Course "Automated Planning: Theory and Practice"
Chapter 01: Introduction to Planning

Teacher: Marco Roveri - marco.roveri@unitn.it

M.S. Course: Artificial Intelligence Systems (LM)

A.A.: 2023-2024

Where: DISI, University of Trento

URL: https://bit.ly/3zOkGk8

Last updated: Thursday 14th September, 2023

Terms of Use and Copyright

The material (text, figures) in these slides is authored by Jonas Kvarnström and Marco Roveri.

What is "Planning"?

A simple structured toy application

Possible actions

Move topmost block from A to B

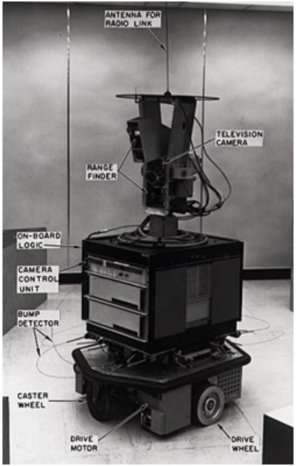
Objective

All blocks in piles of the same color

What is "Planning"?: Shakey

- Classical robot example: Shakey (1969)
  - Available actions:
    - Move from location A to location B
    - Turn light switches on/off
    - Open/Close door
    - Push an object around
    - ...
  - Objective:
    - Be in room X with objects O1, O2 and O3

https://www.youtube.com/watch?v=qXdn6ynwpiI



© WikipediA: https://en.wikipedia.org/wiki/Shakey_the_robot

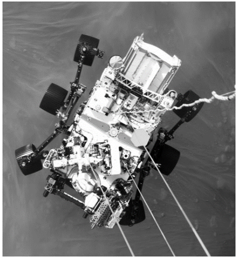
What is "Planning"?: Miconic 10 Elevators

- Shindler Miconic 10 system
  - Tall buildings, equipped with multiple elevators
  - Available actions:
    - Move from one floor to another floor
    - Open/Close door
    - ...
  - Objective:
    - Enter destination before you board
    - Synthesize a "plan" telling which elevator goes to which floor and in which order

https://elevation.fandom.com/wiki/Schindler_Miconic_10

What is “Planning”?: Space missions

https://en.wikipedia.org/wiki/
Samantha_Cristoforetti



Perseverance © NASA 2021

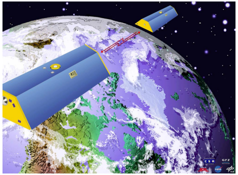
NASA Mapgen/Mars Rover © NASA

https://www.jpl.nasa.gov/

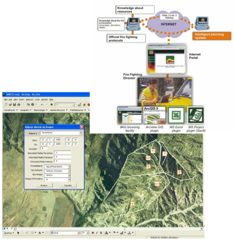
Create daily activity plans for Spirit, Opportunity, Perseverance and for the International Space Station.

Mixed initiative tool with Human in the Loop

What is "Planning"?: Earth missions



© NASA - Gravity Recovery and Climate Experiment



`https://www.researchgate.net/publication/220308820_SIADEX_An_interactive_knowledge-based_planner_for_decision_support_in_forest_fire_fighting`

`https://nms.kcl.ac.uk/planning/currentprojects.html`

- Create daily activity plans for satellite hearth observations
- Plan for firefighting considering limited resources
- Plans for autonomy in hostile environments

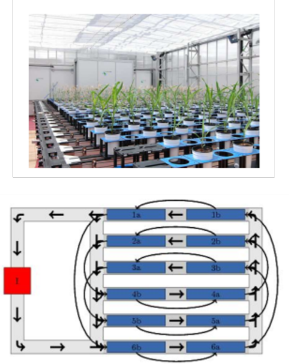
What is "Planning"?: Other applications

F.E.A.R. © Warner Bros. Interactive Entertainment

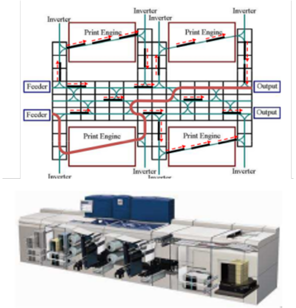
FIFA 2X © EA Sports

What is "Planning"?: Other applications (cont.)

