Lecture slides for

Automated Planning: Theory and Practice

Chapter 1
Introduction

Dana S. Nau

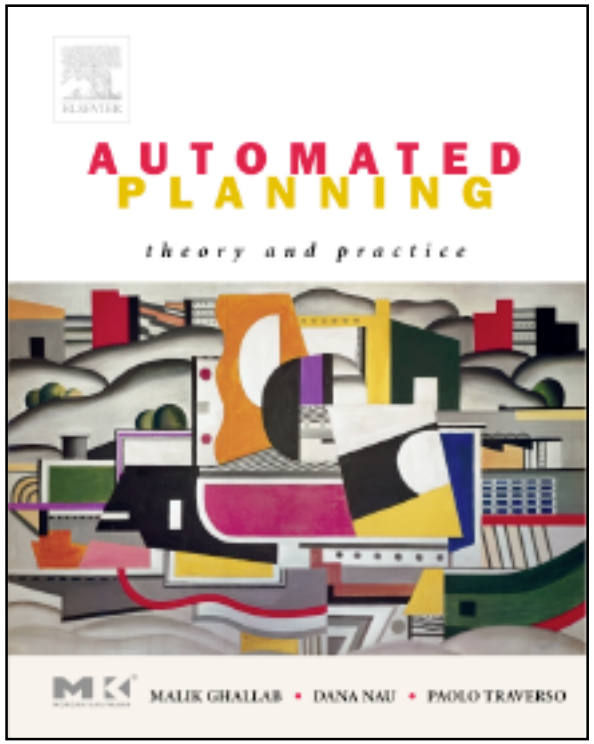
CMSC 722, AI Planning

University of Maryland, Fall 2004

Textbook

- M. Ghallab, D. Nau, and P. Traverso
  Automated Planning: Theory and Practice
  Morgan Kaufmann Publishers
  May 2004
  ISBN 1-55860-856-7

- Web site: http://www.laas.fr/planning



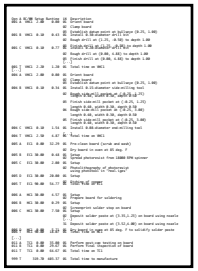
Class Background

- Things I hope you know
  - Asymptotic complexity, e.g., $O(n^3)$, $\Theta(n^5)$, $\Omega(n^2)$
    - » Differences in meaning: worst case, best case, average case
    - » Nondeterministic algorithms
    - » P, NP, NP-completeness, NP-hardness
  - Search algorithms
    - » Depth-first, breadth-first, best-first search
    - » A*, admissible versus admissible heuristics
    - » State-space versus problem reduction (AND/OR graphs)
  - Logic
    - » Propositional logic
    - » First-order logic (predicates and quantifiers)
    - » Horn clauses, Horn-clause theorem proving

Plans and Planning

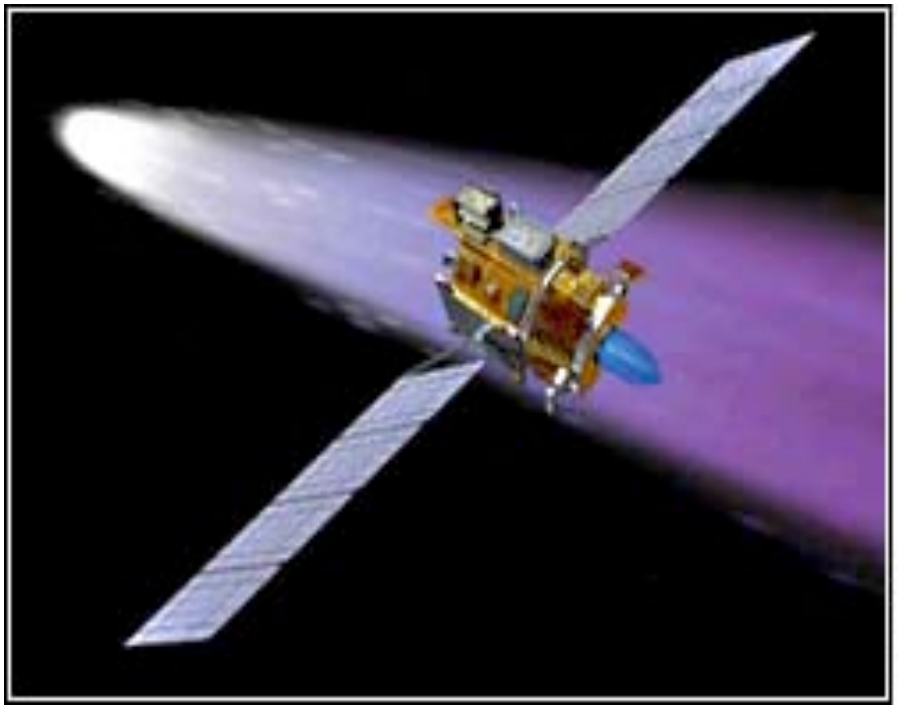
- Plan:
  - A collection of actions for performing some task or achieving some objective
- Planning:
  - There are many programs to aid human planners
    - » Project management, plan storage/retrieval, automatic schedule generation

  - Automatic plan generation is much more difficult
    - » Many research prototypes, fewer practical systems
    - » Research is starting to pay off
      - Several successes on difficult practical problems



NASA Unmanned Spacecraft

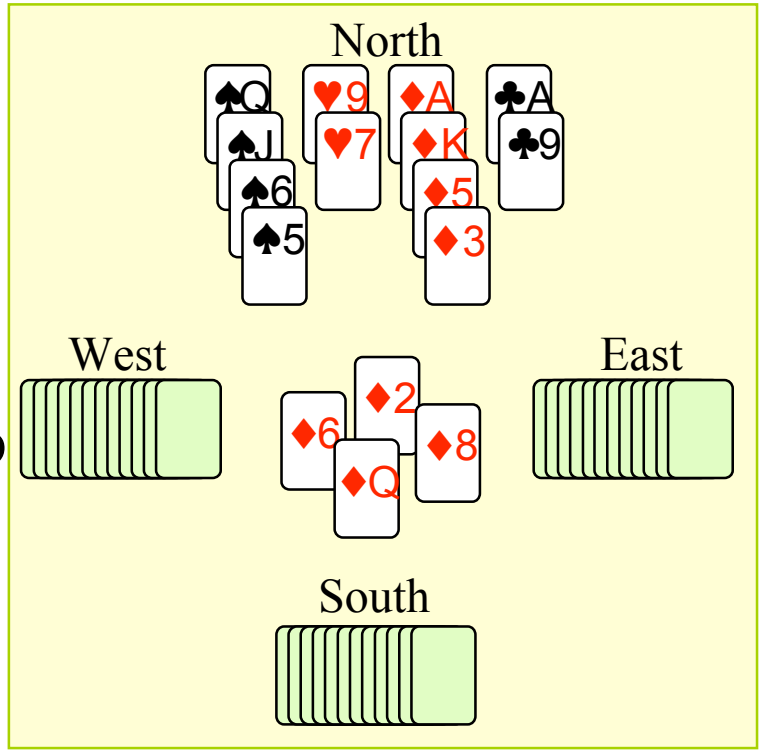
- Remote Agent eXperiment (RAX)
  - autonomous AI software for planning/control
  - ran on the DS1 spacecraft in May 1998
  - For several days it was allowed to control the spacecraft



