CS 1571 Introduction to AI
Lecture 17

Modeling time and actions
Planning

Milos Hauskrecht
milos@cs.pitt.edu
5329 Sennott Square

Administration announcements

Midterm:
- **Tuesday, October 28, 2014**
- **In-class**
- **Closed book**

What does it cover?
- **All material covered by the end of lecture today**

Representation of actions, situations, events

Propositional and first order logic are monotonic

- Once something is true it cannot become false

But, the world is dynamic:

- What is true now may not be true tomorrow
- Changes in the world may be triggered by our activities

Problems:

- How to represent the change in the FOL ?
- How to represent actions we can use to change the world?

Planning

Planning problem:

- find a sequence of actions that achieves some goal
- an instance of a search problem
- the state description is typically **very complex and relies on a logic-based representation**

Methods for modeling and solving planning problems:

- State space search
- Situation calculus based on FOL
- STRIPS – state-space search algorithm based on restricted FOL
- Partial-order planning algorithms

Situation calculus

Provides a framework for representing change, actions and for reasoning about them

- **Situation calculus**
  - based on the **first-order logic**,
  - a **situation variable** models possible states of the world
  - properties and relations depend on different world states (situations)
  - **action objects** model activities
- **Inference:**
  - inference methods developed for FOL to do the reasoning

Situation calculus

Logic for reasoning about changes in the state of the world

The world dynamics is described by:

- Sequences of **situations** of the current state
- Changes from one situation to another are **caused by actions**

The situation calculus allows us to:

- Describe the **initial state and the goal state**
- Build the **KB that describes the effect of actions** (operators)
- Prove that the KB and the initial state can lead to the goal state
  - extracts a plan (sequence of actions) as side-effect of the proof

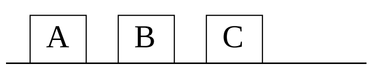
Situation calculus

The language is based on the First-order logic plus:

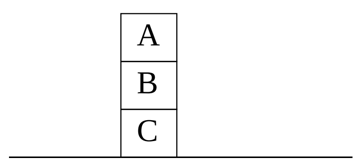
- **Special variables:** s,a – objects of type situation and action
- **Action functions:** return actions (action objects).
  - E.g. $Move(A, TABLE, B)$ represents a move action
  - $Move(x,y,z)$ represents an action schema
- **Special function symbols of type situation**
  - s_0 – initial situation
  - $DO(a,s)$ – represents the situation obtained after performing action a in situation s
- **Situation-dependent predicates, functions**

  (also called **fluents**)
  - **Relation:** $On(x,y,s)$ – object x is on object y in situation s;
  - **Function:** Above(x,s) – object that is above x in situation s.

Situation calculus. Blocks world example.



Initial state



Goal

$On(A,Table, s_0)$
$On(B,Table, s_0)$
$On(C,Table, s_0)$
$Clear(A, s_0)$
$Clear(B, s_0)$
$Clear(C, s_0)$
$Clear(Table, s_0)$

Find a state (situation) s, such that

$On(A,B, s)$
$On(B,C, s)$
$On(C,Table, s)$

Knowledge base: Axioms

Knowledge base is needed to support the reasoning:

- Must represent changes in the world due to actions.

Two types of axioms:

- **Effect axioms**
  - changes in situations that result from actions
- **Frame axioms**
  - things preserved from the previous situation

Example:

- blocks world with *On, Clear* predicates
- *Move* actions

Blocks world example. Effect axioms.

Effect axioms:

- **represent the changes after the action is executed**

Moving x from y to z. $MOVE(x,y,z)$

Effect of move changes on **On** relations

$$On(x,y,s) \wedge Clear(x,s) \wedge Clear(z,s) \rightarrow On(x,z,DO(MOVE(x,y,z),s))$$

$$On(x,y,s) \wedge Clear(x,s) \wedge Clear(z,s) \rightarrow \neg On(x,y,DO(MOVE(x,y,z),s))$$

Effect of move changes on **Clear** relations

$$On(x,y,s) \wedge Clear(x,s) \wedge Clear(z,s) \rightarrow Clear(y,DO(MOVE(x,y,z),s))$$

$$On(x,y,s) \wedge Clear(x,s) \wedge Clear(z,s) \wedge (z \neq Table)$$
$$\rightarrow \neg Clear(z,DO(MOVE(x,y,z),s))$$

Blocks world example. Frame axioms.

