Intelligent Agents

Introduction to Planning

Ute Schmid

Cognitive Systems, Applied Computer Science, Bamberg University

last change: 28. Mai 2014

Intuitions on Planning

- *Intelligent Agents:* Natural or artificial systems which act in an intelligent way
- Intelligent action is rational action, that is, the best possible action in a given situation
- Planning is the reasoning side of acting
- Abstract, explicit deliberation process that chooses and organizes actions by anticipating their expected outcomes
- Some actions require planning, many do not
  - we act more frequently than we explicitly plan
  - performing well-trained behaviors for which we have pre-stored plans
  - acting and adapting in flexible settings

Intuitions on Planning

- Planning is a complicated, time consuming, and costly process
- Planning is needed when
  - new situations, unfamiliar actions are involved
  - complex tasks, complex objectives are addressed
  - actions are constrained by high risks, high costs, joint activities, need for synchronization
- Typically we seek feasible, good plans, not optimal plans
  (cf. Simon's "bounded rationality")

Motivations for Automated Planning

- Practical
  - Designing information processing tools that give access to affordable and efficient planning resources
  - Some professionals face complex changing tasks that involve demanding safety and/or efficiency requirements
  - Example:
    disaster rescue operations large number of actors, deployment of communication and transportation infrastructure, time constrained, demands for immediate decisions relies on careful planning and assessment of several alternate plans
  - Example:
    organizers of social meetings

Motivations for Automated Planning cont.

- Theoretical
  - Planning is an important component of rational behavior
  - Purpose of AI: grasping computational aspects of intelligence
  - planning, as the reasoning side of acting, is a key element
  - Studying planning as abstract process (complexity, efficiency of algorithms, ...)
  - Planning as integrated component of deliberative behavior

Motivations for Automated Planning cont.

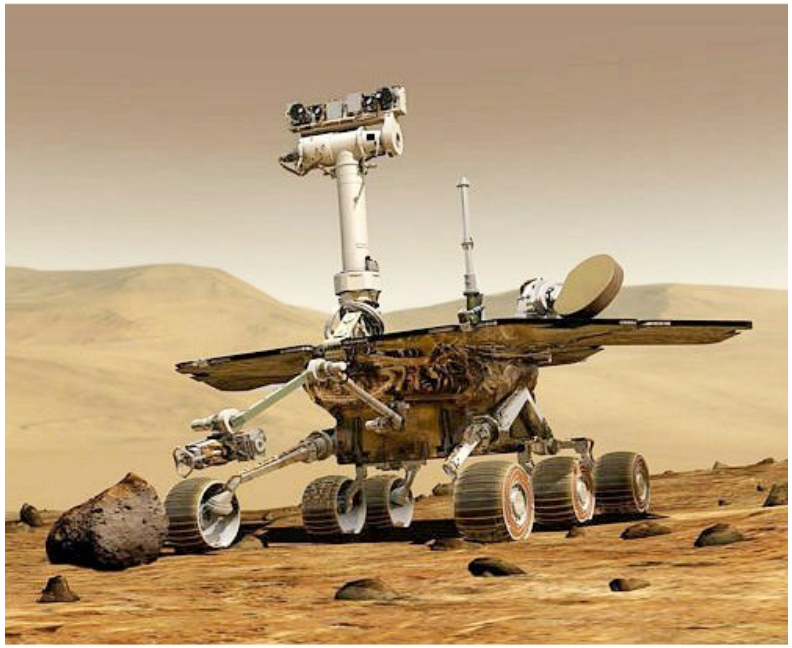
- **Hot topic:** study and design of autonomous intelligent machines
  - satellites, space-crafts, robots cannot always be tele-operated
  - interaction with non-expert humans on task level rather than control signals
  - machines that can sense and act as well as reason on their actions

Automated Planning

- *Plan*: Sequence of actions to achieve a goal
- *Planning*: Computation of such a sequence
- Examples of successful applications
  - Space Exploration
  - Manufacturing
  - Games

