Introduction to Robotics Lesson 8 – Planning and Behavior-Based Robotics

Lecturer: Dr. Yehuda Elmaliah, Mr. Roi Yehoshua

Elmaliahy@colman.ac.il

roiyeho@gmail.com

Agenda

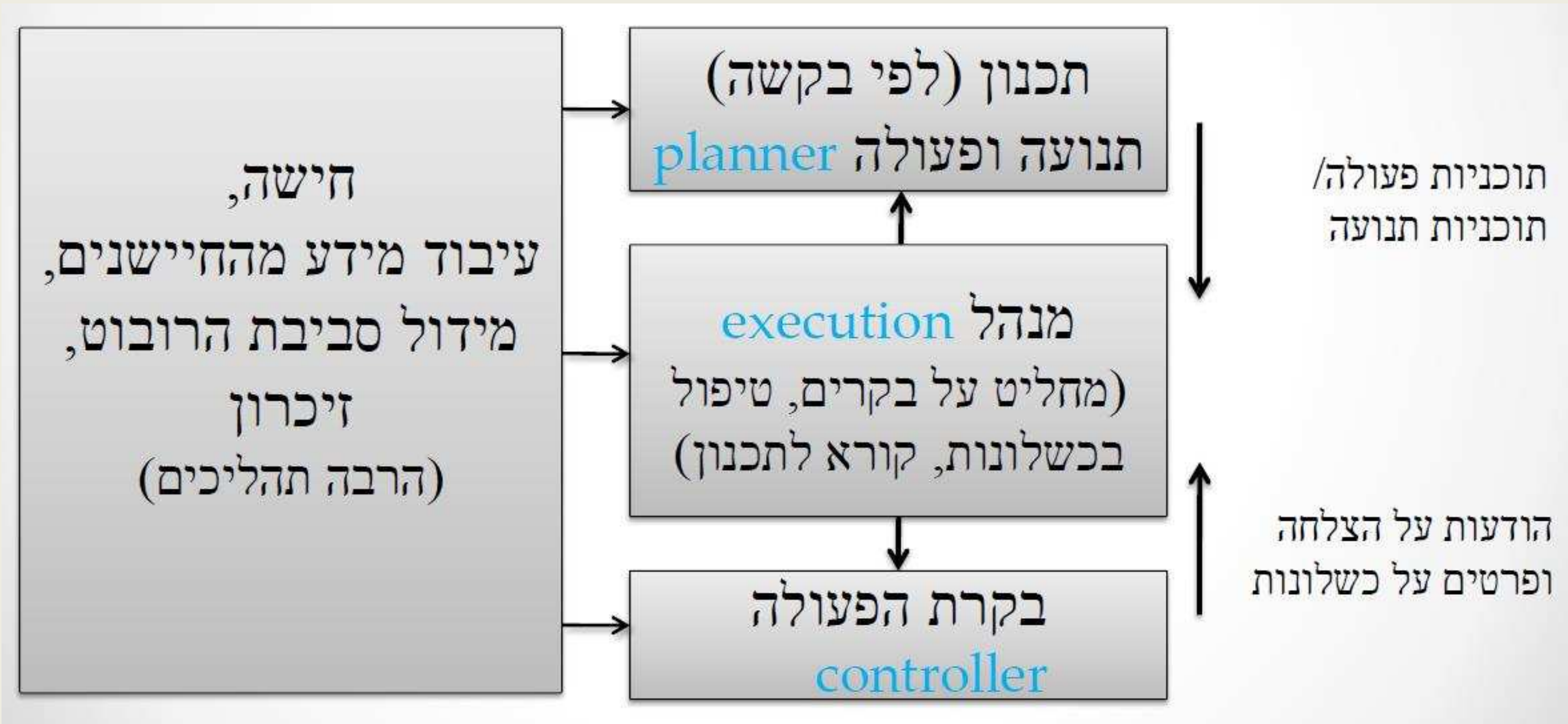
המסלול האקדמי
המכללה למינהל

- The hierarchical paradigm
- AI Planning
- STRIPS
- The reactive paradigm
- Behaviors

Previously, on Robots…

- Sense-Think-Act cycle
- 3-Tier Architecture
- Action-selection (planning) problem:
  - What to do now in service of the task?

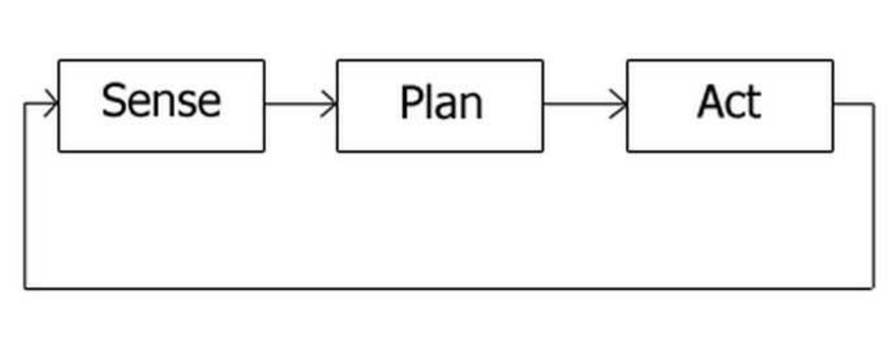
3-Tier Architecture



Robot Paradigms

- A robotic paradigm can be described by the relationship between the three primitives of robotics: Sense, Plan, Act.
- It can also be described by how sensory data is processed and distributed through the system, and where decisions are made.

