STRIPS: A More Efficient Planner

Alan Bundy

School of Informatics

(slides courtesy of Robert Wilensky)

Planning

- Idea: Use special purpose representations and algorithms for more efficient plan production.
  - Representation for situations, actions, goals, plans
  - Algorithms for searching
    - Situation space
    - Plan space

Planning and Acting

- Want algorithms for
  - Making a plan
  - Carrying out the plan
  - Monitoring the execution
    - Why? Plan might not work as expected; world might change
  - Learning from experience

The `STRIPS' Approach

- Represent a situation as knowledge base of (restricted) logical sentences.
- **Represent operators by how they should change the KB.**
- Reason about the implications of actions by changing the KB.
  - reversibly, so we can back-track cheaply.

Representation in STRIPS

- Situations
  - Conjunctions of ground literals (no functions)
    - Systems differ on whether this is supposed to be a complete or partial representation of the world
  - E.g.: **At(Box1,B)** $\wedge$ **At(Box2,C)**
  - Of course, we still have general facts about the world that might be represented using implication, etc., but these won't change from situation to situation.
- Goals
  - Same thing, but allow variables (interpreted existentially)
  - E.g., **At(Box1,x)** $\wedge$ **At(Box2,x)**

Representing Operators in STRIPS

- Action or Operator Schemata
- Components
  - Action description: A way to name the action
  - **Add list**: What becomes true after the action
  - **Delete list**: What ceases to become true after the action
    - Add and Delete lists are sometimes combined into a single **Effects** list.
  - **Precondition**: What must be true to undertake the action.
- We can only execute actions, meaning a fully instantiated schema.

STRIPS Operator (Schema) Example

- Go to one location from another.
  - Action Description: **Goto(m,n)**
  - Add: **At(Robot,n)**
  - Del: **At(Robot,m)**
  - Pre: **At(Robot,m)**
- Pushing a box from one location to another.
  - Action Description: **Push(k,m,n)**
  - Add: **At(Robot,n)** $\wedge$ **At(k,n)**
  - Del: **At(Robot,m)** $\wedge$ **At(k,m)**
  - Pre: **At(Robot,m)** $\wedge$ **At(k,m)** $\wedge$ **Box(k)**
- In general, preconditions might contain more information.

Using Operators to Search Situation Space

- Search the *space of situations*, which is connected by operator instances, for a situation, accessible from the initial situation, in which the goal pertains.
- The sequence of operator instances is the plan.
- Simulate applying an operator instance by changing the knowledge base.
- In principle, one could use any number of different search strategies to find a plan.
  - Forward from initial state
  - Backward from goal state
  - Other

(What STRIPS Actually Did)

- STRIPS used GPS as a search algorithm:
- Newell and Simon's `General Problem Solver'
  - Determine difference between situation and goal
  - Select an operator that reduces the difference
  - Apply the operator
  - Reconsider goal in new situation

STRIPS Algorithm (con't)

- Try to prove the goal in a situation being considered.
  - STRIPS used resolution
- If fail, use incomplete proof as the difference.
- Use Add and Del lists of operators to pick on that may help the proof continue.
  - Heuristic: Look for operator that may help resolve something.
- Turn the preconditions of the operators into new subgoals.
  - Recursively attempt to achieve them.
- Apply operator by modifying the KB.
  - Can reversibly modify the KB, or use Add list as additions and Del list as filters without changing KB.

STRIPS and the Frame Problem

- The computational part of the frame problem is now addressed. How?
  - Everything in initial situation is assumed to remain true unless it is mentioned on an Add or Del list.
  - Add, Del lists are relatively small compared to the KB, reflecting the fact that operators don't change much in the world.
  - I.e., moving from one situation to another is essentially linear in number of operators.

Very Simple STRIPS Example

- Initial Situation: **At(Robot,A)**
- Goal: **At(Robot,B)**
- Operators:

Goto(m,n)

Add: **At(Robot,n)**

Del: **At(Robot,m)**

Pre: **At(Robot,m)**

Push(k,m,n)

Add: **At(Robot,n)** $\wedge$ **At(k,n)**

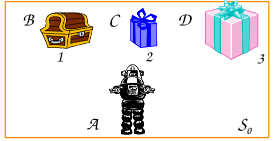
Del: **At(Robot,m)** $\wedge$ **At(k,m)**

Pre: **At(Robot,m)** $\wedge$ **At(k,m)** $\wedge$ **Box(k)**

Simple STRIPS Example Goal: **At(Robot,B)**

- Add negated goal **¬At(Robot,B)** to KB.
- Proof attempt fails, leaving **¬At(Robot,B)** to resolve away.
- Look for operator to reduce difference, i.e., complete proof.
- **Goto(m,n)** looks like it will help the proof.
  - because it adds **At(Robot,n)**, which unifies with negation of **¬At(Robot,B)**, suggesting the (partially instantiated) operator **Goto(m,B)**.
  - Actually, so does **Push(k,m,n)**; heuristic helps select former .
- Precondition becomes new subgoal: **At(Robot,m)**
- Can prove this, because **At(Robot,A)**
- Can now execute (fully instantiated operator) **Goto(A,B)**.
- Simulate execution: Hack KB by adding Adds (**At(Robot,B)**) and deleting Dels (**At(Robot,A)**).
- Afterwards, KB contains **At(Robot,B)** (and not **At(Robot,A)**).
- Now can complete proof, so done!
- Plan is **Goto(A,B)**.