- **Frame axioms.**
  - **Represent relations/properties that remain unchanged by the executed action**
  - Explicitly move the relations to the next situation (after the action)

On relations:

$$On(u,v,s) \wedge (u \neq x) \wedge (v \neq y) \rightarrow On(u,v,DO(MOVE(x,y,z),s))$$

Clear relations:

$$Clear(u,s) \wedge (u \neq z) \rightarrow Clear(u,DO(MOVE(x,y,z),s))$$

Planning in situation calculus

Planning problem:

- find a sequence of actions that lead to the goal

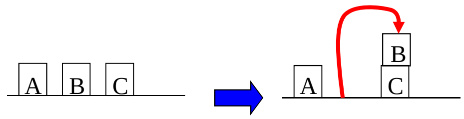
Planning in situation calculus is converted to the theorem proving problem

Goal state:

$$\exists s\ On(A,B,s) \wedge On(B,C,s) \wedge On(C,Table,s)$$

- Possible inference approaches:
  - **Inference rule approach**
  - **Conversion to SAT**
- **Plan** (solution) is a byproduct of theorem proving
- **Example:** blocks world

Planning in the blocks world.



Initial state (s0)   s1

$s_0 =$

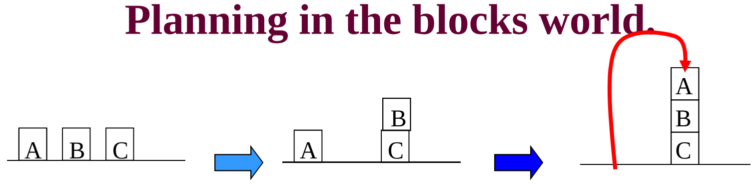
$On(A, Table, s_0)$ $Clear(A, s_0)$ $Clear(Table, s_0)$
$On(B, Table, s_0)$ $Clear(B, s_0)$
$On(C, Table, s_0)$ $Clear(C, s_0)$

Action: $MOVE(B, Table, C)$

$s_1 = DO(MOVE(B, Table, C), s_0)$

$On(A, Table, s_1)$ $Clear(A, s_1)$ $Clear(Table, s_1)$
$On(B, C, s_1)$ $Clear(B, s_1)$
$\neg On(B, Table, s_1)$ $\neg Clear(C, s_1)$
$On(C, Table, s_1)$

Planning in the blocks world.



Initial state (s0)   s1   s2

$s_1 = DO(MOVE(B, Table, C), s_0)$

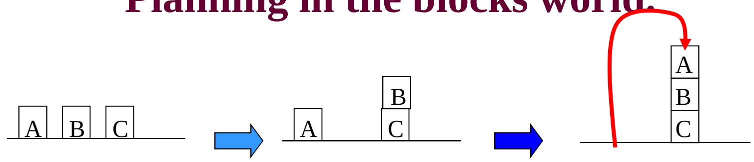
$On(A, Table, s_1)$ $Clear(A, s_1)$ $Clear(Table, s_1)$
$On(B, C, s_1)$ $Clear(B, s_1)$
$\neg On(B, Table, s_1)$ $\neg Clear(C, s_1)$
$On(C, Table, s_1)$

Action: $MOVE(A, Table, B)$

$s_2 = DO(MOVE(A, Table, B), s_1)$
$= DO(MOVE(A, Table, B), DO(MOVE(B, Table, C), s_0))$

$On(A, B, s_2)$ $\neg On(A, Table, s_2)$ $\neg Clear(B, s_2)$
$On(B, C, s_2)$ $\neg On(B, Table, s_2)$ $\neg Clear(C, s_2)$
$On(C, Table, s_2)$ $Clear(A, s_2)$ $Clear(Table, s_2)$

Planning in the blocks world.