- Mars rover
  - guided by autonomous AI planning/control software

Other Examples

- Computer bridge: *Bridge Baron*
  - Used AI planning to win the 1997 world computer bridge championship
  - Commercial software, thousands of copies sold

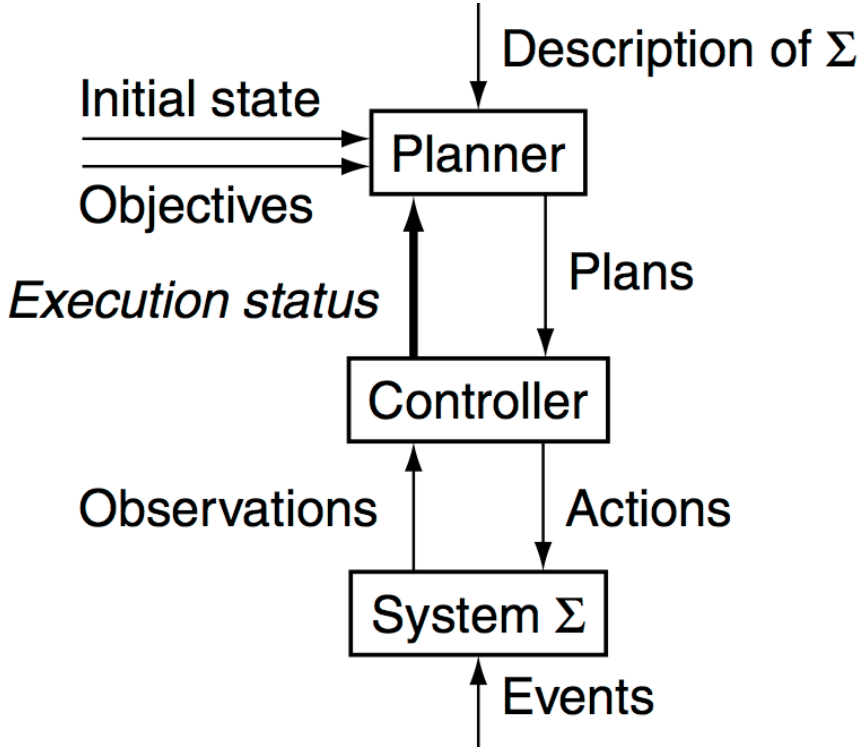


- Manufacturing process planning
  - Software included with Amada's sheet-metal bending machines
  - Used to plan bending operations

Outline

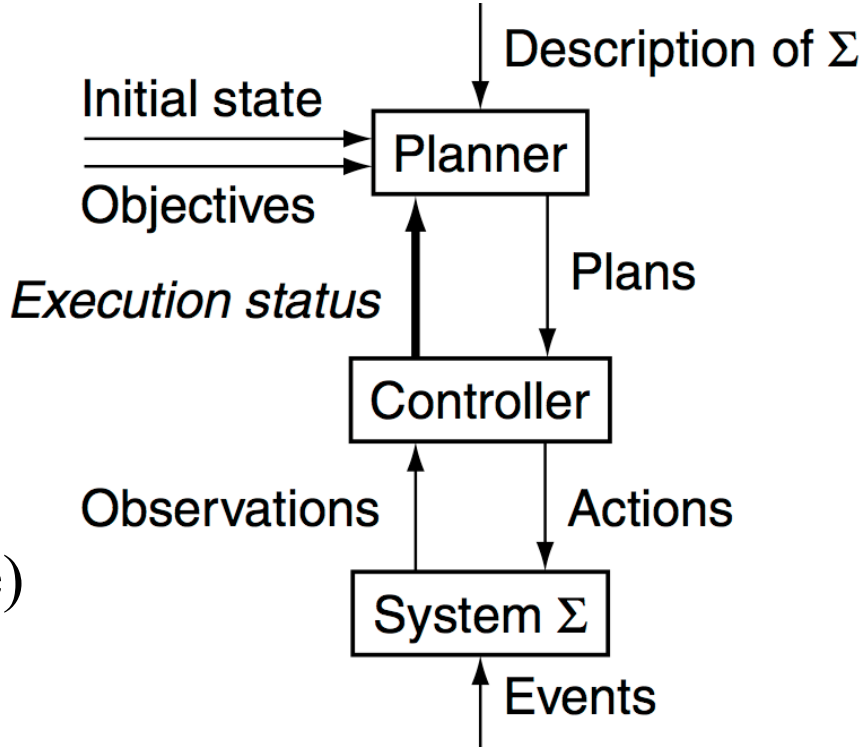
- Conceptual model
- Restrictive assumptions
- Classical planning
- Relaxing the assumptions
- A running example: Dock Worker Robots

Conceptual Model



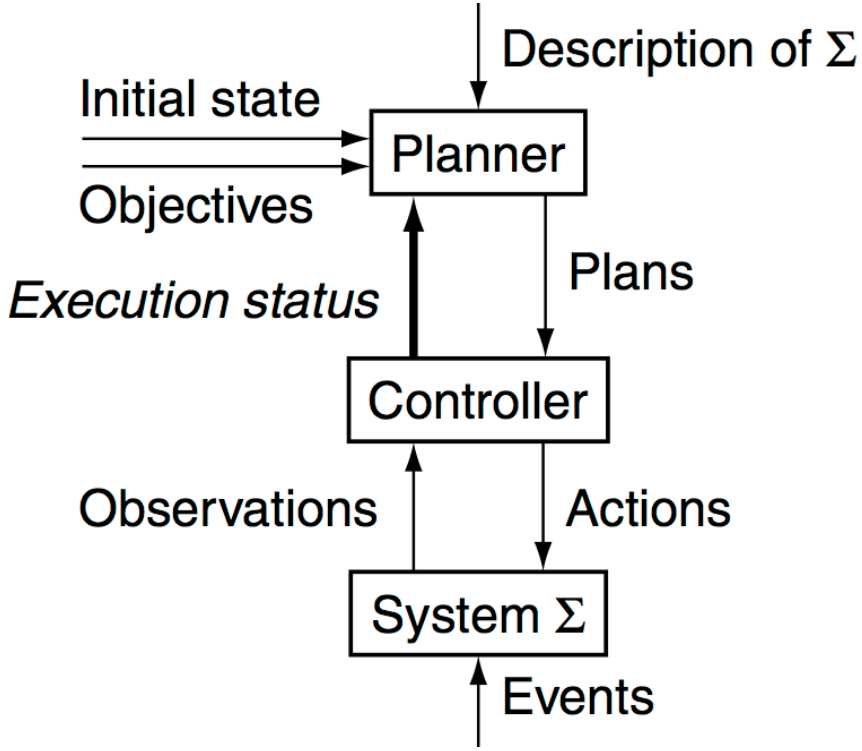
- **Ingredients:**
  - Model of the environment: *possible states*
  - Model of how the environment can change: *effects of actions*
  - Specification of *initial conditions* and *objectives*
  - Plans of actions that are generated by a planner
  - A model of execution of a plan in the environment
  - A model of observation of the environment

