Bridging Planning and Reasoning in Natural Language with Foundational Models

Brief introduction to AI Planning

Shirin Sohrabi

What is AI Planning

Task of finding a procedure course of actions for a declaratively described system to reach its goals while optimizing overall performance measures.

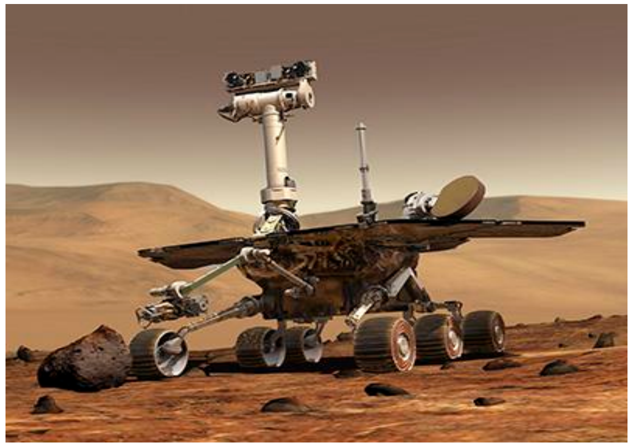
Basic planning problem includes:

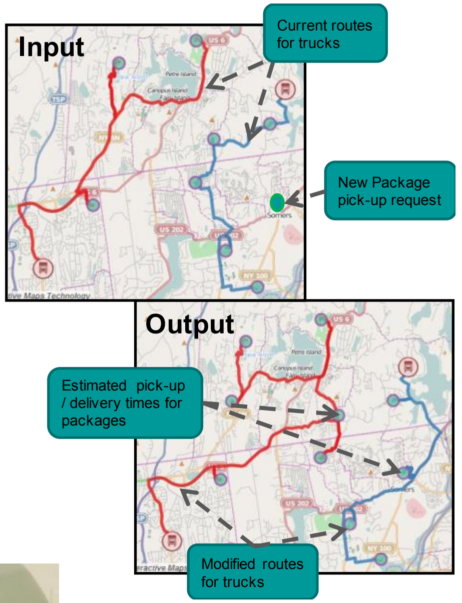
- Initial state of the world
- Desired goals
- A set of possible actions

Synthesize a plan that is guaranteed to generate a state which contains the desired goals.

https://www.odtap.com/2018/10

AI Planning is Everywhere




Input
Current routes for trucks
New Package pick-up request
Output
Estimated pick-up / delivery times for packages
Modified routes for trucks

Source: https://www.livemint.com/Opinion/tijzm8flw2RY98Jvdm8LzJ/Opinion--Cyber-security-a-complex-behaviour-problem.html
Source: https://www.rigzone.com/news/schlumberger_in_400mm_deal_to_sell_drilling_assets-15-may-2019-158842-article/

What is AI Planning

Three approaches to the problem of action selection or control in AI:

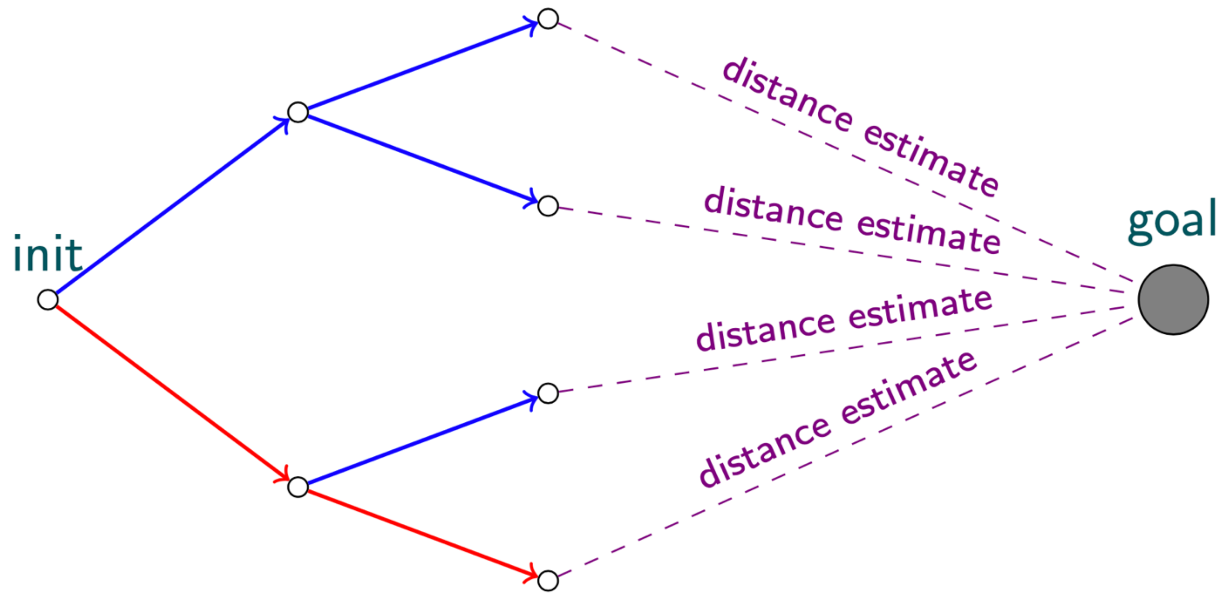
- Learning: learn control from experience
- Programming: specify control by hand
- Planning: specify problem by hand, derive control automatically