Logistics

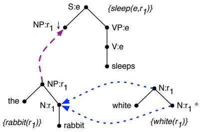


Greenhouse logistics

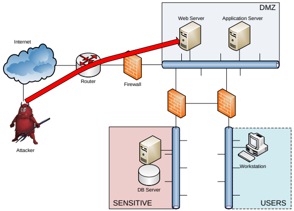


Modular Printing Control

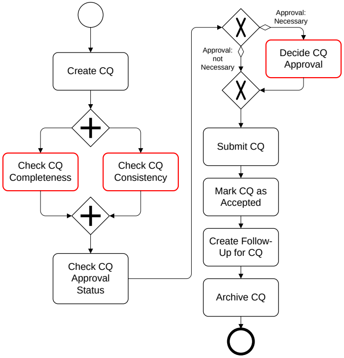
What is "Planning"?: Other applications (cont.)



Natural Language Generation



Penetration Testing



Business Process Templates Generation

©Prof. Jörg Hoffmann

Planning definition

One way to define planning:

Using knowledge about the world, including possible actions and their results, to decide what to do and when in order to achieve an objective, before actually start doing it.

Why "Planning"?

- Manual planning can be boring and inefficient!
- Automated planning can create higher quality plans
  - It is possible to apply systematic optimizations
- Can be applied where the agent is!
  - Satellites/Exploration robots cannot always communicate with Ground operators
  - Spacecraft robots on other planets may be hours away by radio (Earth-Mars about 30min delay)

Planning vs Reacting: Unmanned Aerial Vehicles

- A modern context for planning:
  - Autonomous Unmanned Aerial Vehicles (UAV)

*Using **knowledge** about the world, including possible actions and their results, to **decide** what to do and when in order to achieve an **objective**, **before** actually start doing it.*

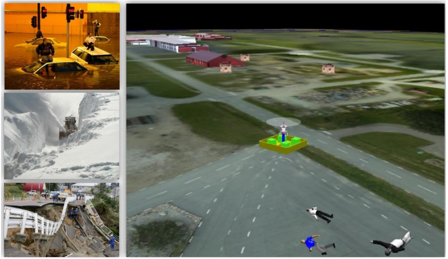
Actions for UAVs

- General knowledge about the world
  - Location of UAVs and objects
  - Fuel levels, ...
- Available actions:
  - Takeoff
    - Before: The UAV must be on the ground
    - Result: The UAV is flying
  - Fly to a point
    - Before: Must have sufficient fuel
    - Result: Will end up at the indicated point
  - Land
  - Fly a trajectory curve
  - Point camera, take picture
  - Transmit data to operator
  - ...

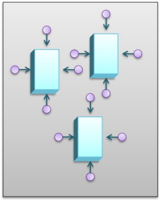
*Using **knowledge** about the world, including possible actions and their results, to **decide** what to do and when in order to achieve an **objective**, **before** actually start doing it.*

UAV Objectives

- Emergency Service Logistics
  - Assist in emergency situations
    - Deliver packages of food, medicines, water
    - Inspect the zone to detect fires, people
    - ...



- Photogrammetry
  - Find best way to take pictures to ensure proper 3D rendering
    - From specific locations
    - In the specific directions



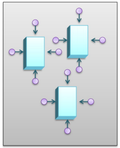
*Using **knowledge** about the world, including possible actions and their results, to **decide** what to do and when in order to achieve an **objective**, **before** actually start doing it.*

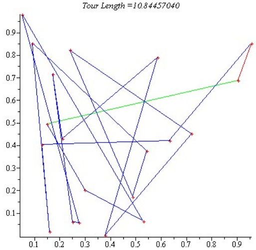
Method 0: Reactive and dumb!

