INTRODUCTION TO MEANS-ENDS PLANNING

Ivan Bratko

FRI, Univerza v Ljubljani

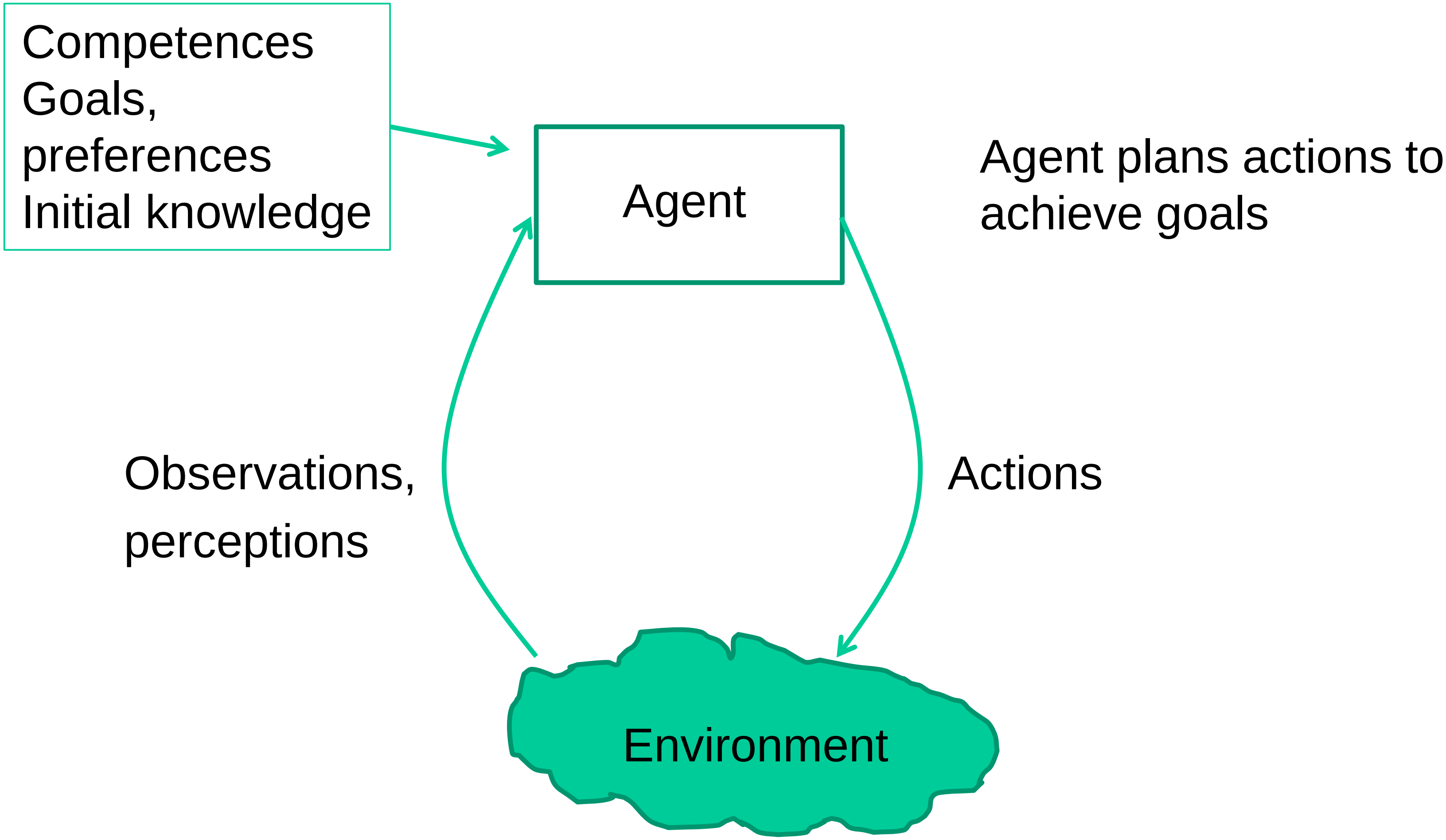
2021/22

A definition of AI makes reference to planning

AI is the field that studies the synthesis and analysis of computational agents that act intelligently.