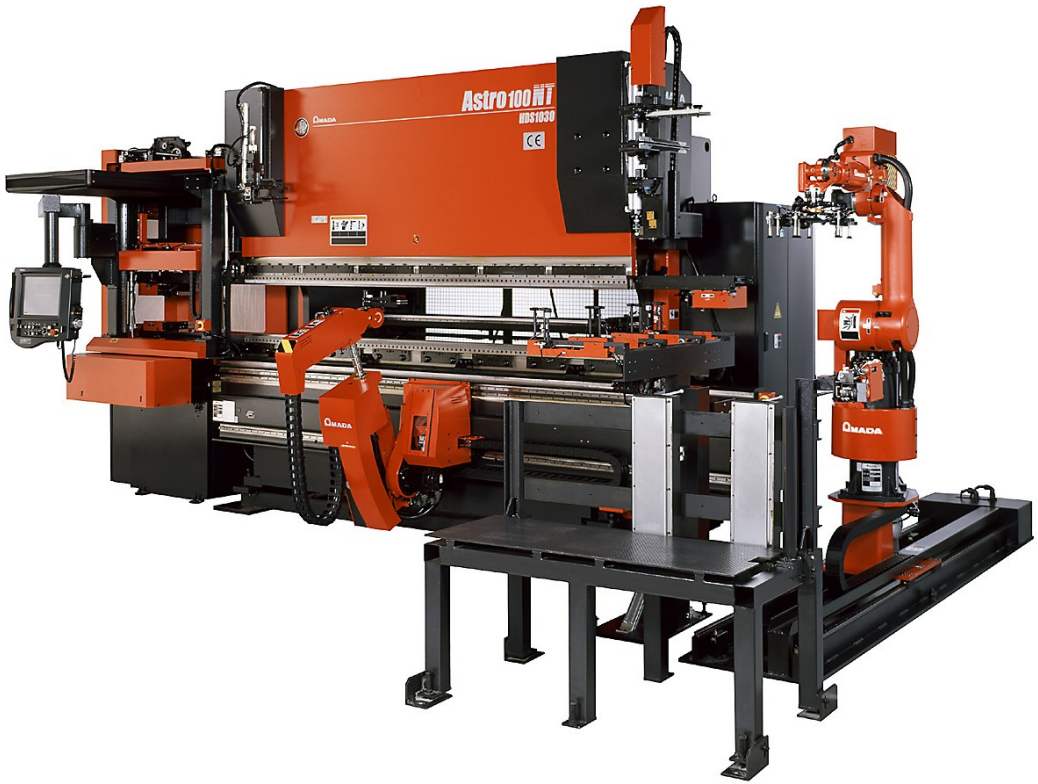
Space Exploration

- Autonomous planning, schedule, control
  - NASA: JPL and AMES
- Remote Agent Experient (RAX)
  - Deep Space 1
- Mars Exploration Rover (MER)



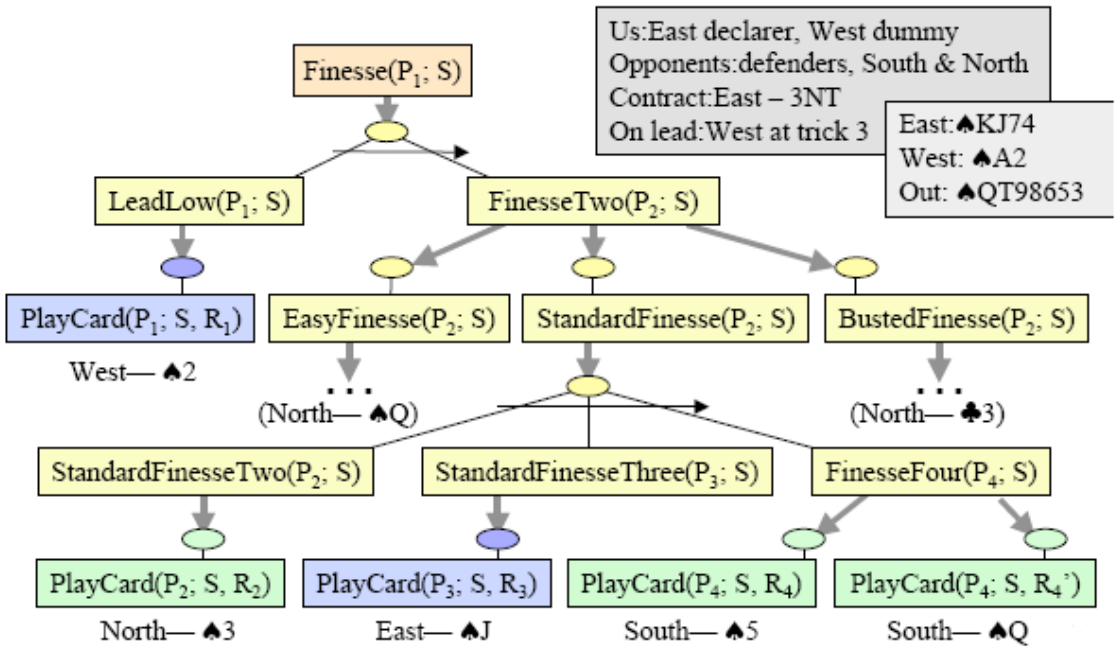
Manufacturing

- Sheet-metal bending machines - Amada Corporation
  - Software to plan the sequence of bends
    [Gupta and Bourne, *J. Manufacturing Sci. and Engr.*, 1999]



