CMRoboBits:
Action Selection: Behaviors, Planning

Manuela Veloso
Scott Lenser

15-491
November 3, 2003

Complete Robot

- Perception
- Cognition
- Action

Complete Robot

- Perception
  - What is it?
  - All the sensors, vision, world modeling
- Cognition
  - What is it?
  - Action selection, problem solving
  - Coordination, learning
- Action
  - What is it?
  - Motion, manipulation
  - Sound, speech

Problem Solving - Planning

Newell and Simon 1956

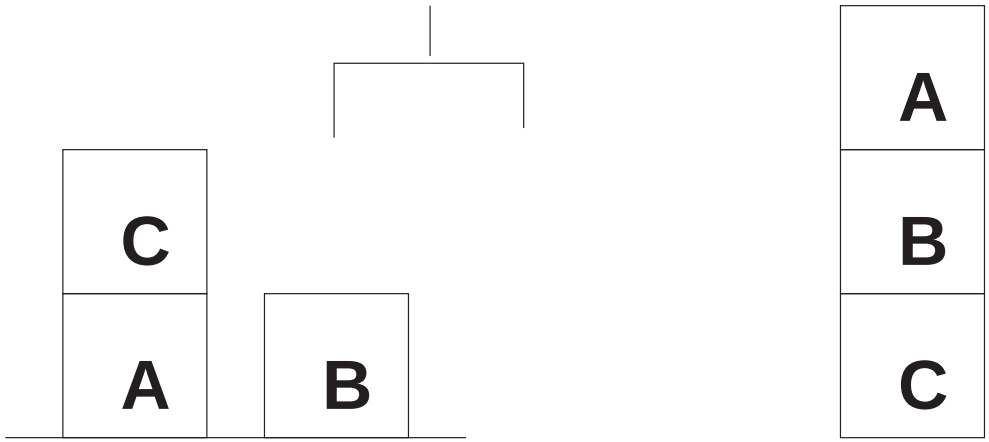
- Given a *problem* specified as:
  - an initial *state* of the world,
  - a *goal statement* to be achieved.
- Given the *actions* available,
- Find a *solution* to the problem, i.e., *the actions* that can transform the initial state into a new state of the world where the goal statement is true.

State, Actions, Goal

Examples of Tasks

- The blocks world
- Towers of Hanoi
- Chess
- 8-puzzle
- Puzzles
- Artificial domains
- Logistics transportation
- Manufacturing planning
- Vacation planning
- .....

Example - Blocks World



The Blocks World

Description of the domain:

- All blocks are labeled and of equal size.
- There is an infinite table where any block can be.
- The block position on the table does not matter.
- There can be at most one block on top of another.
- Blocks are moved by an arm.
- The arm can hold only block at a time.

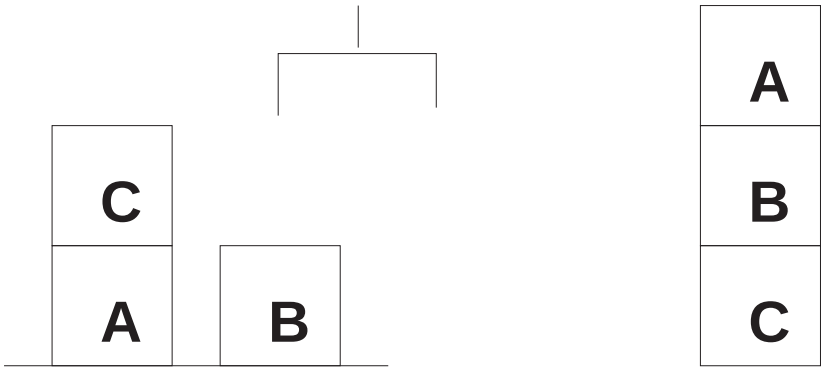
Problems:

- Given an initial set of towers of blocks, find the actions that achieve some specific block positions.

The Blocks World - Definition

- States can be defined as a conjunction of literals.
  - *on(A,B), on(C,table), clear(B), arm-empty,....*
- Goal statement
  - A conjunction of literals; a subset of a complete state
- Actions
  - move, pick-up, put-down
  - We need to represented *legal* actions at states.

