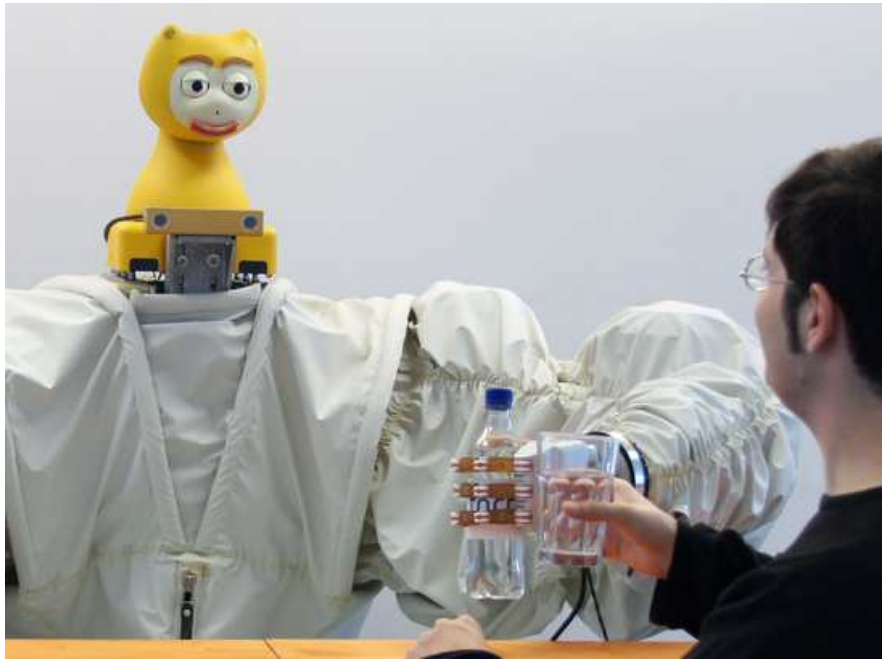
Vorlesung

Grundlagen der Künstlichen Intelligenz

Reinhard Lafrenz / Prof. A. Knoll

Robotics and Embedded Systems
Department of Informatics – I6
Technische Universität München

www6.in.tum.de
lafrenz@in.tum.de
089-289-18136
Room 03.07.055



Wintersemester 2012/13 21.12.2012

Chapter 10 cont‘d (3rd ed.) + 11

Classical Planning, Planning in the Real World

From the last session

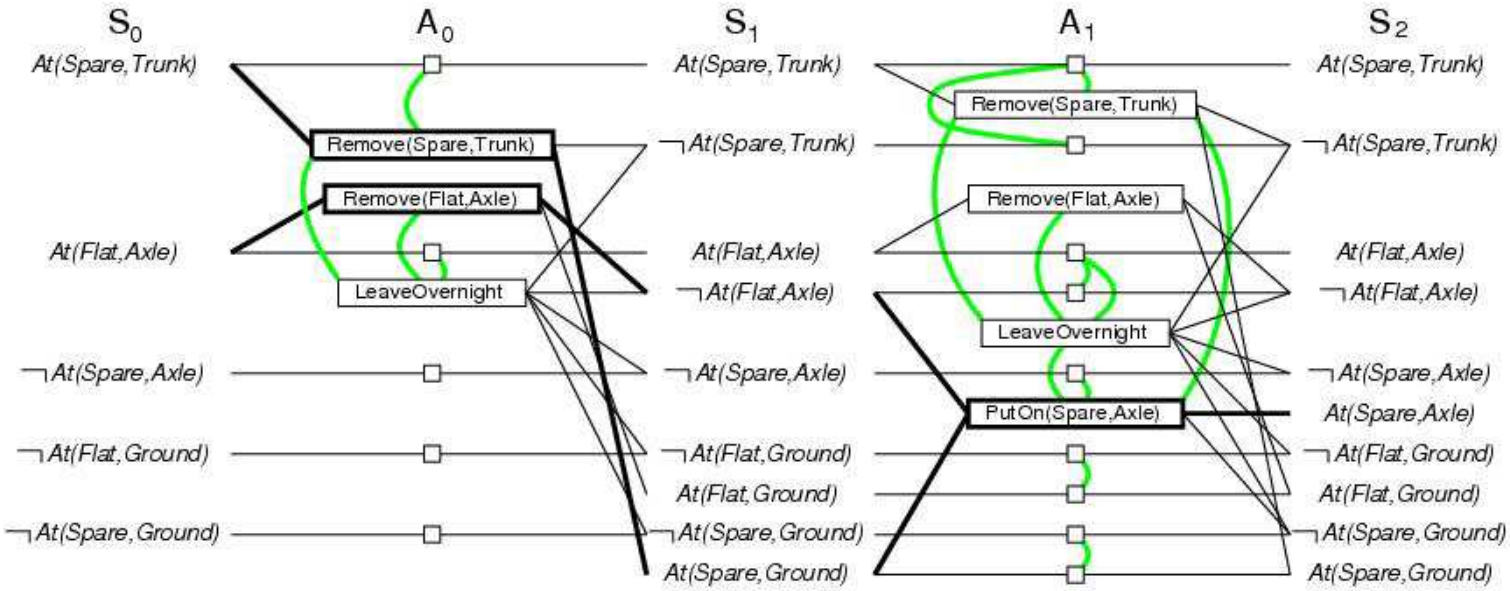
- General description of plans and planning algorithms
- Planning as search for goal states
- GraphPlan algorithm