Hierarchical Paradigm



- Dominated the world of robotics until the late 1980's
- The robot operates in a top-down fashion, heavy on planning
- At each step the robot explicitly plans the next move and then acts upon it
- The sensing data is gathered into a global world model

A Crash Course in AI Planning

המסלול האקדמי
המכללה למינהל

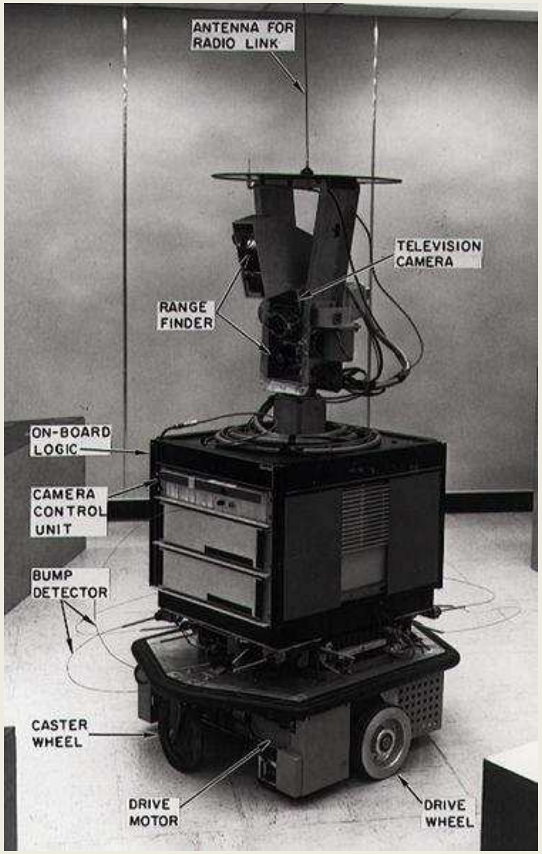
- Planning: An approach to action-selection problem
- Very long history, since the very beginning of AI
- 1971, seminal paper by Fikes and Nillson: **STRIPS** (Stanford Research Institute Problem Solver)
  - Still cited and taught today, despite much progress
- STRIPS originally developed for SRI robot, Shakey

Shakey The Robot

המסלול האקדמי
המכללה למינהל

- The first general-purpose mobile robot to be able to reason about its own actions.
- Shakey could analyze the commands given to it and break them down into basic chunks by itself.
- In 1970, Life magazine referred to Shakey as the "first electronic person"

Shakey The Robot



http://www.youtube.com/watch?v=qXdn6ynwpiI

Planning Problem

- Find a **sequence of actions** that achieves a given **goal** when executed from a given **initial world state**. That is, given
  - a set of operator descriptions (defining the possible primitive actions by the agent),
  - an initial state description, and
  - a goal state description or predicate,

  compute a plan, which is
  - a sequence of operator instances, such that executing them in the initial state will change the world to a state satisfying the goal-state description.
- Goals are usually specified as a conjunction of goals to be achieved

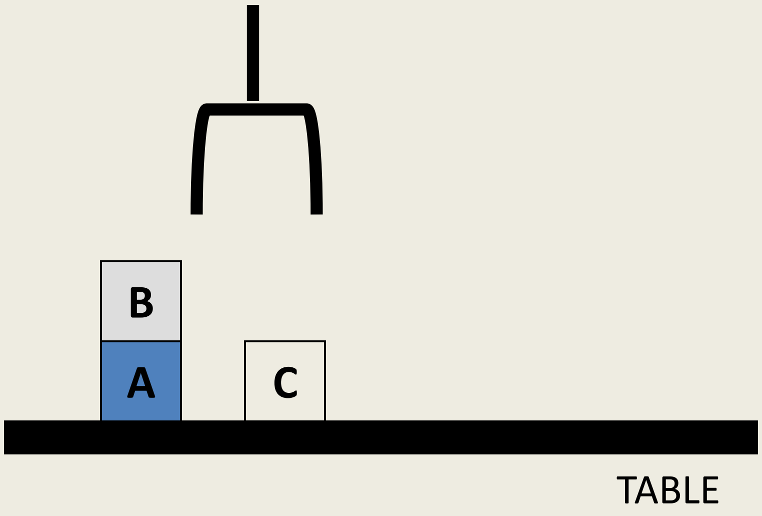
Example for a Planning Problem

- “Pick Up the Trash”
- Sequence of actions needed to achieve the goal:
  - Search for a can
  - Move toward the can when it is found
  - Pick up the can
  - Search for the recycle bin
  - Move toward the recycle bin
  - Drop the can

Blocks World

- The **blocks world** is a micro-world that consists of a table, a set of blocks and a robot hand.
- Some domain constraints:
  - Only one block can be on another block
  - Any number of blocks can be on the table
  - The hand can only hold one block
- Typical representation:

  ontable(A)
  ontable(C)
  on(B,A)
  handempty
  clear(B)
  clear(C)



STRIPS

- STRIPS (Stanford Research Institute Problem Solver) is an automated planner developed by Fikes and Nilsson in 1971 at SRI International
- States represented as a conjunction of ground literals
  - at(Home) ^ ~have(Milk) ^ ~have(bananas) ...
- Goals are conjunctions of literals, but may have variables which are assumed to be existentially quantified
  - at(x) ^ have(Milk) ^ have(bananas) ...
- Do not need to fully specify state
  - Non-specified either don’t-care or assumed false
  - Represent many cases in small storage
  - Often only represent changes in state rather than entire situation

Operator/action representation

- Operators contain three components:
  - **Action description**
  - **Precondition** - conjunction of positive literals
  - **Effect** - conjunction of positive or negative literals which describe how situation changes when operator is applied
- Preconditions must be true in the state immediately before operator is applied; effects are true immediately after