Sussman's Anomaly



- State: on(A,table), on(C,A), on(B,table), arm-empty, clear(C), clear(B)
- Goal: on(A,B), on(B,C)

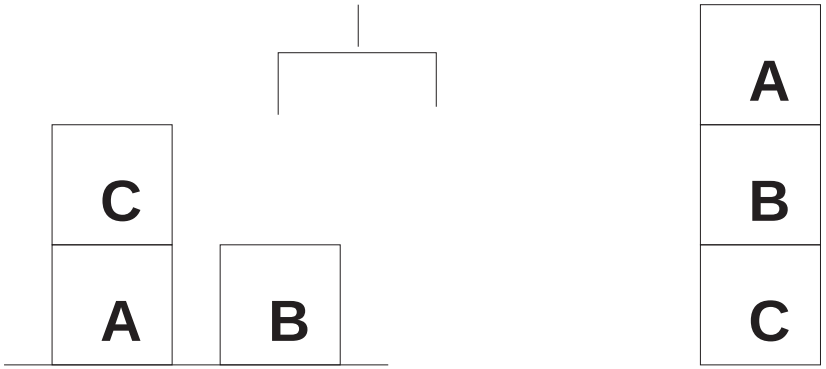
Actions

- Actions - operators – rules – with:
  - Precondition expression – must be satisfied before the operator is applied.
  - Set of effects – describe how the application of the operator changes the state.
- Precondition expression: propositional, typed first order predicate logic, negation, conjunction, disjunction, existential and universal quantification, and functions.
- Effects: add-list and delete-list.
- Conditional effects – dependent on condition on the state *when action takes place*.

Action Representation

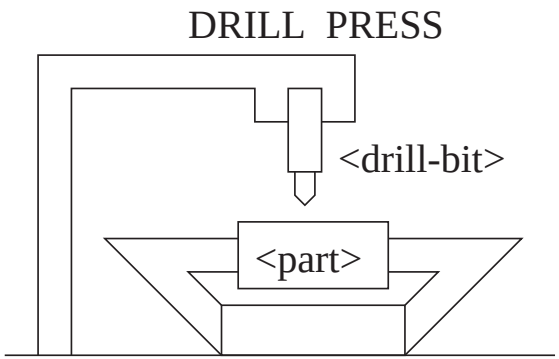
```
(OPERATOR MOVE
   :preconds
      ?block BLOCK
      ?from OBJECT
      ?to OBJECT
      (and (clear ?block)
           (clear ?to)
           (on ?block ?from))
   :effects
      add (on ?block ?to)
      del (on ?block ?from)
      (if (block-p ?from)
          add (clear ?from))
      (if (block-p ?to)
          del (clear ?to)))
```

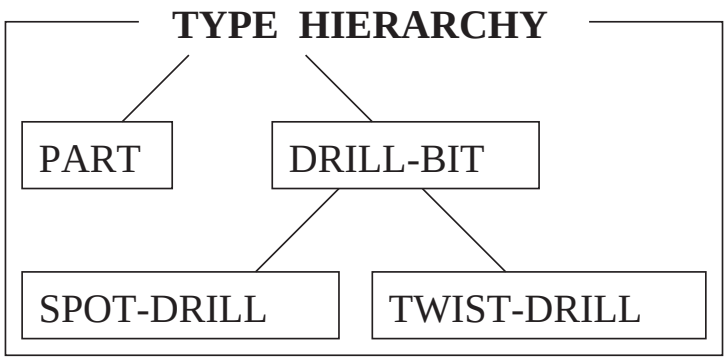
Sussman's Anomaly



- State: on(A,table), on(C,A), on(B,table), arm-empty, clear(C), clear(B)
- Goal: on(A,B), on(B,C)
- Optimal plan:
  - MOVE C A table
  - MOVE B table C
  - MOVE A table B

Another Example - Manufacturing





drill-spot (<part>, <drill-bit>)
<part>: type PART
<drill-bit>: type SPOT-DRILL
Pre: (holding-tool <drill-bit>)
(holding-part <part>)
Add: (has-spot <part>)

put-drill-bit (<drill-bit>)
<drill-bit>: type DRILL-BIT
Pre: tool-holder-empty
Add: (holding-tool <drill-bit>)
Del: tool-holder-empty

put-part(<part>)
<part>: type PART
Pre: part-holder-empty
Add: (holding-part <drill-bit>)
Del: part-holder-empty