The GRAPHPLAN Algorithm

- How to extract a solution directly from the PG

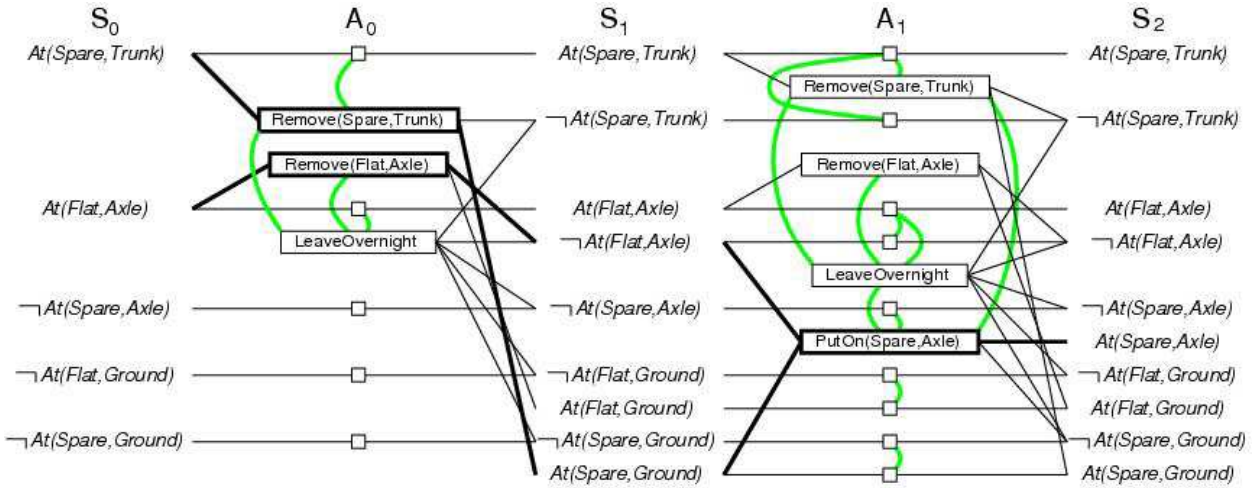
function GRAPHPLAN(*problem*) **return** *solution* or failure
  graph ← INITIAL-PLANNING-GRAPH(*problem*)
  goals ← GOALS[*problem*]
  loop do
    if *goals* all non-mutex in last level of graph **then do**
      solution ← EXTRACT-SOLUTION(*graph, goals,* LENGTH*(graph)*)
      if *solution* ≠ failure **then return** *solution*
      else if NO-SOLUTION-POSSIBLE(*graph*) **then return** failure
    graph ← EXPAND-GRAPH(*graph, problem*)

