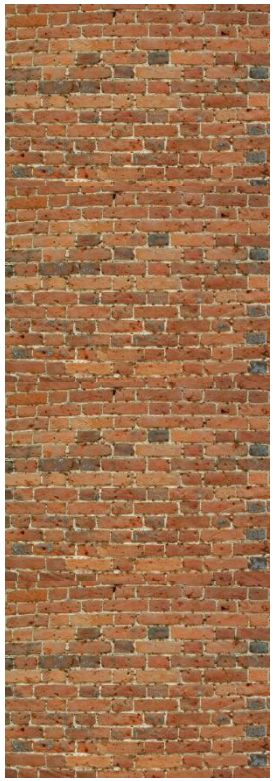
COSC1125/1127 Artificial Intelligence

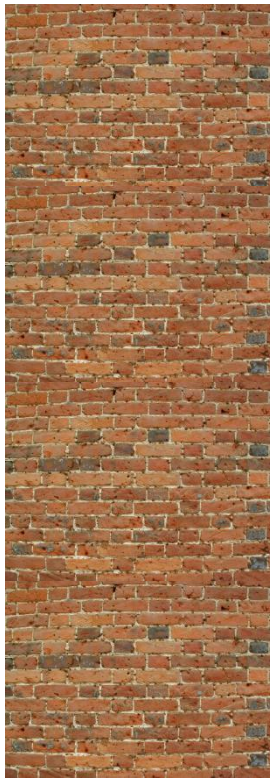
Week 6: Automated Planning

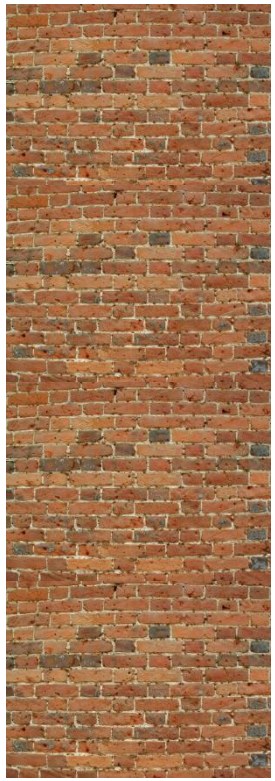
[RN2] Part IV - Chapters 11
[RN3] Part III - Chapters 10

DARPA Grand Challenge



DARPA Grand Challenge





Aircraft control

- Route aircraft between runways and terminals
- Aircraft must be kept safely separated.
- Safe distance depends on aircraft size and mode of travel (pushing or under own power).
- Minimize taxi and wait times.

Planning

Declarative Knowledge focus on the world and what is true (So far)

Procedural Knowledge focus on how to change the world (now)

Need to specify how an action changes the world

Consider a robot that performs complex tasks:

- how to navigate from one room to another;
- how to put together a water pump;
- how to safely shut down a nuclear power plant;
- how to manage the operation of the space station.

To accomplish such tasks, *plans* are needed.

Plans and Planning

A plan is a sequence of actions that perform a certain task or achieves a given goal.

An action is a primitive operation that can be carried out by the agent/robot

- Walk (not lift up right foot, place in front of left foot, …)
- Turn left (not turn left stepper motor 3 revolutions)

Planning is the process of determining a sequence of actions necessary to get some desired state of the world

Planning involves search.

Plans cannot be pre-programmed – a robot/agent must be able to constructs plans for different task.

Plans and Planning…

A planning system is an AI system that constructs plans for different tasks.

A general purpose planning system can be tailored to different sorts of tasks/domains and can deal with novel situations.

Some applications:

- Robot control: many different planning system for many different robots. One of the earliest was the STRIPS system, used to control SRI's robot *Shakey*.
- Factory and part assembly, job-shop scheduling: O-PLAN have been used at Hitashi; ISIS has been used at Westinghouse
- Mission planning: the Hubble space telescope uses a planner for allocating tasks to the telescope's viewing schedule
- Intelligent agents: lots of very recent technology for control of small "intelligent" processes, used in distributed digital libraries, software processes, animated intelligent creatures in games and simulations, etc.