Conceptual Model



- State-transition system
  $\Sigma = (S,A,E,\gamma)$
  - S = {states}
  - A = {actions} (controllable)
  - E = {events} (uncontrollable)
  - state-transition function
    $\gamma: S \times (A \cup E) \rightarrow 2^S$
- Observation function $h: S \rightarrow O$
  - produces observation o about current state s
- Controller: given observation o in O, produces action a in A
- Planner:
  - input: description of Σ, initial state s_0 in S, some objective
  - output: produces a plan to drive the controller

Conceptual Model



- Possible objectives:
  - A set of goal states S_g

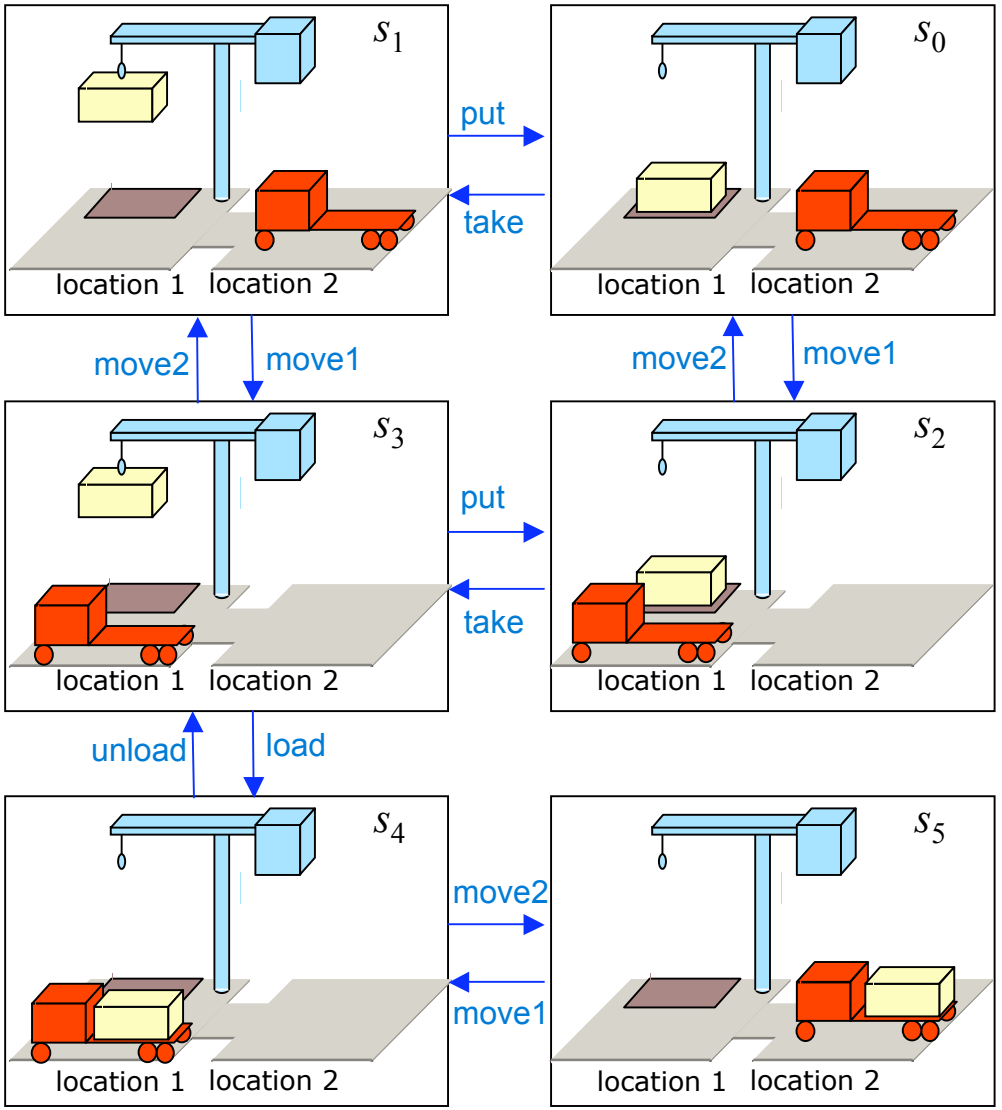
    Find sequence of state transitions ending at a goal
  - Some condition over the set of states followed by the system

    e.g., reach S_g and stay there
  - Utility function attached to states

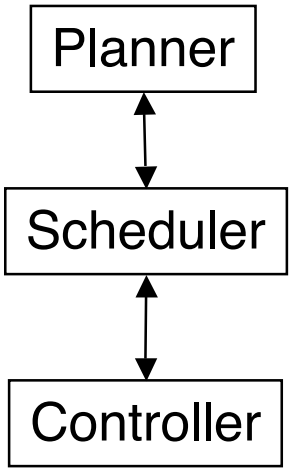
    Optimize some function of the utilities
  - Tasks to perform, specified recursively as sets of sub-tasks and actions

Conceptual Model: Example

- State transition system $\Sigma = (S,A,E,\gamma)$
  - $S = \{s_0, \ldots, s_5\}$
  - $A = \{$move1, move2, put, take, load, unload$\}$
  - $E = \{\}$
  - γ: as shown
- $h(s) = s$ for every s
- Input to planner:
  - system Σ
  - initial state s_0
  - goal state s_5



