Intro to CS 541 (AI Planning)

http://www.isi.edu/~blythe/cs541

Jim Blythe
Jose Luis Ambite
Yolanda Gil
Syllabus outline (roughly)

- Techniques for generating plans (September)
- Representation for plans and time (end of Sept.)
- Controlling search (October)
- Planning and uncertainty (November, 1\textsuperscript{st} half)
- Applications (November, 2\textsuperscript{nd} half)
How grades will be assigned

- Homeworks: 30%
- Exams: 30%
  - mid-term and final
- Project: 30%
- Class participation and quizzes: 10%
Class projects

- We will put some suggested projects on the web page
  - Develop new features for or application of a planning system, e.g. constraint satisfaction planning on the grid
  - Investigate several planning systems on a class of domains
  - Relevant projects connected to your own research

- Talk to us about what you’d like to do.

- Initial proposals are due Sep 30th (~ 1 page)
  We will give feedback in first week of October
  Final proposals due Oct 9th

- Project presentations: Nov 25th and Dec 2nd, a written report due December 12th.
- AAAI deadline is Jan 20.
Sign-up sheet

- Please add your name and email address to our sign-up sheet.

  Jim Blythe, blythe@isi.edu
  Yolanda Gil, gil@isi.edu
  Jose Luis Ambite, ambite@isi.edu
Generating plans

- **Given:**
  - A way to describe the world
  - An initial state of the world
  - A goal description
  - A set of possible actions to change the world

- **Find:**
  - A prescription for actions to change the initial state into one that satisfies the goal
Applications

- Mobile robots
  - An initial motivator, and still being developed

- Simulated environments
  - Goal-directed agents for training or games

- Web and grid environments
  - Composing queries or services
  - Workflows on a computational grid

- Managing crisis situations
  - E.g. oil-spill, forest fires, urban evacuation, in factories, …

- And many more
  - Factory automation, flying autonomous spacecraft, playing bridge, military planning, …
Representing change

As actions change the world OR we consider possible actions, we want to:

- Know how an action will alter the world
- Keep track of the history of world states (have we been here before?)
- Answer questions about potential world states (what would happen if..?)
The situation calculus (McCarthy 63)

- Key idea: represent a snapshot of the world, called a ‘situation’ explicitly.

- ‘Fluents’ are statements that are true or false in any given situation, e.g. ‘I am at home’

- Actions map situations to situations.
S0

\( \text{go(store)} \)

S1

\( \neg \text{holds(at(home), S1)} \)
\( \text{holds(at(store), S1)} \)

S1 = \text{result(go(store), S0)}

S2

\( \text{mow_lawn()} \)

\( \text{holds(at(home), S0)} \)
\( \text{holds(color(door, red), S0)} \)
Frame problem

- I go from home to the store, creating a new situation $S'$. In $S'$:
  - My friend is still at home
  - The store still sells chips
  - My age is still the same
  - Los Angeles is still the largest city in California…

- How can we efficiently represent everything that hasn’t changed?
Successor state axioms

- Normally, things stay true from one state to the next -- unless an action changes them:

  holds(at(X), result(A,S)) iff A = go(X)
  or [holds(at(X), S) and A != go(Y)]

- We need one or more of these for every fluent.

- Now we can use theorem proving to deduce a plan.

- Class dismissed!
Well, not quite..

- Theorem proving can be really inefficient for planning

Strips (Fikes and Nilsson 71)

- For efficiency, separates theorem-proving within a world state from searching the space of possible states.

- Highly influential representation for actions:
  - Preconditions (list of propositions to be true)
  - Delete list (list of propositions that will become false)
  - Add list (list of propositions that will become true)
Example problem:

Initial state: at(home), ¬ have(beer), ¬ have(chips)
Goal: have(beer), have(chips), at(home)

Actions:

Buy (X):
  Pre: at(store)
  Add: have(X)

Go (X, Y):
  Pre: at(X)
  Del: at(X)
  Add: at(Y)
Frame problem (again)

- I go from home to the store, creating a new situation S’. In S’:
  - The store still sells chips
  - My age is still the same
  - Los Angeles is still the largest city in California…

- How can we efficiently represent everything that hasn’t changed?
  - Strips provides a good solution for simple actions
Ramification problem

- I go from home to the store, creating a new situation $S'$. In $S'$:
  - I am now in Marina del Rey
  - The number of people in the store went up by 1
  - The contents of my pockets are now in the store..

- Do we want to say all that in the action definition?
Solutions to the ramification problem

- In Strips, some facts are inferred within a world state,
  - e.g. the number of people in the store

- ‘primitive’ facts, e.g. at(home) persist between states unless changed. ‘inferred’ facts are not carried over and must be re-inferred.
  - Avoids making mistakes, perhaps inefficient.
Questions about Strips

- What would happen if the order of goals was at(home), have(beer), have(chips) ?

- When Strips returns a plan, is it always correct? efficient?

- Can Strips always find a plan if there is one?
Example: blocks world (Sussman anomaly)

Initial:

Goal:

State I: (on-table A) (on C A) (on-table B) (clear B) (clear C)

Goal: (on A B) (on B C)
Noah (Sacerdoti 75)

- Explicitly views plans as a partial order of steps. Add ordering into the plan as needed to guarantee it will succeed.

- Avoids the problem in Strips, that focusing on one subgoal forces the actions that resolve that goal to be contiguous.
Nets Of Action Hierarchies

\[ \text{on}(a, b) \]
\[ \text{on}(b, c) \]

\[ \text{clear}(a) \]
\[ \text{clear}(b) \]
\[ \text{clear}(c) \]

\[ \text{puton}(a, b) \]
\[ \text{puton}(b, c) \]
Nets Of Action Hierarchies

S

on(a, b)

S

on(b, c)

J

S

clear(a)

clear(b)

J

puton(a, b)

S

clear(b)

clear(c)

J

puton(b, c)
Resolve conflicts ‘critic’:
puton(a, b)

puton(b, c)

clear(a)
clear(b)
clear(c)
26

USC INFORMATION SCIENCES INSTITUTE

Intro to Planning

puton(a, b)

S

clear(a)  puton(b, c)

J

S

clear(b)  clear(c)  puton(c, X)

J

clear(c)

S

clear(b)  clear(c)  puton(b, c)

J

puton(a, b)

J

puton(a, b)
Final plan

\[
\text{clear}(c) \quad \text{puton}(c, X) \quad \text{puton}(b, c) \quad \text{puton}(a, b)
\]

\[
\text{clear}(b)
\]
Strips and Noah: assumptions and discussion points