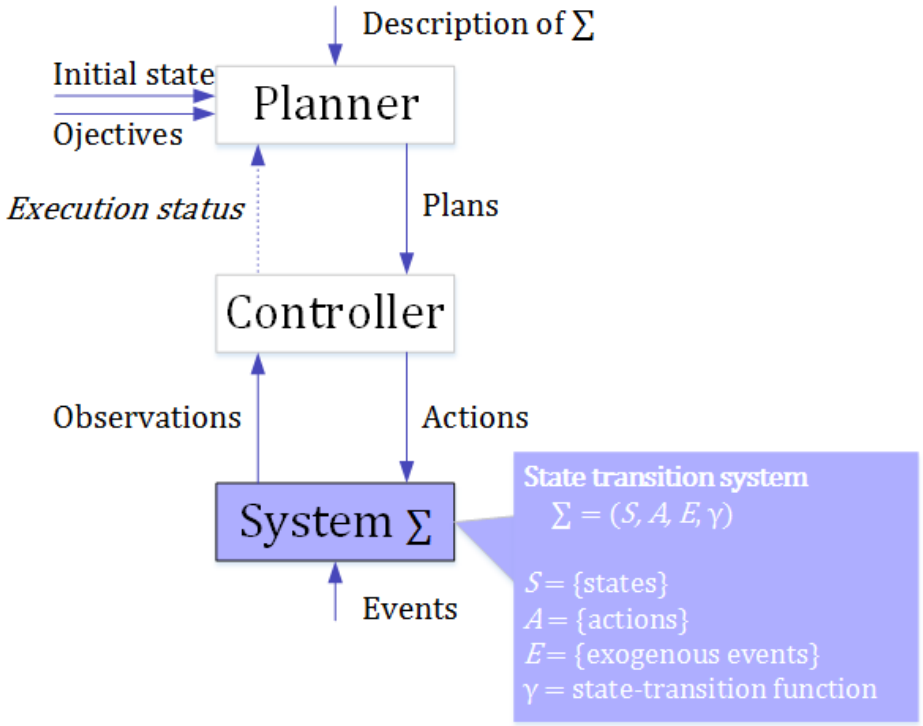
AI in Games

- *Bridge Baron* - Great Game Products
  - 1997 world champion of computer bridge
    [Smith, Nau, and Throop, *AI Magazine,* 1998]
  - 2004: 2nd place



