Intelligent Autonomous Agents and Cognitive Robotics

Rainer Marrone, Ralf Möller,
Hamburg University of Technology

Lecture

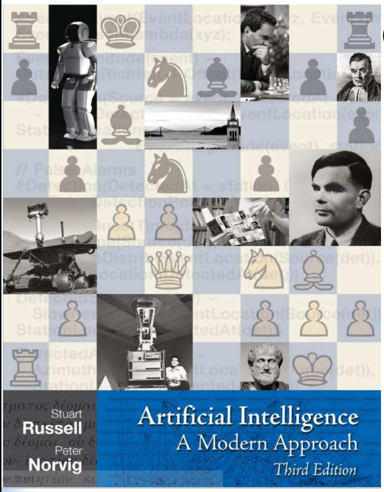
- Lecture: Tuesday, 8:00 – 9:30, D-1.025
- pdf files of the lecture will be available on StudIP.

2

Exercise

- Thursday, 15:00-16:30, H-0.08
  First exercise: 27.10
- I will upload exercise sheets every week, after the lecture.
- After the exercise, I will upload the solution as pdf.

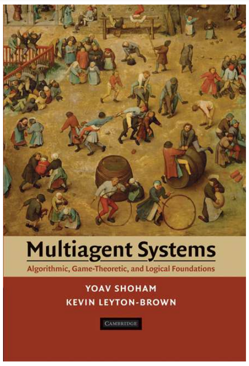
Literature



Chapters 2-5, 13 - 17

http://aima.cs.berkeley.edu
- with code repository
- further readings

Literature



Finally, we ask you not to link directly to the PDF or to distribute it electronically. Instead, we invite you to link to `http://www.masfoundations.org`. This will allow us to gauge the level of interest in the book and to update the PDF to keep it consistent with reprintings of the book.

5

Main Topics

- Solving Problems by Searching
- Adversarial Agents
- Constraint Satisfaction Problems
- Bayesian Networks
- Probabilistic Reasoning Over Time
- Decision Making
- Game Theory
- Mechanism Design

6

What is an Agent? (Wooldridge)

- Trivial (non-interesting) agents:
  - thermostat
  - UNIX daemon (e.g., xbiff)
- An intelligent agent is capable of *flexible autonomous action in some environment*
- By *flexible*, we mean:
  - *reactive*
  - *pro-active*
  - *social*

7

Reactivity

- A *reactive* system is one that maintains an ongoing interaction with its environment, and responds to changes that occur in it (in time for the response to be useful)
- The real world is more complicated: things change, information is incomplete. Many (most?) interesting environments are *dynamic*

8

Proactiveness

- Reacting to an environment is easy (e.g., stimulus → response rules)
- But we generally want agents to *do things for us*
- Hence *goal directed behavior*
- Pro-activeness = generating and attempting to achieve goals
  - Not driven solely by events
  - Taking the initiative

9

Balancing Reactive and Goal-Oriented Behavior

- We want our agents to be reactive, responding to changing conditions in an appropriate (timely) fashion
- We want our agents to systematically work towards long-term goals
- These two considerations can be at odds with one another
- Designing an agent that can balance the two remains an open research problem

10

Social Ability

- The real world is a *multi*-agent environment: we cannot go around attempting to achieve goals without taking others into account
- Some goals can only be achieved with the cooperation of others
- *Social ability* in agents is the ability to interact with other agents (and possibly humans) via some kind of *agent-communication language*. Goal is to fulfill the design objectives commitments/cooperation.

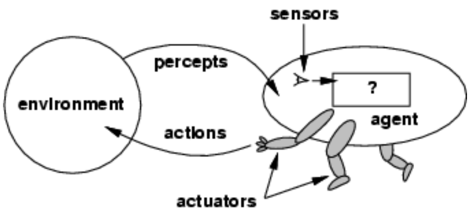
11

Agents (Norvig, Russell)

- An agent is anything that can be viewed as perceiving its environment through sensors and acting upon that environment through actuators
- *Human agent*: eyes, ears, and other organs for sensors; hands, legs, mouth, and other body parts for actuators
- *Robotic agent*: cameras and infrared range finders for sensors; various motors for actuators

12

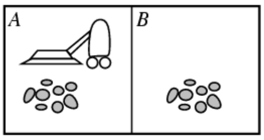
Agents and environments



- The agent function maps from percept histories to actions .
  f : P*→ A
- The agent program runs on the physical architecture to produce *f*
- agent = architecture + program