Initial state (s0) **s1** **s2**

$s_1 = DO(MOVE(B, Table, C), s_0)$

$On(A, Table, s_1)$
$On(B, C, s_1)$
$\neg On(B, Table, s_1)$
$On(C, Table, s_1)$

$Clear(A, s_1)$
$Clear(B, s_1)$
$\neg Clear(C, s_1)$

$Clear(Table, s_1)$

Action: $MOVE(A, Table, B)$

$s_2 = DO(MOVE(A, Table, B), s_1)$
$= DO(MOVE(A, Table, B), DO(MOVE(B, Table, C), s_0))$

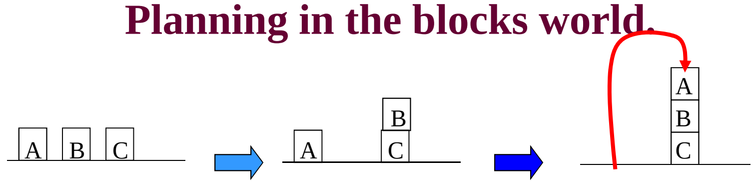
Satisfies the goal

$\boxed{On(A, B, s_2) \quad On(B, C, s_2) \quad On(C, Table, s_2)}$

$\neg On(A, Table, s_2)$
$\neg On(B, Table, s_2)$
$Clear(A, s_2)$

$\neg Clear(B, s_2)$
$\neg Clear(C, s_2)$
$Clear(Table, s_2)$

Planning in the blocks world.



Initial state (s0) **s1** **s2**

$s_1 = DO(MOVE(B, Table, C), s_0)$

$On(A, Table, s_1)$
$On(B, C, s_1)$
$\neg On(B, Table, s_1)$
$On(C, Table, s_1)$

$Clear(A, s_1)$
$Clear(B, s_1)$
$\neg Clear(C, s_1)$

$Clear(Table, s_1)$

Action: $MOVE(A, Table, B)$

DO functions capture the plan

$s_2 = DO(MOVE(A, Table, B), s_1)$
$= \boxed{DO(MOVE(A, Table, B), DO(MOVE(B, Table, C), s_0))}$

$On(A, B, s_2)$
$On(B, C, s_2)$
$On(C, Table, s_2)$

$\neg On(A, Table, s_2)$
$\neg On(B, Table, s_2)$
$Clear(A, s_2)$

$\neg Clear(B, s_2)$
$\neg Clear(C, s_2)$
$Clear(Table, s_2)$

Situation calculus: problems

Frame problem refers to:

- The need to represent a large number of frame axioms

Solution: combine positive and negative effects in one rule

$$On(u,v,DO(MOVE(x,y,z),s)) \Leftrightarrow (\neg((u=x)\wedge(v=y))\wedge On(u,v,s))\vee$$
$$\vee(((u=x)\wedge(v=z))\wedge On(x,y,s)\wedge Clear(x,s)\wedge Clear(z,s))$$

Inferential frame problem:

- We still need to derive properties that remain unchanged

Other problems:

- **Qualification problem** – enumeration of all possibilities under which an action holds
- **Ramification problem** – enumeration of all inferences that follow from some facts

Planning problems

Properties of many (real-world) planning problems:

- The description of the state of the world is very **complex**
- **Many possible** actions to apply in any step
- Actions are typically **local**
  - \- they affect only a small portion of a state description
- Goals are defined as conditions referring only to a small portion of state
- Plans consists of a large number of actions

The situation calculus framework:

- too cumbersome and inefficient to represent and solve the planning problems

Solutions

- **Complex state description and local action effects:**
  - avoid the enumeration and inference of every state component, focus on changes only
- **Many possible actions:**
  - Apply actions that make progress towards the goal
  - Understand what the effect of actions is and reason with the consequences of these action
- **Sequences of actions in the plan can be too long:**
  - Many goals consists of independent or nearly independent sub-goals
  - Allow goal decomposition & divide and conquer strategies

STRIPS planner

Defines a **restricted representation language** when compared to the situation calculus

Advantage: leads to more efficient planning algorithms.

