OPPA European Social Fund
Prague & EU: We invest in your future.

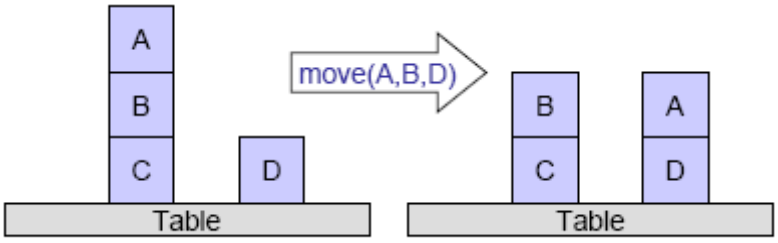
Introduction, representations

Introduction ~ 20 min

Basic notions

Problem → Problem specification/**Planning problem** in a **language** → **Planner** → Solution **= Plan**

- **Problem** (solve rubic cube, navigating mars vehicle, handling containers in dock, …)
- **Planning problems**
  - model of the world: objects, relations, actions
  - state, initial+goal state, legal moves/actions ~ transitions
- **Planner** = a software tool which is able to compute plans given the description of the environment, possible actions, an initial state and goals to achieve.
  - domain-independent ~ general problem solving / domain-specific
  - satisficing / optimal (actions with assigned cost)
- **Plan =** sequence of actions; application of actions: initial → goal state



Properties of the world: Dynamics

- **Deterministic dynamics:** Action + current state uniquely determine successor state.
- **Nondeterministic dynamics:** For each action and current state there may be several possible successor states.
- **Probabilistic dynamics:** For each action and current state there is a probability distribution over possible successor states.
- **Analogy:** deterministic versus nondeterministic automata

Properties of the world: Observability

- **Full observability:** Observations/sensing determine current world state uniquely.
- **Partial observability:** Observations determine current world state only partially: we only know that current state is one of several possible ones.
- **No observability:** There are no observations to narrow down possible current states. However, can use knowledge of action dynamics to deduce which states we might be in.
- **Consequence:** If observability is not full, must represent the knowledge an agent has.

Classical Planning

- dynamics: **deterministic**; observability: full, partial or **none**

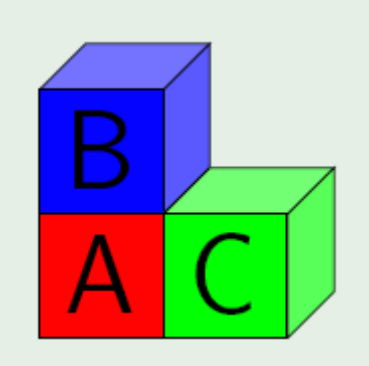
Representation ~ 45 min

Propositional Representation (výroková logika) ~ 10 min

- Propositions (výrokové formule): atomic proposition (atomická formule) / formula (výroková formule)
- **world state** is set of propositions
  - A-on-B, A-on-C, A-on-table, B-on-A, B-on-C, B-on-table, C-on-A, C-on-B, C-on-table
- **action** consists of precondition propositions, propositions to be added and removed

- move-B-A-C … move B which is on A on C

State variables ~ 10 min



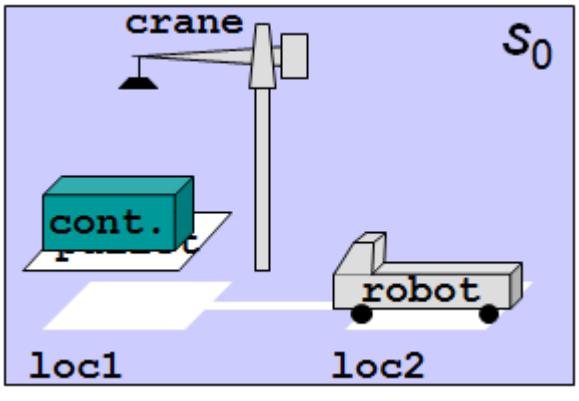
- The state of the world is described in terms of a finite set of finite-valued state variables.
- **Block-world**
  - **State variables:**
    - location-of-A = Table, location-of-B = A, location-of-C = Table
    - location(A) = Table, location(B) = A, location(C) = Table
  - Values/domains
  - Description of the state of the block-world
- Propositional Representation = Boolean state variables