Conceptual Model of Planning

Conceptual Model
1. Environment

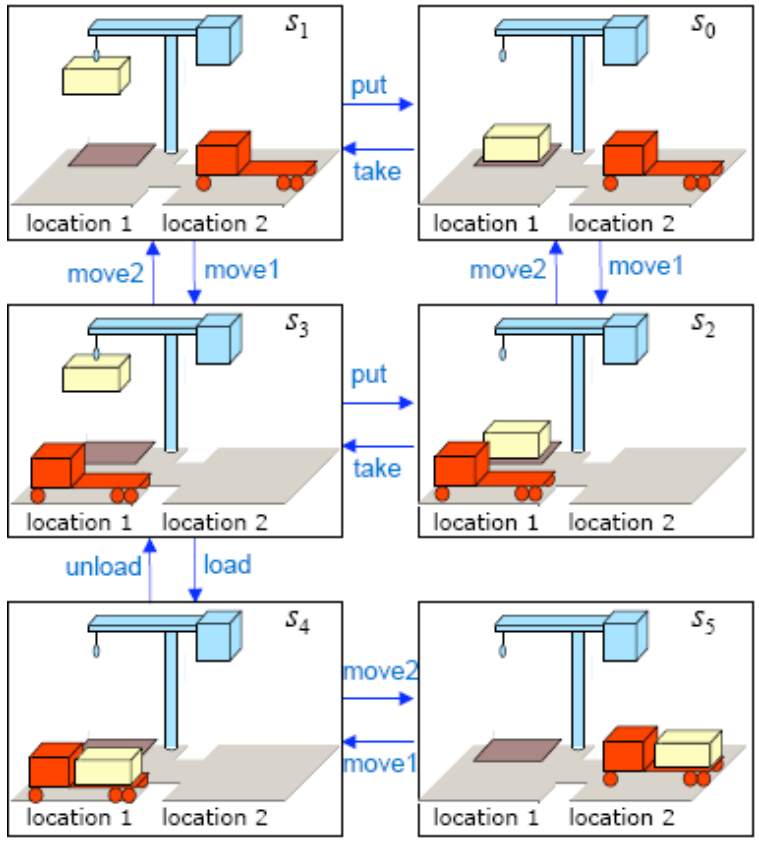


Conceptual Model of Planning

State Transition System

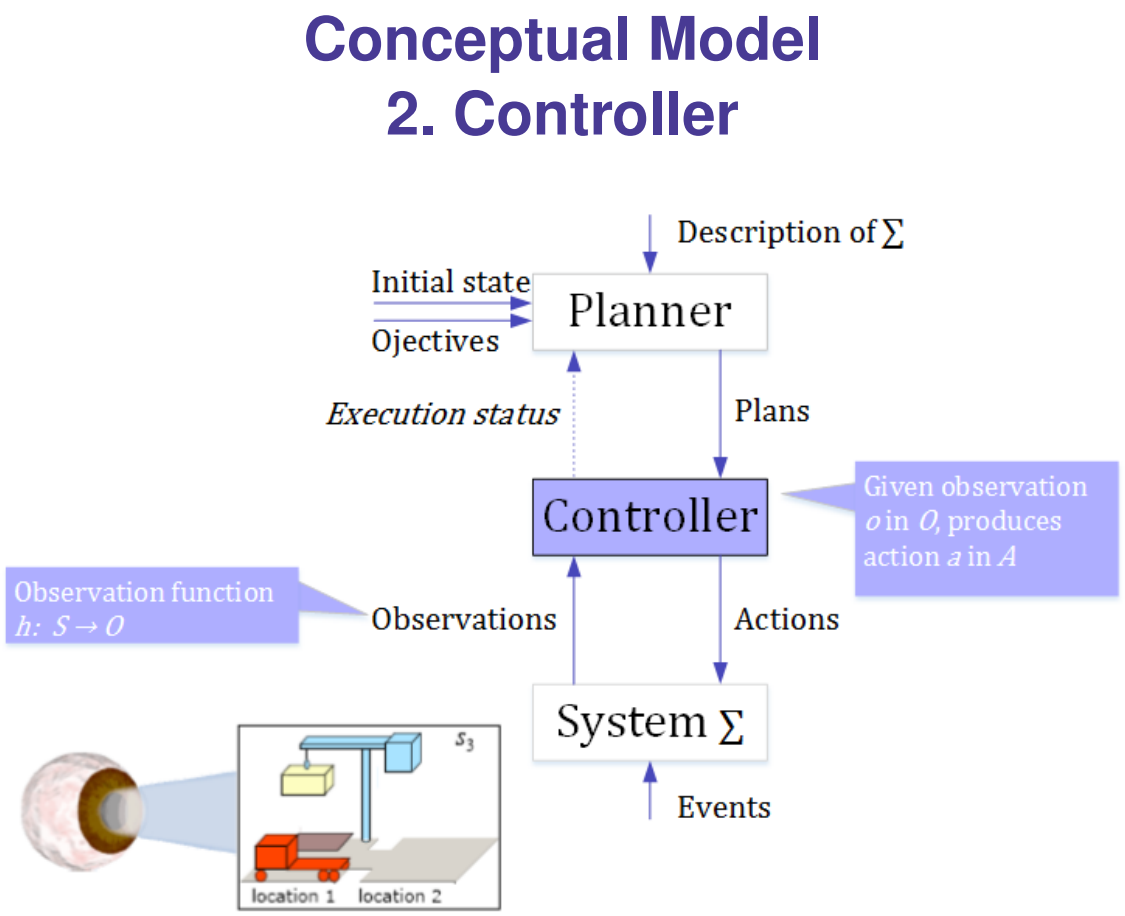
$\Sigma = (S, A, E, \gamma)$

- $S = \{states\}$
- $A = \{actions\}$
- $E = \{exogenous\ events\}$
- State-transition function
  $\gamma : S \times (A \cup E) \rightarrow 2^S$
  - $S = \{s_0, ..., s_5\}$
  - $A = \{move1, move2, put, take, load, unload\}$
  - $E = \{\}$
  - γ : see the arrows