- State-space search with structured representations of states, actions and goals
- Action representation avoids the frame problem

STRIPS planning problem:

- much like a standard search problem
- State descriptions use first order logic

STRIPS planner

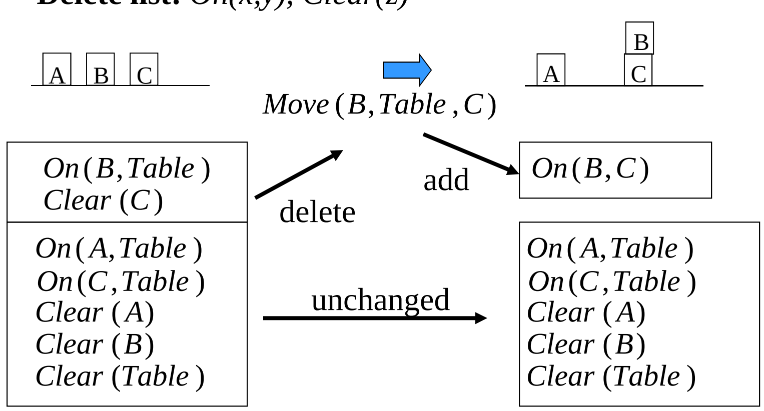
- **States:**
  - **conjunction of literals**, e.g. *On(A,B), On(B,Table), Clear(A)*
  - represent facts that are true at a specific point in time
- **Actions (operators):**
  - **Action:** *Move (x,y,z)*
  - **Preconditions:** conjunctions of literals with variables

    On(x,y), Clear(x), Clear(z)
  - **Effects.** Two lists:
    - **Add list:** *On(x,z), Clear(y)*
    - **Delete list:** *On(x,y), Clear(z)*
    - Everything else remains untouched (is preserved)

STRIPS planning

Operator: *Move (x,y,z)*

- **Preconditions:** *On(x,y), Clear(x), Clear(z)*
- **Add list:** *On(x,z), Clear(y)*
- **Delete list:** *On(x,y), Clear(z)*



STRIPS planning

Initial state:

- Conjunction of literals that are true

Goals in STRIPS:

- A goal is a **partially specified state**
- Is defined by a **conjunction of ground literals**
  - No variables allowed in the description of the goal

Example:

$On(A,B) \quad \wedge \quad On(B,C)$

Search in STRIPS

Objective:

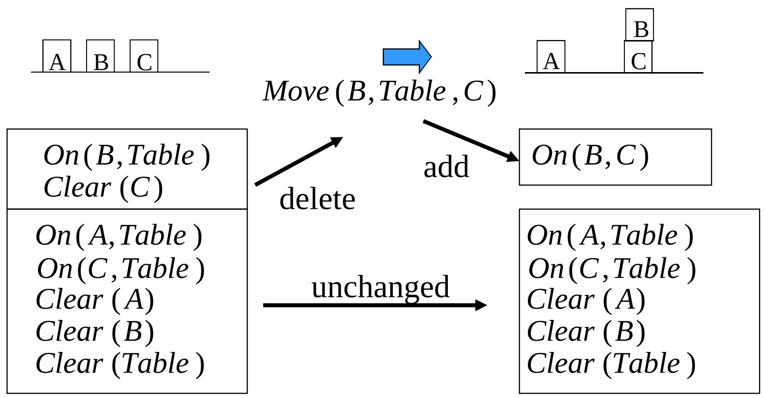
Find a sequence of operators (a plan) from the initial state to the state satisfying the goal

Two approaches to build a plan:

- **Forward state space search (goal progression)**
  - Start from what is known in the initial state and apply operators in the order they are applied
- **Backward state space search (goal regression)**
  - Start from the description of the goal and identify actions that help us to reach the goal

Forward search (goal progression)

