Universidad de Alcalá

Planning Representation

ISG

Intelligent Systems Group

Overview

- **Introduction**
- Planning
- STRIPS
- PDDL
- Conclusions

Introduction

- Planning: selects a sequence of activities that meet one or more goals and a set of constraints imposed by the domain
- Scheduling: Resource allocation and start times of activities, obeying the temporal restrictions on the activities and the capacity limitations of shared resources. It is an optimization task where limited resources are available throughout the time between activities that can be run in serial or in parallel according to different objectives

Overview

- Introduction
- **Planning**
- STRIPS
- PDDL
- Conclusions

Planning

- Representations based on predicates and objects
- Each domain has a specific first-order language containing predicates and functions to describe the domain
- In planning, we define a set of operators that are a parametrized representation of the available transitions domain
- In scheduling, no specific languages

Planning

- Given a planning domain D, a classical planning problem
  - Inputs
    - Initial State
    - Goal(s)
    - Domain
  - State space
    - All states reachable by applying a sequence of actions to the initial state
    - Actions are operators with its parameters instantiated by constants of the initial state
  - Output
    - A sequence of actions that transform the initial state to a state that satisfies the goals

State Representation

- Planners decompose the world in terms of logical conditions and represent a state as a sequence of connected positive literals
  - Propositions: Fast $\wedge$ AirbusA380
  - FOL: At(Plane1,Madrid) $\wedge$ At(Plane2,Paris)
  - Not allowed: At(Padre(Ana),Madrid)
- It is assumed that all conditions that are not mentioned in a state are FALSE (**closed world assumption**)

Initial State

- It can be any logical description:
  - It is efficient to determine the true value of each atomic formula

$$\forall\, x,\ x = c_1 \vee x = c_2 \vee \ldots \vee x = c_n$$

- Implies that it is possible:
  - Determine in the initial state, the true value of the FOL
  - Represent the initial state as a set of propositions

Goals

- To represent a goal
  - A set of literals (*ground literals*)
  - A goal is satisfied if all literals are satisfied
  - Other possibilities:
    - A more complex condition in a state with a formula specified in FOL
    - The difficulty lies in creating efficient methods to find plans that satisfy these more complex goals

Operators

- Operators specify the possible transactions in a domain
- For each problem, we must use both the initial state and the specification of operators to determine possible transactions for the particular problem
- To be problem independent, operators use parameters

Operators

- Instantiated operator = action
- Specified in terms of::
  - preconditions: conditions that must be met before execution
    - All the variables in precond. should appear in the parameter list of the action
  - Effects: conditions that we achieve when the action is executed
    - What is not mentioned, remains unchanged
- Example:

Fly(plane(p,desde,hasta))

precond: En(p,desde) $\wedge$ plane(p) $\wedge$ aeropuerto(desde) $\wedge$ aeropuerto(hasta)

effect: En(p,hasta) $\wedge$ $\neg$En(p,desde)

Operators

- We say that an action is applicable in any state that satisfies its preconditions, otherwise the action has no effect
- The application consists of replacing the variables
- In general, given a FOL (with free variable) it is possible to determine the set of instances of these variables (constant) that satisfy the formula in the initial state

  En(IB101,Madrid) ∧ plane(IB101) ∧ plane(KLM101) ∧

  aeropuerto(Madrid) ∧ aeropuerto(Paris) ∧
  En(KLM101,Paris)

Overview

- Introduction
- Planning
- **STRIPS**
- PDDL
- Conclusions

STRIPS

- STRIPS is in the simplest and oldest representation of operators in AI
- The initial state is represented by a DB of positive facts
- STRIPS can be seen as a simple way to specify updates to the DB
- Little expressiveness for real domains so that other languages have emerged: ADL, Prodigy language (PDL), UCPOP language , SIPE-2 language ,…, PDDL (standar)

STRIPS

```
(def-strips-operator (pickup ?x)
   (pre (handempty) (clear ?x) (ontable ?x))
   (add (holding ?x))
   (del (handempty) (clear ?x) (ontable ?x)))
```