More Interesting STRIPS Example



- Initial Situation (**S_0**):

  Box(Box1) ∧ Box(Box2) ∧ Box(Box3)

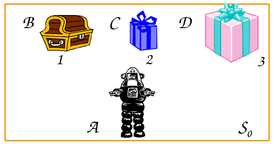
  ∧ At(Robot,A)

  ∧ At(Box1,B) ∧ At(Box2,C) ∧ At(Box3,D)
- Initial Goal (**G_0**):

  ∃x At(Box1,x) ∧ At(Box2,x) ∧ At(Box3,x)

  (I.e., all three boxes should be at the same place.)
- For this example, we'll use the notation **(S_i (G_n G_{n-1} ...))** to designate being in situation **S_i** and have goal stack **(G_n G_{n-1} ...)**.
- So the initial state of affairs is **(S_0 (G_0))**.

STRIPS Example (con't)



- Negated goal in clausal form:

  {¬At(Box1,x),¬At(Box2,x),¬At(Box3,x)}
- Resolve with any of the box location predications:

  {¬At(Box1,x),¬At(Box2,x),¬At(Box3,x)}

  {At(Box1,B)}

  {¬At(Box2,B),¬At(Box3,B)}
  - This is our incomplete proof.
- Select operator that might let proof continue.
  - Add list of **Push(k,m,n)** contains **At(k,n)**, which looks promising:

  {¬At(Box2,B),¬At(Box3,B)}

  {At(k,n)}

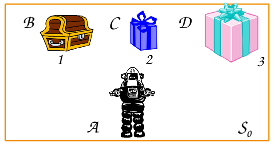
  {¬At(Box3,B)}

  with unifier **{k/Box2,n/B}**

  giving us (partially instantiated) operator **OP_1: Push(Box2,m,B)**

  Moreover, Del list is **At(Box2,m) ∧ At(Robot,m)**; nothing resembling either conjunct is used in the proof so far.

STRIPS Example (con't)



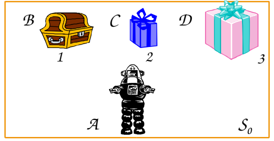
- New goal is precondition of **Push(Box2,m,B)**:

  G_1: At(Box2,m) ∧ At(Robot,m) ∧ Box(Box2)

  State of affairs is **(S_0 (G_1,G_0))**
- Negated goal is

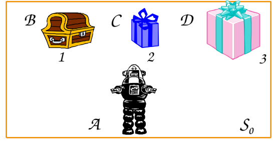
  {¬At(Box2,m),¬At(Robot,m), ¬Box(Box2)}
  - In KB is **Box(Box2)**, so can cancel out **¬Box(Box2)**.
  - Since **At(Robot,A)**, can deduce **{¬At(Box2,A)}**.
  - Since **At(Box2,C)**, can deduce **{¬At(Robot,C)}**.
- Stuck; need strategy for operator selection.
  - First suggests pushing **Box2** again, a bad idea.
  - Second suggests **Goto** operator; less objectionable.
- Specifically:
  - Operator **OP_2: Goto(m,C)**
  - Precondition **At(Robot,m)** becomes new goal **G_2**.
  - State of affairs is **(S_0 (G_2,G_1,G_0))**

STRIPS Example (con't)

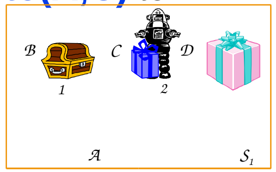


- But we can prove G_2: **At(Robot,m)**, because **At(Robot,A)** is true.
- Apply (fully instantiated) OP_2: **Goto(A,C)** to produce new situation, S_1

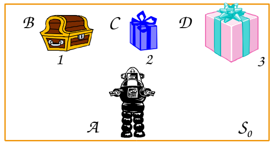
STRIPS Example (con't)



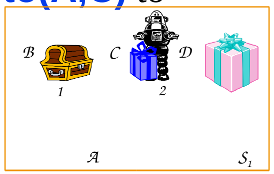
- But we can prove G_2: **At(Robot,m)**, because **At(Robot,A)** is true.
- Apply (fully instantiated) OP_2: **Goto(A,C)** to produce new situation, S_1