13

Vacuum-Cleaner World



- Percepts: location and contents, e.g., [A,Dirty]
- Actions: *Left*, *Right*, *Suck*, *NoOp*

14

A Vacuum-Cleaner Agent

Percept sequence	Action
$[A, Clean]$	$Right$
$[A, Dirty]$	$Suck$
$[B, Clean]$	$Left$
$[B, Dirty]$	$Suck$
$[A, Clean]$, $[A, Clean]$	$Right$
$[A, Clean]$, $[A, Dirty]$	$Suck$
⋮	⋮

Performance measure

- An agent should strive to "do the right thing", based on what it can perceive and the actions it can perform.
- Success to be measured w.r.t. an agent-local perspective of *environment states*.
- Performance measure: An objective criterion for success of an agent's behavior.
  - Performance measure of a vacuum-cleaner agent could be amount of dirt cleaned up, amount of time taken, amount of electricity consumed, amount of noise generated, etc.

Rational Agents

- Rational Agent: For each possible percept sequence, a rational agent
  - should select an action that is expected to maximize its *performance measure*,
  - given the evidence provided by the percept sequence and whatever built-in knowledge the agent has.
- Rational = Intelligent
- Rationality is distinct from omniscience (all-knowing with infinite knowledge)

17

Autonoumous Agents

- Agents can perform actions in order to obtain useful information (information gathering, exploration)
- An agent is autonomous if its behavior is determined by its own experience (with ability to learn and adapt)

18

Applications

- Robotics: Drone, Explorer, Rescue BOT
- Web Agents: Personalized Search Egines
- Logistics: Tour planning
- Medicine: Diagnosis, Surgery, …
- …

19

First task in agent design: PEAS

Must first specify the setting/task environment for intelligent agent design.

- **P**erformance measure
- **E**nvironment
- **A**ctuators
- **S**ensors

20

PEAS

- Consider, e.g., the task of designing an automated taxi driver:
  - *Performance measure*:
    - Safe, fast, legal, comfortable trip, maximize profits, …
  - *Environment*:
    - Roads, other traffic, pedestrians, customers, …
  - *Actuators*:
    - Steering wheel, accelerator, brake, signal horn, …
  - *Sensors*:
    - Cameras, sonar, speedometer, GPS, odometer, engine sensors, …

21

PEAS

- Agent: Part-picking and sorting robot
  - *Performance measure*:
    - Percentage of parts in correct bins
  - *Environment*:
    - Conveyor belt with parts, bins
  - *Actuators*:
    - Jointed arm and hand, …
  - *Sensors*:
    - Camera, joint angle sensors. …

22

Environment Types

- Fully observable vs. partially observable: An agent's sensors give it access to the state of the environment at each point in time.
- Deterministic vs. stochastic: The next state of the environment is completely determined by the current state and the action executed by the agent. If the environment is deterministic except for the actions of other agents, then the environment is strategic.
- Episodic vs. sequential: The agent's experience is divided into atomic "episodes" (each episode consists of the agent perceiving and then performing a single action), and the choice of an action in each episode depends only on the episode itself.

23

Environment Types

- Static vs. dynamic: The environment is unchanged while an agent is deliberating. (The environment is semidynamic if the environment itself does not change with the passage of time but the agent's performance score does)
- Discrete vs. continuous: Discrete if there are a limited number of distinct, clearly defined percepts, states and actions.
- Single agent vs. multiagent: An agent operating by itself in an environment.

24

Environment Types

	Chess with a clock	Chess without a clock	Taxi driving
Fully observable			
Deterministic			
Episodic			
Static			
Discrete			
Single agent			

- The environment type largely determines the agent design
- The real world is (of course) partially observable, stochastic, sequential, dynamic, continuous, multi-agent

25

Environment Types

	Chess with a clock	Chess without a clock	Taxi driving
Fully observable	Yes	Yes	No
Deterministic			
Episodic			
Static			
Discrete			
Single agent			

- The environment type largely determines the agent design
- The real world is (of course) partially observable, stochastic, sequential, dynamic, continuous, multi-agent

26

Environment Types

	Chess with a clock	Chess without a clock	Taxi driving
Fully observable	Yes	Yes	No
Deterministic	Strategic	Strategic	No
Episodic			
Static			
Discrete			
Single agent			

- The environment type largely determines the agent design
- The real world is (of course) partially observable, stochastic, sequential, dynamic, continuous, multi-agent