STRIPS

Operator name & parameters

```
(def-strips-operator (pickup ?x)
   (pre (handempty) (clear ?x) (ontable ?x))
   (add (holding ?x))
   (del (handempty) (clear ?x) (ontable ?x)))
```

STRIPS

```
(def-strips-operator (pickup ?x)
   (pre (handempty) (clear ?x) (ontable ?x))
```

List of predicates that must be true in the current state for aplying the action

```
   (add (holding ?x))
   (del (handempty) (clear ?x) (ontable ?x)))
```

STRIPS

```
(def-strips-operator (pickup ?x)
   (pre (handempty) (clear ?x) (ontable ?x))
   (add (holding ?x))
```

List of predicates that must be true in the next state

```
   (del (handempty) (clear ?x) (ontable ?x)))
```

STRIPS

```
(def-strips-operator (pickup ?x)
   (pre (handempty) (clear ?x) (ontable ?x))
   (add (holding ?x))
   (del (handempty) (clear ?x) (ontable ?x)))
   List of predicates that must be false
   in the next state
```

STRIPS

- Given the initial state
  - All instantiations of the `?x` parameter that satisfy the precondition

    ```
    (and (handempty) (clear ?x) (ontable ?x))
    ```

    produced a different action (transition) that can be applied to the initial state
  - Actions whose preconditions are not met are illegal transitions

STRIPS

- Actions are **determinististics**:
  - Given a particular instantiation of the parameters, the action specifies a finite collection of *ground atomic formula* that must be true and another collection of formulas to be false in the next state
- ¡Nothing else changes more! (This often has algorithmic consequences)

STRIPS

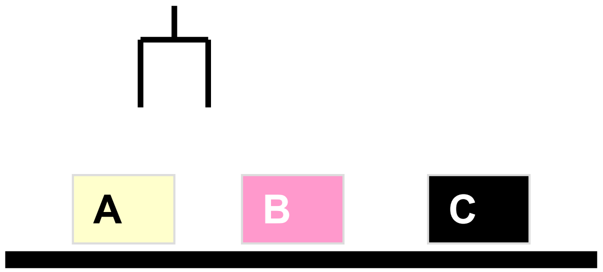
handempty
clear(A)
ontable(A)
clear(B)
ontable(B)
clear(C)
ontable(C)

pickup(A)

~~handempty~~
~~clear(A)~~
~~ontable(A)~~
holding(A)
clear(B)
ontable(B)
clear(C)
ontable(C)

pickup(B)

~~handempty~~
clear(A)
ontable(A)
~~clear(B)~~
~~ontable(B)~~
holding(B)
clear(C)
ontable(C)



Initial State

STRIPS

- From the properties of the initial state and the set of operators is possible to determine:
  - The finite set of actions that can be applied to the initial state
  - In each successor state generated by these actions, you can evaluate all logical formulas, and determine the set of available actions

Overview

- Introduction
- Planning
- STRIPS
- **PDDL**
- Conclusions

PDDL

- PDDL means Planning Domain Definition Language
- Emerges as an attempt to unify previous formalisms and to compare the efficiency of planners
- AIPS-98 Planning Competition Committee
- Intended to represent the physics of a domain: what predicates there are, what actions are possible, the structure of such actions and their effects
- New versions have currently emerged, actually in the 3.1

PDDL2.1 Variants (Wikipedia)

- **PDDL+:** provides a more flexible model of continuous changes through the use of autonomous processes and events
- **MAPL** (Multi-Agent Planning Language, pronounced "maple”)
  - Non-propositional state-variables (which may be n-ary: true, false, unknown, or anything else)
  - Temporal model given with modal operators (before, after, etc.).
  - Actions whose duration will be determined in runtime and explicit plan synchronization which is realized through speech act based communication among agents

PDDL2.1 Variants (Wikipedia)

