Introduction to Automated Planning

Kyle Usbeck
http://www.cs.drexel.edu/~kfu22

Slides adapted from G. Wickler / A. Tate
Overview

• Introduction and Motivation
• Formalization
• Case Study
• Conclusion
What is Planning?

- **General:**
  - explicit deliberation process that chooses and organizes actions by anticipating their outcomes
- **Human:**
  - complex (or high-risk) tasks only
- **AI:**
  - computational study of the general deliberation process
Agents in Planning

- Task Assignment
- Planner
- Controller (Execution Agent)

Note: Control Loop
Types of Planning

• Domain-Dependent
  – use specific representations and techniques adapted to each problem
  – e.g. Path Planning

• Domain-Independent
  – uses generic action models
  – helps in study and design of autonomous intelligent machines
Motivation: Unix Sys Admin Aid

Volume groups example

This example has O-Plan produce a shell script for removing a volume group.

To remove a volume group with the vg_remove command, it is first necessary to remove all logical volumes from the group and all but one physical volume. To remove a logical volume, it is necessary to unmount any file system on it.

The volume group in the example is named vg0. The logical and physical volumes in the group have names lv1, lv2, ..., and pv1, pv2, ..., respectively. File systems have names fs1, fs2, ..., and we assume that they are mounted on the corresponding logical volumes (fs1 on lv1, fs2 on lv2, and so on).

The volume group contents:

2 logical volumes.
2 physical volumes.

Plan

1. Produce a plan to remove the volume group.
2. Close all changes to the form.

VG results

O-Plan version 3.35
Release date: 01-Aug-00
Build date: 16-Aug-00

Running statistics:

- in-cycles = 57
- in-alto-chosen = 2
- in-site-remaining = 3
- in-poisons = 2

Script:

#!/bin/sh
/usr/mount fs1
/usr/mount lvremove -f lv1
/usr/mount fs2
/usr/mount lvremove -f lv2
/usr/mount vgremove vg0 pv1
/usr/mount vgremove vg0 pv2
# physical volume pv0 will be removed automatically
/usr/shutdown
rm -rf vg0

The TG files
Mail a comment
Motivation: Evacuation Planning

Note: recall “Agents in Planning” – Task Assignment, Planner, Executor
Overview

• Introduction and Motivation
• Formalization
• Case Study
• Conclusion
Conceptual Model

- A **planning domain** is a **state-transition system**, a 4-tuple $\Sigma = (S,A,E,\gamma)$, where:
  - $S$=set of states;
  - $A$=set of actions (controlled by planner);
  - $E$=set of events (system dynamics); and
  - $\gamma: S \times A \times E \rightarrow 2^S$ is a state transition function.
Restrictive Assumptions

• A0: Finite $\Sigma$
• A1: Fully Observable $\Sigma$
• A2: Deterministic $\Sigma$
• A3: Static $\Sigma$
• A4: Restricted Goals
• A5: Sequential Plans
• A6: Implicit Time
• A7: Offline Planning
Restrictive Model

• Make Restrictive Assumptions (A0 – A7)

• Given a planning problem, \( P = (\Sigma, s_0, S_g) \), where:
  – \( s_0 \in S \) = initial state; and
  – \( S_g \subset S \) = set of possible goal states.

• find a sequence of actions \( \langle a_1, a_2, \ldots, a_k \rangle \)
  – corresponding to a sequence of state transitions \( \langle s_0, s_1, \ldots, s_k \rangle \)
  such that
  – \( s_1 = \gamma(s_0, a_1), \ s_2 = \gamma(s_1, a_2), \ldots, s_k = \gamma(s_{k-1}, a_k), \) and \( s_k \in S_g \).
Overview

• Introduction and Motivation
• Formalization
• Case Study
• Conclusion
An Example: Dock Worker Robots

- Dock Worker Robots (DWR) Domain ($\Sigma$):
  - harbour with several locations (docks), docked ships, storage areas for containers, and parking areas for trucks and trains
  - cranes to load and unload ships etc., and robot carts to move containers around
DWR Objects