D. Poole & A. Mackworth, Artificial Intelligence: Foundations of Computational Agents, Cambridge University Press, 2010

Agent acting in its environment



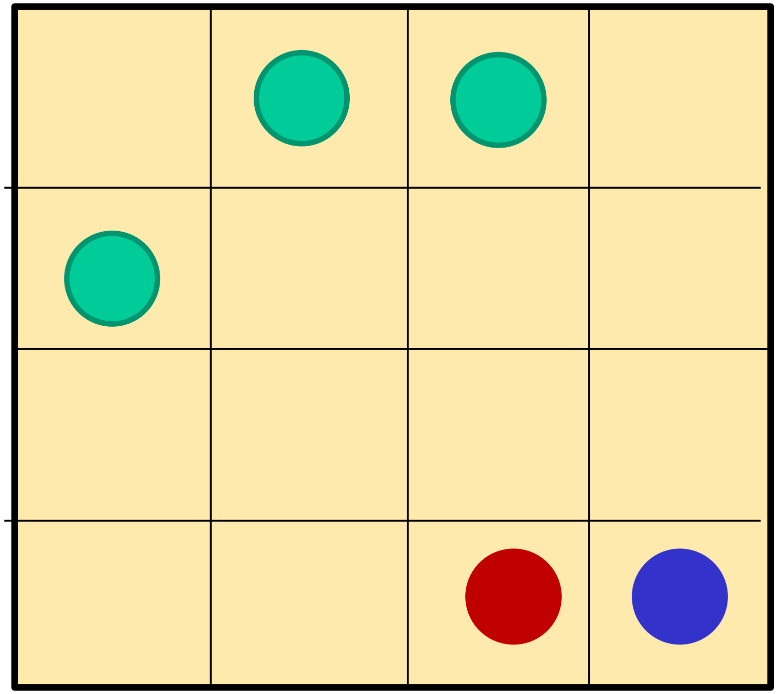
MEANS-ENDS PLANNING

- Word “planning” is used in two senses
- “PLANNING” in general sense includes problem solving in state space
- “PLANNING” in narrow sense is:

  “means-ends planning”
- Means-ends planning is topic of this presentation

EXAMPLE

- Five robots on grid; red robot wants to move to the bottom right corner



EXAMPLE CONTINUED: FINDING A PLAN

- With state space search: search possible movements of **all five robots**

- With means-ends planning: realize that green robots don't matter

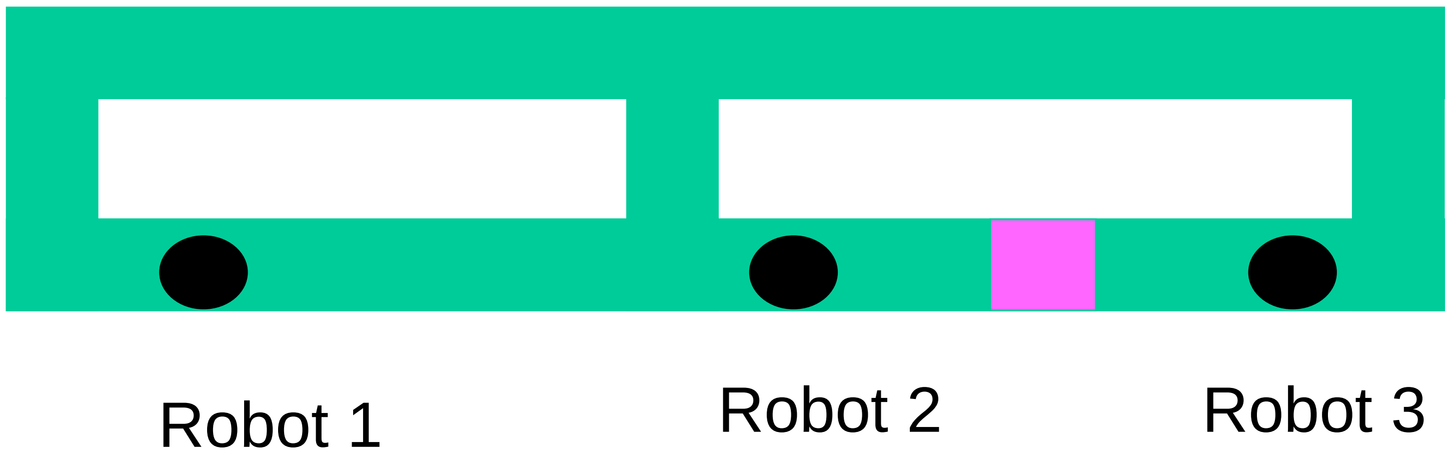
- Means-ends reasoning:

  (a) red robot moving right requires:

  right-bottom corner must be empty

  (b) to empty right-bottom corner, blue robot moves up

Example: mobile robots

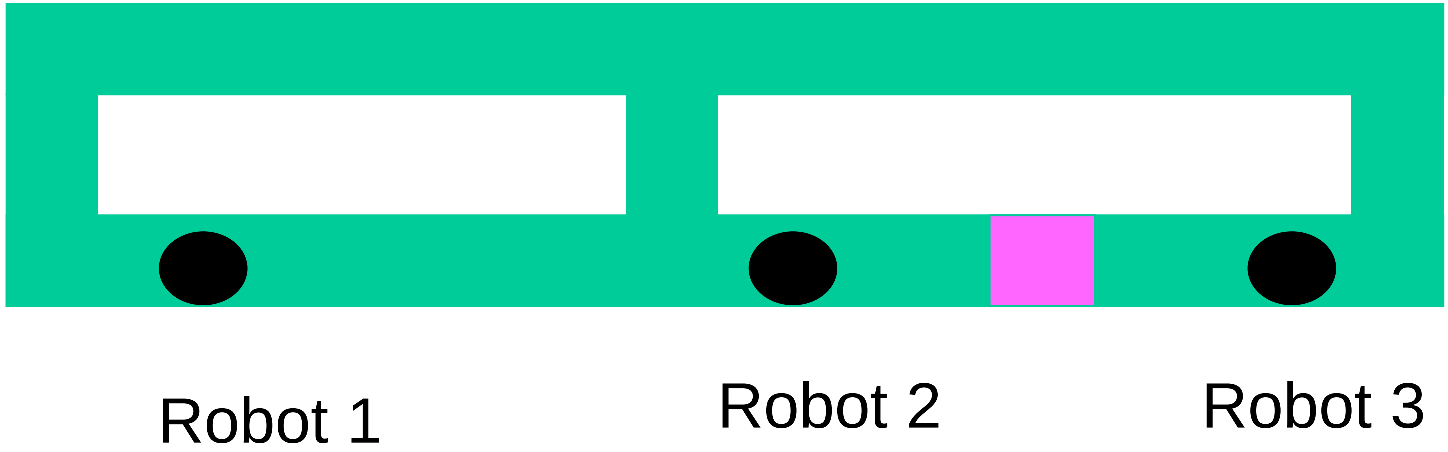


Task: Robot 1 wants to move into pink

How can plan be found with state-space search?

Means-ends planning avoids irrelevant actions

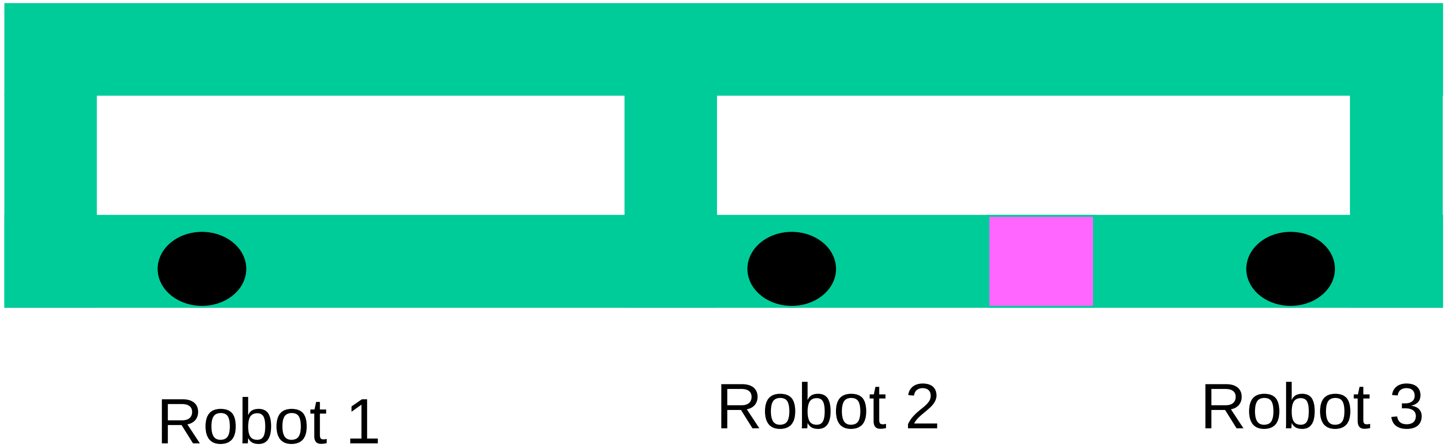
Solving with state-space



Task: Robot 1 wants to move into pink

Construct state-space search graph:
  states + successor relation among states

Solving by means-ends planner



Task: Robot 1 wants to move into pink

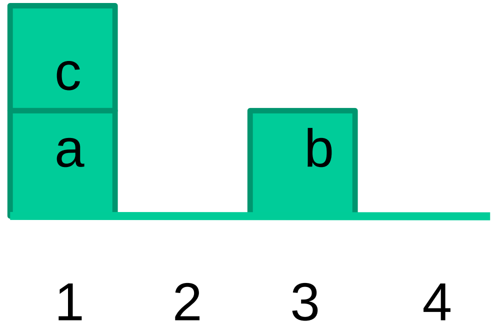
Formulate goal
Formulate actions in terms of preconditions and effects

Representation

- How to represent a classical planning problem?