- Let's be **reactive** and dumb!
  - Reactive: No planning, don't explicitly consider future!
  - Very fast decision and execution algorithm

```
while (exists unvisited position) {
    if (greedy) {
        pos := some random unvisited position;
    } else {
        pos := some nearest unvisited position;
    }
    flyto(pos);
    aim();
    take-picture();
}
```

  - Somewhat sub-optimal for flight in both cases
  - Fly too far → run out of fuel!



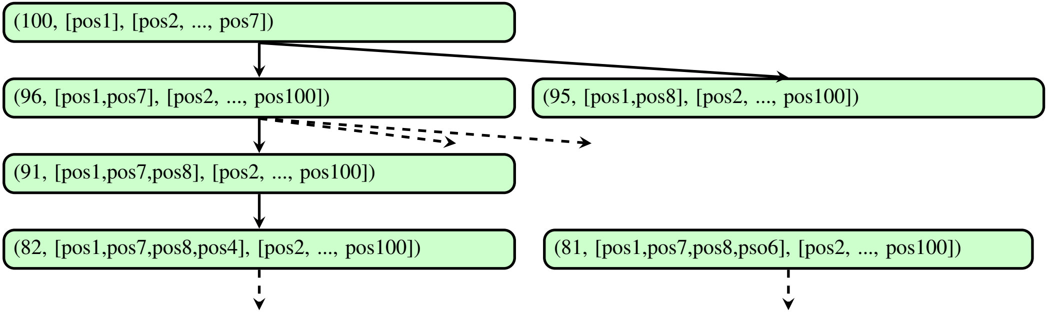


https://www.maplesoft.com/applications/view.aspx?SID=6873&view=html

*Using **knowledge** about the world, including possible actions and their results, to **decide** what to do and when in order to achieve an **objective**, **before** actually start doing it.*

Method 1: think ahead!

- First create a complete plan by searching, considering multiple choices
  - Keeping track of (fuel, visited, remaining)



- Second execute the plan after verifying it

Method 1: first step: Search (e.g. depth first)

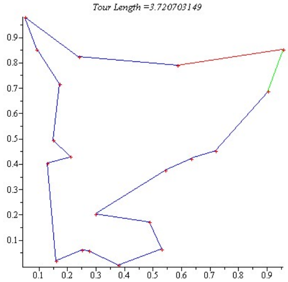
```
call solve(100, [pos1], {pos2, ..., pos100})
solve (fuel-before, plan, remaining) {
  if (remaining == ∅) return plan;
  currpos := last(plan)
  foreach position nextpos in remaining
          in order of increasing
          distance(curpos, nextpos) {
    after := fuel-before - fuel_usage(curpos, nextpos);
    if (after > 0) {
      plan := plan+[nextpos];
      remaining := remaining-[nextpos];
      plan2 := solve(after, plan, remaining);
      if (plan2 != NULL) return plan2;
    }
  }
  return NULL;
}
```

Have we already achieved the goal?

First choice: as before (greedy heuristic)
If not feasible: try next rearest pos
Usually won't have to try each position

Check fuel in simulation not in reality!

Backtrack if there is no feasible continuation!

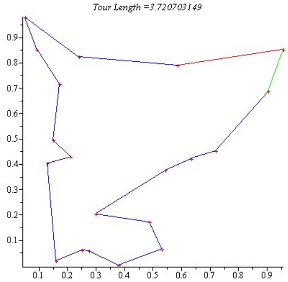


This is (one form of) planning!

https://www.maplesoft.com/applications/view.aspx?SID=6873&view=html

Method 1: second step: execution

```
seq := solve(100, [pos1], {pos2, ..., pos100})
if (seq != NULL) {
   signal error!;
   foreach position pos in seq) {
      flyto(pos);
      aim();
      take-picture();
   } }
```

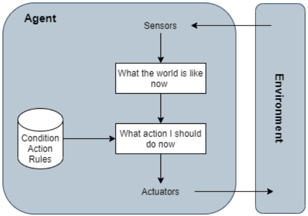


https://www.maplesoft.com/applications/view.aspx?SID=6873&view=html

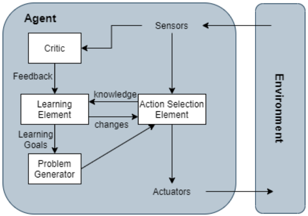
Planning as deliberation: three main approaches

Using **knowledge** about the world, including possible actions and their results, to **decide** what to do and when in order to achieve an **objective**, **before** actually start doing it.