Operator/action representation

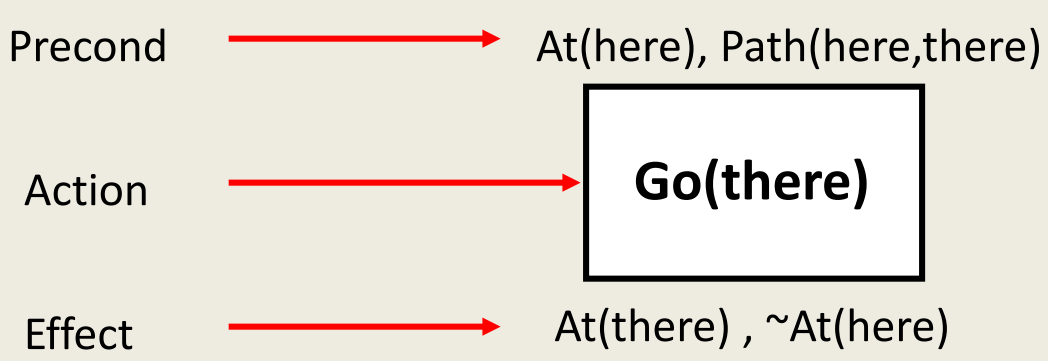
המסלול האקדמי
המכללה למינהל

- Example:

  Op[Action: Go(there),

  Precond: At(here) ^ Path(here,there),

  Effect: At(there) ^ ~At(here)]



Operators

- Each operator is represented by:
  - a list of preconditions
  - a list of new facts to be added (add-effects)
  - a list of facts to be removed (delete-effects)
  - optionally, a set of (simple) variable constraints

המסלול האקדמי
המכללה למינהל

Block World Operators

- The basic classic operations for the blocks world:
  - stack(X,Y): put block X on block Y
  - unstack(X,Y): remove block X from block Y
  - pickup(X): pickup block X from the table
  - putdown(X): put block X on the table

Block World Operators

- operator(stack(X,Y),
    Precond [holding(X), clear(Y)],
    Add [handempty, on(X,Y), clear(X)],
    Delete [holding(X), clear(Y)],
    Constraints [X\=Y, Y\=table, X\=table]).
- Now describe the operator pickup(X)
  operator(pickup(X),
    Precond [ontable(X), clear(X), handempty],
    Add [holding(X)],
    Delete [ontable(X), clear(X), handempty],
    Constraints [X\=table]).

Block World Operators

- operator(unstack(X,Y),

  Precond [on(X,Y), clear(X), handempty],

  Add [holding(X), clear(Y)],

  Delete [handempty, clear(X), on(X,Y)],

  Constraints [X\=Y, Y\=table, X\=table]).
- operator(putdown(X),

  Precond [holding(X)],

  Add [ontable(X),handempty,clear(X)],

  Delete [holding(X)],

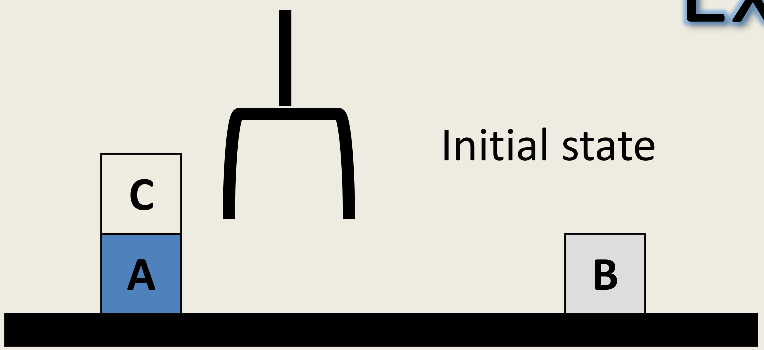
  Constraints [X\=table]).

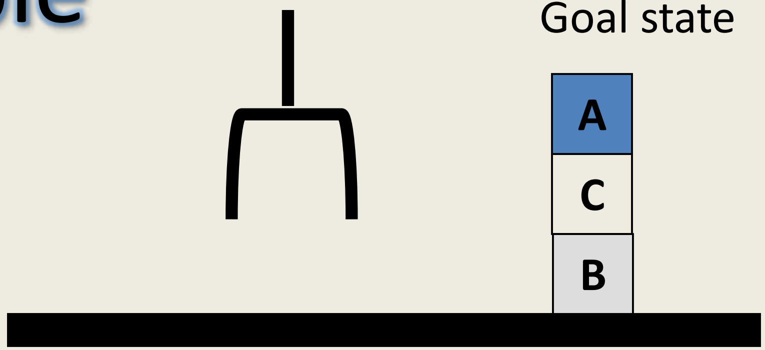
STRIPS Planning

המסלול האקדמי
המכללה למינהל

- STRIPS maintains two additional data structures:
  - **State List** - all currently true predicates.
  - **Goal Stack** - a push down stack of goals to be solved, with current goal on top of stack.
- If current goal is not satisfied by present state, examine add lists of operators, and push operator and preconditions list on stack. (Subgoals)
- When a current goal is satisfied, POP it from stack.
- When an operator is on top stack, record the application of that operator on the plan sequence and use the operator's add and delete lists to update the current state.

Example





1. Place on stack original goal:

Stack:

On(A,C) & On(C,B)

Database:

CLEAR(B)
ON(C,A)
CLEAR(C)
ONTABLE(A)
ONTABLE(B)
HANDEMPTY

Example

2. Since top goal is unsatisfied compound goal, list its unsatisfied subgoals on top of it:

Stack:

On(C,B)
On(A,C)
On(A,C) & On(C,B)

Database (unchanged):

CLEAR(B)
ON(C,A)
CLEAR(C)
ONTABLE(A)
ONTABLE(B)
HANDEMPTY

Example

3. Since top goal is unsatisfied single-literal goal, find rule whose instantiated add-list includes the goal, and: a. Replace the goal with the instantiated rule; b. Place the rule's instantiated precondition formula on top of stack

Stack:

Holding(C) & Clear(B)

stack(C,B)

On(C,B)

On(A,C)

On(A,C) & On(C,B)

Database (unchanged):