- **OPT** (Ontology with Polymorphic Types):
  - An attempt to create a general-purpose notation for creating ontologies, defined as formalized conceptual frameworks for planning domains
- **PPDDL** (Probabilistic PDDL):
  - Probabilistic effects (discrete, general probability distributions over possible effects of an action)
  - Reward fluents (for incrementing or decrementing the total reward of a plan in the effects of the actions)
  - Goal rewards (for rewarding a state-trajectory, which incorporates at least one goal-state)
  - Goal-achieved fluents (which were true, if the state-trajectory incorporated at least one goal-state)

Variants (Wikipedia)

- **RDDL** (Relational Dynamic influence Diagram Language): the 7th IPC in 2011
  - Based on PPDDL1.0 and PDDL3.0, is a completely different language both syntactically and semantically
  - Introduces partial observability (allows efficient description of MDPs and POMDPs by representing everything with variables
- **NDDL** (New Domain Definition Language) is NASA's planning language based on HSTS (NRMA)
  - Use variable representation (timelines/activities) rather than a propositional/first-order logic, and
  - No concept of states or actions, only of intervals (activities) and constraints between activities

PDDL

- Domain definition

```
(define (domain briefcase-world)
  (:requirements :strips :equality :typing :conditional-effects)
  (:types location physob)
  (:constants (B - physob))
  (:predicates (at ?x - physob ?l - location)
               (in ?x ?y - physob))
```

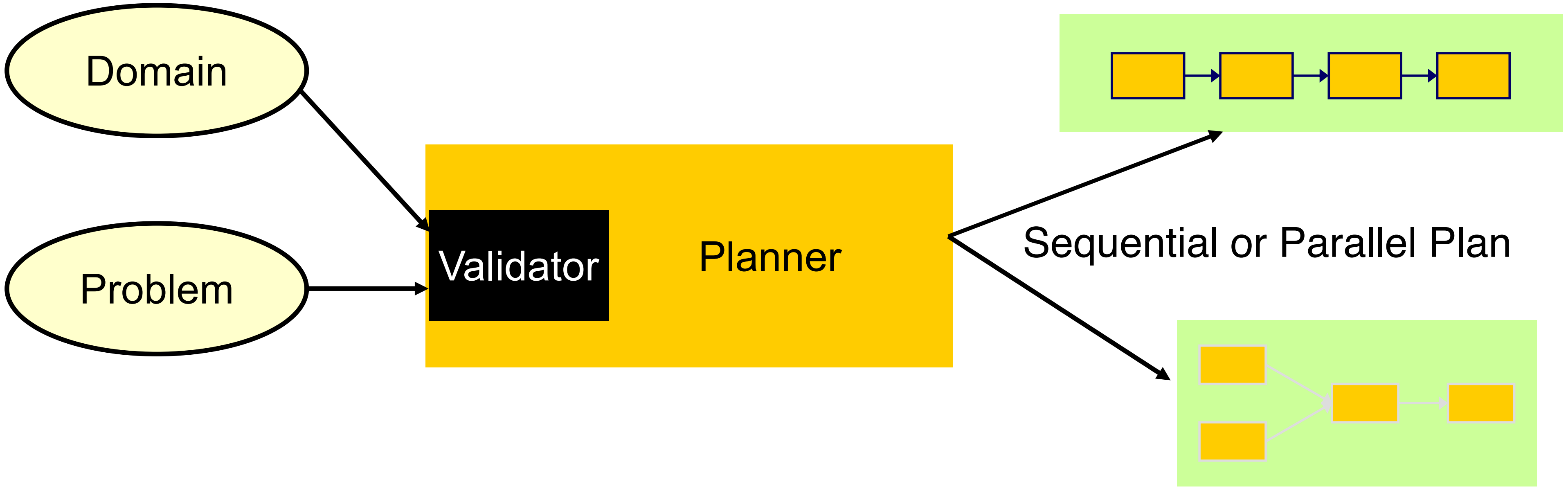
PDDL versions

- We will study:
  - PDDL
  - PDDL 2.X

Overview

Conclusions

- Represent knowledge → FOL
- Languages: STRIPS, ADL… PDDL (standar)
- Inputs & outputs of a planner


Domain
Problem
Validator
Planner
Sequential or Parallel Plan