GRAPHPLAN example



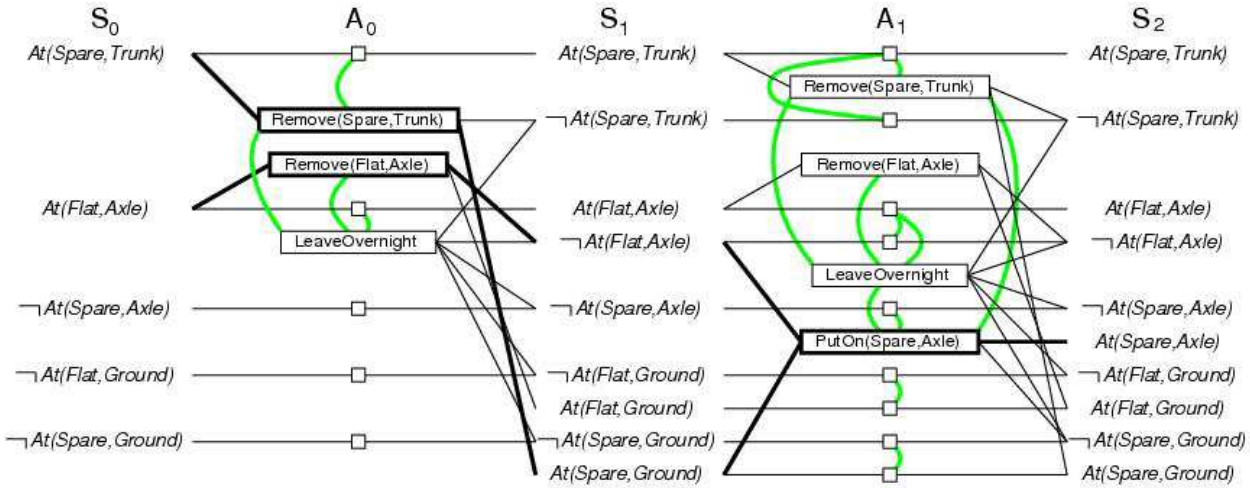
- Initially the plan consist of 5 literals from the initial state and the literals resulting from the closed-world-assumption (CWA) (S0).
- Add actions whose preconditions are satisfied by EXPAND-GRAPH (A0)
- Also add persistence actions and mutex relations.
- Add the effects at level S1
- Repeat until goal is in level Si

GRAPHPLAN example



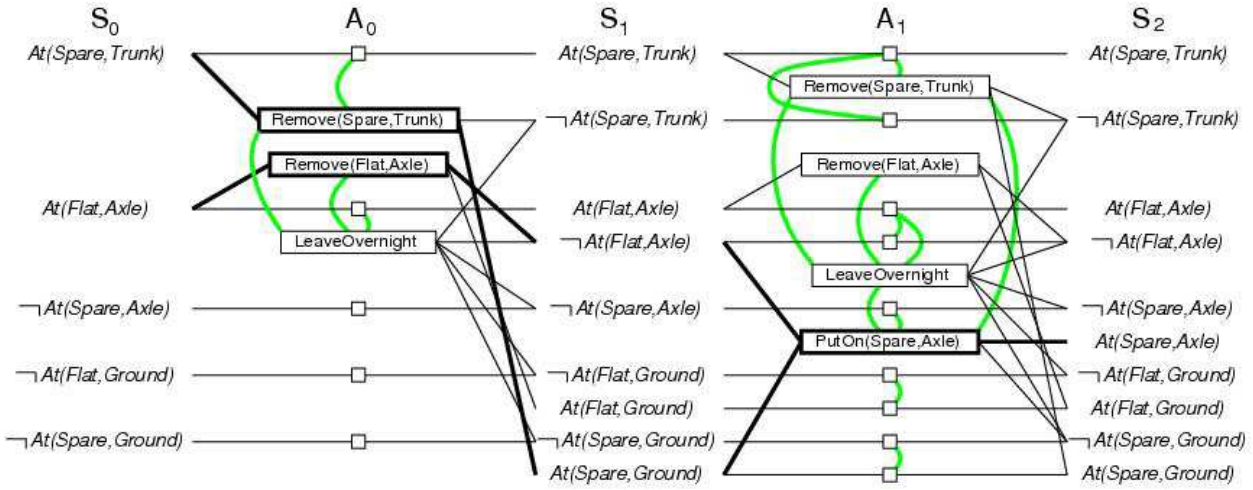
- EXPAND-GRAPH also looks for mutex relations
  - Inconsistent effects
    - E.g. Remove(Spare, Trunk) and LeaveOverNight
  - Interference
    - E.g. Remove(Flat, Axle) and LeaveOverNight
  - Competing needs
    - E.g. PutOn(Spare,Axle) and Remove(Flat, Axle)
  - Inconsistent support
    - E.g. in S2, At(Spare,Axle) and At(Flat,Axle)