CLEAR(B)
ON(C,A)
CLEAR(C)
ONTABLE(A)
ONTABLE(B)
HANDEMPTY

Example

4. Since top goal is unsatisfied compound goal, list its subgoals on top of it:

Stack:

Clear(B)

Holding(C)

Holding(C) & Clear(B)

stack(C,B)

On(A,C)

On(A,C) & On(C,B)

Database (unchanged):

CLEAR(B)
ON(C,A)
CLEAR(C)
ONTABLE(A)
ONTABLE(B)
HANDEMPTY

Example

5. Single goal on top of stack matches data base, so remove it:

Stack:

~~**Clear(B)**~~

Holding(C)

Holding(C) & Clear(B)

stack(C,B)

On(A,C)

On(A,C) & On(C,B)

Database (unchanged):

CLEAR(B)
ON(C,A)
CLEAR(C)
ONTABLE(A)
ONTABLE(B)
HANDEMPTY

Example

המסלול האקדמי
המכללה למינהל

6. Since top goal is unsatisfied single-literal goal, find rule whose instantiated add-list includes the goal, and: a. Replace the goal with the instantiated rule; b. Place the rule's instantiated precondition formula on top of stack

Stack:

On(C,A) &Clear(C) & Handempty

unstack(C,A) **Holding(C)**

Holding(C) & Clear(B)

stack(C,B)

On(A,C)

On(A,C) &On(C,B)

Database: (unchanged)

Example

7. Compound goal on top of stack matches data base, so remove it:

~~On(C,A) &Clear(C) & Handempty~~

Stack:

unstack(C,A)
Holding(C) & Clear(B)
stack(C,B)
On(A,C)
On(A,C) &On(C,B)

Database (unchanged):

CLEAR(B)
ON(C,A)
CLEAR(C)
ONTABLE(A)
ONTABLE(B)
HANDEMPTY

Example

המסלול האקדמי
המכללה למינהל

8. Top item is rule, so:
   a. Remove rule from stack;
   b. Update database using rule;
   c. Keep track of rule (for solution)

Stack:

~~**unstack(C,A)**~~

Holding(C) & Clear(B)

stack(C,B)

On(A,C)

On(A,C) &On(C,B)

Database:

unstack(X,Y):

Add - [holding(X),clear(Y)]

Delete -[handempty,clear(X),on(X,Y)]

CLEAR(B)
ONTABLE(A)
ONTABLE(B)
HOLDING(C)
CLEAR(A)

Solution: {unstack(C,A)}

Example

9. Compound goal on top of stack matches data base, so remove it:

Stack:

~~**Holding(C) & Clear(B)**~~

stack(C,B)

On(A,C)

On(A,C) &On(C,B)

Database: (unchanged)

CLEAR(B)
ONTABLE(A)
ONTABLE(B)
HOLDING(C)
CLEAR(A)

Solution: {unstack(C)}

29

Example

10. Top item is rule, so:
    a. Remove rule from stack;
    b. Update database using rule;
    c. Keep track of rule (for solution)

Stack:

~~**stack(C,B)**~~

On(A,C)

On(A,C) &On(C,B)

Database:

stack(X,Y):

Add - [handempty,on(X,Y),clear(X)]

Delete - [holding(X),clear(Y)]

ONTABLE(A)
ONTABLE(B)
HANDEMPTY
CLEAR(A)
CLEAR(C)
ON(C,B)

Solution: {unstack(C), stack(C,B)}

Example

המסלול האקדמי
המכללה למינהל

11. Since top goal is unsatisfied single-literal goal, find rule whose instantiated add-list includes the goal, and:
    a. Replace the goal with the instantiated rule;
    b. Place the rule's instantiated precondition formula on top of stack

Stack:

Holding(A) &Clear(C)

stack(A,C)

On(A,C)

On(A,C) &On(C,B)

Database:
(unchanged)

ONTABLE(A)
ONTABLE(B)
HANDEMPTY
CLEAR(A)
CLEAR(C)
ON(C,B)

Solution: {unstack(C), stack(C,B)}

Example

12. Since top goal is unsatisfied compound goal, list its unsatisfied sub-goals on top of it:

Stack:

Holding(A)

Holding(A) &Clear(C)

stack(A,C)

On(A,C) &On(C,B)

Database: (unchanged)

ONTABLE(A)
ONTABLE(B)
HANDEMPTY
CLEAR(A)
CLEAR(C)
ON(C,B)

Solution: {unstack(C), stack(C,B)}

Example

13. Since top goal is unsatisfied single-literal goal, find rule whose instantiated add-list includes the goal, and:
   a. Replace the goal with the instantiated rule;
   b. Place the rule's instantiated precondition formula on top of stack

Stack:

Ontable(A) &Clear(A) &Handempty

pickup(A)

Holding(A) &Clear(C)

stack(A,C)

On(A,C) &On(C,B)

Holding(A)

Database:
(unchanged)

Solution: {unstack(C), stack(C,B)}

Example

המסלול האקדמי
המכללה למינהל

14. Compound goal on top of stack matches data base, so remove it:

Stack:

~~Ontable(A) &Clear(A) &Handempty~~

pickup(A)

Holding(A) &Clear(C)

stack(A,C)

On(A,C) &On(C,B)

Database: (unchanged)

ONTABLE(A)
ONTABLE(B)
HANDEMPTY
CLEAR(A)
CLEAR(C)
ON(C,B)

Solution: {unstack(C), stack(C,B)}

34

Example

המסלול האקדמי
המכללה למינהל

15. Top item is rule, so:
    a. Remove rule from stack;
    b. Update database using rule;
    c. Keep track of rule (for solution)

Stack:

~~pickup(A)~~
Holding(A) &Clear(C)
stack(A,C)
On(A,C) &On(C,B)

Database:

pickup(X):

Add - [holding(X)]

