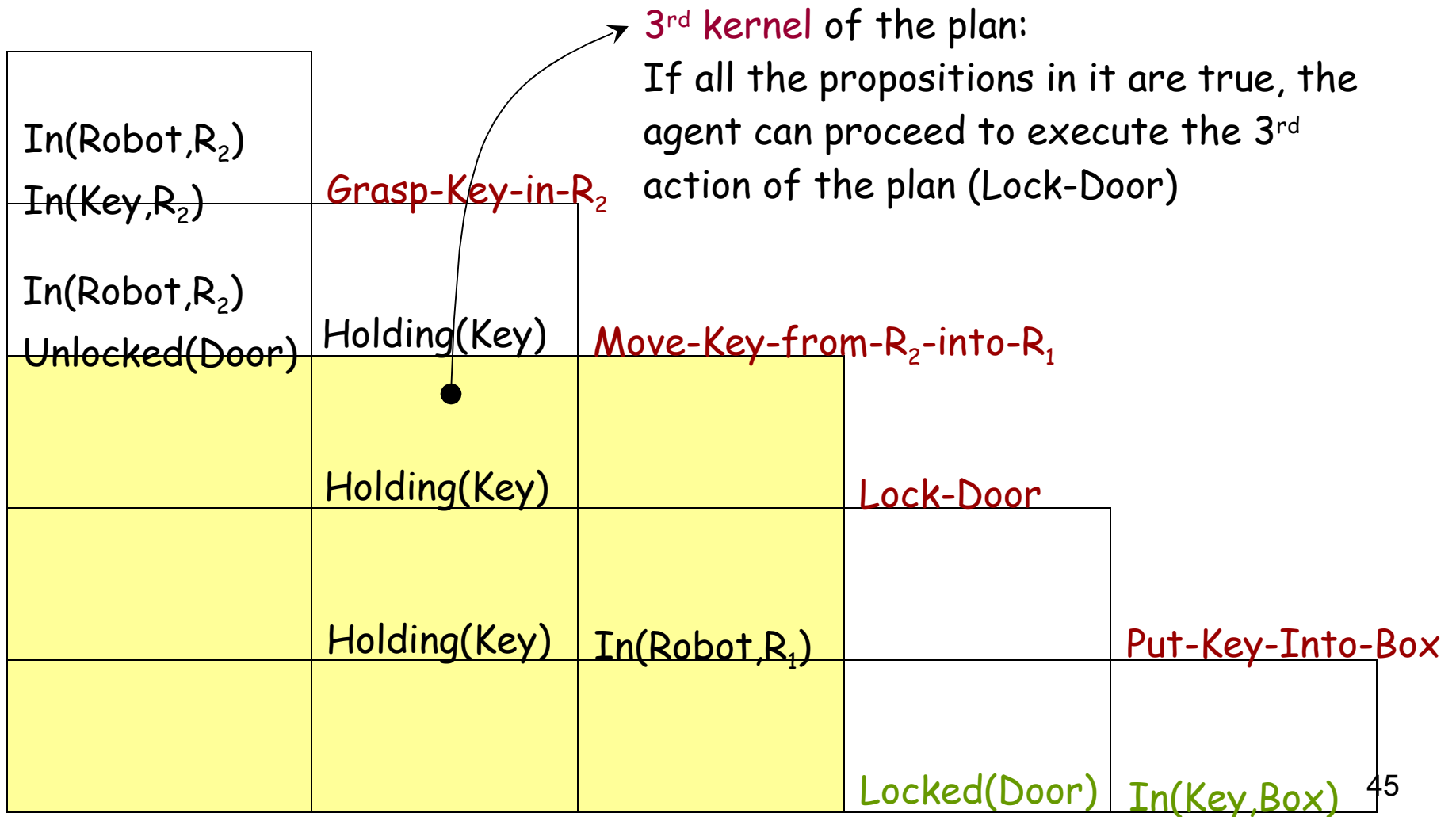
In(Robot, R_2)				
In(Key, R_2)	Grasp-Key-in- R_2			
In(Robot, R_2)	Holding(Key)			
Unlocked(Door)	Holding(Key)		Move-Key-from- R_2 -into- R_1	
	Holding(Key)			Lock-Door
	Holding(Key)	In(Robot, R_1)		Put-Key-Into-Box
			Locked(Door)	In(Key, Box)

Propositions achieved by Grasp-Key-in- R_2 that are necessary to the applicability of subsequent actions and the achievement of the goal