STRIPS Example (con't)

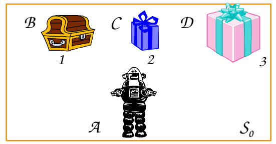


- But we can prove G_2: **At(Robot,m)**, because **At(Robot,A)** is true.
- Apply (fully instantiated) OP_2: **Goto(A,C)** to produce new situation, S_1
  - *Add* **At(Robot,C)**
  - *Delete* **At(Robot,A)**
  - *State of affairs is* **(S_1 (G_1,G_0))**

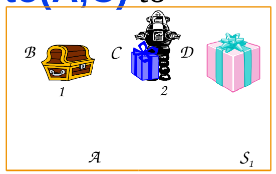


- *Now we can prove* G_1*, and hence apply* OP_1 *getting situation* S_2.

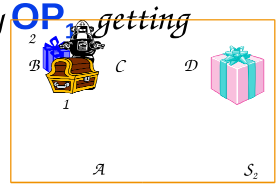
STRIPS Example (con't)



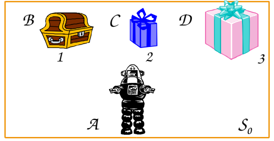
- But we can prove G_2: **At(Robot,m)**, because **At(Robot,A)** is true.
- Apply (fully instantiated) OP_2: **Goto(A,C)** to produce new situation, S_1
  - *Add* **At(Robot,C)**
  - *Delete* **At(Robot,A)**
  - *State of affairs is* **(S_1 (G_1,G_0))**



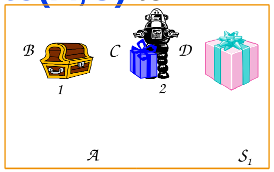
- *Now we can prove* G_1*, and hence apply* OP_1 *getting situation* S_2.



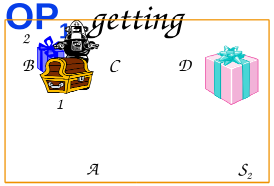
STRIPS Example (con't)



- But we can prove **G_2: At(Robot,m)**, because **At(Robot,A)** is true.
- Apply (fully instantiated) **OP_2: Goto(A,C)** to produce new situation, **S_1**

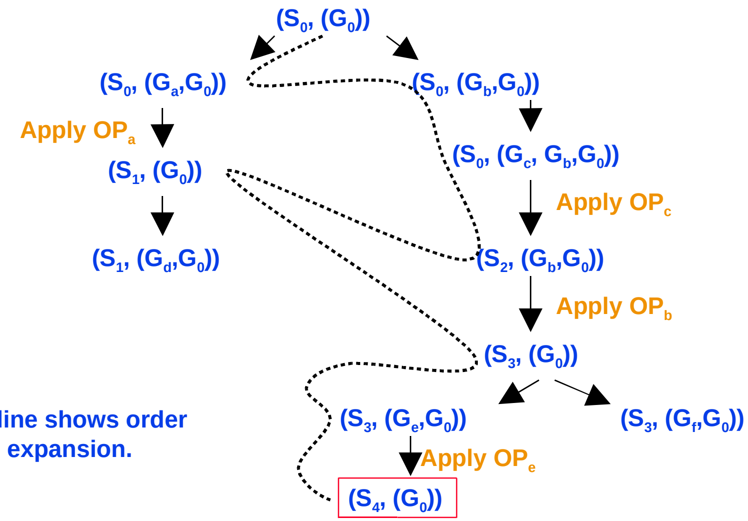


  - *Add* **At(Robot,C)**
  - *Delete* **At(Robot,A)**
  - *State of affairs is* **(S_1 (G_1,G_0))**
- *Now we can prove* **G_1**, *and hence apply* **OP_1**, *getting situation* **S_2**.
  - *State of affairs is* **(S_2 (G_0))**



- *And so on.*

STRIPS Searches a tree



Dotted line shows order of node expansion.

STRIPS Running Time

- STRIPS spent most of its time in the theorem prover:

	Time	TP-time	No. of nodes in sol.	No. of nodes in search tree	No. of operators in sol.	No. of operators in search tree
3 boxes	66	49.6	9	9	4	4
M&B	113	83	13	21	6	6

- It didn't necessarily find a good solution.
  - For M&B problem: get on box, get off box, push box under bananas, get on box, get bananas
- It didn't guarantee finding a solution.
- It was inefficient for handling conjunctive subgoals.
  - We'll return to this problem later.

Summary

- It is more efficient to represent actions by how they modify a KB, than by using the purely logical situation calculus.
- We can do so using a STRIPS-style operator representation.
- This uses add and delete lists.
- STRIPS avoids the need for frame axioms.