STATES ARE REPRESENTED WITH RELATIONS

Example state from blocks world



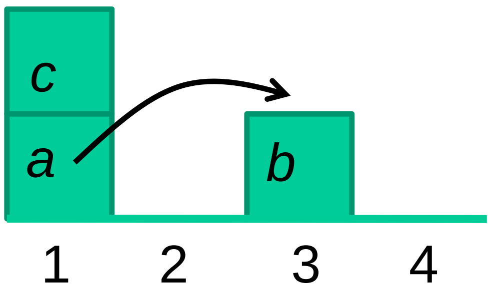
This state can be represented by the following set of relationships:

on(c,a), on(a,1), on(b,3), clear(2), clear(4), clear(b), clear(c)

DEFINING GOALS AND POSSIBLE ACTIONS

- Example of goals:
  on(a,b), on(b,c)

- Example of action:
  move(a, 1, b)
  (Move block a from 1 to b)



- Action preconditions:
  clear(a), on(a,1), clear(b)

- Action effects:
  on(a,b), clear(1), ~on(a,1), ~clear(b)

"add" (true after action)

"delete" (no longer true after action)

ACTION SCHEMA

- Action schema represents a set of actions using variables (variable names here written with capital initials)

- **move(X, Y, Z)**

  X is any block

  Y and Z are any block or location

- Precondition: **on(X,Y), clear(X), clear(Z)**
- Adds: **on(X,Z), clear(Y)**
- Deletes: **on(X,Y), clear(Z)**

STRIPS language for problem definition

- STRIPS = Stanford Research Institute Problem Solver

- STRIPS – traditional representation
  “STRIPS-like representation”

- STRIPS makes some simplifications:

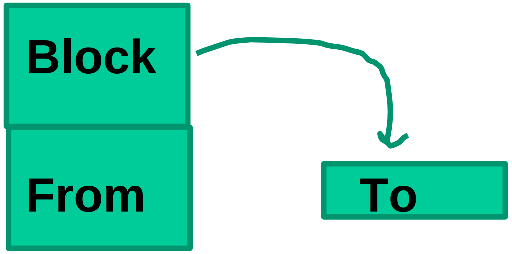
  - no variables in goals
  - positive relations given only
  - unmentioned relations are assumed false (c.w.a. – closed world assumption)
  - effects are conjunctions of relations

- There are several other “STRIPS-like” planning problem definition languages

DOMAIN SPECIFICATION FOR BLOCKS WORLD

Action:

move(Block, From, To)



Action precondition:

clear(Block), clear(To), on(Block, From)

Positive effects (“add“):

on(Block,To), clear(From)

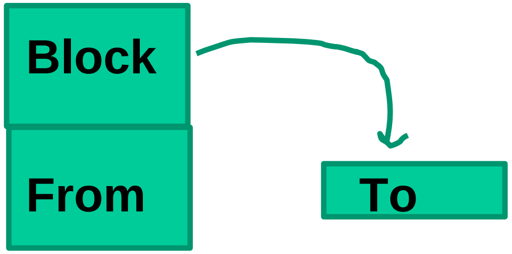
Negative effects (“del”)

on(Block,From), clear(To)

BETTER WITH ADDITIONAL CONSTRAINTS

Action:

move(Block, From, To)



Precondition for action:

clear(Block), clear(To), on(Block, From)

Additional constraints:

```
block( Block),        % Object Block to be moved must be a block
object( To),          % "To" is an object, i.e. a block or a place
To \= Block,          % Block cannot be moved to itself
object( From),        % "From" is a block or a place
From \= To,           % Move to new position
Block \= From
```

SPECIFICATION OF BLOCKS AND LOCATIONS

% Our blocks world: three blocks a, b and c, and 4 locations

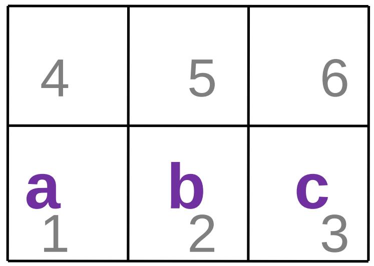
block(a).   block(b).   block(c).

place(1).   place(2).   place(3).   place(4).

% X is an object if X is a block or a place:

object(X) $\leftarrow$ (block(X) V place(X))

ROBOTS ON GRID IN STRIPS



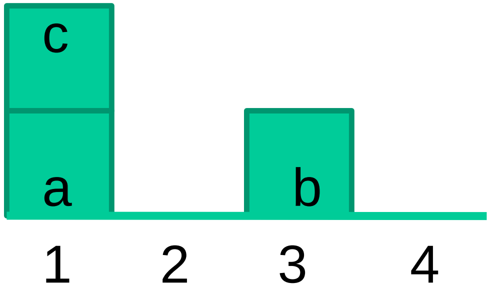
Robots: a, b, c, cells 1, ..., 6

Goal: at(a,3)

A plan: m(b,2,5) $\longrightarrow$ m(a,1,2) $\longrightarrow$ m(c,3,6) $\longrightarrow$ m(a,2,3)

Question: Propose a STRIPS-like representation for this planning problem

PRINCIPLE OF MEANS-ENDS ANALYSIS



In this state, the following relations hold:

on(c,a), on(a,1), on(b,3), clear(2), clear(4), clear(b), clear(c)

Let goal of plan be **on(a,b)**; find plan:

What action establishes **on(a,b)**? Such action is: **move(a,X,b)**

What is precondition COND for this action?