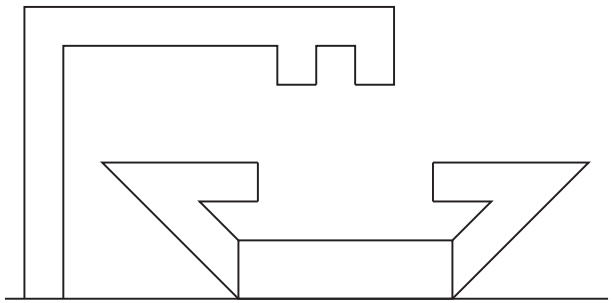
drill-hole(<part>, <drill-bit>)
<part>: type PART
<drill-bit>: type TWIST-DRILL
Pre: (has-spot <part>)
(holding-tool <drill-bit>)
(holding-part <part>)
Add: (has-hole <part>)

remove-drill-bit(<drill-bit>)
<drill-bit>: type DRILL-BIT
Pre: (holding-tool <drill-bit>)
Add: tool-holder-empty
Del: (holding-tool <drill-bit>)

remove-part(<part>)
<part>: type PART
Pre: (holding-part <drill-bit>)
Add: part-holder-empty
Del: (holding-part <drill-bit>)

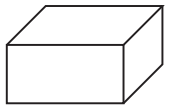
Example - Problem and Plan

Initial State

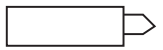


part-holder-empty
drill-holder-empty

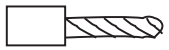
Set of Objects



part-1, part-2 : type PART

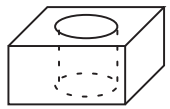


drill-1: type SPOT-DRILL



drill-2, drill-3 : type TWIST-DRILL

Goal Statement



has-hole (part-1)

```
put-part(part-1)
put-drill-bit(drill-1)
drill-spot(part-1, drill-1)
remove-drill-bit(drill-1)
put-drill-bit(drill-2)
drill-hole(part-1, drill-2)
```

Why is Planning Hard?

Planning involves a complex search:

- Alternative operators to achieve a goal
- Multiple goals that interact
- Solution optimality, quality
- Planning efficiency, soundness, completeness

Planning Strategies

- Forward chaining – progression
  - From the initial state,
  - Apply operators with preconditions true in the state,
  - Get new states.
- Backward-chaining – regression
  - From the goal state,
  - Find operators that can add goal,
  - Set its preconditions as new goals.

One-Way Rocket Domain

```
(OPERATOR LOAD-ROCKET
   :preconds
      ?roc ROCKET
      ?obj OBJECT
      ?loc LOCATION
      (and (at ?obj ?loc)
           (at ?roc ?loc))
   :effects
      add (inside ?obj ?roc)
      del (at ?obj ?loc))
```

```
(OPERATOR MOVE-ROCKET
   :preconds
      ?roc ROCKET
      ?from-l LOCATION
      ?to-l LOCATION
      (and (at ?roc ?from-l)
           (has-fuel ?roc))
   :effects
      add (at ?roc ?to-l)
      del (at ?roc ?from-l)
      del (has-fuel ?roc))
```

```
(OPERATOR UNLOAD-ROCKET
   :preconds
      ?roc ROCKET
      ?obj OBJECT
      ?loc LOCATION
      (and (inside ?obj ?roc)
           (at ?roc ?loc))
   :effects
      add (at ?obj ?loc)
      del (inside ?obj ?roc))
```

Planning with Stack of Goals: Incompleteness

Initial state:

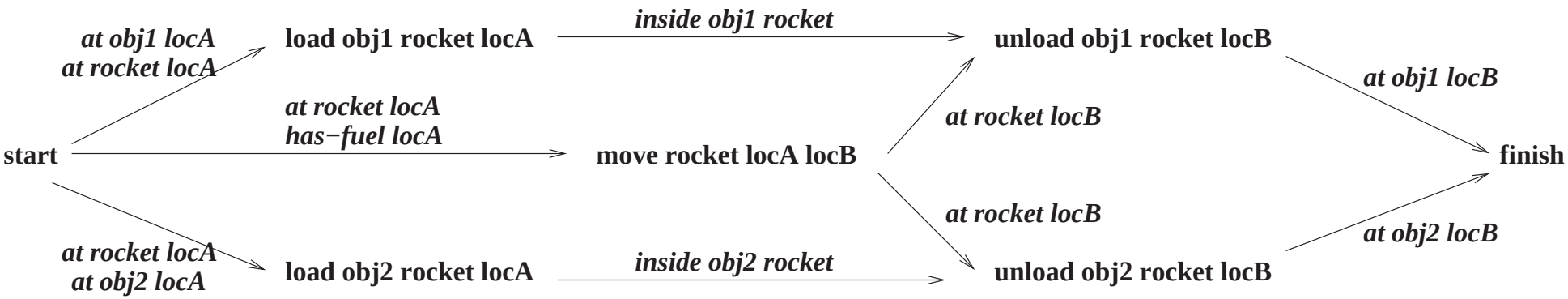
(at obj1 locA)
(at obj2 locA)
(at ROCKET locA)
(has-fuel ROCKET)