27

Environment Types

	Chess with a clock	Chess without a clock	Taxi driving
Fully observable	Yes	Yes	No
Deterministic	Strategic	Strategic	No
Episodic	No	No	No
Static			
Discrete			
Single agent			

- The environment type largely determines the agent design
- The real world is (of course) partially observable, stochastic, sequential, dynamic, continuous, multi-agent

28

Environment Types

	Chess with a clock	Chess without a clock	Taxi driving
Fully observable	Yes	Yes	No
Deterministic	Strategic	Strategic	No
Episodic	No	No	No
Static	Semi	Yes	No
Discrete			
Single agent			

- The environment type largely determines the agent design
- The real world is (of course) partially observable, stochastic, sequential, dynamic, continuous, multi-agent

29

Environment Types

	Chess with a clock	Chess without a clock	Taxi driving
Fully observable	Yes	Yes	No
Deterministic	Strategic	Strategic	No
Episodic	No	No	No
Static	Semi	Yes	No
Discrete	Yes	Yes	No
Single agent			

- The environment type largely determines the agent design
- The real world is (of course) partially observable, stochastic, sequential, dynamic, continuous, multi-agent

30

Environment Types

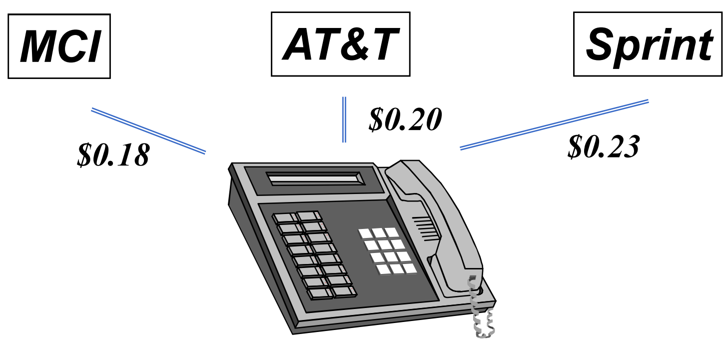
	Chess with a clock	Chess without a clock	Taxi driving
Fully observable	Yes	Yes	No
Deterministic	Strategic	Strategic	No
Episodic	No	No	No
Static	Semi	Yes	No
Discrete	Yes	Yes	No
Single agent	No	No	No

- The environment type largely determines the agent design
- The real world is (of course) partially observable, stochastic, sequential, dynamic, continuous, multi-agent

31

Mechanisms for multi-agent environments

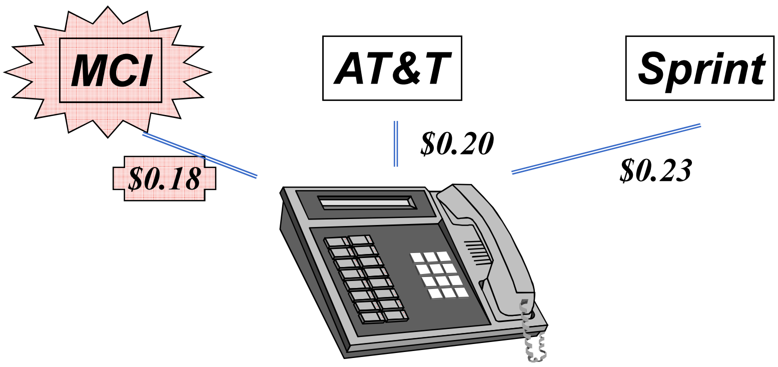
- Customer wishes to place long-distance call
- Carriers simultaneously bid, sending proposed prices
- Phone automatically chooses the carrier (dynamically)