COND: **on(a,X), clear(a), clear(b)**

Set intermediate goal COND, find plan for COND

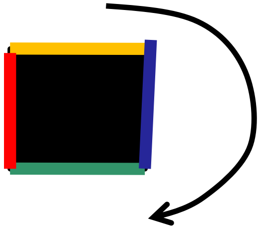
Etc.

COMPARISON WITH STATE SPACE

- In state-space: search state space
- In means-ends planning: search space of sets of goals
- Space of sets of goals = abstraction of state space (part of description of a state is ignored)
- What is better? Means-ends planning may be able to avoid searching useless actions, see example on next slide

BLOCKS A LITTLE DIFFERENTLY: ADD COLORS + ROTATIONS

- Block, top view; sides have different colors

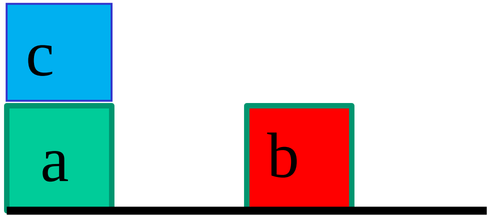


- Possible action is also: rotation
- Result of rotation: block changes color (if viewed from the side)

- rot_clockwise(Block, Color, NewColor1)
- rot_anticlock(Block, Color, NewColor2)

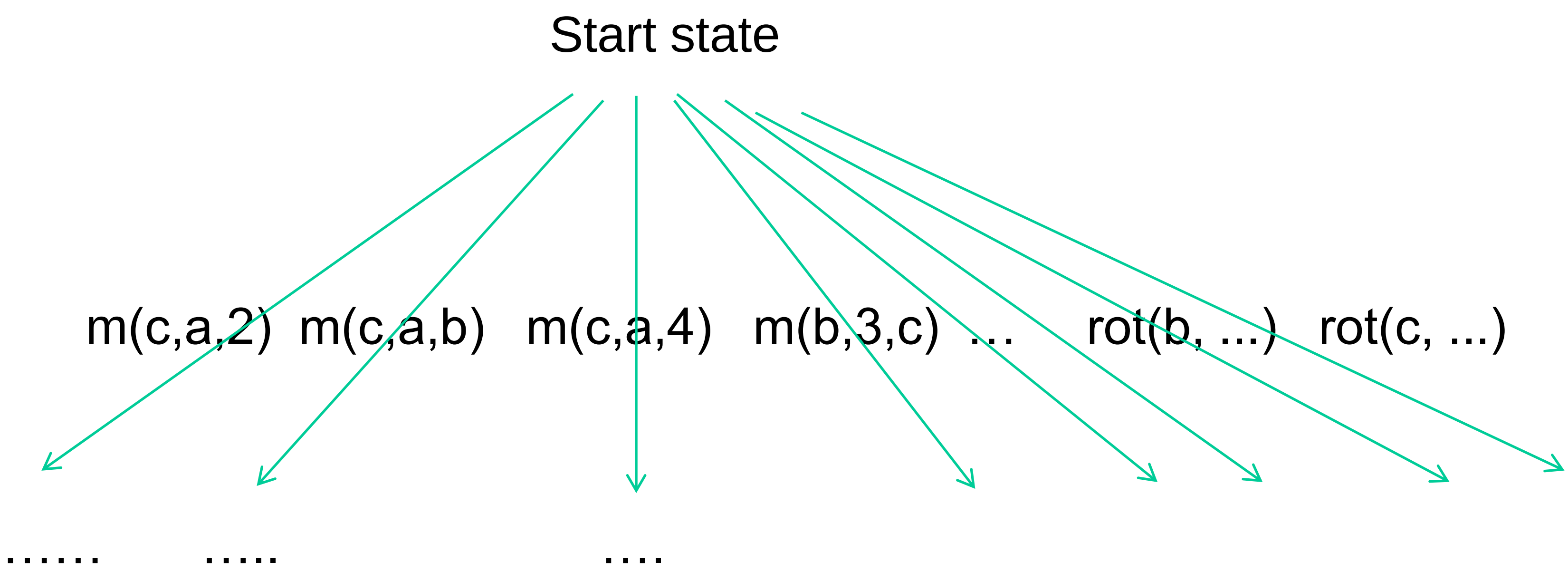
COMBINATORIAL SPACE

- Start state



- Goal: on(a,b)
- State space: takes into account actions **move** and **rotate**
- Means-ends principle finds on its own that only **move** actions are relevant to this problem; rotations can be ignored

Actions in state space



Means-ends planning in blocks world with actions move and rotate

- Let goal be: on(a,b)

  Which actions achieve on(a,b)?