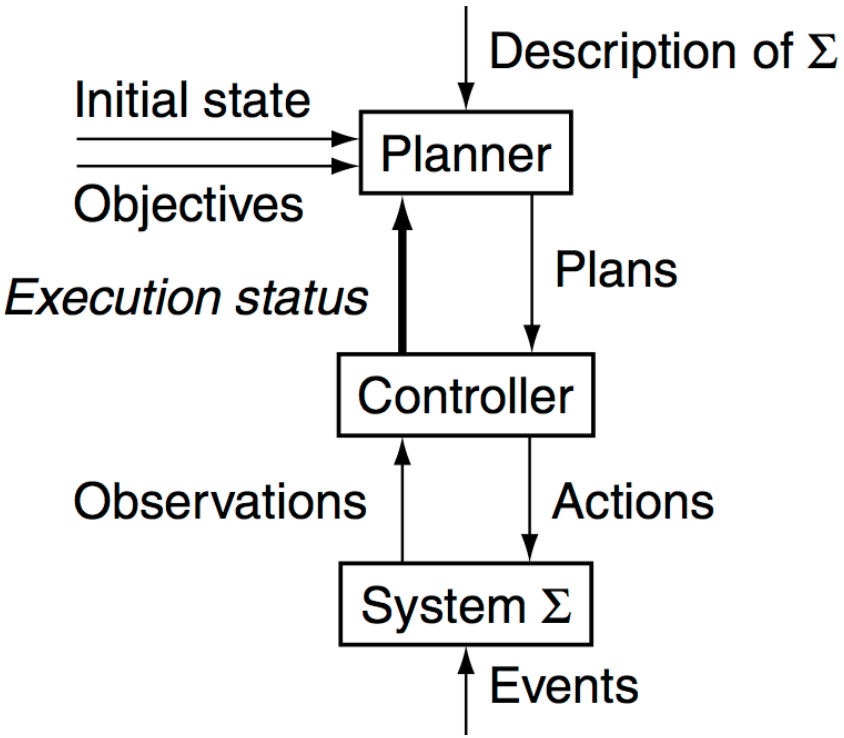
Planning Versus Scheduling



- Scheduling
  - Decide how to perform a given set of actions using a limited number of resources in a limited amount of time
  - Typically NP-complete
- Planning
  - Decide what actions to use to achieve some set of objectives
  - Can be much worse than NP-complete
    - » In the most general case, it is undecidable
    - » Most research assumes various collections of restrictions to guarantee decidability
  - We'll now look at some of the restrictions

Restrictive Assumptions

- A0 (finite Σ):
  - the state space S is finite
  - $S = \{s_0, s_1, s_2, \ldots s_k\}$ for some k
- A1 (fully observable Σ):
  - the observation function $h: S \rightarrow O$ is the identity function
  - i.e., the controller always knows what state Σ is in



$\Sigma = (S, A, E, \gamma)$

$S = \{\text{states}\}$

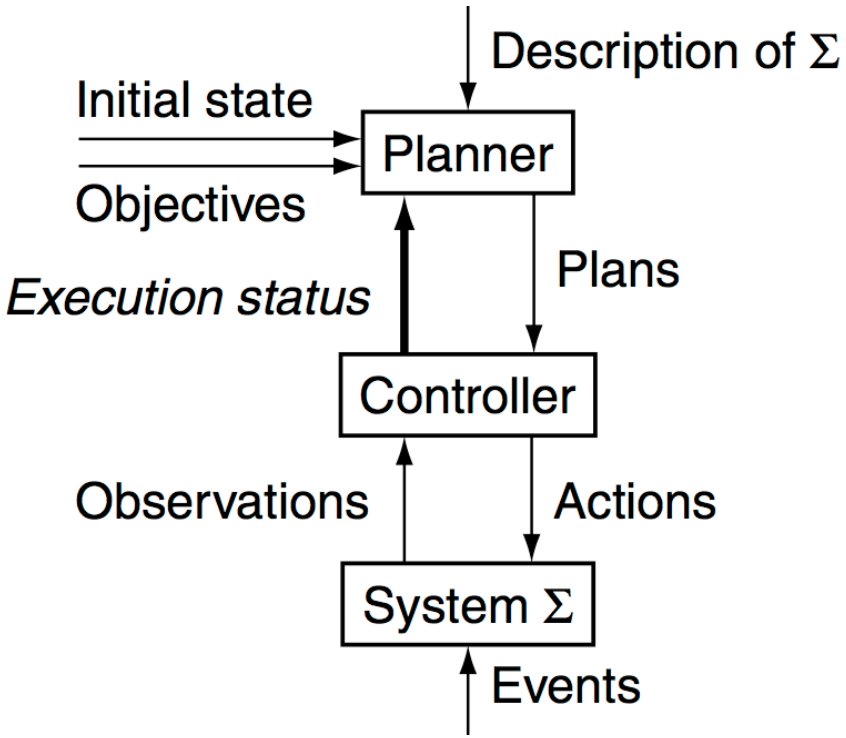
$A = \{\text{actions}\}$

$E = \{\text{events}\}$

$\gamma: S \times (A \cup E) \rightarrow 2^S$

Restrictive Assumptions

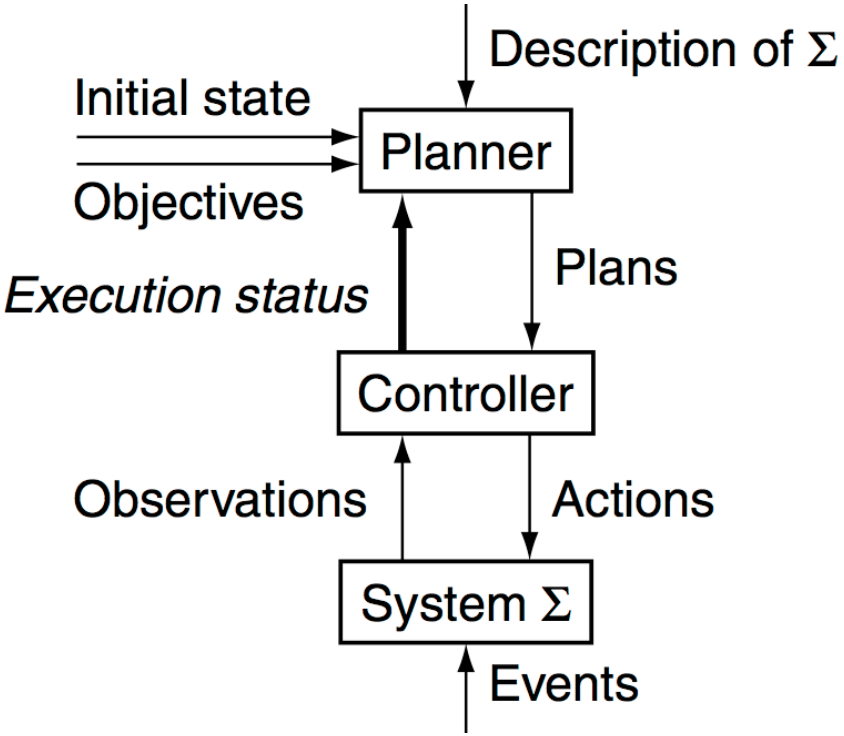
- A2 (deterministic Σ):
  - for all u in $A \cup E$, $|\gamma(s,u)| = 1$
  - each action or event has only one possible outcome
- A3 (static Σ):
  - E is empty: no changes except those performed by the controller
- A4 (attainment goals):
  - a goal state s_g or a set of goal states S_g



$$\Sigma = (S,A,E,\gamma)$$
$$S = \{\text{states}\}$$
$$A = \{\text{actions}\}$$
$$E = \{\text{events}\}$$
$$\gamma: S \times (A \cup E) \rightarrow 2^S$$