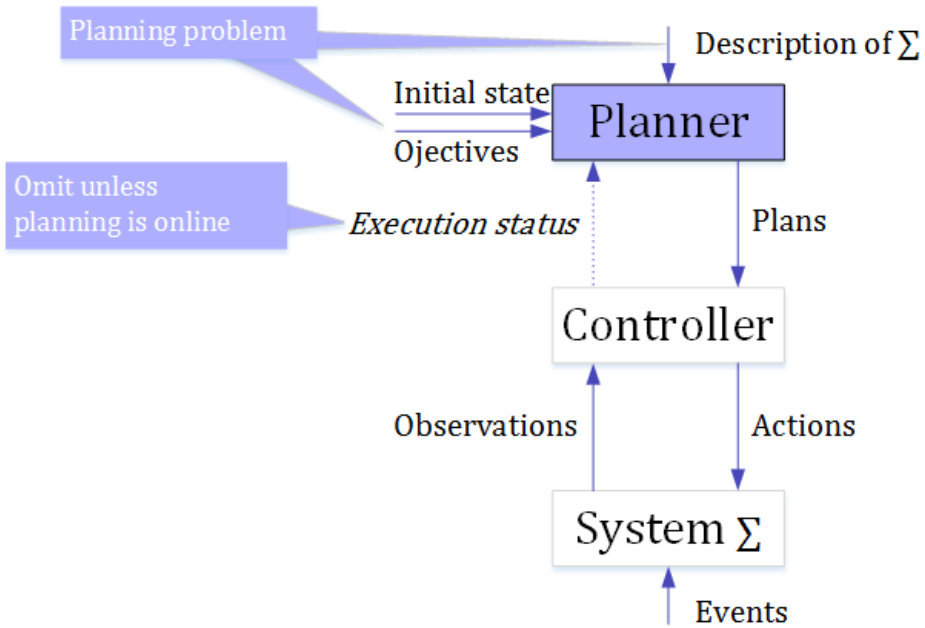
The Dock Worker Robots (DWR) domain

Conceptual Model of Planning



Conceptual Model of Planning

Conceptual Model
3. Planner's Input



Conceptual Model of Planning

Planning Problem

Description of Σ

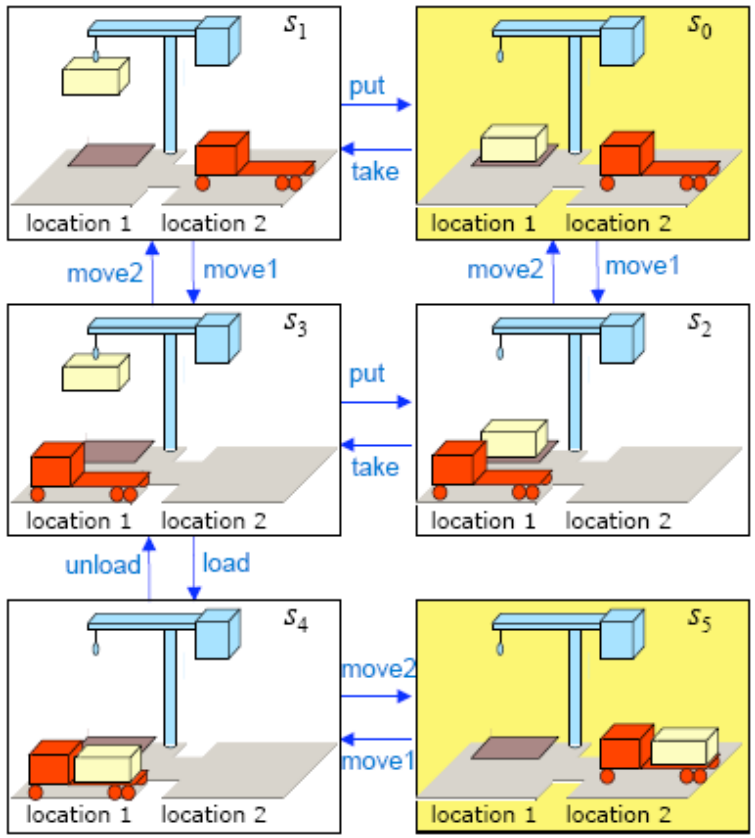
Initial state or set of states

Initial state $= s_0$

Objective

Goal state, set of goal states, set of tasks, "trajectory" of states, objective function, ...