32

Best Bid Wins

- Phone chooses carrier with lowest bid
- Carrier gets amount that it bid

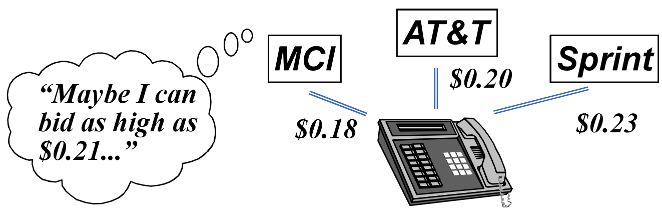


33

Attributes of the Mechanism

- ✓ Distributed
- ✓ Symmetric
- × Stable
- × Simple

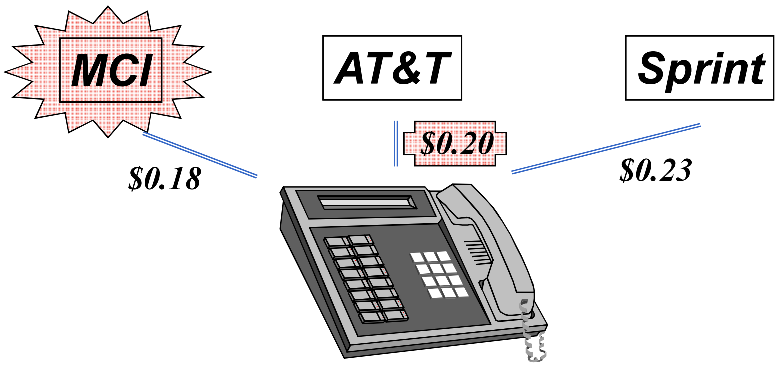
Carriers have an incentive to invest effort in strategic behavior



34

Best Bid Wins, Gets Second Price (Vickrey Auction)

- Phone chooses carrier with lowest bid
- Carrier gets amount of second-best price

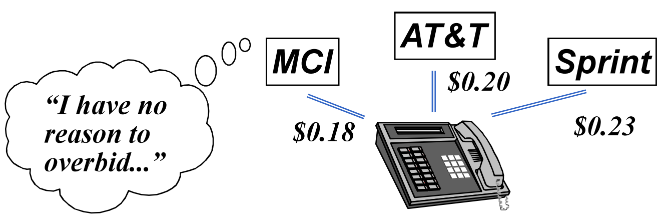


35

Attributes of the Vickrey Mechanism

- ✓ *Distributed*
- ✓ *Symmetric*
- ✓ *Stable*
- ✓ *Simple*

Carriers have no incentive to invest effort in strategic behavior



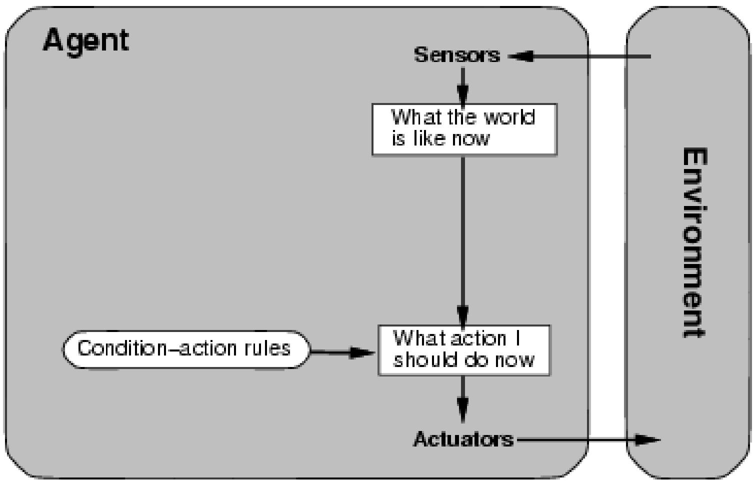
36

Agent Types

- Five basic types in order of increasing generality:
  - Simple reflex agents
  - Model-based reflex agents
  - Goal-based agents
  - Utility-based agents
  - Learning agents
    see lecture *Machine Learning*

add features

Simple Reflex Agents

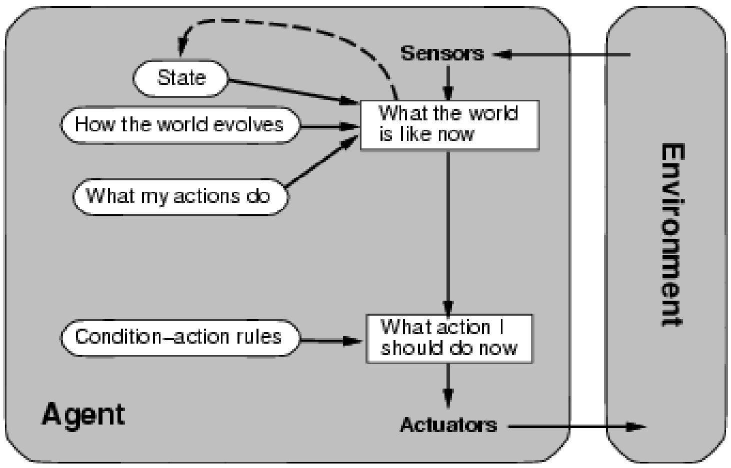


Simple Reflex Agent

- Drawbacks:
  - No autonomy
  - Decision depends on current percepts.
  - Sensitive to sensor fault