Restrictive Assumptions

- A5 (sequential plans):
  - solution is a linearly ordered sequence of actions $(a_1, a_2, \ldots a_n)$
- A6 (implicit time):
  - no durations, instantaneous state-transitions
- A7 (off-line planning):
  - planner doesn't know the execution status



$$\Sigma = (S, A, E, \gamma)$$
$$S = \{\text{states}\}$$
$$A = \{\text{actions}\}$$
$$E = \{\text{events}\}$$
$$\gamma: S \times (A \cup E) \rightarrow 2^S$$

Classical Planning

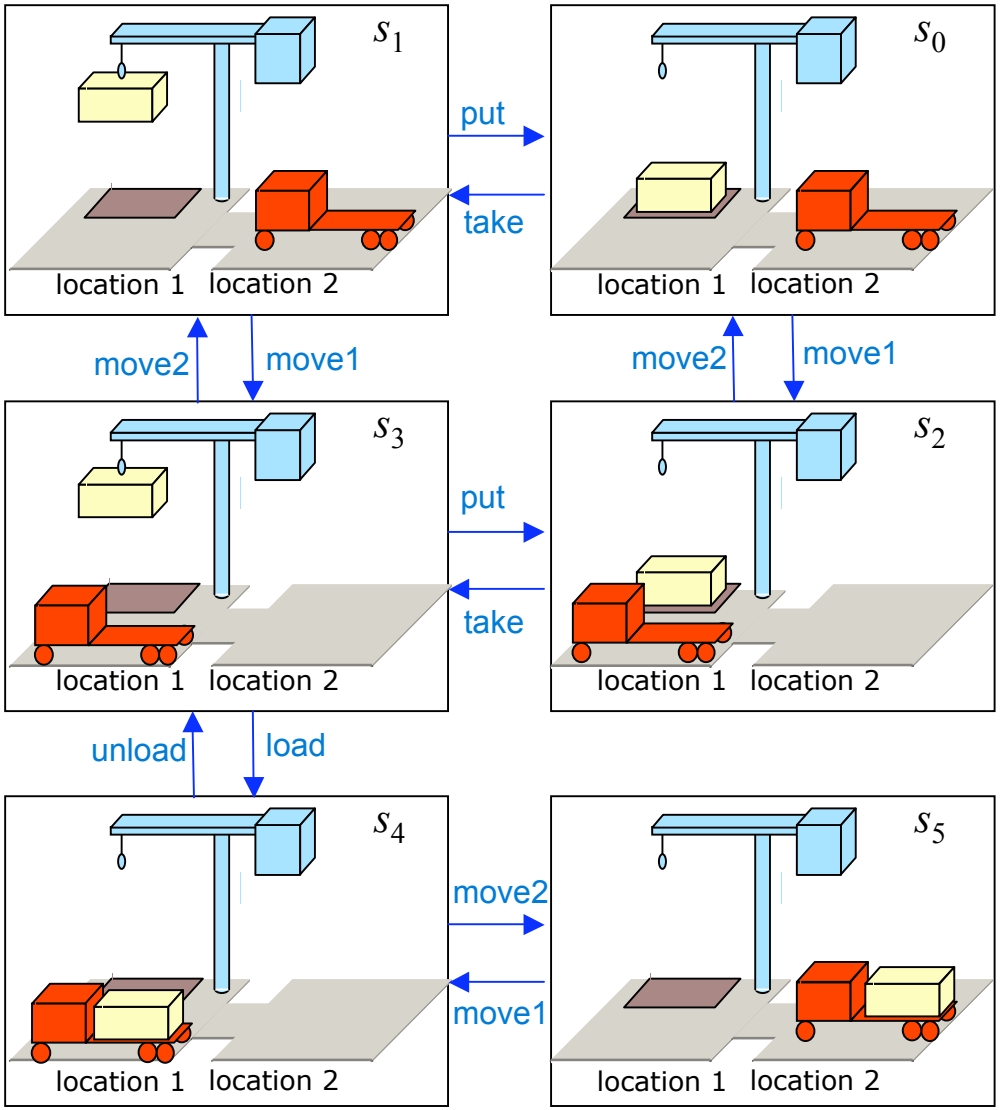
- Classical planning requires all eight restrictive assumptions
  - complete knowledge about a deterministic, static, finite-state system with attainment goals and implicit time
- Reduces to the following problem:
  - Given (Σ, s_0, S_g), find a sequence of actions $(a_1, a_2, \ldots a_n)$ that produces a sequence of state transitions

$$s_1 = \gamma(s_0, a_1),$$
$$s_2 = \gamma(s_1, a_2),$$
$$\ldots,$$
$$s_n = \gamma(s_{n-1}, a_n)$$

such that s_n is in S_g.

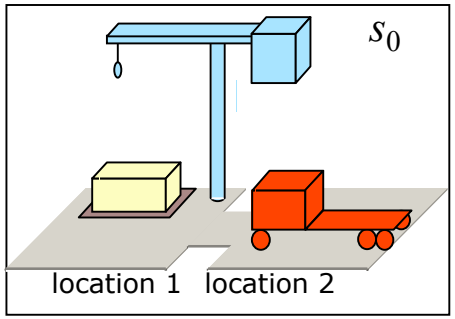
Classical Planning: Example

- Same example as before:
  - System is finite, deterministic, static
  - Complete knowledge
  - Attainment goals
  - Implicit time
  - Offline planning
- Classical planning is just path-searching in a graph
  - states are nodes
  - actions are edges
- Is this trivial?