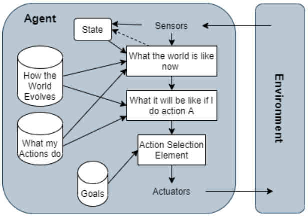
- *Programming-based agent* - writing the agent control by hand
- *Learning-based agent* - infer the control by experience (Reinforcement Learning)
- *Model-based approach* - derive the control automatically by reasoning



Programming Based



Learning-Based

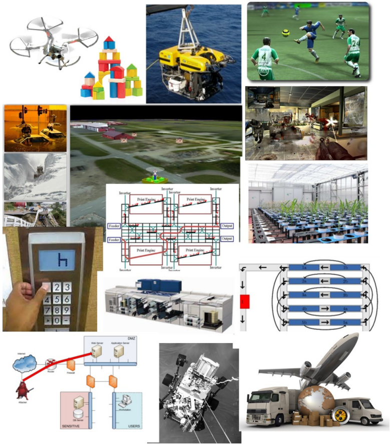


Model-Based

These approaches are not orthogonal!

What is Planning?

These solutions are domain specific, highly specialized, less flexible, efficiency might be difficult



- Programming-based:
  - Puts all the burden on the programmer
  - No formalisation of the solution proposed
  - Programs are brittle and error-prone
- Learning-based:
  - Need of data to learn and generalize behavior
  - Difficulties in generalising to different domains
  - Difficulties in defining high level preferences, goals
- Model-based
  - The model of the environment is not written, or not available in the language we want!
  - Models can be complex, burden on the machine

But at least you can reason about it.

AI Planning is Model Based Planning.

What is Model-Based Planning?

"Planning is the reasoning side of acting. It is an explicit deliberation process that chooses and organises actions, on the basis of their expected outcomes, in order to achieve some objective as best as possible." Ghallab et al. [3]

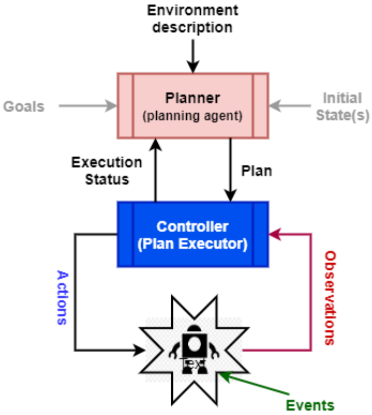
"...planning is the model based approach to action selection..." Geffner and Bonet [2]

"AI Planning is general problem solving" Sievers et al. [6], Newell et al. [5]

What is Model-Based Planning?

Main ingredients about model-based planning:

- In AI Planning the agent has access to world dynamic
  - actions describe how the world changes
  - sensor-model describe how to update the knowledge of the world
  - goals denote what the agent wants
  - states capture evolving relevant conditions of the world the agent is operating in
- A Planner is a *domain independent* program that can solve (find a *plan* for) all the planning problems starting from
  - an *environment* description (including action capabilities)
  - a *problem* description (initial state(s) and goal(s))



AI Planning Problem

Given:

- A description of (possible) **initial state(s)**
- A description of **desired goal states**
- A description of a set of **possible actions**

Generate:

- A **sequence of actions** that **leads to** one of **the goal states**.

Planning Problem: Blocks

Planning Problem: Tower of Hanoi (ToH)

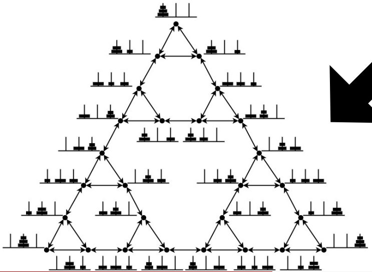
Why is planning so difficult?