  Such actions are of form: move(a,X,b)

  What is precondition COND for move(a,X,b)?

  COND: on(a,X), clear(a), clear(b)

  In start state, clear(b) and on(a,X) are true if we choose X=1

  To achieve COND, it remains to achieve clear(a)

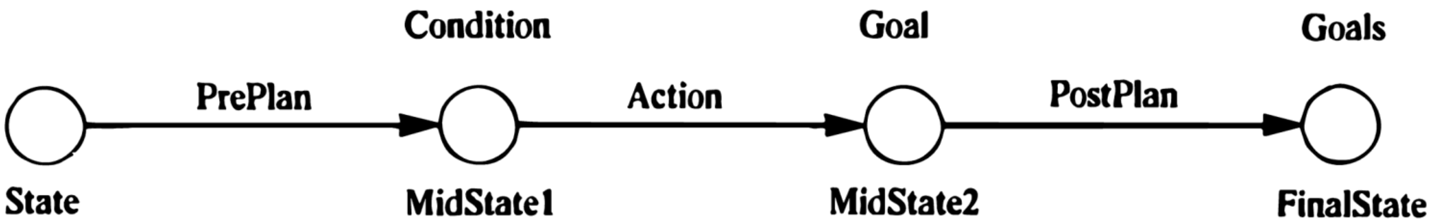
  Which actions achieve clear(a)?

  Such actions are of form:

  move(Y,a,Z)

- Note: For this planning problem, there is never any reason to consider actions of form rotate(…)! Planner may just ignore rotations

MEANS-ENDS PLANNING IN STRIPS
One possible realisation of means-ends planning



NONDETERMINISTIC STRIPS ALGORITHM

```
procedure  plan( InitialState, Goals, Plan, FinalState)
  if  Goals  ⊆  InitialState  then  Plan = [ ]   else        % All goals achieved
  begin
     Select   a  goal G from Goals;
     Select   an  action A that  achieves  G;              % adds( A, G)
     PreCond  =  preconditions of A;
     plan( InitialState, PreCond, PrePlan, MidState1) ;    % Enable A
    Apply A to MidState1 giving MidState2;
    plan( MidState2, Goals, PostPlan, FinalState);   % Achieve remaining goals
    Plan   =  concatenate( PrePlan,  [ Action ], PostPlan)
  end
```

STRIPS IMPLEMENTED IN PROLOG

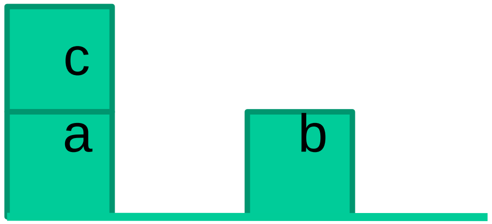
```
%   plan( State, Goals, Plan, FinalState)

plan( State, Goals, [ ], State)  :-
  satisfied( State, Goals).

plan( State, Goals, Plan, FinalState)  :-
  conc( PrePlan, [Action | PostPlan], Plan),          % Divide plan
  select( State, Goals, Goal),                        % Select a goal
  achieves( Action, Goal),                            % Relevant action
  can( Action, Condition),
  plan( State, Condition, PrePlan, MidState1),        % Enable Action
  apply( MidState1, Action, MidState2),               % Apply Action
  plan( MidState2, Goals, PostPlan, FinalState).      % Remaining goals
```

Not for exam!

PROCEDURAL ASPECTS:
WHAT SEARCH STRATEGY DO WE USE?

Example of a very awkward search strategy:

?- start(S), plan(S, [on(a,b), on(b,c)], P).



Start state S

```
P = [ move(b,3,c),    % To achieve on(b,c)
      move(b,c,3),
      move(c,a,2),
      move(a,1,b),    % To achieve on(a,b)
      move(a,b,1),
      move(b,3,c) ,   % To achieve on(b,c) again
      move(a,1,b)]    % To achieve on(a,b) again
```

Why 7 steps, why not 3?

We need an appropriate search strategy!

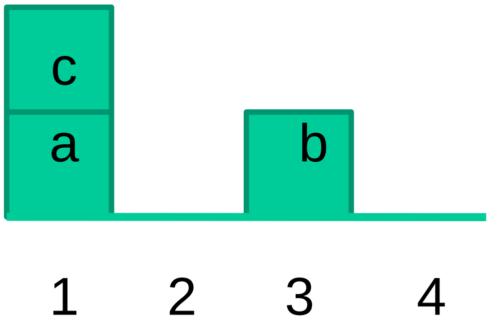
ADDITIONAL DETAILS

- Search strategy (depth-first, breadth-first, ...)
- Goal protection: do not destroy what you already achieved!
- But: goal protection is not always possible! Sometimes it is unavoidable to (temporarily) destroy what has already been achieved.