39

Model-Based Reflex Agents

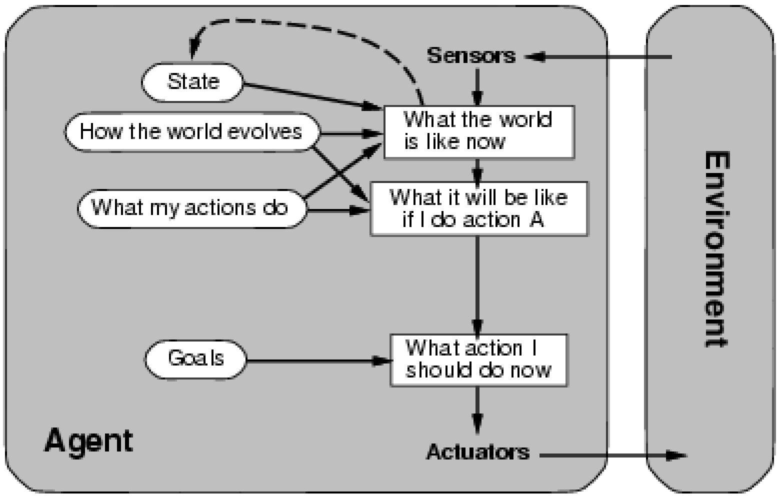


40

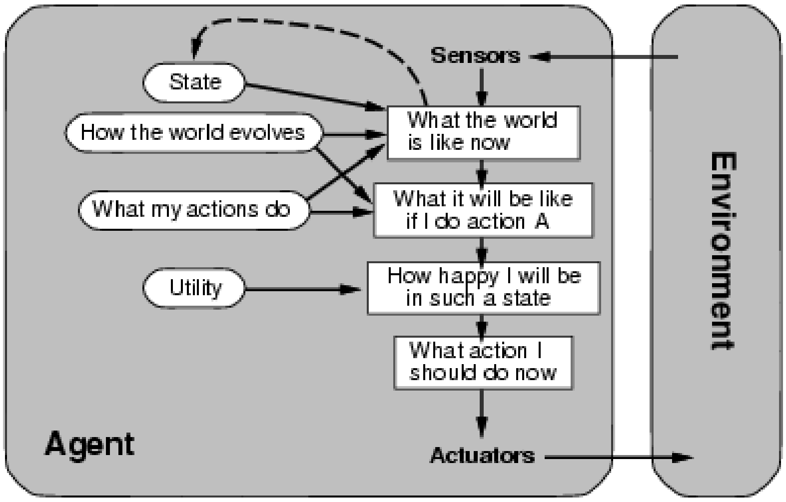
Goals for Agents

- We build agents in order to reach a goal for us
- The goals must be *specified* by us…
- But we want to tell agents what to do *without* telling them how to do it
  → Planning

Goal-Based Agents

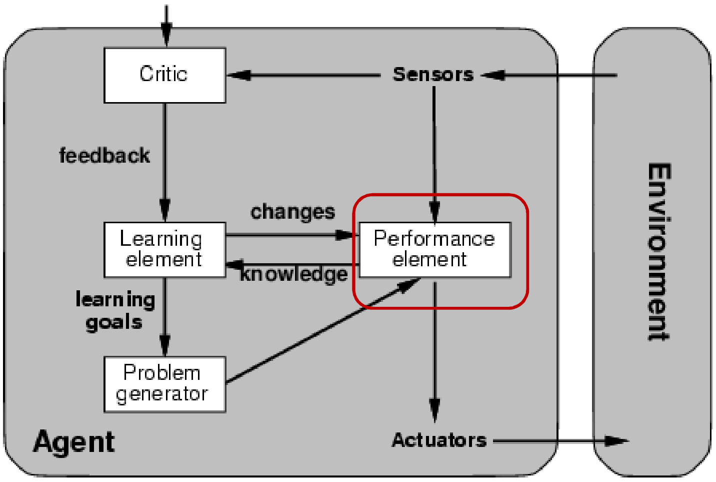


Utility-Based Agents



44

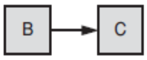
Learning Agents



45

Representation of agent states

- How to represent the states of agents?
- Atomic

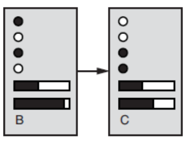


- State is a black box.
- Example:
  Want to travel from city B to C. Cities are represented as names.