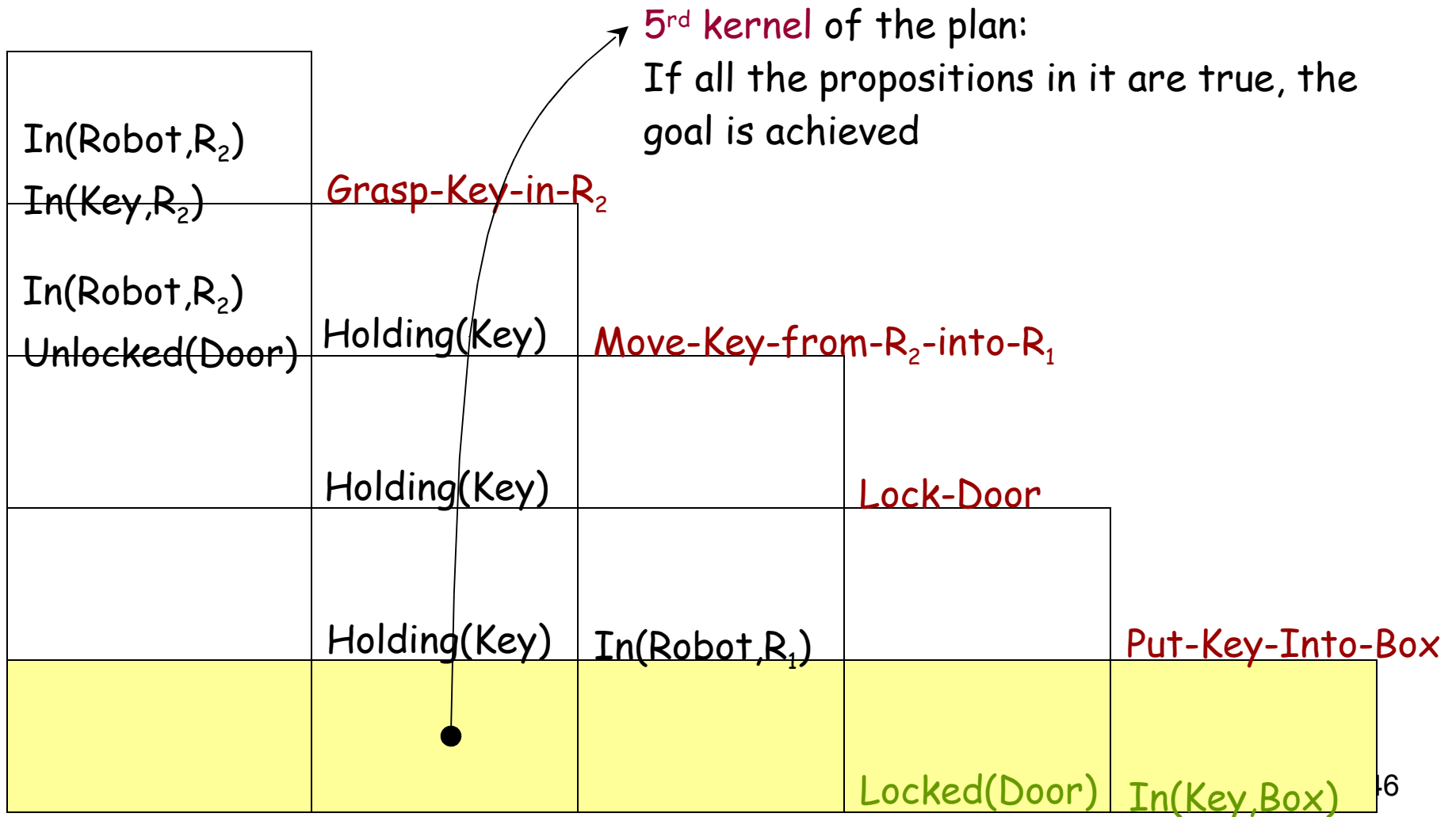
Triangle Table



Triangle Table



Triangle Table



Execution Monitoring with Triangle Tables

Repeat:

- Observe the world and identify the largest k such that all the propositions in the k^{th} kernel are true
 - If $k = 0$ then re-plan
 - Else execute the k^{th} action of the plan
- Actions that fail are repeated
- Actions that are not needed are skipped