Planning is a form of general problem solving

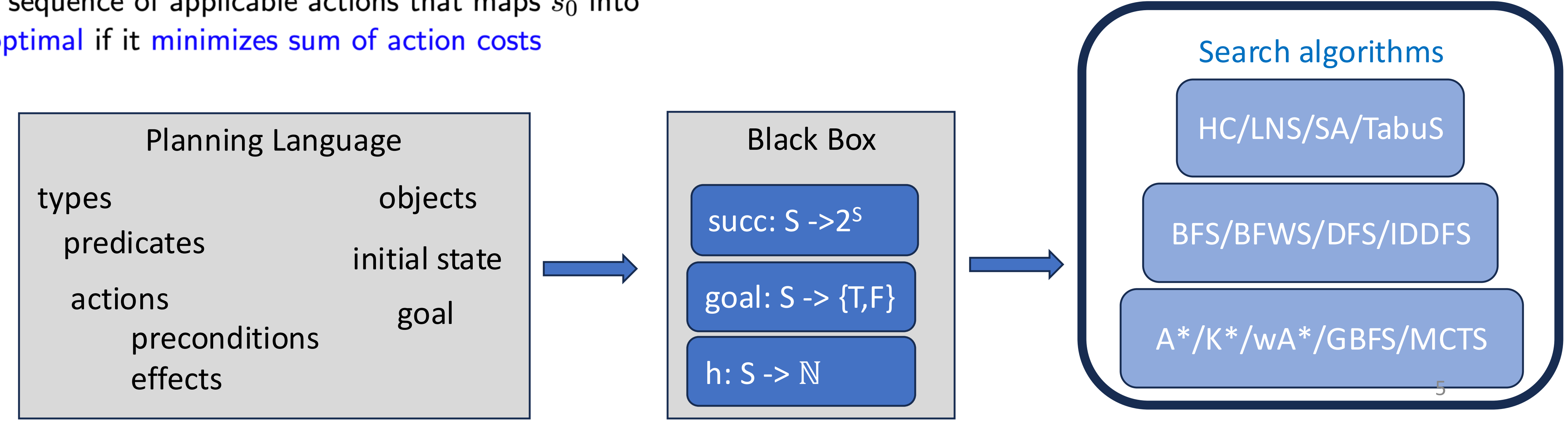
1. Models: define and classify the planning problem and its solution (plan)
2. Languages: represent the problem
3. Algorithms: solve the problem

What is AI Planning

- finite and discrete state space S
- a known initial state $s_0 \in S$
- a set $S_G \subseteq S$ of goal states
- actions $A(s) \subseteq A$ applicable in each $s \in S$
- a deterministic transition function
  $s' = f(a, s)$ for $a \in A(s)$
- non-negative action costs $c(a, s)$



A solution is a sequence of applicable actions that maps s_0 into S_G, and it is optimal if it minimizes sum of action costs



Language for Classical Planning: Strips

A Strips Planning task is 5-tuple $\Pi = \langle F, O, c, I, G \rangle$:

- F: finite set of atoms (boolean variables)
- O: finite set of operators (actions) of form $\langle Add, Del, Pre \rangle$ (Add/Delete/Preconditions; subsets of atoms)
- $c : O \mapsto \mathbb{R}^{0+}$ captures operator cost
- I: initial state (subset of atoms)
- G: goal description (subset of atoms)

Plan: sequence of applicable actions that maps I into a state consistent with G

Variants of Planning Problem

Action dynamics:	deterministic / non-deterministic / probabilistic
Observability:	partial / full
Horizon:	finite / infinite

- Conformant planning (uncertain initial state)
- Probabilistic planning
- Non-deterministic planning (many possible outcomes)
- Temporal Planning (durative actions)
- Hierarchical Task Network (HTN) planning
- Fully observable non-deterministic planning (FOND)
- Partially observable planning (POMDPs)

….