Planning as Search

The task of constructing a plan is a search problem

- Initial state: current state of the world.
- Goal state: state of the world we need to achieve.
- Operations: actions which change the world.
- Constructing a plan is an answer to the question:

  This is how the world is (initial state) – what do I do to make it so that my goal is achieved (i.e. how do I change it to a goal state)?

Constructing a plan involves

- Considering the actions an agent could perform
- Predicting the effects of each, and then
- Constructing a sequence of actions (i.e. a plan) which will achieve the goal.

Knowledge Representation for Planning

In using search for planning we need to represent:

- **State knowledge** (*Declarative knowledge*)
- **Operator knowledge** (*Procedural knowledge*)

For state knowledge use logic

- Description of a state could be complex and require a large number of clauses.
- Contrast to 8-puzzle which has simple state description.

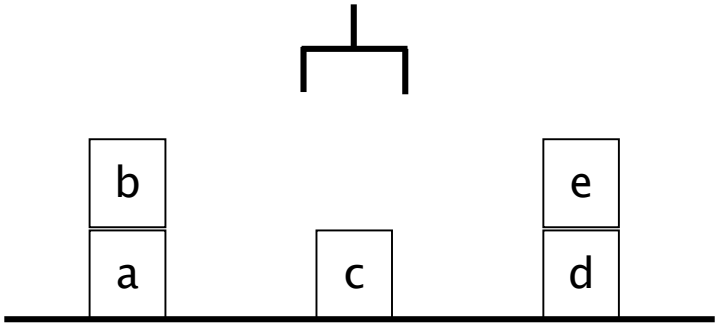
For operator knowledge use

- Logic
- **STRIPS** Operators

Any of the state space searches can be used.

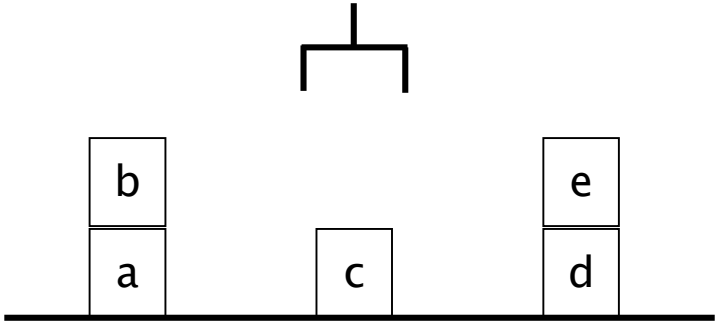
Sophisticated planners use much more sophisticated search, based on AND/OR trees.

Blocks World



Location(W, X, Y, Z)	Block *W* is at coordinates *X*, *Y*, *Z*
On(X, Y)	Block *X* is directly on top of block *Y*
Clear(X)	Block *X* has nothing on top of it
Gripping(X)	The robot arm is holding block *X*
Gripping()	The robot arm is empty
Ontable(W)	Block *W* is on the table