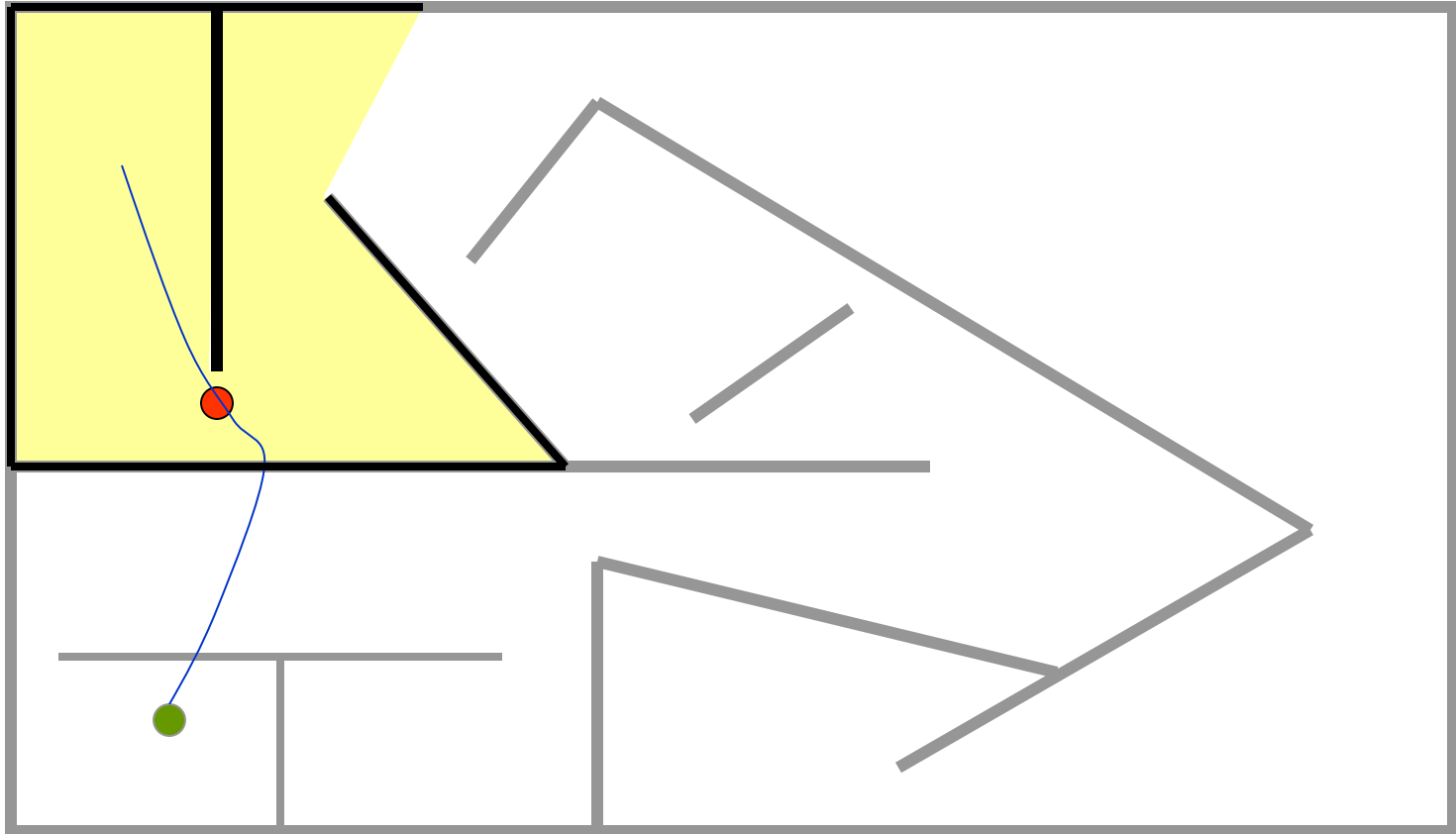
But ...