Realisation with iterative deepening

- Start with plan of length 0 and keep increasing maximal allowed length of plan, until plan is found
- On each iteration (for each maximal plan length) search all possible plans with depth-first search
- Surprise is possible, for example in the case of Sussman's anomaly (on next slide)

SUSSMAN'S ANOMALY



Goals: on(a,b), on(b,c)

Basic STRIPS planner with breadth-first search produces:

move(c, a, 2)

move(b, 3, a) ??? What is the point of this ???

move(b, a, c)

move(a, 1, b)

Problem is: STRIPS concentrates on solving a single goal at a time

Here STRIPS was pursuing just on(a,b) and achieved on(b,c) by chance

EXPLANATION

move(c, a, 2)	achieves clear(a) for move(b,3,a)
move(b, 3, a)	achieves on(b,a) for move(b,a,c)
move(b, a, c)	achieves clear(a) for move(a,1,b)
move(a, 1, b)	achieves on(a,b)

First three actions achieve clear(a) – precond. for move(a,1,b):

move(b,a,c) achieves clear(a), precond. for move(b,a,c) is on(b,a);

move(b,3,a) achieves precond. for move(b,a,c);

precond. for move(b,3,a) is clear(a), which is achieved by move(c,a,2)

Then planner tries to achieve on(b,c), which was in the meantime already achieved by luck (!) with move(b,a,c)

There is no motivation for **move(b,3,c)** in 2nd step w.r.t. on(a,b)! STRIPS is short-sighted – it is only concerned with the current, *local goal*, in this case on(a,b)

COMPLETENESS

- Even with global iterative deepening, STRIPS planner still has problems.
- E.g. it finds a four step plan above for our example blocks task
- Why STRIPS cannot find the optimal, 3-step plan? Basic STRIPS is **incomplete**! It does not consider all possible plans.
- Problem: **locality** (only work towards achieving **one** goal G at a time, temporarily ignoring other goals until G is achieved)
- Sometimes referred to as “linearity” (goals are achieved in “linear order”)

- Basic STRIPS algorithm does not consider everything that makes sense!
- Problem: locality in achieving goals
- “Linear planning”: first goal1, then goal 2, ...
- In previous example: first on(a,b), then on(b,c)
- Better idea instead of STRIPS algoritma: **Goal regression**

GOAL REGRESSION

- STRIPS solves goals one after another “locally” (when solving one goal it does not consider other goals)
- Better: “global planning” (keep in mind all the goals all the time)
- One idea to achieve global planning is ***goal regression***
- This is based on concept of “Regressing Goals through Action”

$$\text{RegressedGoals} \xrightarrow[\text{Action}]{} \text{Goals}$$

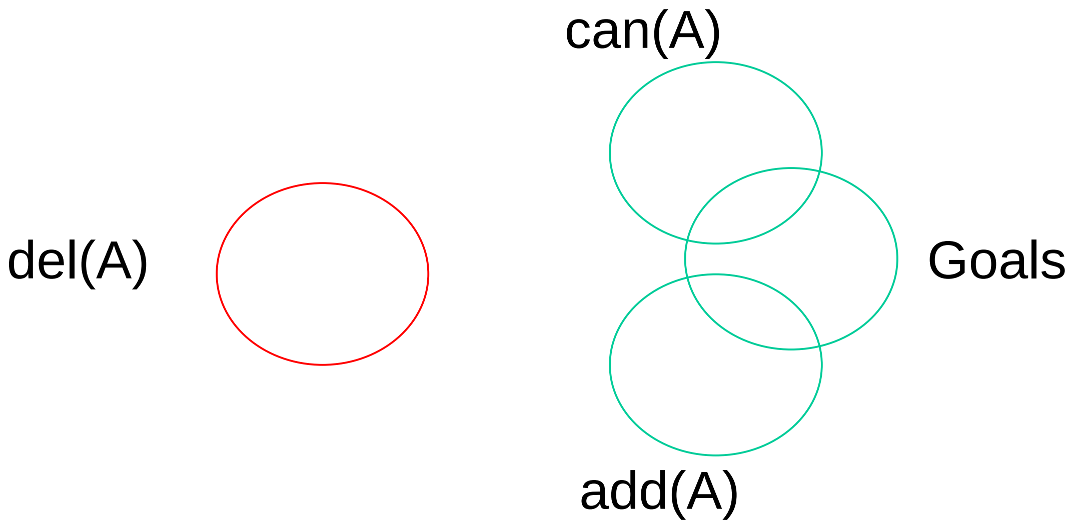
- What (RegressedGoals) must be true before Action, in order that Goals are true after Action?