- **locations** \{loc1, loc2, ...\}:
  - storage area, dock, docked ship, or parking or passing area
- **robots** \{robot1, robot2, ...\}:
  - container carrier carts for one container
  - can move between adjacent locations
- **cranes** \{crane1, crane2, ...\}:
  - belongs to a single location
  - can move containers between robots and piles at same location
- **piles** \{pile1, pile2, ...\}:
  - attached to a single location
  - pallet at the bottom, possibly with containers stacked on top of it
- **containers** \{cont1, cont2, ...\}:
  - stacked in some pile on some pallet, loaded onto robot, or held by crane
- **pallet**:
  - at the bottom of a pile
DWR Relations

Relations

- occupied($l$)
- at($r,l$)
- loaded($r,c$)
- unloaded($r$)
- holding($k,c$)
- empty($k$)
- in($c,p$)
- on($c,c'$)
- top($c,p$)

Object symbols

- $l = Location$
- $r = Robot$
- $c = Container$
- $k = Crane$
- $p = Pile$
DWR Actions

• **move** robot \( r \) from location \( l \) to some adjacent and unoccupied location \( l' \)
• **take** container \( c \) with empty crane \( k \) from the top of pile \( p \), all located at the same location \( l \)
• **put** down container \( c \) held by crane \( k \) on top of pile \( p \), all located at location \( l \)
• **load** container \( c \) held by crane \( k \) onto unloaded robot \( r \), all located at location \( l \)
• **unload** container \( c \) with empty crane \( k \) from loaded robot \( r \), all located at location \( l \)
DWR State-Transitions
DWR in Action

• Create DWR domain from States, Actions, and State-Transitions
• Define the Planning Problem with Initial State, and Goal State(s)
• Construct a Plan that solves the Planning Problem
• Execute the Plan on the real system
DWR Example Planning Problem

Initial state = $S_0$

Goal state(s) = \{$S_4$\}
Plan = { take( crane, cont. ); move( robot, location1 ); load( cont., crane, robot ) }
Overview

• Introduction and Motivation
• Formalization
• Case Study
• Conclusion
Summary

- Agents Involved in Planning
- Types of Planning
- Conceptual Model $\Sigma = (S,A,E,\gamma)$
- Restrictive Model $P = (\Sigma,s_0,S_g)$
  - Restrictive Assumptions
- DWR Domain Example
### Planning Research Areas/Techniques

| Domain Modelling | HTN, SIPE |
| Domain Description | PDDL, NIST PSL |
| Domain Analysis | TIMS |
| Plan Repair | O-Plan |
| Re-planning | O-Plan |
| Plan Monitoring | O-Plan, IPEM |
| Plan Generalisation | Macrops, EBL |
| Case-Based Planning | CHEF, PRODIGY |
| Plan Learning | SOAR, PRODIGY |
| Plan Advice | SRI/Myers |
| Mixed-Initiative Plans | TRIPS/TRAINS |
| User Interfaces | SIPE, O-Plan |
| Graph Planning Algorithms | GraphPlan |
| Search Methods | Heuristics, A* |
| Partial-Order Planning | Nonlin, UCPOP |
| Plan Generalisation | Macrops, EBL |
| Hierarchical Planning | NOAH, Nonlin, O-Plan |
| Case-Based Planning | CHEF, PRODIGY |
| Refinement Planning | Kambhampati |
| Plan Learning | SOAR, PRODIGY |
| Opportunistic Search | OPM |
| Constraint Satisfaction | CSP, OR, TMMS |
| Plan Advice | SRI/Myers |
| Mixed-Initiative Plans | TRIPS/TRAINS |
| Issue/Flaw Handling | O-Plan |
| Planning Web Services | O-Plan, SHOP2 |
| Plan Analysis | NOAH, Critics |
| Plan Sharing & Comms | I-X, <I-N-C-A> |
| Plan Simulation | QinetiQ |
| NL Generation ... | ... |
| Plan Qualitative Modelling | Excalibur |
| Dialogue Management | ... |

**Deals with whole life cycle of plans**

**Problem is to make sense of all these techniques**
Thank you