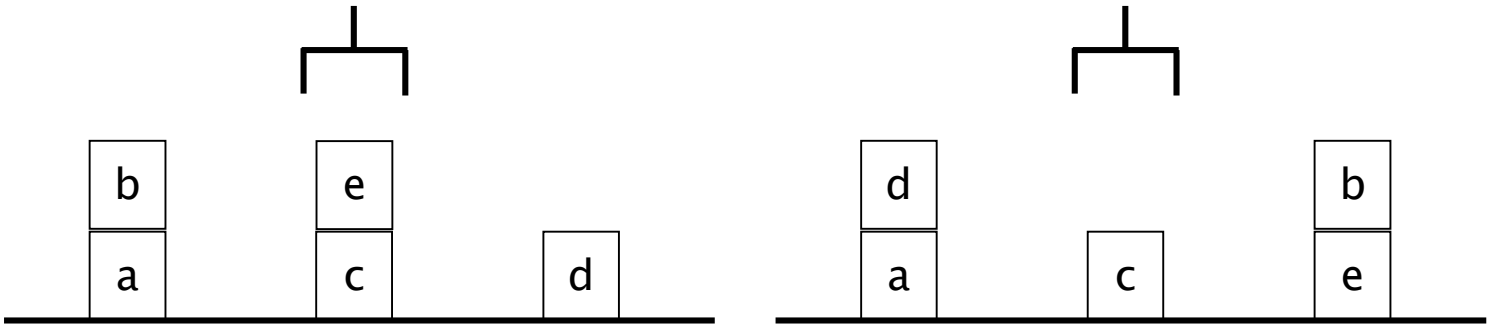
Blocks World



The above state can be represented as

Ontable(a) $\wedge$ *Ontable(c)* $\wedge$ *Ontable(d)* $\wedge$ *On(b, a)* $\wedge$ *On(e, d)* $\wedge$ *Clear(b)* $\wedge$ *Clear(c)* $\wedge$ *Clear(e)* $\wedge$ *Gripping()*

How about the following two goal states?



Blocks World Operators

Goto(X, Y, Z)	Go to location (X, Y, Z). Usually implicit as in *Pickup*(*W*)
Pickup(W)	Pick up block *W* and hold it. *W* must be clear, gripper empty
Putdown(W)	Place block *W* down on table. *W* must be in gripper.
Stack(U, V)	Place block *U* on top of block *V*. Gripper must be holding *U* and *V* must be clear.
Unstack(U, V)	Remove block *U* from block *V*. *U* must be clear. *U* must be on *V*, gripper must be empty.

These are primitive operators

- It is assumed that the gripper controller can perform these actions
- and it is not necessary to break actions into smaller steps.

Logic Representation of Actions

1. $\forall X\ (Clear(X) \Leftarrow \neg(\exists Y)\ (On(Y,X))$

- X is clear if there is no block Y such that Y is on top of X
- Procedural interpretation: to clear X go and remove any block that might be top of X
- It could be written as:

  $\forall X \neg(\exists Y)\ ((On(Y,X) \Rightarrow Clear(X))$

2. $\forall Y \forall X\ (Ontable(Y) \Rightarrow \neg On(Y,X))$

- If any block is on the table it is not on any other block

3. $\forall Y\ Gripping() \Leftrightarrow \neg Gripping(Y)$

- If the hand is empty it is not gripping any block
- If the hand is not gripping any block it is empty

Logic Representation of Actions…

4. $\forall X\ (Pickup(X) \Rightarrow (\ Gripping(X) \Leftarrow Gripping() \wedge Clear(X)\ )$

- Pickup Operator
- When the gripper is empty and block X is clear we get a new state with the hand gripping X using operator *Pickup*
- $A \Rightarrow (B \Leftarrow C)$ means

  Operator A produces new predicate(s) B when condition(s) C is true.

5. $\forall X\ (Putdown(X) \Rightarrow (\ Gripping() \wedge Ontable(X) \wedge Clear(X) \Leftarrow Gripping(X)))$

- Putdown Operator
- When the hand is gripping X we get a new state with the hand empty, and X on the table and X clear using operator *Putdown*

Logic Representation of Actions…

6. $\forall X \forall Y\ (Stack(X,Y) \Rightarrow (On(X,Y) \wedge Gripping() \wedge Clear(X) \Leftarrow (Gripping(X) \wedge Clear(Y))))$

- Stack Operator
- When Y is clear and the gripper is holding X we get a new state where X is on Y and the gripper is empty and X is clear by using the *Stack* operator

7. $\forall X \forall Y\ (Unstack(X,Y) \Rightarrow ((Gripping(X) \wedge Clear(Y)) \Leftarrow (On(X,Y) \wedge Gripping() \wedge Clear(X))))$

- Unstack Operator
- When X is on Y, X is clear and the gripper is empty we get a new state where Y is clear and the gripper is holding X by using the *Unstack* operator

Frame Axioms

Frame axioms say what does not change after an operator is applied.