46

Representation of agent states

- Factored

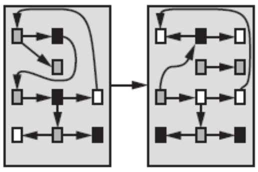


- A state consists of a vector of attribute values
- Example: A car with GPS location, Fuel, radio station, …

47

Representation of agent states

- Structured



- A state includes objects, each of which may have attributes of its own as well as relationships to other objects
- Example: natural language understanding

AI first approach: Deductive Reasoning Agents

- How can an agent decide what to do using theorem proving?
- Basic idea is to use logic to encode a theory stating the *best* action to perform in any given situation
- Let:
  - ρ be this theory (e.g. a set of rules)
  - Δ be a logical database that describes the current state of the world
  - Ac be the set of actions the agent can perform
  - $\Delta \mid\!\!-\!-_{\rho} \phi$ mean that ϕ can be proven from Δ using ρ

Deductive Reasoning Agents

```
/* try to find an action explicitly prescribed */
for each a ∈ Ac do
        if Δ |--ρ Do(a) then
                return a
        end-if
end-for
/* try to find an action not excluded */
for each a ∈ Ac do
        if Δ |-/-ρ ¬Do(a) then
                return a
        end-if
end-for
return null /* no action found */
```

50

Deductive Reasoning Agents

- Problems:
  - how to convert video camera input to logical description?
  - decision making assumes a *static* environment
  - decision making using first-order logic is *undecidable*!
- Even when we use *propositional* logic, decision making in the worst case means solving NP-complete problems (= bad news!)

51

Practical Reasoning

- Practical reasoning consists of two activities:
  - *deliberation*
    deciding *what* state of affairs we want to achieve
  - *means-ends reasoning*
    deciding *how* to achieve these states of affairs

52

What is Means-End Reasoning?

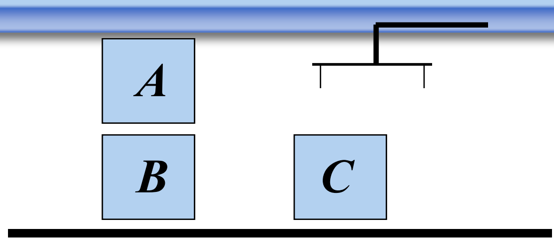
- Basic idea is to give an agent:
  - representation of goal to achieve
  - representation of actions it can perform
  - representation of the environment

and have it generate a *plan* to achieve the goal

- Essentially, this is
  automatic programming

53

The Blocks World



- We'll illustrate the techniques with the *blocks world*
- Contains a robot arm, 3 blocks (A, B, and C) of equal size, and a table

54

The Blocks World Ontology

- To represent this environment, need an *ontology*

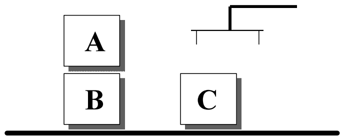
$On(x, y)$	obj x on top of obj y
$OnTable(x)$	obj x is on the table
$Clear(x)$	nothing is on top of obj x
$Holding(x)$	arm is holding x

55

The Blocks World

- Here is a representation of the blocks world described above:

  Clear(A)
  On(A, B)
  OnTable(B)
  OnTable(C)



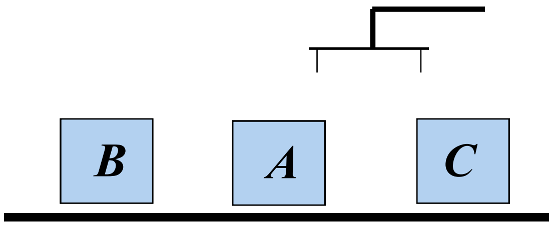
- Use the *closed world assumption*: anything not stated is assumed to be *false*

56

The Blocks World

- A *goal* is represented as a set of formula
- Here is a goal:

  $OnTable(A) \wedge OnTable(B) \wedge OnTable(C)$



57

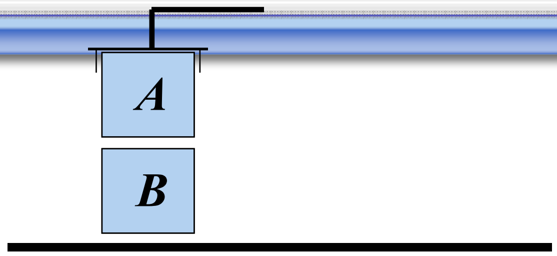
The Blocks World

- *Actions* are represented using a technique that was developed in the STRIPS planner
- Each action has:
  - a *name*
    which may have arguments
  - a *pre-condition list*
    list of facts which must be true for action to be executed
  - a *delete list*
    list of facts that are no longer true after action is performed
  - an *add list*
    list of facts made true by executing the action

Each of these may contain *variables*

58

The Blocks World Operators



- Example 1:
  The *stack* action occurs when the robot arm places the object x it is holding is placed on top of object y.