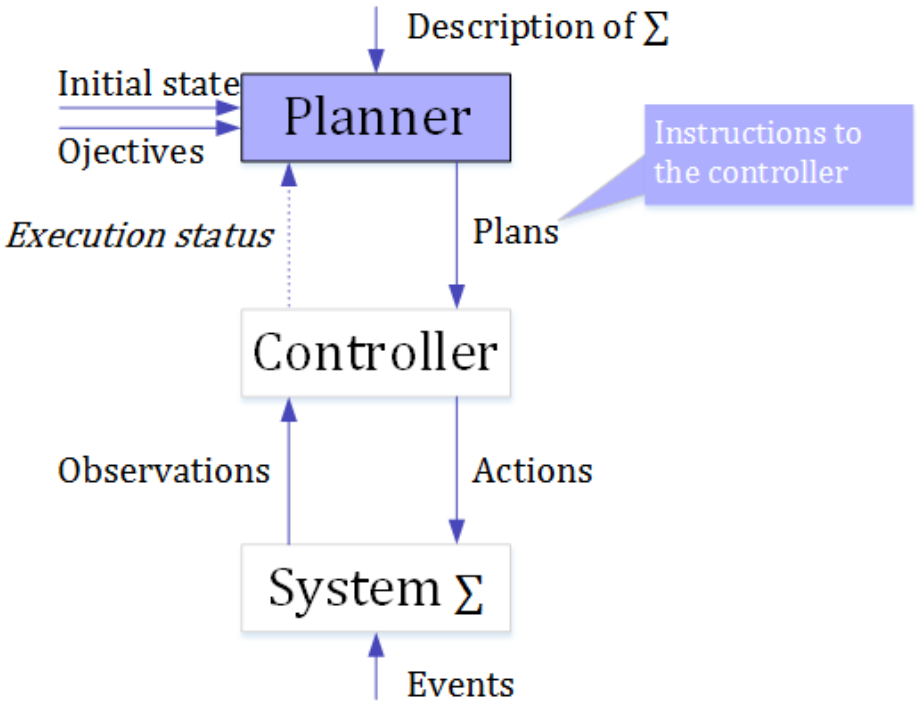
Goal state $= s_5$



The Dock Worker Robots (DWR) domain

Conceptual Model of Planning

Conceptual Model
4. Planner's Output



Conceptual Model of Planning

Plans

Classical plan:

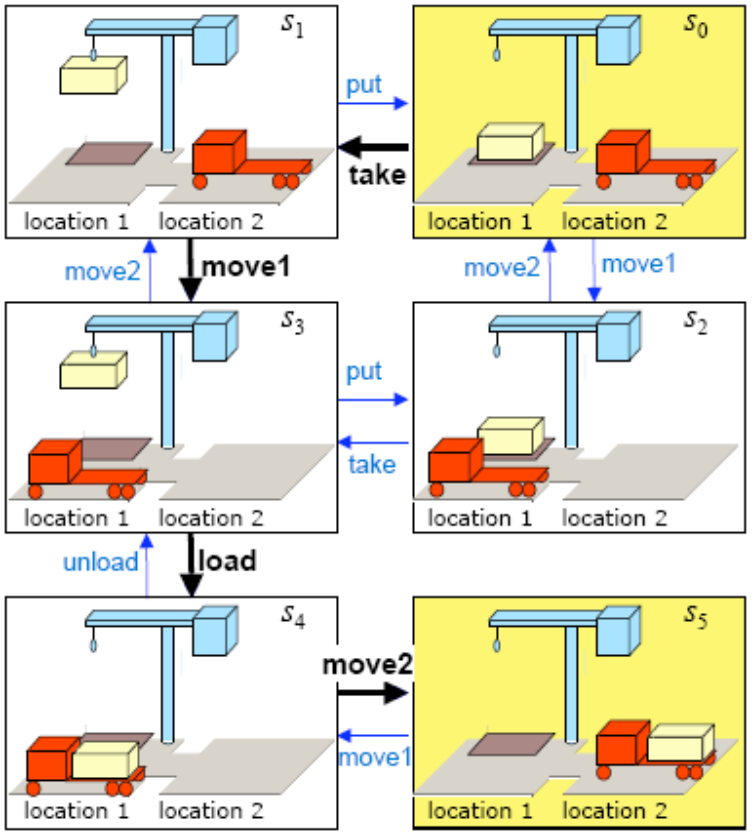
a sequence of actions

$\langle take, move1, load, move2 \rangle$

Policy:

partial function from S into A

$\{(s_0, take),$
$(s_1, move1),$
$(s_3, load),$
$(s_4, moves2)\}$



The Dock Worker Robots (DWR) domain

Representation of States

- Usually, representation of problem states are based on some restrictive variant of first order logic (FOL)
- {occupied(L1), loaded(R1,C1), on(C2,C3), in(C2,P1), ...}
- A state representation is a set of predicate symbols which represent relations which hold in the situation.
- Typically, arguments of predicate symbols are restricted to constants and variables.
- A single predicate symbol is called **atom**.
- Representations will be introduced in a later lecture.
- FOL will also be introduced later in some detail.