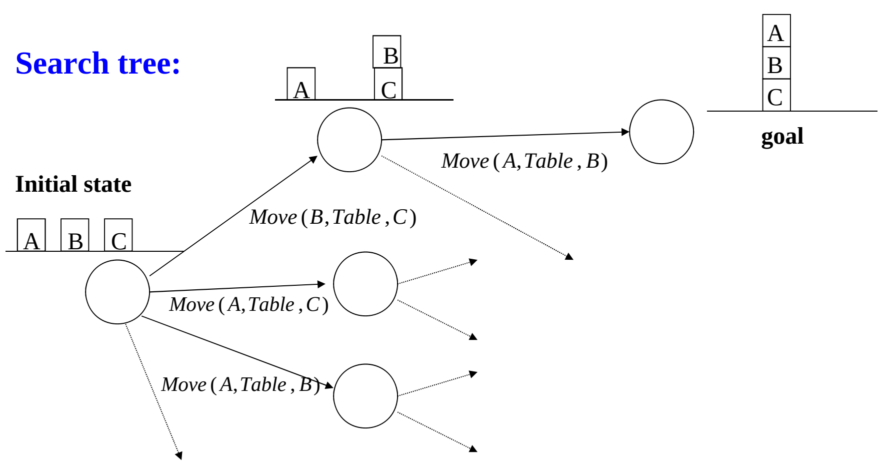
- **Idea:** Given a state *s*
  - Unify the preconditions of some operator *a* with *s*
  - Add and delete sentences from the add and delete list of an operator *a* from *s* to get a new state



Forward search (goal progression)

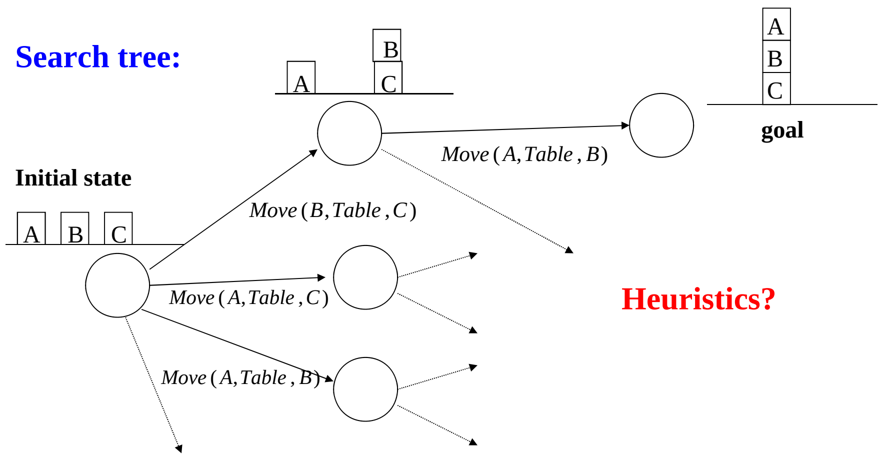
- Use operators to generate new states to search
- Check new states whether they satisfy the goal

Search tree:



Forward search (goal progression)

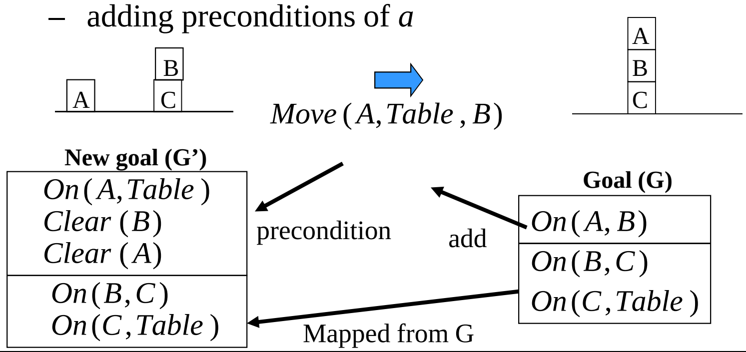
- Use operators to generate new states to search
- Check new states whether they satisfy the goal



Backward search (goal regression)