GRAPHPLAN example



- In S2, the goal literal exists and is not mutex with any other
  - Solution might exist and EXTRACT-SOLUTION will try to find it
- EXTRACT-SOLUTION can use Boolean CSP to solve the problem or a search process:
  - Initial state = last level of PG and goal goals of planning problem
  - Actions = select any set of non-conflicting actions that cover the goals in the state
  - Goal = reach level S0 such that all goals are satisfied
  - Cost = 1 for each action.

GRAPHPLAN example



- Termination? YES
- PG are monotonically increasing or decreasing:
  - Literals increase monotonically
  - Actions increase monotonically
  - Mutexes decrease monotonically
- Because of these properties and because there is a finite number of actions and literals, every PG will eventually level off !

Planning with propositional logic

- Planning can be done by proving theorem in situation calculus.
- Here: test the *satisfiability* of a logical sentence:

$$initial\ state \wedge all\ possible\ action\ descriptions \wedge goal$$

- Sentence contains propositions for every action occurrence.
  - A model will assign true to the actions that are part of the correct plan and false to the others
  - An assignment that corresponds to an incorrect plan will not be a model because of inconsistency with the assertion that the goal is true.
  - If the planning is unsolvable the sentence will be unsatisfiable.

SATPLAN algorithm

function SATPLAN(*problem,* T_{max}) **return** *solution* or failure

inputs: *problem*, a planning problem

T_{max}, an upper limit to the plan length

for *T= 0* **to** T_{max} **do**

cnf, mapping ← TRANSLATE-TO_SAT(*problem, T*)

assignment ← SAT-SOLVER(*cnf*)

if *assignment* is not null **then**

return EXTRACT-SOLUTION(*assignment, mapping*)

return failure

cnf, mapping ← TRANSLATE-TO_SAT(*problem, T*)

- Distinct propositions for assertions about each time step.
  - Superscripts denote the time step

    $At(P1,SFO)^0 \wedge At(P2,JFK)^0$
  - No CWA thus specify which propositions are not true

    $\neg At(P1,SFO)^0 \wedge \neg At(P2,JFK)^0$
  - Unknown propositions are left unspecified.

- The goal is associated with a particular time-step
  - But which one?

cnf, mapping ← TRANSLATE-TO_SAT(*problem, T*)

- How to determine the time step where the goal will be reached?
  - Start at *T=0*
    - Assert $At(P1,SFO)^0 \wedge At(P2,JFK)^0$
  - Failure .. Try *T=1*
    - Assert $At(P1,SFO)^1 \wedge At(P2,JFK)^1$
  - …
  - Repeat this until some minimal path length is reached.
  - Termination is ensured by T_{max}

cnf, mapping ← TRANSLATE-TO_SAT(*problem, T*)

- How to encode actions into PL?
  - Propositional versions of successor-state axioms

    $At(P1,JFK)^1 \Leftrightarrow$

    $(At(P1,JFK)^0 \wedge \neg(Fly(P1,JFK,SFO)^0 \wedge At(P1,JFK)^0)) \vee$
    $(Fly(P1,SFO,JFK)^0 \wedge At(P1,SFO)^0)$
  - Such an axiom is required for each plane, airport and time step
  - If more airports add another way to travel than additional disjuncts are required

- Once all these axioms are in place, the satisfiability algorithm can start to find a plan.

assignment ← SAT-SOLVER(*cnf*)

- Multiple models can be found
- They are NOT satisfactory: (for T=1)