- Uncertainty about effects of actions
- Uncertainty about environment
- Uncertain sensing and perception
- Agents own actions may have bad effects
- Time and resources are limited!
- Preferences
- Complex goals
- Optimality

Introduction

Formal Model

$$\langle \Sigma = \langle S, A, \gamma \rangle, s_0, S_g \rangle$$



Language to describe models

```
;;The Towers of Hanoi problem
;;  (formalisation by Hector Geffner).
(define (domain hanoi)
  (:requirements :strips)
  (:predicates (clear ?x) (on ?x ?y) (smaller ?x ?y))
  (:action move
     :parameters (?disc ?from ?to)
     :precondition (and (smaller ?to ?disc)
                        (on ?disc ?from)
                        (clear ?disc) (clear ?to))
     :effect (and (clear ?from) (on ?disc ?to)
                  (not (on ?disc ?from))
                  (not (clear ?to))))
)
```

Classical planning

Assumption 0: Finite number of states
Can't model continuous positions of disks in ToH
We are interested in some discrete alternatives: On peg 1, on peg 2, above disk 3, ...

The world is always in a given state, which we want to affect

Photogrammetry state:
(fuel left, visited positions, remaining positions)

- We need **states of the world**
  - $S = \{s_0, s_1, \ldots, s_S\}$ is a finite set of states

Classical planning (II)

Assumption 2: The world can only be affected by executing an action
No random changes in the world
No other agents acting in the world
At least not in the part of the world we model!

The world is always in a given state, which we want to affect

- We need **actions to affect the world**
  - $A = \{a_0, a_1, ..., s_A\}$ is a finite set of actions;

Classical planning (III)

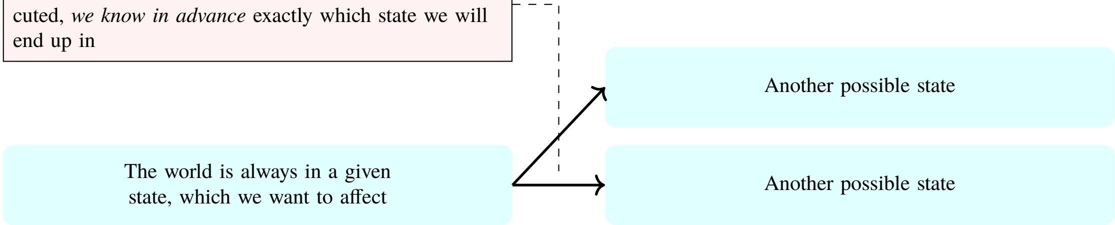
Assumption 3: Every action results in a discrete state transition
No concept of time
No concept of continuous change (crane swinging from A to B), only:
before pickup, the container is on truck y;
after, the container is carried by crane z

The world is always in a given state, which we want to affect → Another possible state

- We must know **when an action is executable** and **what it achieves**
  - $\gamma : S \times A \rightarrow 2^S$ — $\gamma(s, a)$ is the set of states you may end-up in if you execute a in s;
  - $\gamma(s, a) = \emptyset$ — means a cannot be executed in s;
  - $\gamma(s, a) = \{s'\}$ — means executing a in s leads to s';

Classical planning (IV)

Assumption 4: Deterministic actions
If we know the current state and the action that is executed, *we know in advance* exactly which state we will end up in



- We must know **when an action is executable** and **what it achieves**
  - $\gamma : S \times A \rightarrow 2^S$ — $\gamma(s, a)$ is the set of states you may end-up in if you execute a in s;
  - $\gamma(s, a) = \emptyset$ — means a cannot be executed in s;
  - $\gamma(s, a) = \{s_1, s_2, \ldots\}$ — means executing a in s leads to either s_1 or s_2 or...;
  - $|\gamma(s, a)| > 1$ — impossible in classical planning, due to Assumption 4;