Planning, Problem Solving, Scheduling

- Problem solving:
  using domain-specific heuristics to search for a (optimal) sequence of actions
- Scheduling:
  decide when and how to perform a given set of actions obeying time constraints, resource constraints, objective functions
- Planning:
  decide what actions to use in what sequence to achieve some set of objectives biggest algorithmical challenge: often worse than NP-complete, worst case is undecidable (see lecture about complexity of classical planning)

Domain-Independent Planning

- In principle, a domain-independent planner works in any planning domain
- Uses no domain-specific knowledge except the definition of the basic actions
- In practice, it is not feasible to develop a planner that works in every possible domain
- Make simplifying assumptions to restrict the set of domains: mostly classical planning
- Domain-specific planners can be very successful in specific domains but one needs to write an entire program (lots of work)

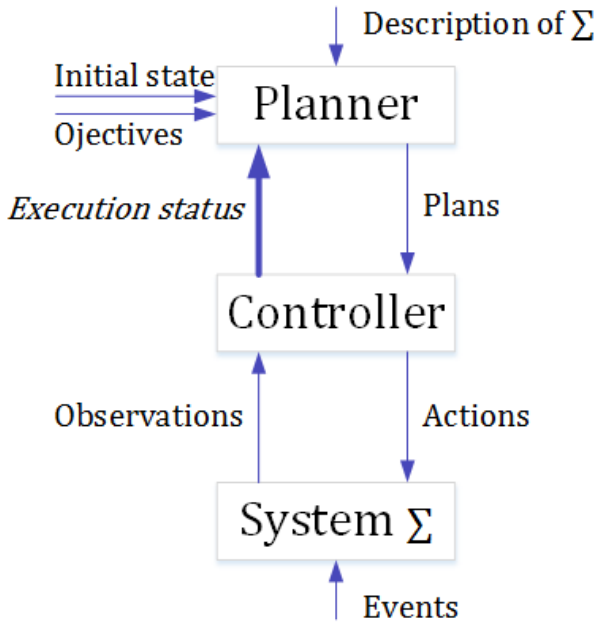
Classical Planning

- Restrictive assumptions (see next slide):
  finite set of states and actions; fully observable states;
  deterministic outcome of actions, ...
- Reduces to the problem of path searching in a graph with nodes as states and edges as actions (which is still hard enough):
  - Generalize the earlier example to 5 locations, 3 robot carts, 100 containers, 3 piles: 10^{277} states
  - Number of particles in the universe is about 10^{87}
- Most research is on classical planning with many different algorithms
- Planning Competition (AIPS 1998, AIPS 2000, IPC 2002, ...) shows the progress every two years

Restrictions

Restrictive Assumptions

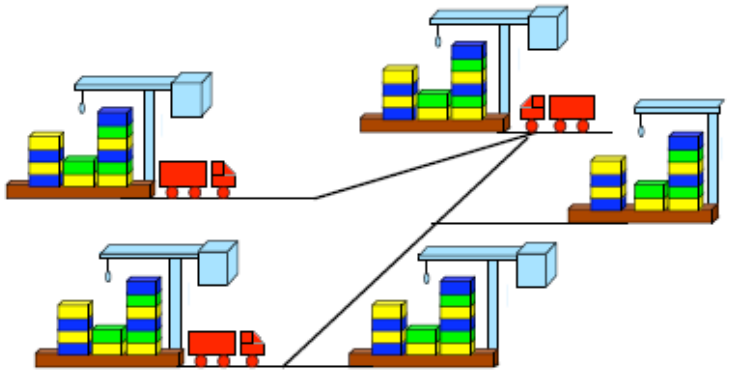
- **A0: Finite system:**
  - ↪ finitely many states, actions, events
- **A1: Fully observable:**
  - ↪ the controller always Σ' s current state
- **A2: Deterministic:**
  - ↪ each action has only one outcome
- **A3: Static** (no exogenous events):
  - ↪ no changes but the controller's actions
- **A4: Attainment goals:**
  - ↪ a set of goal states S_g
- **A5: Sequential plans:**
  - ↪ a plan is a linearly ordered sequence of actions $(a_1, a_2, \ldots, a_n)$
- **A6: Implicit time:**
  - ↪ no time durations; linear sequence of instantaneous states
- **A7: Off-line planning:**
  - ↪ planner doesn't know the execution status



Running Example

A running example: Dock Worker Robots

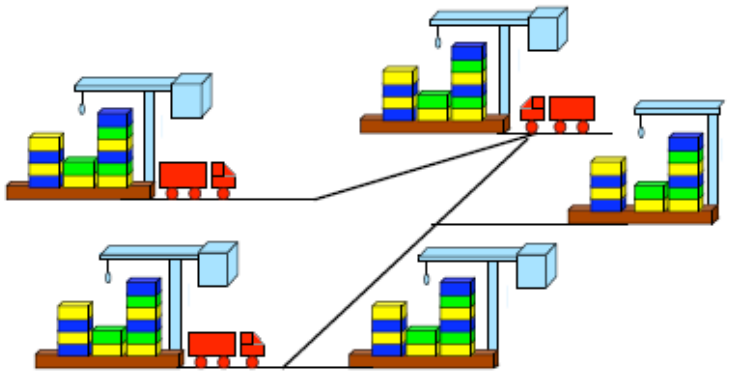
- Generalization of the earlier example
  - A harbor with several locations
    - ↪ e. g., docks, docked ships, storage areas, parking areas
  - Containers
    - ↪ going to/from ships
  - Robot carts
    - ↪ can move containers
  - Cranes
    - ↪ can load and unload containers