  $Fly(P1,SFO,JFK)^0 \wedge Fly(P1,JFK,SFO)^0 \wedge Fly(P2,JFK, SFO)^0$

  The second action is infeasible

  Yet the plan IS a model of the sentence

$$initial\ state \wedge all\ possible\ action\ descriptions \wedge goal^1$$

- Avoiding illegal actions: pre-condition axioms

$$Fly(P1,SFO,JFK)^0 \Rightarrow At(P1,JFK)$$

- Exactly one model now satisfies all the axioms where the goal is achieved at *T=1*.

assignment ← SAT-SOLVER(*cnf*)

- A plane can fly at two destinations at once
- They are NOT satisfactory: (for T=1)

  $Fly(P1,SFO,JFK)^0 \wedge Fly(P2,JFK,SFO)^0 \wedge Fly(P2,JFK, LAX)^0$

  The second action is infeasible

  Yet the plan allows spurious relations
- Avoid spurious solutions: action-exclusion axioms

  $\neg(Fly(P2,JFK,SFO)^0 \wedge Fly(P2,JFK,LAX))$

  Prevents simultaneous actions
- Lost of flexibility since plan becomes totally ordered : no actions are allowed to occur at the same time.
  - Restrict exclusion to preconditions

Plannning in the Real World

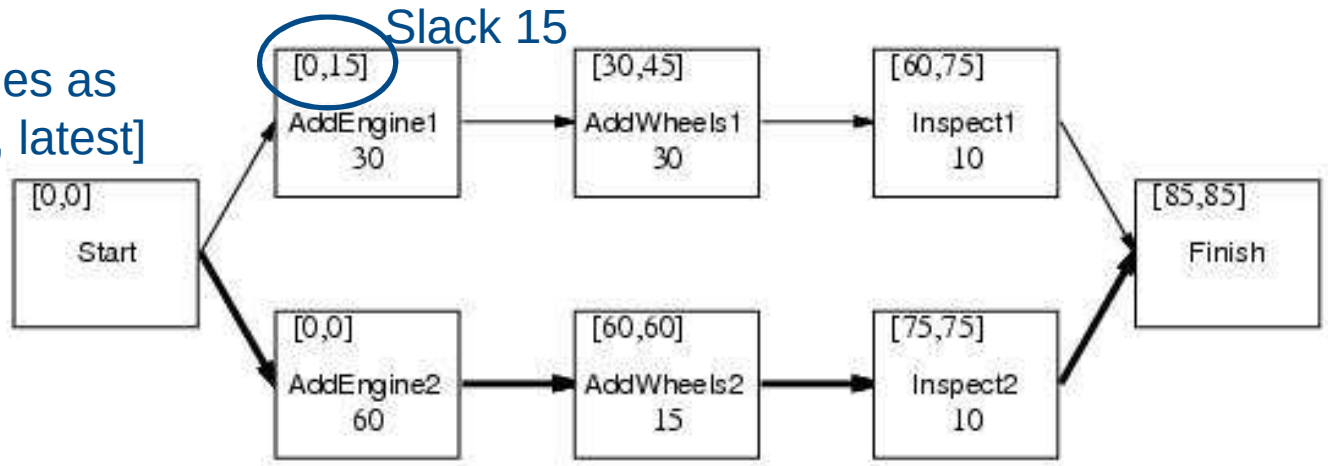
Until now: planning considered as search for goal states