Idea: Given a goal *G*

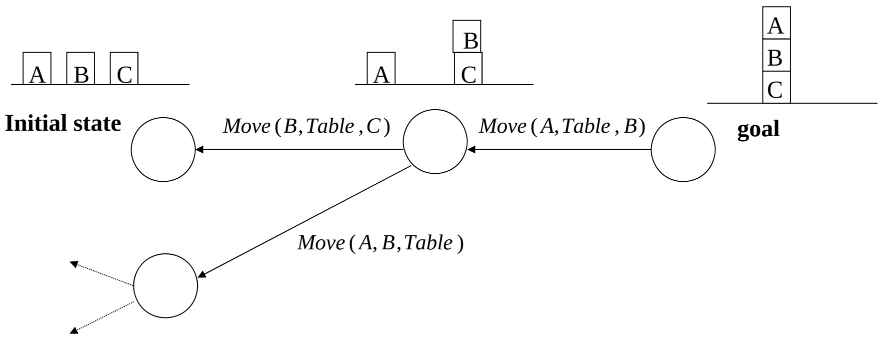
- Unify the add list of some operator *a* with a subset of *G*
- If the delete list of *a* does not remove elements of *G*, then the goal regresses to a new goal *G'* that is obtained from *G* by:
  - deleting add list of *a*
  - adding preconditions of *a*



Backward search (goal regression)

- Use operators to generate new goals
- Check whether the initial state satisfies the goal

Search tree:



State-space search

- **Forward and backward state-space planning approaches:**
  - Work with strictly linear sequences of actions

- **Disadvantages:**
  - They cannot take advantage of the **problem decompositions** in which the goal we want to reach consists of a set of independent or nearly independent sub-goals
  - Action sequences cannot be **built from the middle**
  - No mechanism to represent **least commitment** in terms of the action ordering

Divide and conquer

- **Divide and conquer strategy**:
  - divide the problem to a set of smaller sub-problems,
  - solve each sub-problem independently
  - combine the results to form the solution

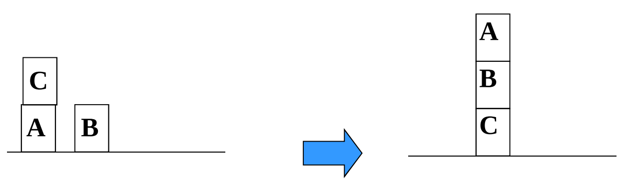
In planning we would like to satisfy a set of goals

- **Divide and conquer in planning:**
  - Divide the planning goals along individual goals
  - Solve (find a plan for) each of them independently
  - Combine the plan solutions in the resulting plan

- Is it always safe to use divide and conquer?
  - No. There can be interacting goals.

Sussman's anomaly.

- An example from the blocks world in which the divide and conquer strategy for goals fails due to interacting goals

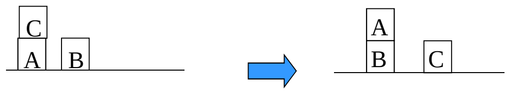


Initial state **Goal**

$$On(A,B)$$
$$On(B,C)$$

Sussman's anomaly

1. Assume we want to satisfy $On(A,B)$ first

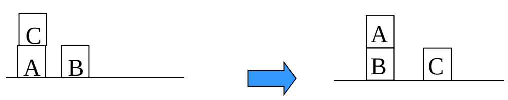


Initial state

But now we cannot satisfy $On(B,C)$ without undoing $On(A,B)$

Sussman's anomaly

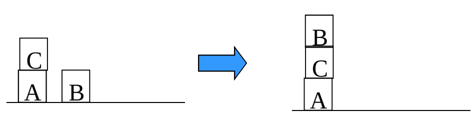
1. Assume we want to satisfy $On(A,B)$ first



Initial state

But now we cannot satisfy $On(B,C)$ without undoing $On(A,B)$

2. Assume we want to satisfy $On(B,C)$ first.



Initial state

But now we cannot satisfy $On(A,B)$ without undoing $On(B,C)$