8. $\forall X \forall Y \forall Z\ (Unstack(Y,Z) \Rightarrow (Ontable(X) \Leftarrow Ontable(X)))$

- When X is on the table we get a new state with X on the table using operator *Unstack(Y, Z)*
- If X is on the table its position does not change when we remove Y from Z

9. $\forall X \forall Y \forall Z\ (Stack(Y,Z) \Rightarrow (Ontable(X) \Leftarrow Ontable(X)))$

- Stacking Y on Z does not change the position of a block on the table

Frame problem

- To be able to carry out inferencing using logic it is necessary to have a frame axiom for each operator predicate pair.
- It is tractable only for small problems.

Applying Unstack Operator

Initial state

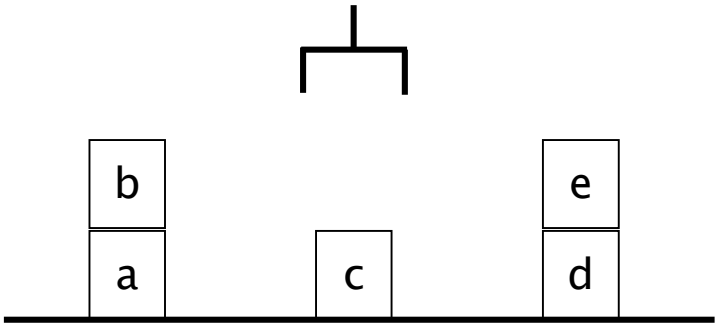
Ontable(a) ∧ Ontable(c) ∧ Ontable(d) ∧ On(b, a) ∧ On(e, d) ∧ Clear(b)
∧ Clear(c) ∧ Clear(e) ∧ Gripping()

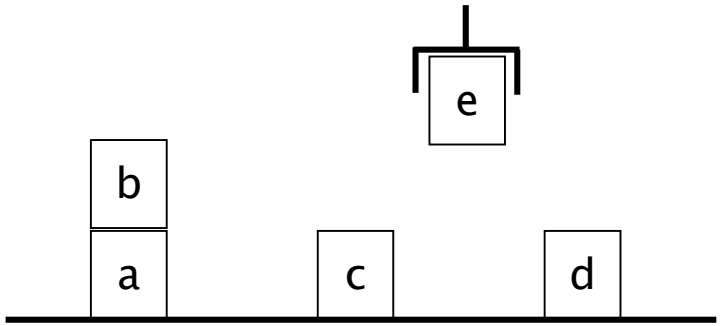
Unstack operator

7. $\forall X \forall Y\ (Unstack(X,Y) \Rightarrow ((Gripping(X) \wedge Clear(Y)) \Leftarrow (On(X,Y) \wedge Gripping() \wedge Clear(X))))$

New state after Unstack(e)

Ontable(a) ∧ Ontable(c) ∧ Ontable(d) ∧ On(b, a) ∧ Clear(b)
∧ Clear(d) ∧ Clear(c) ∧ Gripping(e)





Problems with Planning via Logic and Search

- Using frame axioms adds exponentially to the search
- Frame axioms are needed for each attribute/action pair
- Assumes sub problems are independent and can be solved in any order

  (Usually not the case, in building a house the walls must be built before the roof)
- These are still major research problems
- STRIPS approach assists with some of the problems.