- Repeating an action that failed assumes that it may succeed next time. But what if the agent picked the wrong key in R_2 ?
- Either the agent has more knowledge or sensors than it used so far, and it's time to use them
- Or it doesn't have any of these, and it has no choice - fail or call another agent
[I do the same when my car does not start and I can't figure out why]

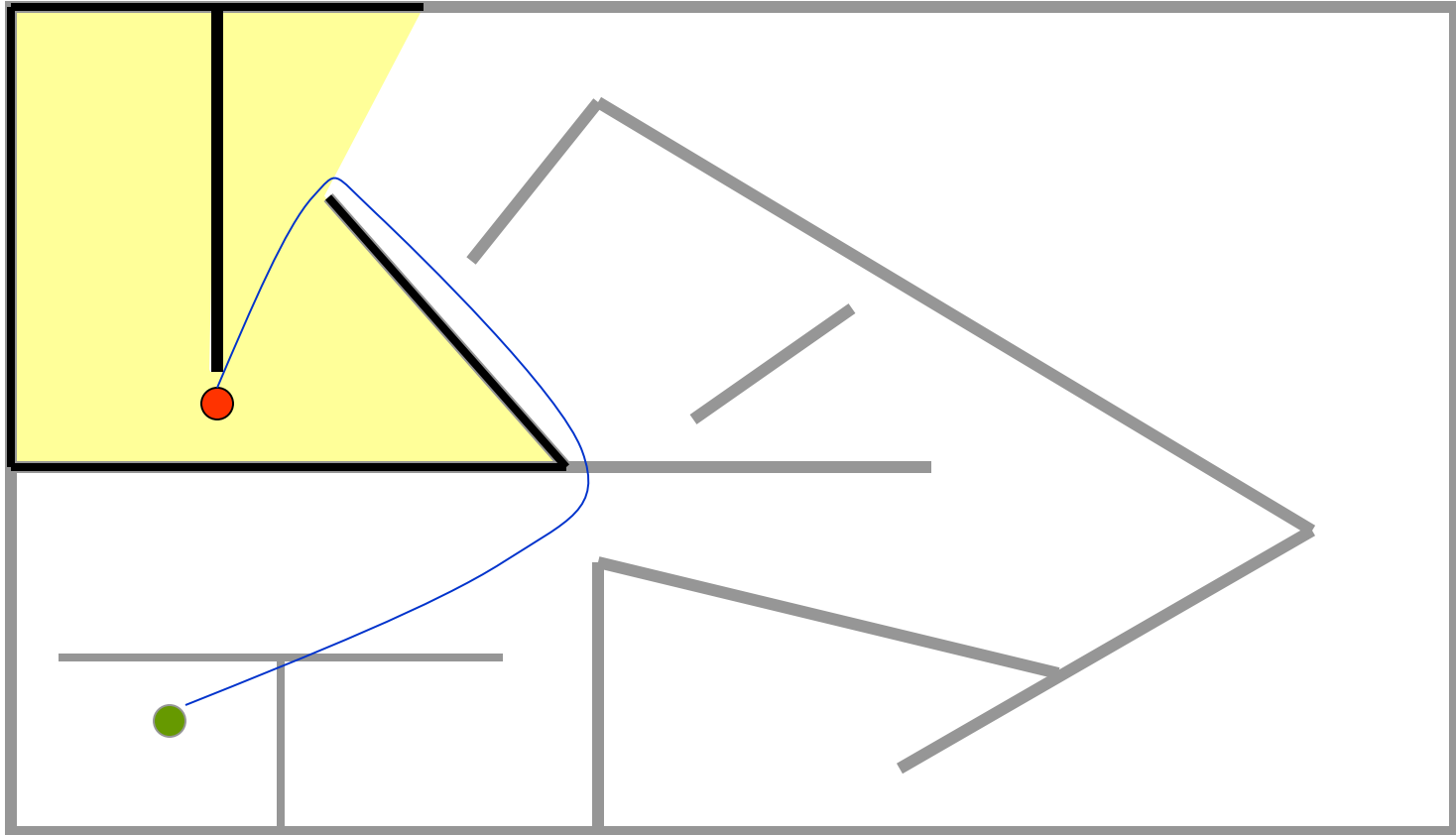
On-Line Search

- Sometimes uncertainty is so large that actions need to be executed for the agent to know their effects
- Example: A robot must reach a goal position. It has no prior map of the obstacles, but its vision system can detect all the obstacles visible from a the robot's current position

Assuming no obstacles in the unknown region and taking the shortest path to the goal is similar to searching with an admissible (optimistic) heuristics



Assuming no obstacles in the unknown region and taking the shortest path to the goal is similar to searching with an admissible (optimistic) heuristics



Just as with classical search, **on-line search may detect dead-ends** and move to a more promising position (~ node of search tree)

Conclusion

- Uncertainty is an unavoidable part of most real-world applications of AI
- We must deal with this in order to have success

- Last class: uncertainty from adversary
- Today: non-deterministic uncertainty
- Next time: probabilistic uncertainty