	$Stack(x, y)$
pre	$Clear(y) \wedge Holding(x)$
del	$Clear(y) \wedge Holding(x)$
add	$ArmEmpty \wedge On(x, y)$

59

The Basic STRIPS Idea

- Place goal on goal stack:

Goal1

- Considering top Goal1, place onto it its subgoals:

GoalS1-2
GoalS1-1
Goal1

- Then try to solve subgoal GoalS1-2, and continue…

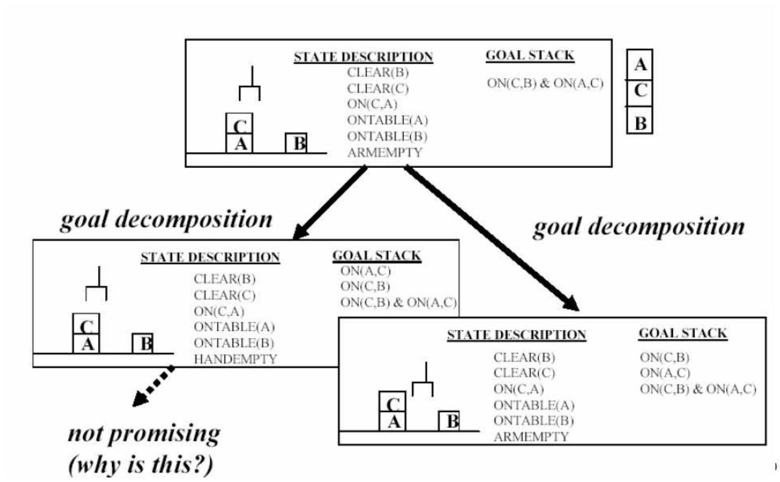
61

Stack Manipulation Rules, STRIPS

If on top of goal stack:	Then do:
Compound or single goal matching the current state description	Remove it
Compound goal *not* matching the current state description	1. Keep original compound goal on stack 2. List the unsatisfied component goals on the stack in some *new* order
Single-literal goal not matching the current state description	Find rule whose instantiated add-list includes the goal, and 1. Replace the goal with the instantiated rule; 2. Place the rule's instantiated precondition formula on top of stack
Rule	1. Remove rule from stack; 2. Update database using rule; 3. Keep track of rule (for solution)
Nothing	Stop