In real-world applications, additional constraints apply

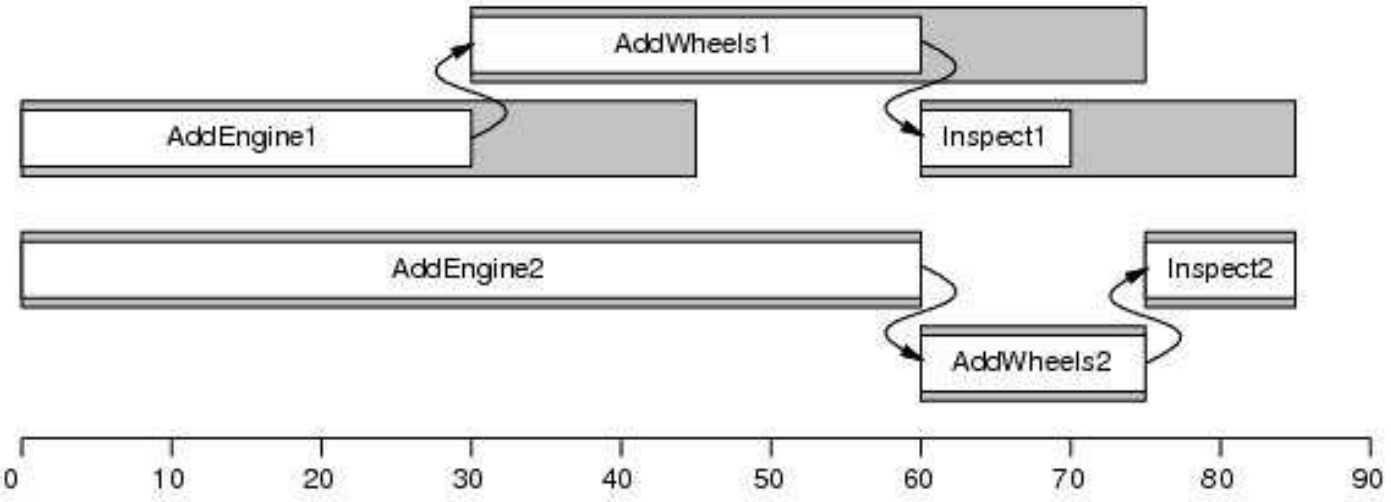
- Time
  - Execution times are relevant, especially in concurrent plans
- Resources
  - Availability of reusable resources, e.g. machines, robots, …
  - Availability of consumable resources, e.g. fuel, screws, …
- In addition to finding a valid plan, scheduling is important
  - Execution order in concurrent plans determines overall execution time

Scheduling example – critical path method

Start times as
[earliest, latest]

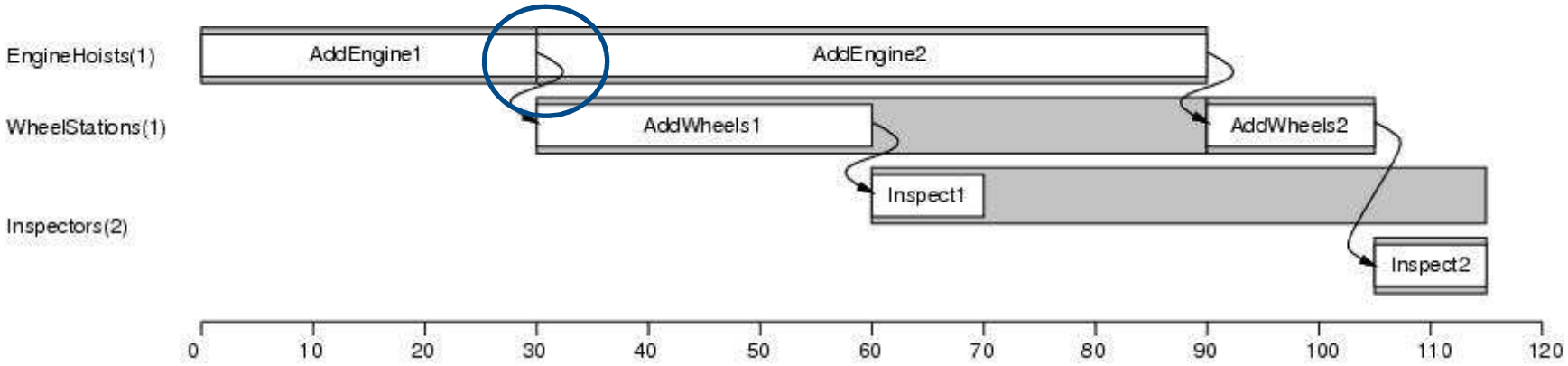


Bold arrows: critical path, i.e. maximal duration, actions with slack zero



Scheduling example – critical path method

- Example with limited resources
  - Only 1 engine hoist leads to sequentialisation of AddEngine