Classical Planning

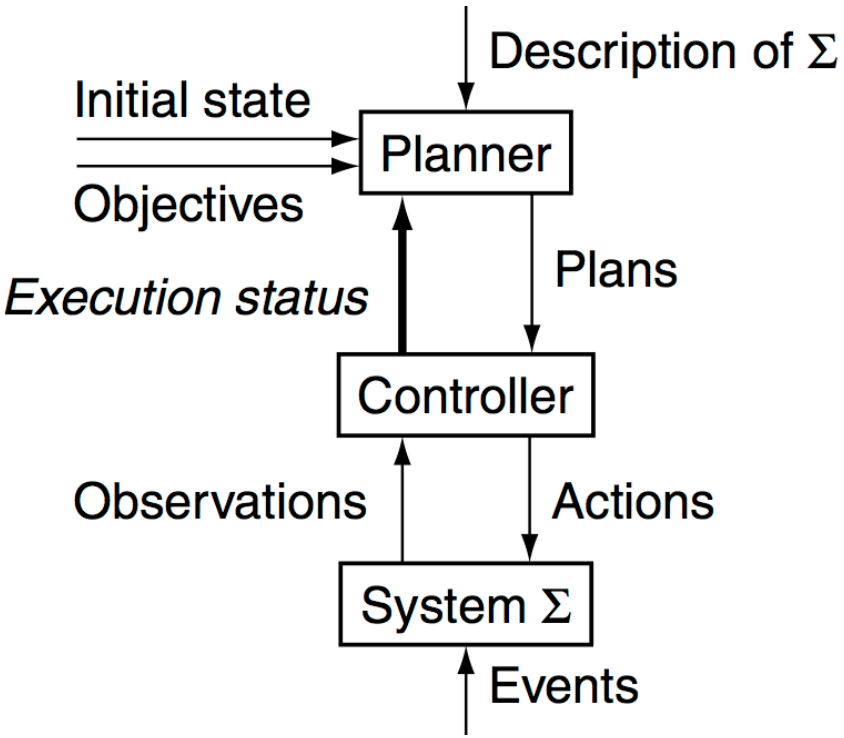
- Very difficult computationally
  - Generalize the earlier example:
    - » five locations, three piles, three robots, 100 containers
  - Then there are 10^{277} states
    - » More than 10^{190} times as many states as the number of particles in the universe!



- The vast majority of AI research has been on classical planning
  - Parts I and II of the book
- Too restricted to fit most problems of practical interest
  - But the ideas can sometimes be useful in those problems

Relax the Assumptions

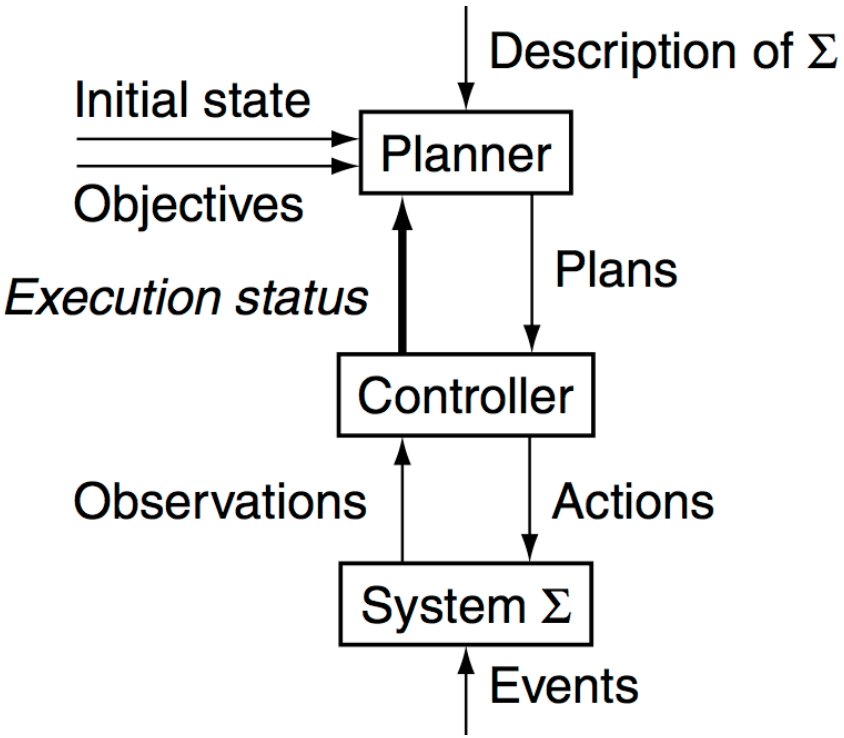
- Relax A0 (finite Σ):
  - Discrete, *e.g.* 1st-order logic:
  - Continuous, *e.g.* numeric variables
  - Sections:
    - » 2.4 (extensions to classical)
    - » 10.5 (control-rule planners)
    - » 11.7 (HTN planning)
  - Case study: Chapter 21 (manufacturability analysis)
- Relax A1 (fully observable Σ):
  - If we don't relax any other restrictions, then the only uncertainty is about s_0



$$\Sigma = (S,A,E,\gamma)$$
$$S = \{\text{states}\}$$
$$A = \{\text{actions}\}$$
$$E = \{\text{events}\}$$
$$\gamma: S \times (A \cup E) \rightarrow 2^S$$

Relax the Assumptions

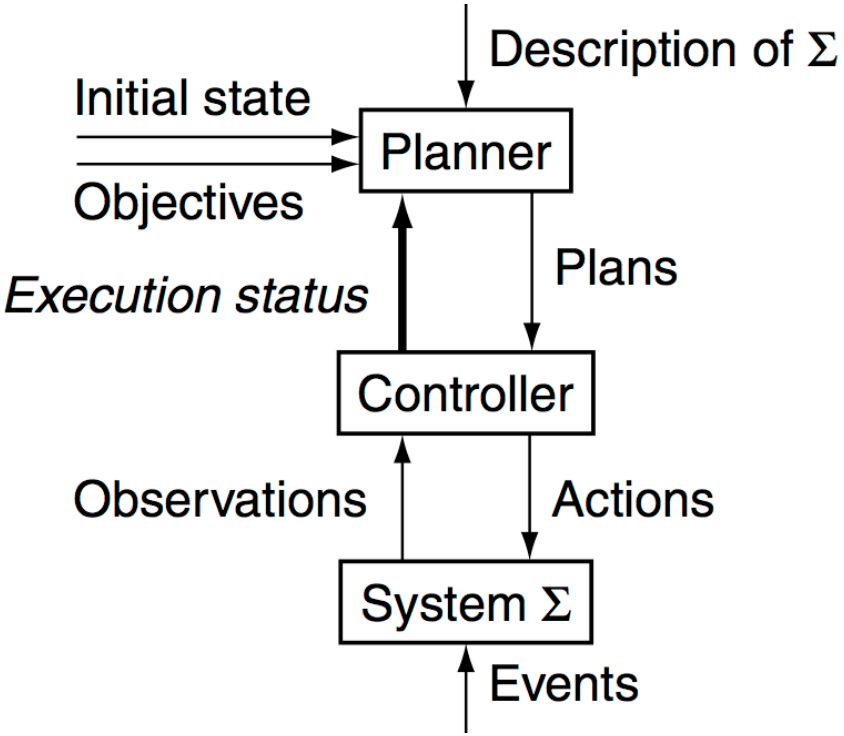
- Relax A2 (deterministic Σ):
  - Actions have more than one possible outcome
  - Seek policy or contingency plan
  - With probabilities:
    - » Discrete Markov Decision Processes (MDPs)
    - » Chapter 11
  - Without probabilities:
    - » Nondeterministic transition systems
    - » Chapters 12, 18



$$\Sigma = (S, A, E, \gamma)$$
$$S = \{\text{states}\}$$
$$A = \{\text{actions}\}$$
$$E = \{\text{events}\}$$
$$\gamma: S \times (A \cup E) \rightarrow 2^S$$