EXAMPLE OF GOAL REGRESSION

- Let goals of plan be: {on(a,b), on(b,c)}
- A relevant action to these goals is: move(a,1,b)
- We regress goals {on(a,b), on(b,c)} through action move(a,1,b):

RegressedGoals		Goals
{on(a,1), clear(a), clear(b), on(b,c)}	$\xrightarrow[\text{move(a,1,b)}]{}$	{on(a,b), on(b,c)}

GOAL REGRESSION



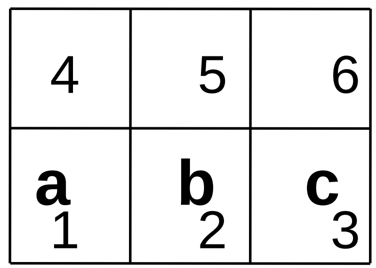
RegressedGoals = Goals + can(A) - add(A)

Goals and del(A) must be disjoint: Goals ∩ del(A) = { }

Goal regression enables “global” planning:

Planner can see all relevant goals at any point of planning

EXAMPLE: ROBOTS ON GRID



Robots a, b, c; cells 1, ..., 6

Goal:: at(a,3)

Plan: m(b,2,5) → m(a,1,2) → m(c,3,6) → m(a,2,3)

DOMAIN DEFINITION

Action: Robot R moves from A to B

m(R,A,B)

Preconditions:

at(R,A), c(B)

Additional constraints:

robot(R), adjacent(A,B) % R is a robot, A and B are adjacent

Positive effects:

at(R,B), c(A)

Negative effects:

at(R,A), c(B)

Finding a plan for goal at(a,3)

Initial state:: at(a,1),at(b,2),at(c,3),c(4),c(5),c(6)

at(b,2),c(5),at(a,1),c(6),at(c,3)

m(b,2,5)

at(a,1),c(2),c(6),at(c,3)

m(a,1,2)

at(a,2),at(c,3),c(6)

m(c,3,6)

at(a,2),c(3)

m(a,2,3)

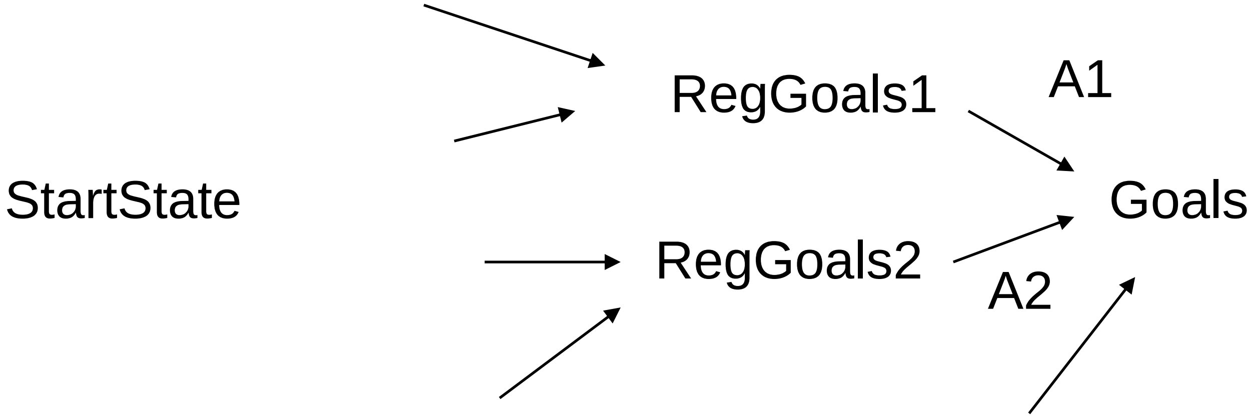
at(a,3)

m(a,6,3)

at(a,6),c(3)

Start here, work backwards

STATE SPACE FOR PLANNING WITH GOAL REGRESSION



Goal state and heuristic

- “Goal” condition for this search space:

  RegressedGoals is a subset of StartState

- A possible heuristic function for this search space: # regressed goals that are not true in StartState:

$$h = |\text{RegressedGoals} - \text{StartState}|$$

QUESTION

- Is this heuristic function for the blocks world optimistic? That is, does it satisfy the condition of admissibility theorem for best-first search?

- If not, can it be modified to become optimistic?

CAN THIS HEURISTIC BE REFINED?

- Can we take into account the difficulty of individual goals?
- In robots on grid example with goal regression, how could the difficulty of regressed goal sets {at(a,2),c(3)} and {at(a,6),c(3)} be compared?
- How about distance between start state and at(a,2), and between start state and at(a,6)?

SUMMARY

- STRIPS representation for planning
- STRIPS planning algorithm
- STRIPS algorithm is not complete – problem with locality w.r.t. current goal
- Planning with goal regression
- Note: All algorithms discussed here produce **totally ordered** plans; that is sequences of actions that are executed one after another, no parallel actions are possible
- For parallel actions, **partial-order planning** is needed (e.g. POP algorithm or GRAPHPLAN – our next topic)