Delete - [ontable(X),clear(X),handempty]

ONTABLE(B) ON(C,B) CLEAR(C) HOLDING(A)

Solution: {unstack(C), stack(C,B),pickup(A)}

Example

16. Compound goal on top of stack matches data base, so remove it:

Stack:

~~**Holding(A) &Clear(C)**~~

stack(A,C)

On(A,C) &On(C,B)

Database:

(unchanged)

ONTABLE(B)
ON(C,B)
CLEAR(C)
HOLDING(A)

Solution: {unstack(C), stack(C,B),pickup(A)}

Example

המסלול האקדמי
המכללה למינהל

17. Top item is rule, so:
    a. Remove rule from stack;
    b. Update database using rule;
    c. Keep track of rule (for solution)

Stack:

~~**stack(A,C)**~~

On(A,C) &On(C,B)

Database:

stack(X,Y):

Add - [handempty,on(X,Y),clear(X)]

Delete - [holding(X),clear(Y)]

ONTABLE(B)
ON(C,B)
ON(A,C)
CLEAR(A)
HANDEMPTY

Solution: {unstack(C), stack(C,B),pickup(A), stack(A,C)}

המסלול האקדמי
המכללה למינהל

Example

18. Compound goal on top of stack matches data base, so remove it:

Stack:

~~On(A,C) &On(C,B)~~

Database:

stack(X,Y):

Add - [handempty,on(X,Y),clear(X)]

Delete - [holding(X),clear(Y)]

ONTABLE(B)
ON(C,B)
ON(A,C)
CLEAR(A)
HANDEMPTY

Solution: {unstack(C), stack(C,B),pickup(A), stack(A,C)}

---

19. Stack is empty, so stop.

Solution: {unstack(C), stack(C,B), pickup(A), stack(A,C)}

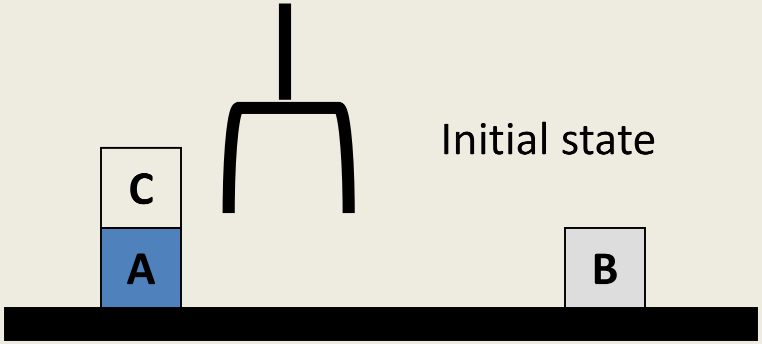
Example

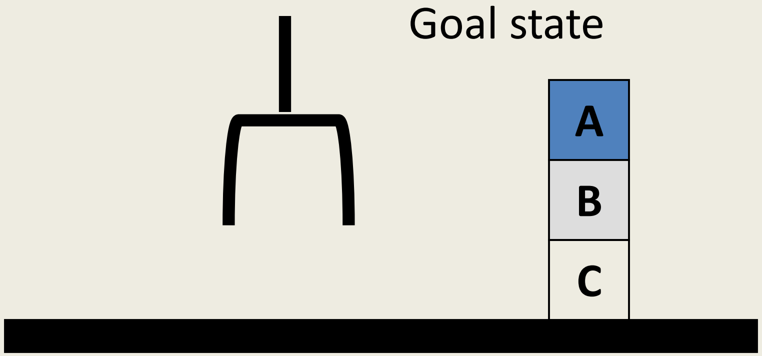
In solving this problem, we took some shortcuts—we branched in the right direction every time.

In practice, searching can be guided by:

1. Heuristic information (e.g., try to achieve "HOLDING(x)" last)
2. Detecting unprofitable paths (e.g., when the newest goal set has become a superset of the original goal set)
3. Considering useful operator side effects (by scanning down the stack).

Sussman Anomaly





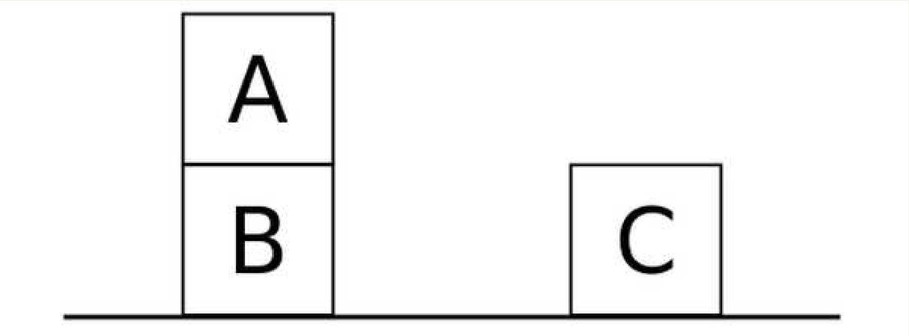
I: ON(C,A)
ONTABLE(A)
ONTABLE(B)
ARMEMPTY

G: ON(A,B)
ON(B,C)