Relax the Assumptions

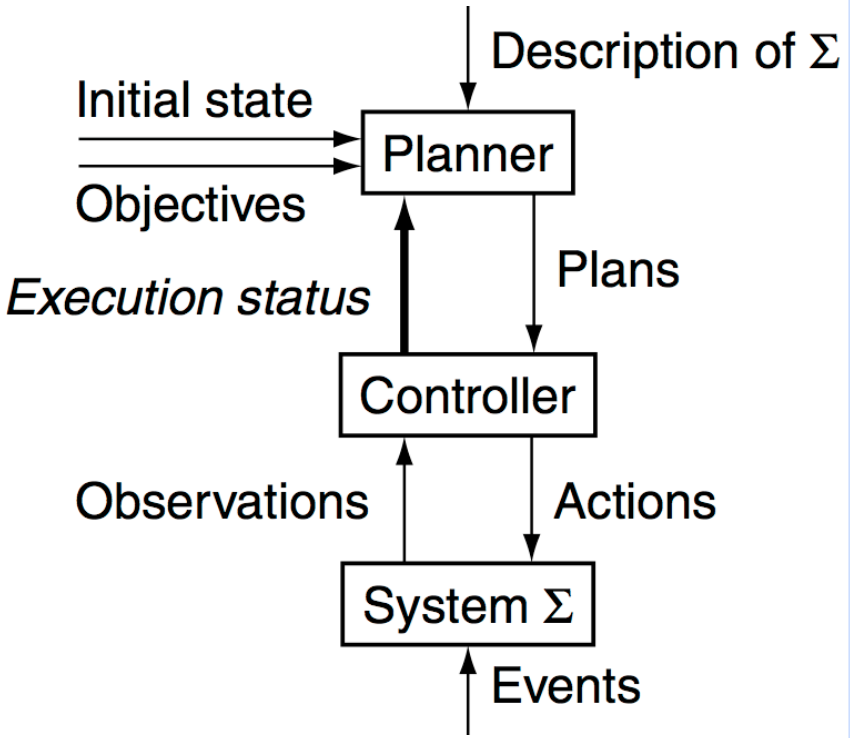
- Relax A1 and A2:
  - Finite POMDPs
    - » Plan over *belief states*
    - » Exponential time & space
    - » Section 16.3
- Relax A0 and A2:
  - Continuous or hybrid MDPs
    - » Control theory (see engineering courses)
- Relax A0, A1, and A2
  - Continuous or hybrid POMDPs
    - » Case study: Chapter 20 (robotics)



$$\Sigma = (S,A,E,\gamma)$$
$$S = \{\text{states}\}$$
$$A = \{\text{actions}\}$$
$$E = \{\text{events}\}$$
$$\gamma: S \times (A \cup E) \rightarrow 2^S$$

Relax the Assumptions

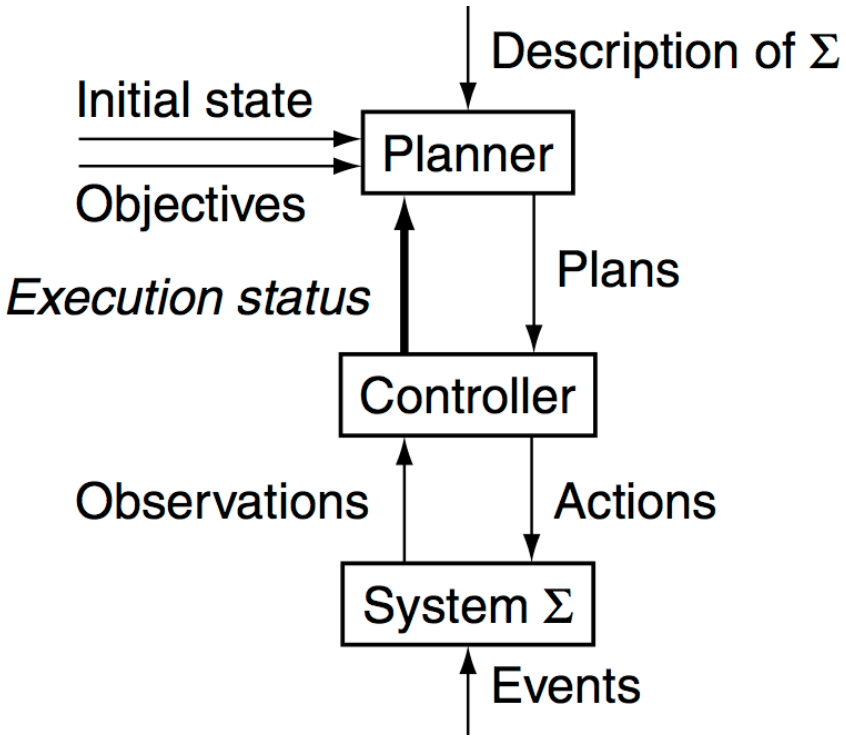
- Relax A3 (static Σ):
  - Other agents or dynamic environment
    - » Finite perfect-info zero-sum games (introductory AI courses)
  - Randomly behaving environment
    - » Decision analysis (business, operations research)
    - » Can sometimes map this into MDPs or POMDPs
  - Case studies: Chapters 19 (space), 22 (emergency evacuation)
- Relax A1 and A3
  - Imperfect-information games
  - Case study: Chapter 23 (bridge)



$$\Sigma = (S,A,E,\gamma)$$
$$S = \{\text{states}\}$$
$$A = \{\text{actions}\}$$
$$E = \{\text{events}\}$$
$$\gamma: S \times (A \cup E) \rightarrow 2^S$$