Computational Problems

- Cost-optimal planning: find a plan that minimizes summed operator cost
- Satisficing planning: find a plan, cheaper plans a better
- Agile planning: find a plan, quicker is better
- Top-k planning: find k plans such that no cheaper plans exist
- Top-quality planning: find all plans up to a certain cost
- Diverse planning: variety of problems, aiming at obtaining diverse set of plans, considering plan quality as well

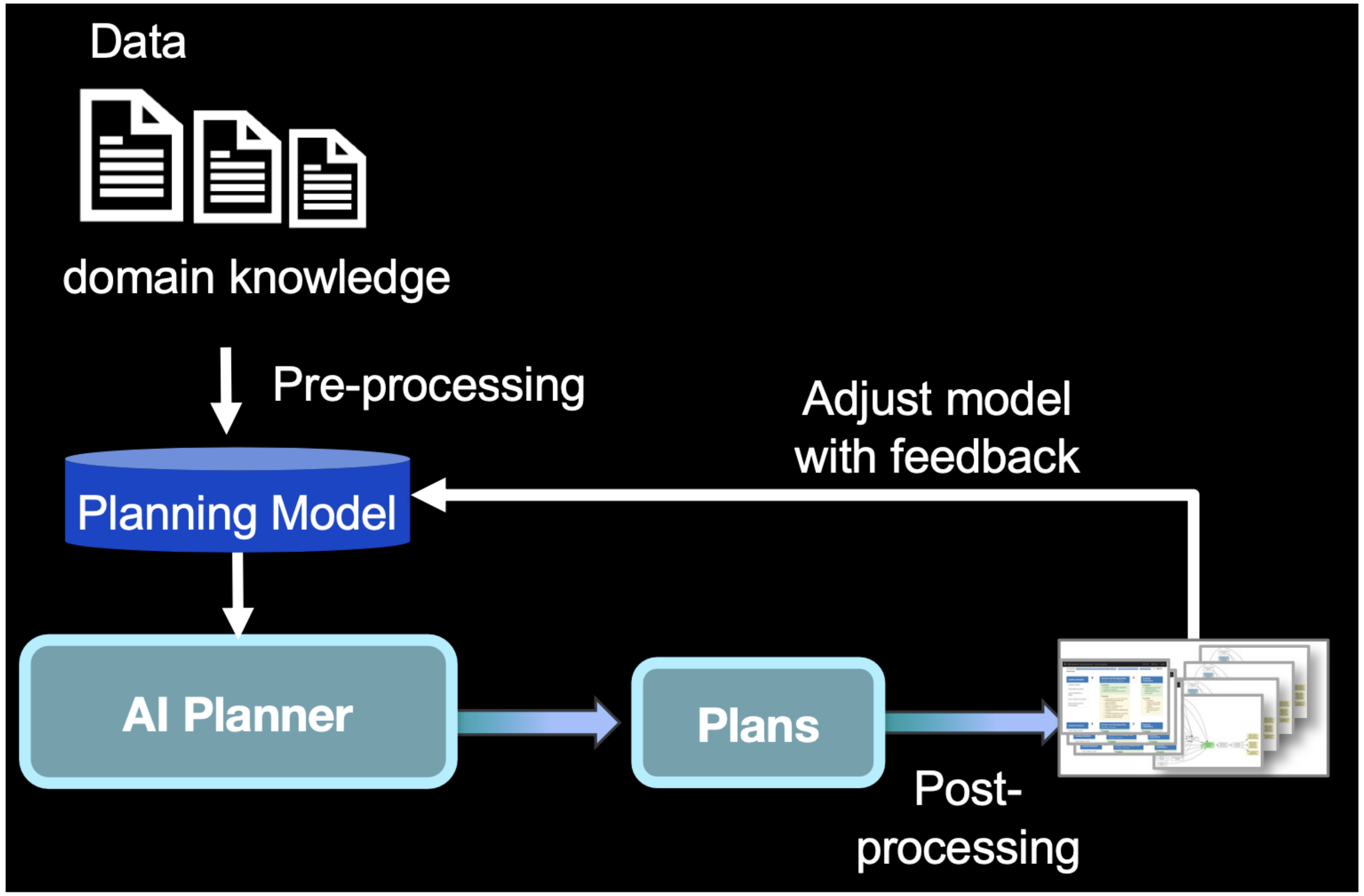
Major Planners/tools

- Fast-Forward (FF): classical satisficing, numeric, conformant, contingent (Hoffmann & Nebel, 2001)
- Fast Downward: classical, cost-optimal, satisficing, agile, cost-bouned, OSP, FOND, probabilistic, temporal (Helmert et al., 2006)
- SHOP2: HTN planning (Nau et al., 2003)
- LPG: classical, satisficing, numeric, temporal, diverse (Gerevini & Serina 2002)
- FOND planner PRP (Muise et al., 2012, 2014)
- OSP planners (Katz & Keyder 2019, Katz & Speck 2021)
- Top-k planners: K* (Katz et al., 2018) SymK (Speck et al., 2020)
- Forbid-iterative collection of planners for top-k, top-quality, diverse (Katz & Sohrabi 2020, Katz et al., 2020)

….

General Framework / Problems

- Model acquisition – how to derive / learn the planning model
- Planning – how to efficiently find solutions in the model
- Execution – how to efficiently execute model solutions in the environment



Representing the Knowledge (Example: PDDL)

http://editor.planning.domains/

```
(define (problem mixed)
   (:domain miconic)
   (:objects p0 p1 p2 p3 f0 f1 f2 f3 f4 f5 f6 f7)

(:init
(passenger p0)(passenger p1)(passenger p2)(passenger p3)
(floor f0)(floor f1)(floor f2)(floor f3)(floor f4)
(floor f5)(floor f6)(floor f7)

(above f0 f1)(above f0 f2)(above f0 f3)(above f0 f4)
(above f0 f5)(above f0 f6)(above f0 f7)(above f1 f2)
(above f1 f3)(above f1 f4)(above f1 f5)(above f1 f6)
(above f1 f7)(above f2 f3)(above f2 f4)(above f2 f5)