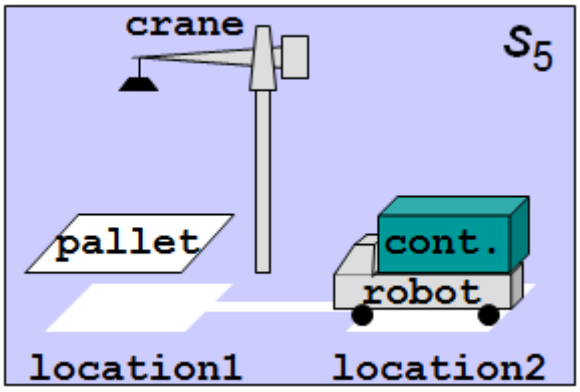
STRIPS ~ 10 min

- First-order logic (predikátová logika)
- STRIPS is a tuple <*P*, *A*, *I*, *G*>
  - *P* … set of ground atoms in first-order logic, e.g., at(r1,loc2)
  - $I \subseteq P$ … initial situation
  - $G \subseteq P$ … goal situation
  - *A* … finite set of actions *a* specified via pre(*a*), add(*a*), del(*a*)
- **states:** collections of atoms
- **applicable action:** *a* applicable in state *s* iff $pre(a) \subseteq s$
- **applying an applicable action *a* in *s*:** $s' = (s \setminus del(a)) \cup add(a)$

Describe the dock-worker robot domain ~ 15 min

- **objects:** loc1, loc2, pallet, robot, cont, crane
- **s0** = {attached(pile,loc1), on(cont,pallet), top(cont,pile), belong(crane,loc1), empty(crane), adjacent(loc1,loc2), adjacent(loc2,loc1), at(robot,loc2), free(loc1), unloaded(robot)}
- **action move:** move(r,l,m)
  - pre: adjacent(l,m), at(r,l), free(m)
  - add: at(r,m), free(m)
  - del: free(m), at(r,l)
- **plan to reach state s5:** <move(robot,loc2,loc1), take(crane,loc1,cont,pallet), load(crane,loc1,cont,robot), move(robot,loc1,loc2)>





STRIPS Example ~ 25 min

Monkey and bananas: A monkey is at location A in a lab. There is a box in location C. The monkey wants the bananas that are hanging from the ceiling in location B, but it needs to move the box and climb onto it in order to reach them.

→ **Describe predicates, initial state, goal state, actions, plan to solve the problem.**

Solution

Predicates: at, level, boxAt, bananasAt
Initial state: At(A), Level(low), BoxAt(C), BananasAt(B)
Goal state: Have(Bananas)
Actions:

```
// move from X to Y
_Move(X, Y)_
Preconditions:  At(X), Level(low)
Postconditions: not At(X), At(Y)

// climb up on the box
_ClimbUp(Location)_
Preconditions:  At(Location), BoxAt(Location), Level(low)
Postconditions: Level(high), not Level(low)

// climb down from the box
_ClimbDown(Location)_
Preconditions:  At(Location), BoxAt(Location), Level(high)
Postconditions: Level(low), not Level(high)

// move monkey and box from X to Y
_MoveBox(X, Y)_
Preconditions:  At(X), BoxAt(X), Level(low)
Postconditions: BoxAt(Y), not BoxAt(X), At(Y), not At(X)

// take the bananas
_TakeBananas(Location)_
Preconditions:  At(Location), BananasAt(Location), Level(high)
Postconditions: Have(bananas)
```

OPPA European Social Fund
Prague & EU: We invest in your future.