Automated Planning

Jacopo Mauro

Slide Based on Slides of the Artificial Intelligence: A Modern Approach book

Definition of Classical Planning

Classical planning is defined as the task of finding a sequence of actions to accomplish a goal in a discrete, deterministic, static, fully observable environment.

Planning Domain Definition Language (PDDL) is the de-facto language

- Basic PDDL can handle classical planning domains
- Extensions can handle non-classical domains that are continuous, partially observable, concurrent, and multi-agent.

State: represented as a conjunction of ground atomic fluents

Closed-world assumption: any fluent that are not mentioned is false

STRIPS Planning Language

- Developed for the Stanford Research Institute Problem Solver (STRIPS). Predates PDDL

Core Components:

- States: Represented as sets of propositional facts. {At(Home), Has(Car)}
- Actions (Operators): Defined preconditions and effects Example:

  Action: Drive(Home, Store)
  Preconditions: At(Home), Has(Car)
  Effects: ¬At(Home), At(Store)

- Goals: Desired conditions to achieve. Example: {At(Store)}
- Initial State

Algorithms for Classical Planning

Forward state-space search for planning

- Start at initial state
- Compute the set of actions in which preconditions are satisfied
- For all actions compute the next state applying effects
- Repeat until goal is found

Backward search also possible

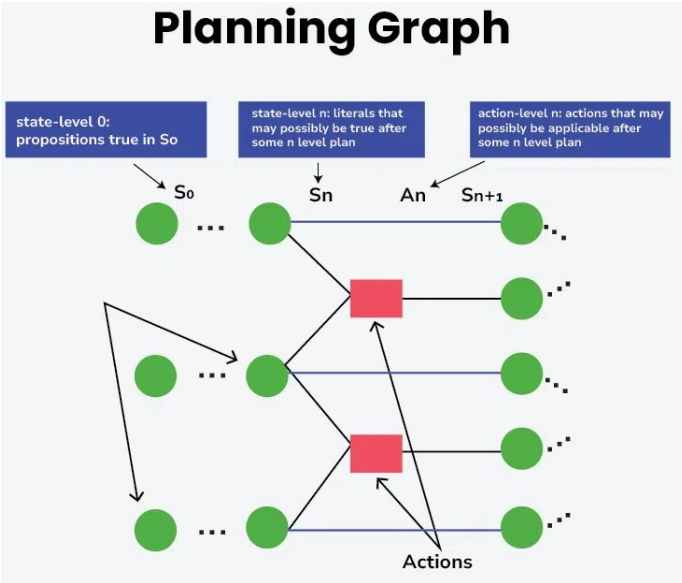
Encoding into SAT

Graph plan uses a specialized data structure, a planning graph

- Backward search with smaller search space

GraphPlan

- Forward Expansion: build graph level until all goal conditions appear in a proposition layer without mutual exclusions
- Mutexes
  - E.g., Drive(Home, Store) causes ¬At(Home), which conflicts with another action requiring At(Home)
- Backward Search: extract a valid plan by recursively selecting actions that achieve goals while avoiding mutexes
- Advantages
  - Reduces search space with compact structure
  - Provides a clear bound on plan length



Hierarchical Planning

A planning approach that breaks down complex tasks into simpler subtasks.

Mimics human problem-solving by addressing tasks at different levels of abstraction.

Key Characteristics

- Top-Down Approach: Starts with high-level goals and refines into detailed plans.
- Hierarchical Task Networks (HTN): Represents planning problems as tasks with dependencies and subtasks.

Why Use Hierarchical Planning?

- Efficiency: Reduces search space by focusing on abstract solutions first
  - Use task hierarchies to provide domain-specific knowledge about how to achieve goals
- Scalability: Handles large, complex domains effectively.

Example

High-Level Task: deliver(package, location_A, location_B)

Decomposed Tasks:

- Move the truck to the package's location.
- Load the package onto the truck.
- Drive the truck to the destination.
- Unload the package at the destination.

Example (e.g., SHOP2 or JSHOP)

```
;; Define the HTN domain
(defdomain transport-domain

  ;; Primitive task: Move the truck
  (:operator (!move-truck ?truck ?from ?to)
    ;; Preconditions
    ((at ?truck ?from))
    ;; Effects
    ((not (at ?truck ?from))
     (at ?truck ?to)))

  ;; Primitive task: Load the package
  (:operator (!load-package ?package ?truck ?location)
    ;; Preconditions
    ((at ?truck ?location)
     (at ?package ?location))
    ;; Effects
    ((not (at ?package ?location))
     (in ?package ?truck)))
```

...

```
;; High-level task: Delivering a package
(:method (deliver ?package ?from ?to)
  ;; Decomposition into subtasks
  ((at ?package ?from))               ;; Precondition
  ((!move-truck truck ?from)          ;; Subtasks
   (!load ?package truck ?from)
   (!drive truck ?from ?to)
   (!unload ?package truck ?to))))
```

```
(defproblem delivery-problem transport-domain
  ((at truck Inglewood)               ;; Initial state
   (at package Inglewood))
  ((at package Hollywood)))          ;; Goal state
```

Output

```
!move-truck(truck, location_A)
!load-package(package, truck, location_A)
!move-truck(truck, location_B)
!unload-package(package, truck, location_B)
```

History

Early Foundations (1950s–1960s)

- Influenced by work on state-space search, such as the *General Problem Solver (GPS)* by Newell and Simon (1957).
- Focus on formalizing intelligent behavior as sequences of operations.

STRIPS and Formalization (1970s)

Graphplan and SAT-based Planning (1990s)

- Graphplan introduced by Blum & Furst (1995) — fast planning using planning graphs.
- Encoding planning problems as **propositional satisfiability (SAT)** became practical and efficient.

Modern and Hierarchical Approaches (2000s–today)

- PDDL and Standardization (late 1990s)
- Growth of Hierarchical Task Network (HTN) planning.
- Advances in multi-agent, temporal, probabilistic, and partially observable planning.

Homework

Read Sections 11.1, 11.2, 11.4 (you can skip the hierarchical planning with angelic nondeterminism - pag 379)