(above f2 f6)(above f2 f7)(above f3 f4)(above f3 f5)
(above f3 f6)(above f3 f7)(above f4 f5)(above f4 f6)
(above f4 f7)(above f5 f6)(above f5 f7)(above f6 f7)

(origin p0 f0)(destin p0 f5)(origin p1 f7)(destin p1 f4)
(origin p2 f0)(destin p2 f7)(origin p3 f1)(destin p3 f6)

(lift-at f0))

(:goal (and
(served p0)(served p1)(served p2)(served p3))))
```

```
(define (domain miconic)
  (:requirements :strips)

(:predicates
(origin ?person ?floor)
(floor ?floor)
(passenger ?passenger)
(destin ?person ?floor)
(above ?floor1 ?floor2)
(boarded ?person)
(served ?person)
(lift-at ?floor))

(:action board
  :parameters (?f ?p)
  :precondition (and (floor ?f) (passenger ?p)(lift-at ?f) (origin ?p ?f))
  :effect (boarded ?p))

(:action depart
  :parameters (?f  ?p)
  :precondition (and (floor ?f) (passenger ?p) (lift-at ?f) (destin ?p ?f)
                (boarded ?p))
  :effect (and (not (boarded ?p))
              (served ?p)))

(:action up
  :parameters (?f1 ?f2)
  :precondition (and (floor ?f1) (floor ?f2) (lift-at ?f1) (above ?f1 ?f2))
  :effect (and (lift-at ?f2) (not (lift-at ?f1))))

(:action down
  :parameters (?f1 ?f2)
  :precondition (and (floor ?f1) (floor ?f2) (lift-at ?f1) (above ?f2 ?f1))
  :effect (and (lift-at ?f2) (not (lift-at ?f1)))))
```

(up f0 f1)
(up f1 f7)
(board f7 p1)
(down f7 f4)
(depart f4 p1)
(down f4 f0)
(board f0 p0)
(up f0 f5)
(depart f5 p0)
(up f5 f6)
(down f6 f0)
(board f0 p2)
(up f0 f7)
(depart f7 p2)
(down f7 f1)
(board f1 p3)
(up f1 f6)
(depart f6 p3)

Want to Find out more:

- Tutorial link:
  - https://aiplanning-tutorial.github.io/, AI Planning: Theory and Practice, AAAI 2022
  - https://mp-tutorial.github.io/, Finding multiple plans for classical planning problems, ICAPS 2024

- Planners available on github:
  - https://github.com/aibasel/downward
  - https://github.com/IBM/forbiditerative
  - https://github.com/IBM/kstar
  - https://github.com/speckdavid/symk
  - …