 40

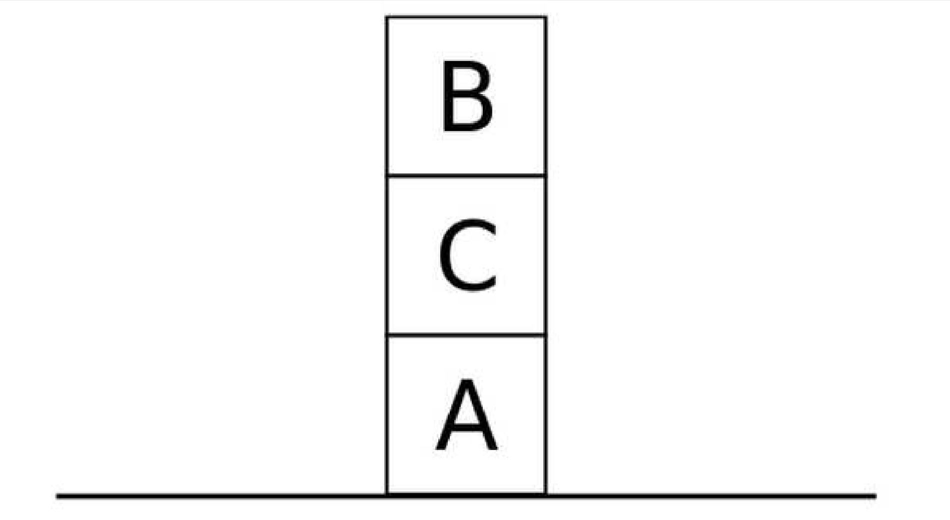
Sussman Anomaly

- Suppose the planner starts by pursuing Goal 1.
- The straightforward solution is to move C out of the way, then move A atop B. But while this sequence accomplishes Goal 1, the agent cannot now pursue Goal 2 without undoing Goal 1, since both A and B must be moved atop C:



Sussman Anomaly

- If instead the planner starts with Goal 2, the most efficient solution is to move B.
- But again, the planner cannot pursue Goal 1 without undoing Goal 2:



Planning on robots

- Sense initial state using sensors
- Create a full plan given goal state (given task)
- Feed plan, step-by-step to motors
  - No need to sense again

What’s wrong with this?

When plan goes wrong

- Dynamic environment
  - State changes even if no operator applied
- Non-deterministic
  - State changes not according to operator specs
- Inaccessible
  - Cannot sense entire state of the world
- Continuous
  - Predicate-based description of world is discrete
  - This means robot may be in a state not describable to the planner
  - e.g., between two grid cells

Closed World Assumption

- **Closed World Assumption**: everything known to be true in the world is included in the state description. Anything not listed is false.
- The opposite of the closed world assumption is the **open world assumption**, stating that lack of knowledge does not imply falsity.

The Frame Problem

- The frame problem is the problem of finding adequate collections of axioms for a viable description of a robot environment.
- Representing the state of a robot requires the use of many axioms that simply imply that things in the environment don't change arbitrarily.
- For example, in the blocks worlds additional axioms are required to infer facts such as a block does not change position if it's not moved.

שאלות?

The Reactive Paradigm

- ***<u>Reactive:</u>***
  - No PLAN component
  - No internal state (no global world model)
  - Direct connection from sensors to actions
  - S-R (stimulus response) systems
  - No choices, no alternatives

The Reactive Paradigm

- Pros
  - Rapid execution times
  - No need for memory
  - Supports good software design principles
- Cons
  - Reactive behaviors are not amenable to mathematical proofs showing they are sufficient and correct for a task

Behavior-Based Robots

המסלול האקדמי
המכללה למינהל

- “There are no general purpose animals… why should there be general purpose robots?” D. MacFarland

Behavior

Behavior

- A **behavior** is a mapping of sensory inputs to a pattern of motor actions
- Similar to a fully-instantiated planning operator
  - No variables (i.e, pick-up-A, not pick-up(A))
- Preconditions (what must be true to be executable)
- Add/delete list (what changes once behavior executes)
- Stop conditions (when the behavior terminates)