$\Sigma = \langle S, A, \gamma \rangle$ is a *State Transition System (STS)*

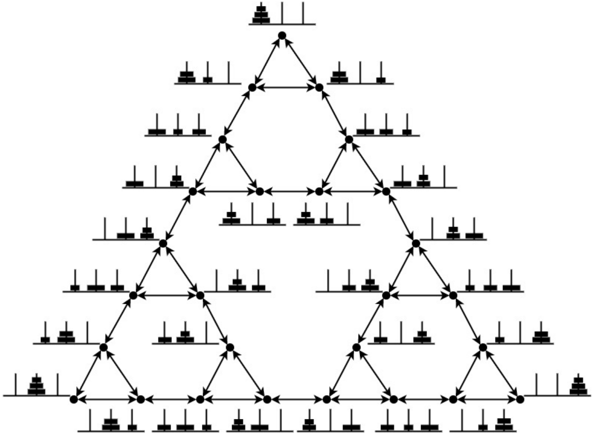
Classical planning (V): STS example

- Part of an STS for an instance of the Hanoi problem

27 states, 78 transitions
How many actions?

Can always have one action per transition: $a_1, \ldots, a_{78}$

Each action only executable in a single state

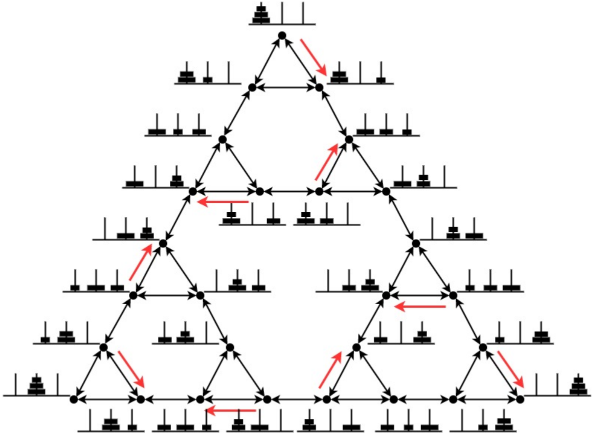


Classical planning (VI): STS example, cont.

- Part of an STS for an instance of the Hanoi problem

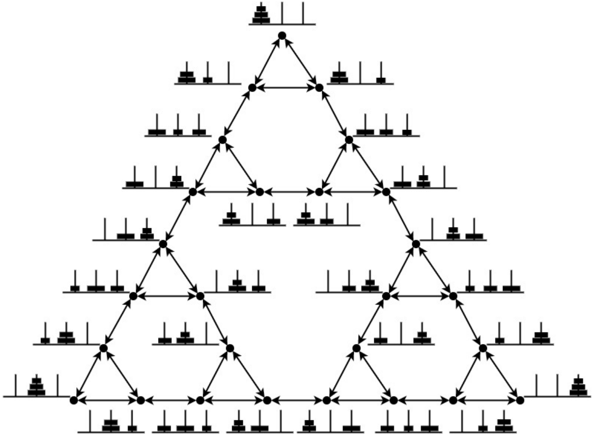
Common meaningful model:
Every red arrow is the same action,
move-diskA-from-peg1-to-peg3
18 context dependent actions

Or we could get by using three actions: at most three transitions from any state → Action "option1", "option2", "option3"