Running Example

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



Running Example

A running example: Dock Worker Robots

- Fixed relations:

  same in all states

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

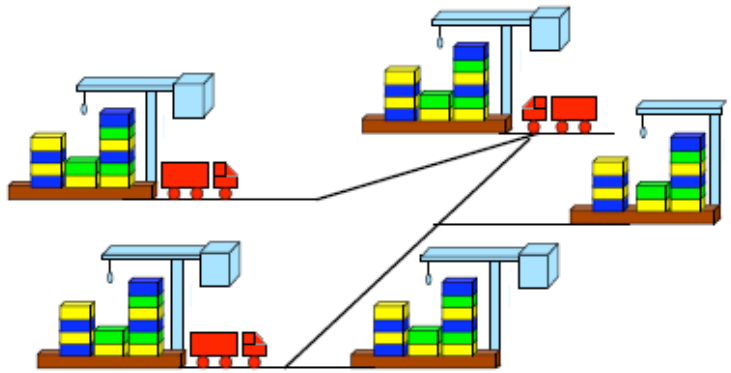
- Dynamic relations:

  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*) unloaded(*r*) move(*r,l,l'*)



Closed-World Assumption (CWA)

- An atom that is not explicitly given in a state does not hold in the state
- That is: Assumption of the value *false* for every atom which is not explicitly stated
- Classical, set-theoretical and state-variable representation all rely on the CWA
- CWA is a restriction of the logic calculus:
  no true negation but *negation by failure*
  (If a proposition cannot be proven to be true, it is assumed to be false.)

CWA cont

- This restriction makes state-based planning more efficient than deductive planning in full FOL where the *frame problem* exists
- Frame problem:
  not only the propositions which change by an action must be specified but also all propositions which are not affected by an action
  (e.g. If I put block *x* from block *y* on the table, *on(y,z) still holds)*

Extended Representation

- Typed variables and relations
- Conditional Operators
- Quantified Expressions
- Equality Constraints
- Disjunctive Preconditions
- Function Symbols
- Axiomatic Inference
- Attached Procedures

PDDL

- Problem Domain Definition Language as common language for most modern planners (see PDDL Specification)
- Example: Equality constraints and conditioned effects

```
(define (domain blocks-world-domain)
  (:requirements :strips :equality :conditional-effects)
  (:constants Table)
  (:predicates (on ?x ?y)
               (clear ?x)
               (block ?b)
               )
  ;; Define step for placing one block on another.
  (:action puton
            :parameters (?X ?Y ?Z)
            :precondition (and (on ?X ?Z) (clear ?X) (clear ?Y)
                               (not (= ?Y ?Z)) (not (= ?X ?Z))
                               (not (= ?X ?Y)) (not (= ?X Table)))
            :effect
            (and (on ?X ?Y) (not (on ?X ?Z))
                 (when (not (= ?Z Table)) (clear ?Z))
                 (when (not (= ?Y Table)) (not (clear ?Y))))))
```

Preview: PDDL

- The domain blocksworld is modeled with the *puton* operator
  “Put block ?x from entity ?z (a block or the table) on entity ?y (a block or the table)
- Requirements specify which language features a planner dealing with this domain must support
- The operator is specified with application conditions (precondition) and effects

Preview: PDDL

- Precondition:
  If block ?x is on ?z and ?x and ?y are clear (and if no two objects are identical and ?x is of type block) then the operator can be applied
- Effect:
  Putting ?x from ?z on ?y results in a state where ?x is on ?y (addition of atoms which hold after application) and no longer on ?z (deletion of atoms which do no longer hold after application)
- A conditioned effect (when) allows to write operators more compactly (without these conditions, there would be three different operators: putBlockonBlock, putBlockFromTableToBlock, putBlockFromBlockonTable)

Summary

- Planning is the reasoning side of action
- There are many application domain, e.g, in the domain of autonomous robotics, manufacturing, and games
- The conceptual model of planning defines an environment as state-transition system
- Many approaches concern domain-independent planning (contrast to domain-dependent approaches where specific domain knowledge is included)
- For classical planning there are restrictive assumptions
- Dock Worker Robots is introduced as running example
- The closed-world assumption means that if an atom is not explicitly given in a state description, it is assumed to not hold in the state
- There are extended representations
- An example for a blocksworld-domain represented in PDDL was introduced.