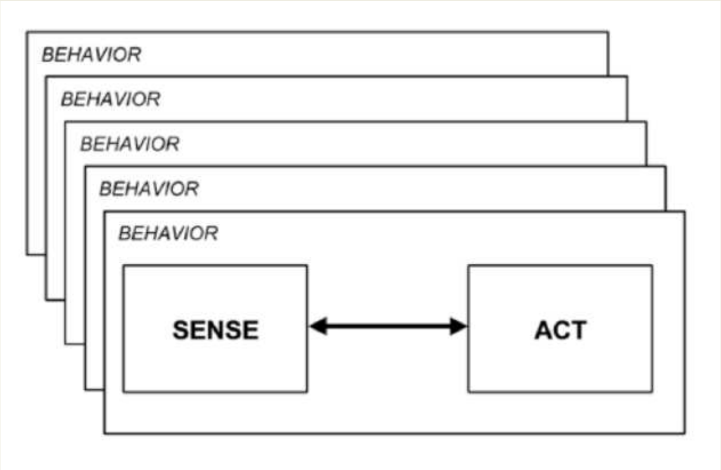
Behavior Example

Behavior

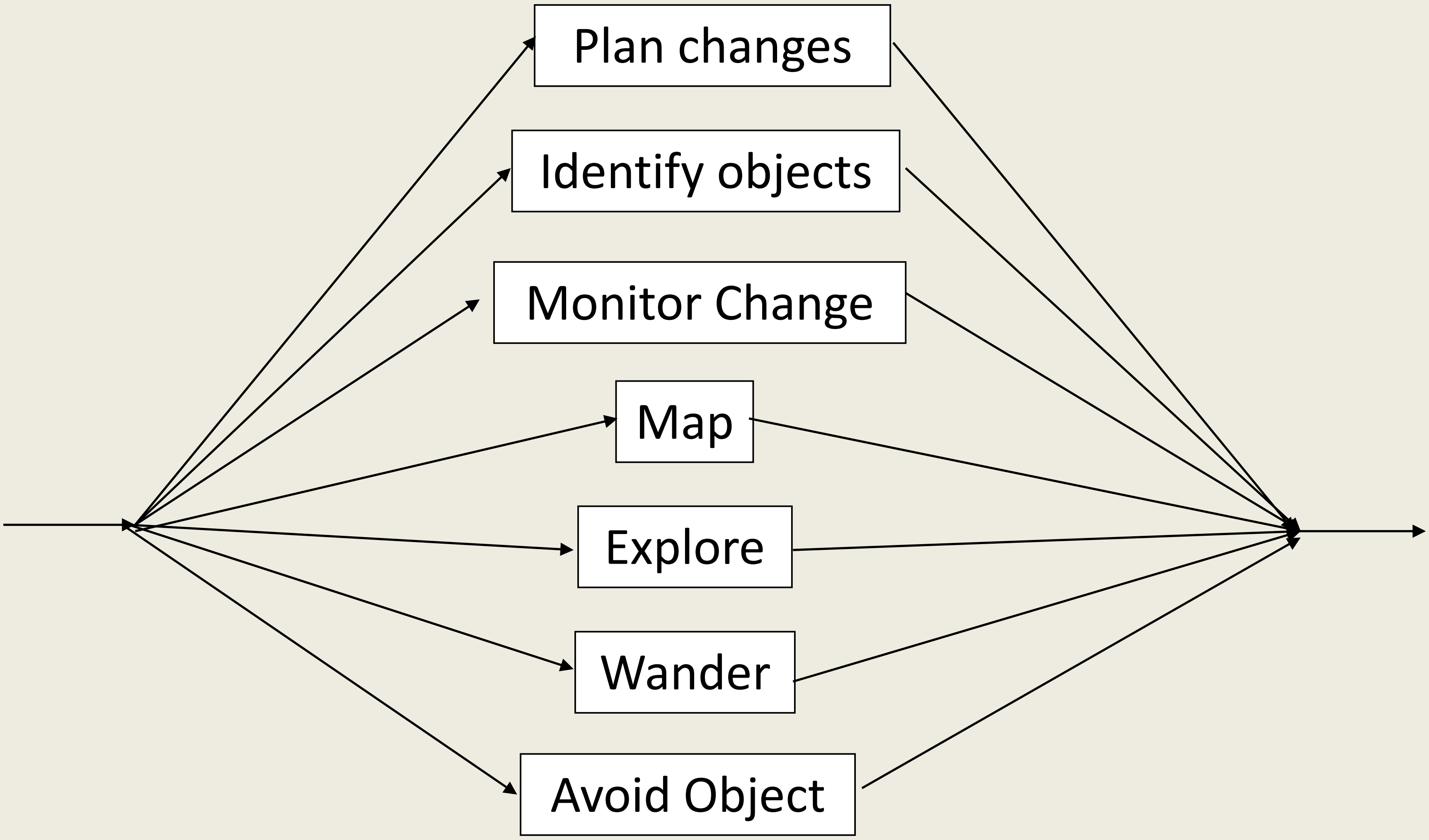
- A **behavior** is a mapping of sensory inputs to a pattern of motor actions
- Similar to a fully-instantiated planning operator
  - No variables (i.e, pick-up-A, not pick-up(A))
- Preconditions (what must be true to be executable)
- Add/delete list (what changes once behavior executes)
- Stop conditions (when the behavior terminates)

Behaviors

- Behaviors are independent and operate concurrently
  - One behavior does not know what another behavior is doing or perceiving
- The overall behavior of the robot is emergent
  - There is no explicit "controller" module which determines what will be done