STRIPS

- STanford Research Institute Planning System (STRIPS)
- Used for control of SHAKEY robot
- Have explicit **precondition**, **add** and **delete** lists for each operator
- Everything else stays the same
- Frame axioms not needed

STRIPS operators for block world

Pickup(X)

PRE: $Gripping() \wedge Clear(X) \wedge Ontable(X)$

ADD: $Gripping(X)$

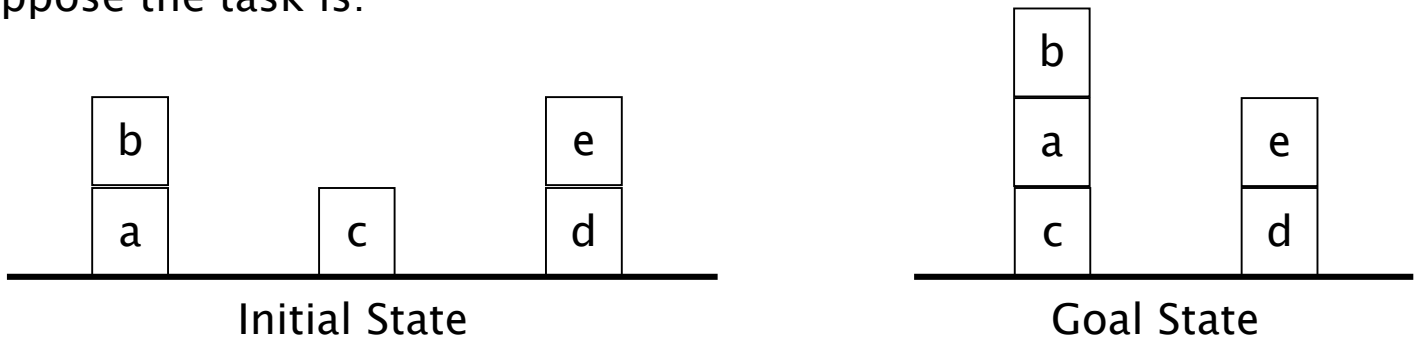
DEL: $Gripping() \wedge Clear(X) \wedge Ontable(X)$

STRIPS…

Putdown(X)	PRE:	*Gripping(X)*
	ADD:	*Gripping() ∧ Ontable(X) ∧ Clear(X)*
	DEL:	*Gripping(X)*
Stack(X, Y)	PRE:	*Gripping(X) ∧ Clear(Y)*
	ADD:	*On(X, Y) ∧ Gripping() ∧ Clear(X)*
	DEL:	*Gripping(X) ∧ Clear(Y)*
Unstack(X, Y)	PRE:	*Gripping() ∧ Clear(X) ∧ On(X, Y)*
	ADD:	*Gripping(X) ∧ Clear(Y)*
	DEL:	*Gripping() ∧ Clear(X) ∧ On(X, Y)*

Incompatible Subgoals

Suppose the task is:



In the goal we require $On(b, a) \wedge On(a,c)$

$On(b,a)$ is already true, so we might think we just have to arrange $On(a,c)$

This is impossible without first undoing $On(b,a)$ by unstack operator

The *triangle table* assists with these problems.