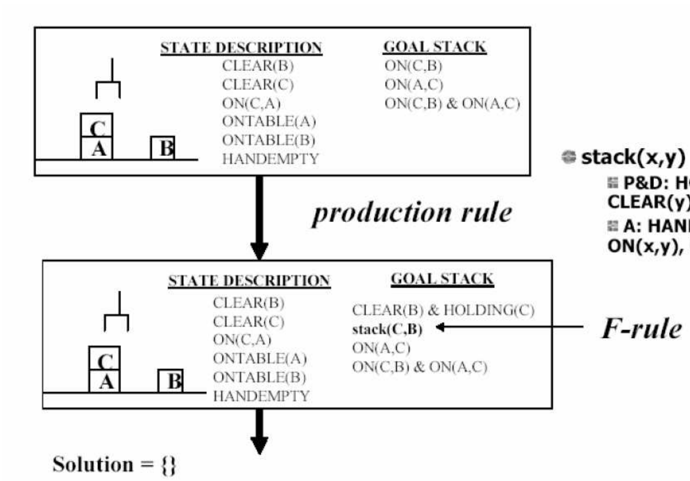
62

STRIPS in Action



63

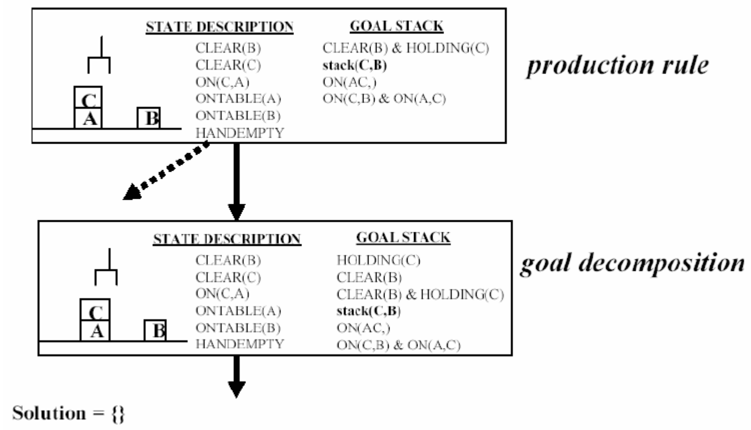
STRIPS in Action



- **stack(x,y)**
  - **P&D: HOLDING(x), CLEAR(y)**
  - **A: HANDEMPTY, ON(x,y), CLEAR(x)**

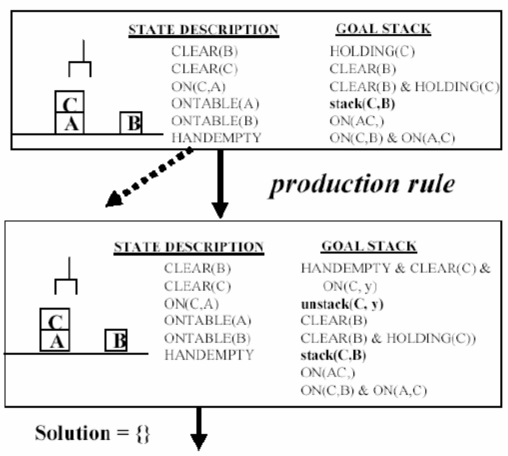
64

STRIPS in Action



65

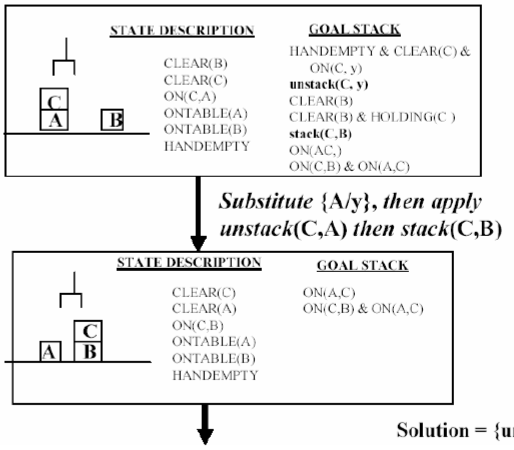
STRIPS in Action



- **unstack(x,y)**
  - **P&D: HANDEMPTY, CLEAR(x), ON(x,y)**
  - **A: HOLDING(x), CLEAR(y)**

66

STRIPS in Action

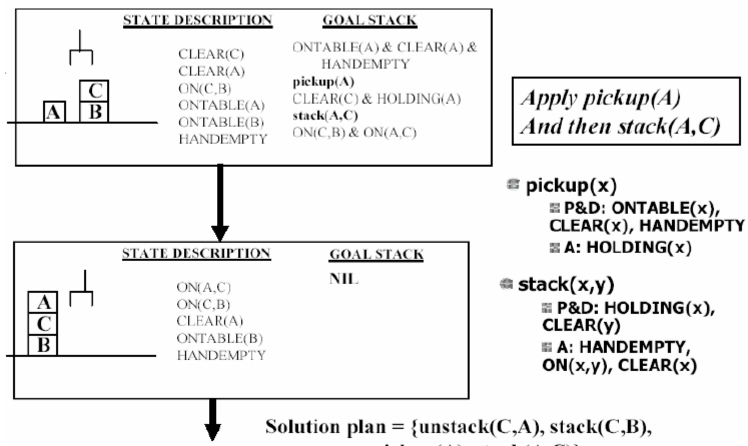


- unstack(x,y)
  - P&D: HANDEMPTY, CLEAR(x), ON(x,y)
  - A: HOLDING(x), CLEAR(y)
- stack(x,y)
  - P&D: HOLDING(x), CLEAR(y)
  - A: HANDEMPTY, ON(x,y), CLEAR(x)

Solution = {unstack(C,A), stack(C,B)}

67

STRIPS in Action



- pickup(x)
  - P&D: ONTABLE(x), CLEAR(x), HANDEMPTY
  - A: HOLDING(x)
- stack(x,y)
  - P&D: HOLDING(x), CLEAR(y)
  - A: HANDEMPTY, ON(x,y), CLEAR(x)

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

68