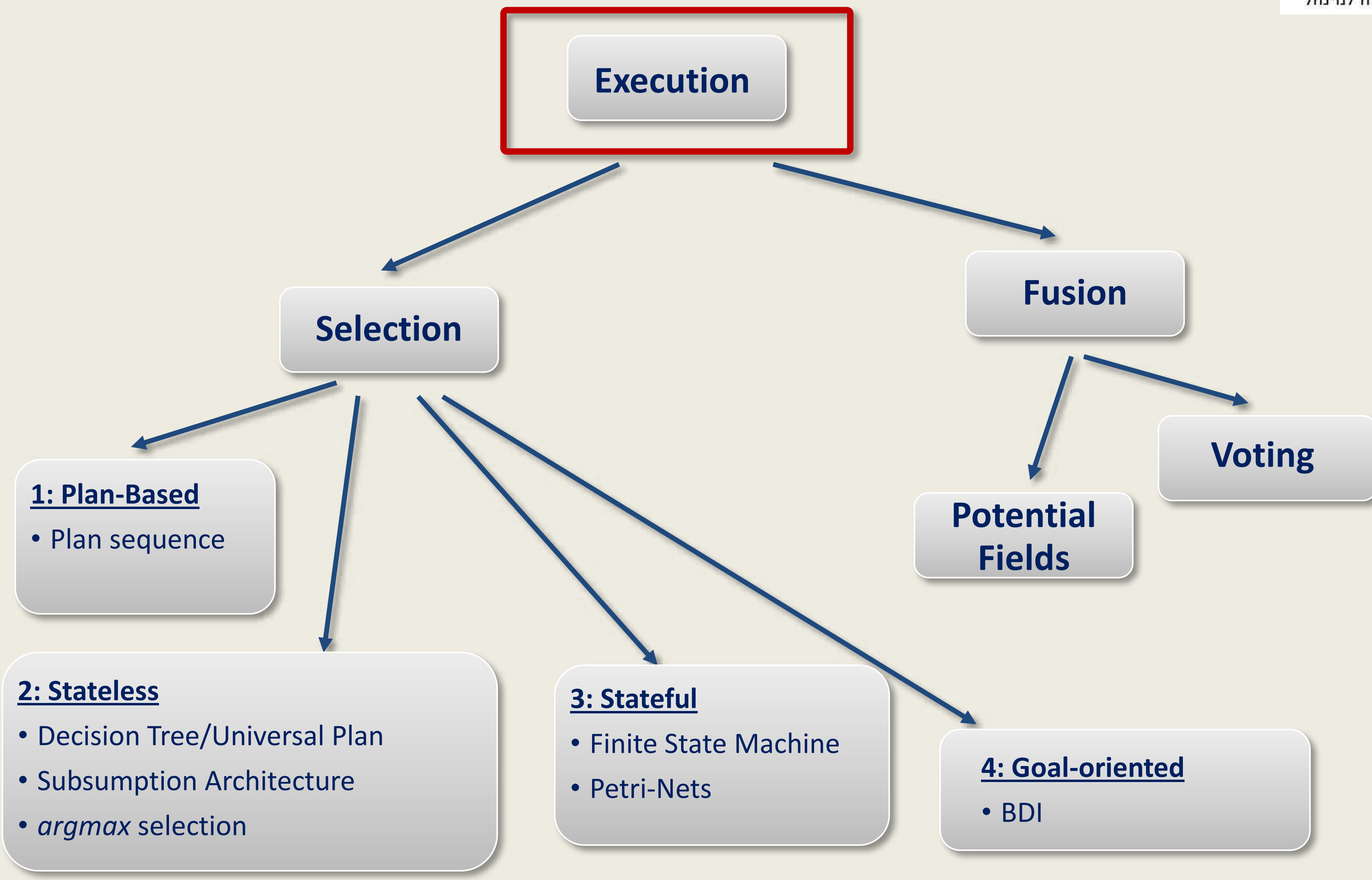


Behaviors

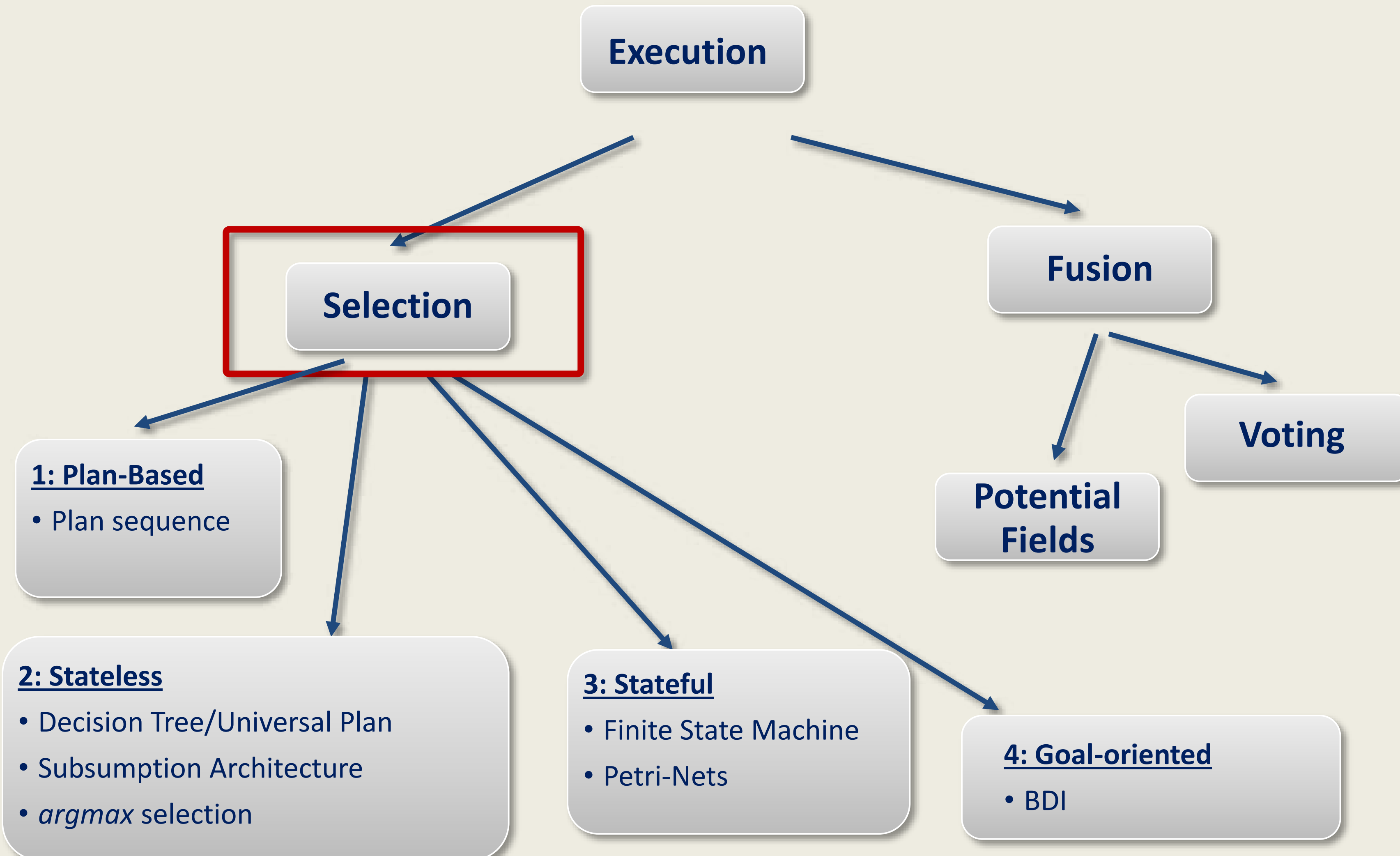


Plan changes
Identify objects
Monitor Change
Map
Explore
Wander
Avoid Object

Plan Execution Languages



Plan Execution Languages



Behavior Selection

- Behaviors compete for control

Key questions:

- How do we select the correct behavior?
- When do we terminate the selected behavior?

Example: Office Delivery

המסלול האקדמי
המכללה למינהל

- Behaviors: Go-In-Room, Go-Out-Room, Travel-Hallway
- Which one to apply?
  - Check whether in room, in hallway, … ?
  - Or, must start in known location, track motion
- When to terminate?
  - Must know success or failure (how?)
  - Must not terminate while progress still made?