Triangle Table

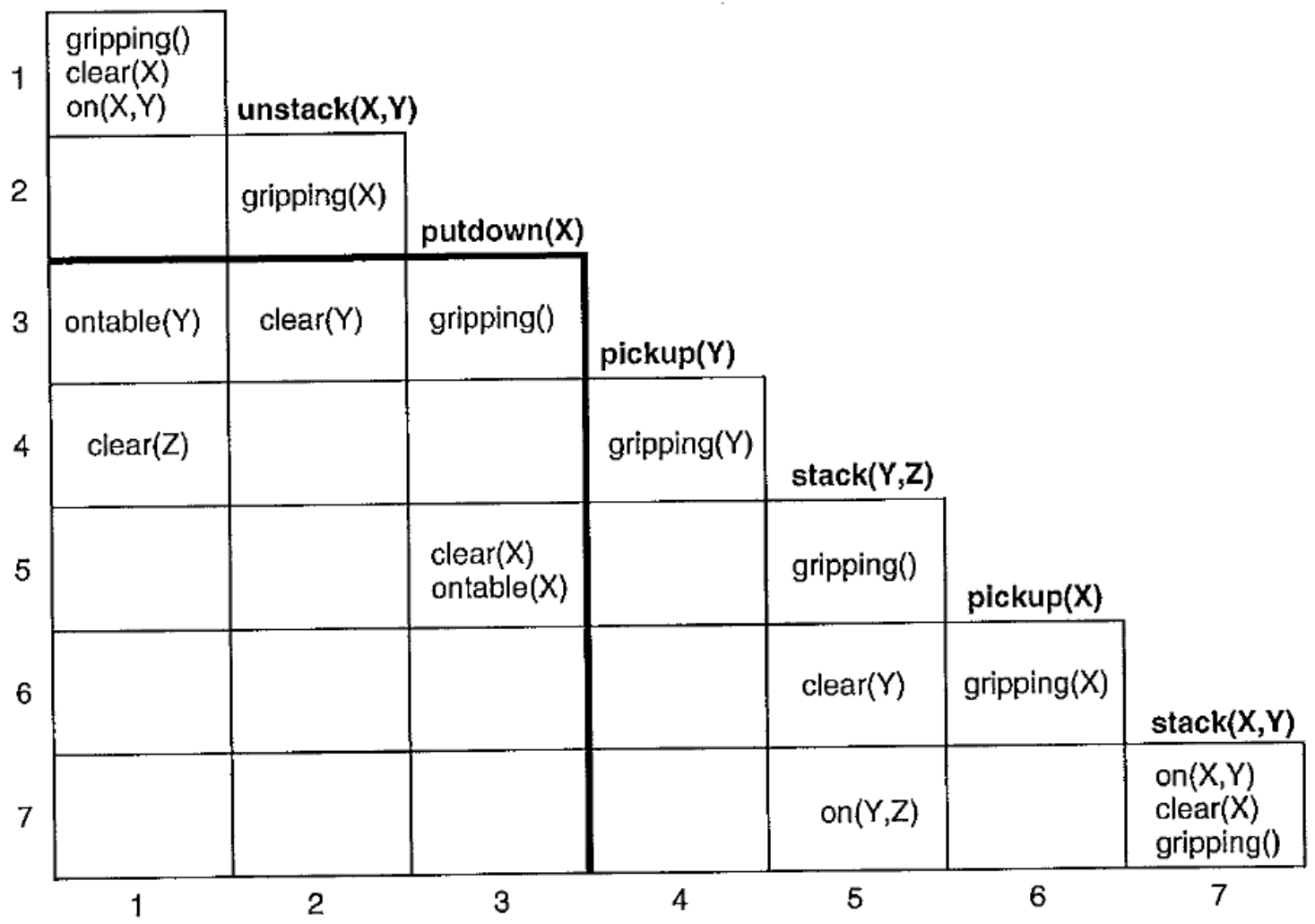


Figure 8.21 A triangle table, adapted from Nilsson (1971).

Triangle Table

- A way of re-using previously developed plans
- Store "Macro Actions" such as $Stack(X, Y) \wedge Stack(Y, Z)$
- Fits initial state: X = b, Y = a, Z = c
- Macro action is interchangeable with normal action

- Atomic acts on diagonal
- Preconditions in rows
- Post conditions in columns
- Can be used in recovering from accident
  - Find which kernel is true
  - Continue with the next action

Difficult Issues in Planning

- Re-planning if plan unexpectedly fails
- Reasoning about actions that consume time
- Resources that are shared between different processes
- Planning to meet deadlines
- Uncertainty in the world
- Planning in a dynamic and unpredictable world
- Planning by co-operating agents
- Planning is not a solved problem, still lots of research in progress

Acknowledgement: the slides were developed based on notes from Russell & Norvig's text, George Luger's text, and several RMIT computer science staff members over the years.