Relax the Assumptions

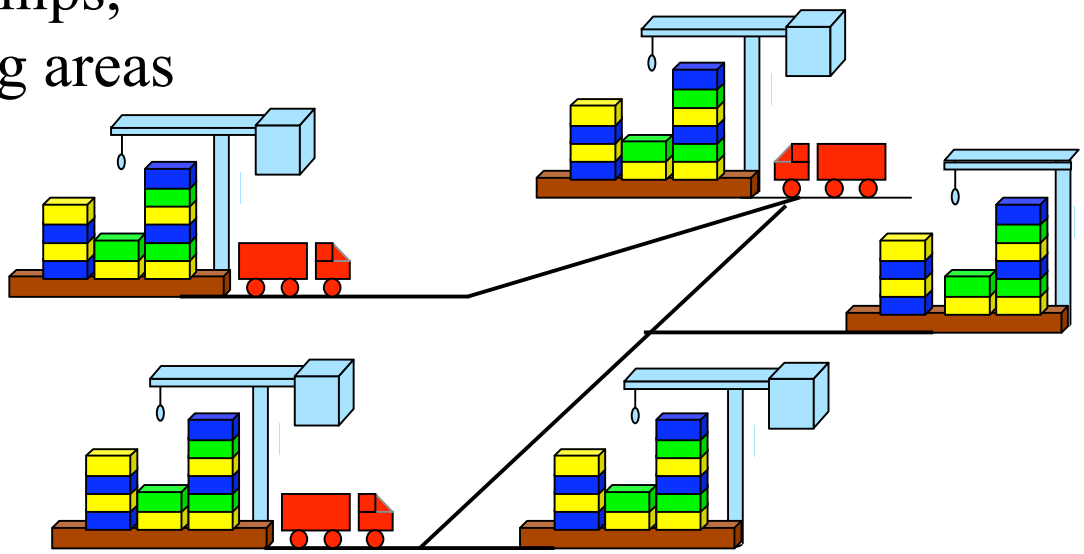
- Relax A5 (sequential plans) and A6 (implicit time):
  - Temporal planning
  - Chapters 13, 14
- Relax A0, A5, A5
  - Planning and resource scheduling
  - Chapter 15
- 247 other combinations
  - I won't discuss them!



$$\Sigma = (S,A,E,\gamma)$$
$$S = \{\text{states}\}$$
$$A = \{\text{actions}\}$$
$$E = \{\text{events}\}$$
$$\gamma: S \times (A \cup E) \rightarrow 2^S$$

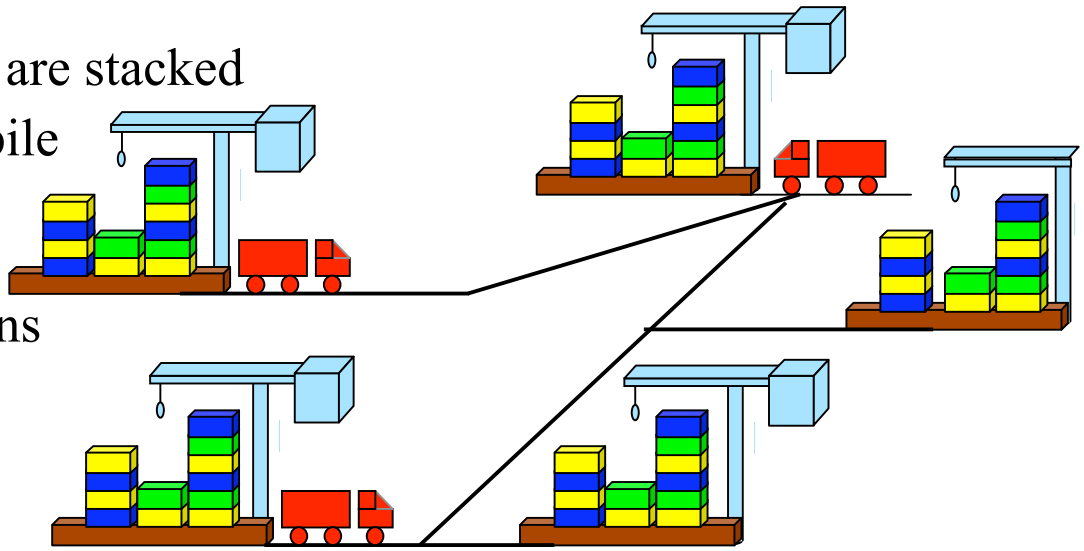
A running example: Dock Worker Robots

- Generalization of the earlier example
  - A harbor with several locations
    - » e.g., docks, docked ships, storage areas, parking areas
  - Containers
    - » going to/from ships
  - Robot carts
    - » can move containers
  - Cranes
    - » can load and unload containers



A running example: Dock Worker Robots

- **Locations**: l1, l2, …
- **Containers**: c1, c2, …
  - can be stacked in piles, loaded onto robots, or held by cranes
- **Piles**: p1, p2, …
  - fixed areas where containers are stacked
  - pallet at the bottom of each pile
- **Robot carts**: r1, r2, …
  - can move to adjacent locations
  - carry at most one container
- **Cranes**: k1, k2, …
  - each belongs to a single location
  - move containers between piles and robots
  - if there is a pile at a location, there must also be a crane there



A running example: Dock Worker Robots

- Fixed relations: same in all states

  adjacent(l,l') attached(p,l) belong(k,l)

- Dynamic relations: differ from one state to another

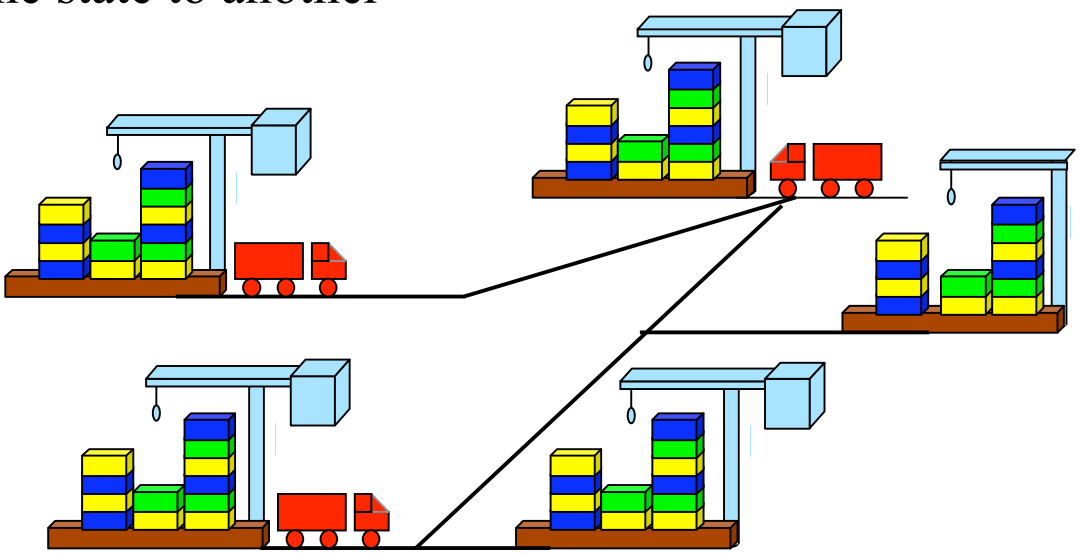
  occupied(l) at(r,l)

  loaded(r,c) unloaded(r)

  holding(k,c) empty(k)

  in(c,p) on(c,c')

  top(c,p) top(pallet,p)



- Actions:

  take(c,k,p) put(c,k,p)

  load(r,c,k) unload(r) move(r,l,l')