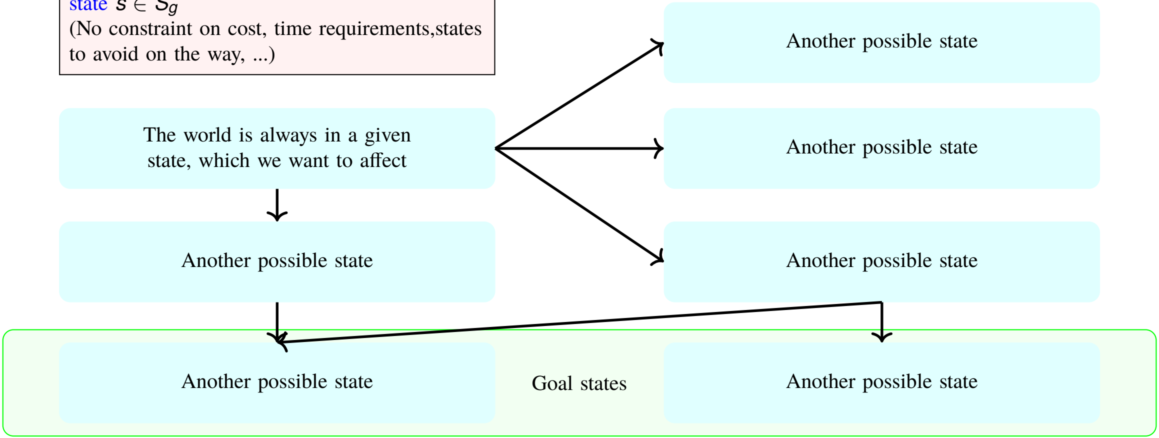
STS: How the world evolves in general

What's the formal model to solve?



Problem: Objectives

Assumption 5: Restricted objectives
The objective is always to end up in a goal state $s \in S_g$
(No constraint on cost, time requirements,states to avoid on the way, ...)



Problem: Initial state

Assumption 6: We can always detect the initial state

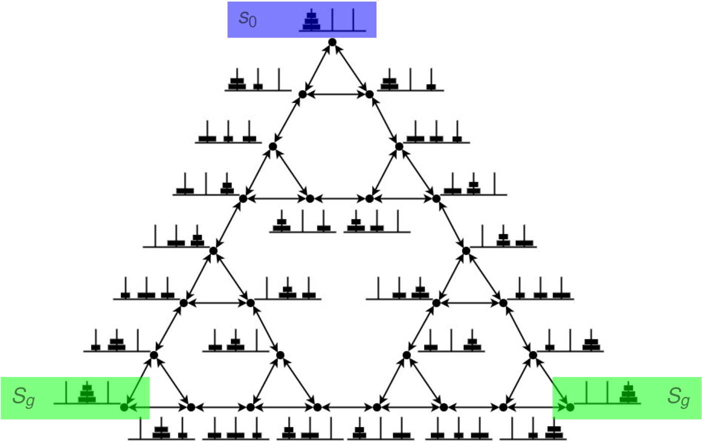
The world is always in a given state, which we want to affect

Assumption 7: Offline planning
No need to consider changes that may happen while generating plans.

We know now what the state of the world will be when we start executing a plan!
Initial state: s_0

PROBLEM DEFINITION

- A complete classical planning problem: $\langle \Sigma, s_0, S_g \rangle$



Transition System and Problem

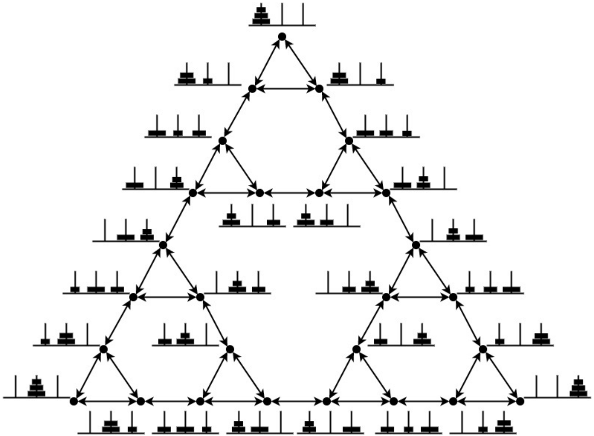
State Transition System $\langle S, A, \gamma \rangle$

Tells us: **How the world works** (Only those aspects that we need in our model in order to solve interesting problems!)

Planning Problem $\langle \langle S, A, \gamma \rangle, s_0, S_g \rangle$

Tells us: **Which specific problem to solve**

What's a solution?