Hierarchical planning

- Several levels of abstraction for the plans
- Detailing out the plan during the planning steps, can be deferred to plan execution phase for the sake of flexibility
  - Off-line vs. on-line planning
  - E.g. in driving, route with cities planned off-line, exact steering parameters on-line based on sensor information
- Plan refinement
  - Simplest form: Description as tuple

(original plan, refined plan)

Hierarchical planning

Plan refinement example

- List of possible substitutions
  (start and termination conditions not shown)

$$((transport), (road - transport))$$
$$((transport), (road - transport, railroad - transport, road - transport))$$
$$((transport), (road - transport, sea - Transport, road - transport))$$
$$((transport), (road - transport, railroad - transport, sea - transport, road - transport))$$
$$((transport), (road - transport, air - transport, road - transport))$$

- Choice of actual substitutions needs semantic information
- Choice can be restricted by pre-conditions

Hierarchical planning

Plan refinement example with resource limitations

- Move object on table
- Resources are 1 mobile manipulator, 1 mobile base

- Abstract operation move can be re-written as

$$((move), (attach, push)) \quad \text{mobile base}$$
$$((move), (grasp, pull)) \quad \text{mobile manipulator}$$

- The mobile manipulator can perform both alternatives, the mobile base only the first

Hierarchical planning

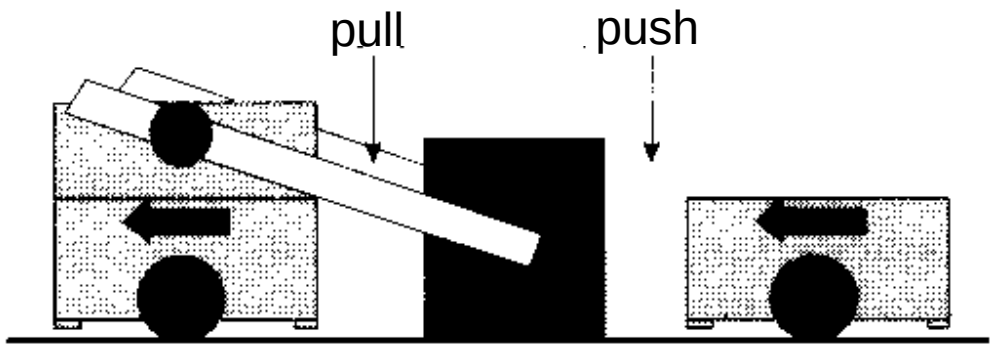
Plan refinement example with resource limitations

- Refinement of move operation

$$((move), (push))$$
$$((move), (attach,\ push))$$
$$((move), (approach, attach,\ push))$$

alternatively stepwise refinement

$$((move), (push))$$
$$((push), (attach,\ push))$$
$$((attach), (approach, attach))$$



Hierarchical planning

Plan refinement example with resource limitations

- Formal description of the refinement by replace operator

$$replace(original,\ substitute,\ precondition,\ aditional\_property)$$

- In the example (off-line variant)

$$\text{replace}\left(\begin{array}{c}(move),\\ (\text{grasp}(manipulator), \text{pull}(manipulator)),\\ (\text{available}(manipulator)),\\ (\text{addcost}(rentalfee))\end{array}\right)$$

Hierarchical planning

Plan refinement example with resource limitations

- On-line variant with dynamic resource allocation during run-time:

$$\text{replace}\left(\begin{array}{c}(move),\\ (\text{request}(manipulator), \text{grasp}(manipulator), \text{pull}(manipulator)),\\ (\text{own}(rentalfee)),\\ (\text{consumed}(rentalfee), \text{addcost}(rentalfee))\end{array}\right)$$

Additional aspects of planning

Not considered in the context of this course

- Single vs. multi agent plans
- Centralized vs. decentralized planning
- Plan coordination
- Fault-tolerance aspects
  - Concurrent alternatives
  - Plan repair techniques

- Motion planning techniques
  - See module IN2138 Robot Motion Planning

Summary

- Planning is an area of great interest within AI
- Biggest problem is the combinatorial explosion in states.

- Planning described as set of preconditions, actions, and postconditions
- Use of search strategies to create plans
- Use of theorem proving

- Consideration of limited resources and time

- Hierarchical planning approaches

Evaluation

Please fill in the evaluation forms


Frohes Fest