ACTION SEQUENCE

> **Assumption 9**: Sequential execution
> A solution never executes two actions concurrently (Many planners do allow concurrency → "semi-classical")

- **Action sequence** $\sigma = \langle a_0, ..., a_N \rangle$ where $\forall i . a_i \in A$
    - Sometimes called plan
- An action sequence is executable in state $s \in S$ if $\exists s_1, ..., s_{N+1}$ such that:
    - $\gamma(s, a_0) = \{s_1\}$
    - $\gamma(s_1, a_1) = \{s_2\}$
    - ...
    - $\gamma(s_N, a_N) = \{s_{N+1}\}$
        - Sometimes called "executable action sequence", "plan", "executable plan", ..

Solution

- An action sequence is a solution to $\langle\langle S, A, \gamma\rangle, s_0, S_g\rangle$ if:
  - It is executable in s_0
  - It results in a state $s_{N+1} \in S_g$
  - Sometimes called "plan", "solution plan", ...
- A good solution:
  - Add a cost function $c : A \rightarrow \mathcal{R}$ to the STS
  - Minimize total plan cost:

$$\sum_{a_i \in \sigma} c(a_i)$$

PLAN GENERATION

- Is classical planning simply graph search?
  - Can be, but:
    - **Graphs are enormous**
      Requires advanced heuristics, adapted to planning
      Requires advanced search methods
    - **Alternatives to searching the STS** can be used to "indirectly" find paths!
    - Many forms of **non-classical** planning do not map into searching an STS

Is STS useful?

Very useful

- As a conceptual model, explaining important concepts
- To analyze espressivity, clarify restrictions
- To prove properties

Very useless

- As a way of actually **writing down** realistic planning problems (enumerate all possible states?)
- As an implementation structure for planners

References I

[1] Hector Geffner and Blai Bonet. *A Concise Introduction to Models and Methods for Automated Planning*. Synthesis Lectures on Artificial Intelligence and Machine Learning. Morgan & Claypool Publishers, 2013. ISBN 9781608459698. doi: 10.2200/S00513ED1V01Y201306AIM022. URL `https://doi.org/10.2200/S00513ED1V01Y201306AIM022`.

[2] Hector Geffner and Blai Bonet. *A Concise Introduction to Models and Methods for Automated Planning*. Synthesis Lectures on Artificial Intelligence and Machine Learning. Morgan & Claypool Publishers, 2013. ISBN 9781608459698. doi: 10.2200/S00513ED1V01Y201306AIM022. URL `https://doi.org/10.2200/S00513ED1V01Y201306AIM022`. 26

[3] Malik Ghallab, Dana S. Nau, and Paolo Traverso. *Automated planning - theory and practice*. Elsevier, 2004. ISBN 978-1-55860-856-6. 26

[4] Malik Ghallab, Dana S. Nau, and Paolo Traverso. *Automated Planning and Acting*. Cambridge University Press, 2016. ISBN 978-1-107-03727-4. URL `http://www.cambridge.org/de/academic/subjects/computer-science/artificial-intelligence-and-natural-language-processing/automated-planning-and-acting?format=HB`.

[5] Allen Newell, J. C. Shaw, and Herbert A. Simon. Report on a general problem-solving program. In *Information Processing, Proceedings of the 1st International Conference on Information Processing, UNESCO, Paris 15-20 June 1959*, pages 256–264. UNESCO (Paris), 1959. 26

References II

[6] Silvan Sievers, Martin Wehrle, and Malte Helmert. Bounded intention planning revisited. In Torsten Schaub, Gerhard Friedrich, and Barry O'Sullivan, editors, *ECAI 2014 - 21st European Conference on Artificial Intelligence, 18-22 August 2014, Prague, Czech Republic - Including Prestigious Applications of Intelligent Systems (PAIS 2014)*, volume 263 of *Frontiers in Artificial Intelligence and Applications*, pages 1097–1098. IOS Press, 2014. doi: 10.3233/978-1-61499-419-0-1097. URL `https://doi.org/10.3233/978-1-61499-419-0-1097`. 26