Goal statement:

(and
(at obj1 locB)
(at obj2 locB))

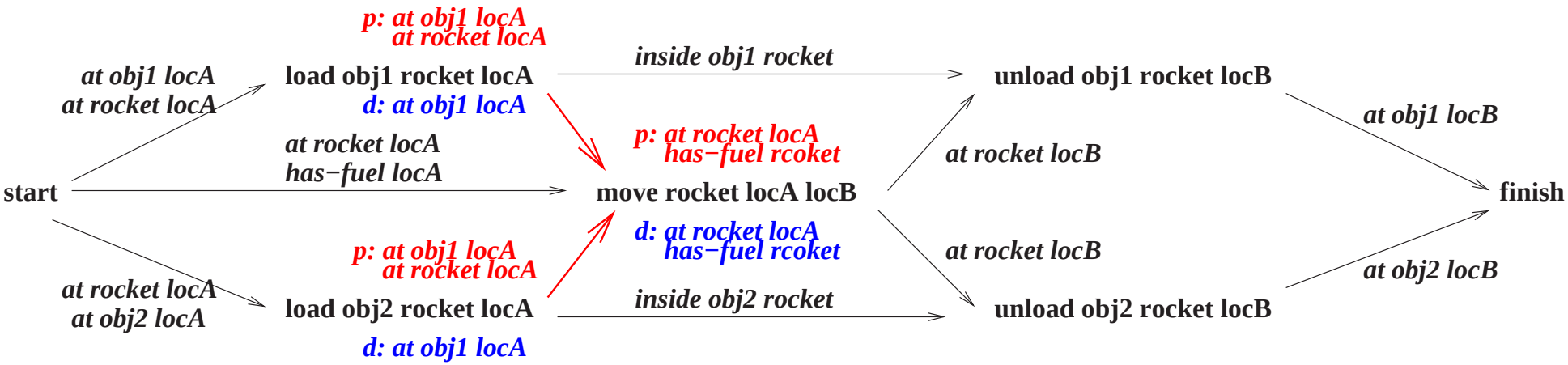
Goal	*Plan*
(at obj1 locB)	(LOAD-ROCKET obj1 locA) (MOVE-ROCKET) (UNLOAD-ROCKET obj1 locB)
(at obj2 locB)	*failure*

Partial Order Planning



Example − LINKING

Partially Ordered Plan



Example − THREATS

Many Issues To Resolve...

- What knowledge defines the task domain?
- How to represent the planning action model?
- What is the (sufficient) initial state of the world?
- What are the (prioritized) goals?
- How to acquire domain knowledge efficiently from expert human planners?
- What algorithm to generate the solution plan?
- How to generate plans in a computationally tractable way?
- How to create plans of *good* quality?
- How to scale up to real-world problems?

Uncertainty in the World

- The goal of planning is *to execute* the plans generated!
- Planning models the world - actions, predicates, objects, choices for plan generation
- Sources of uncertainty
  - State may be unknown
  - Exact effects of actions may be unknown
  - External events may change the world

Handling Uncertainty

- **Replanning:**
  1. Assume model is correct and plan
  2. **Wait** for eventual execution failures
  3. React to failures - Plan again

- **Conditional planning:**
  1. Add/represent observable contingencies
  2. Create a *branching plan* for contingencies
  3. Observe and execute appropriate branch

- **Probabilistic planning:** Plan for the *most probable* contingencies.

The Flat Tire Operators

```
Operator REMOVE (x)
p: On (x)

a: Off (x) and
   ClearHub (x)
d: On (x)
```

```
Operator PUT-ON (x)
p: Off (x) and
   ClearHub (x)
a: On (x)

d: ClearHub (x) and
   Off (x)
```

```
Operator INFLATE (x)
p: Intact (x) and
   Flat (x)
a: Inflated (x)

d: Flat (x)
```

A Flat Tire Problem

- Initial state:
  - On (Tire1) Flat (Tire1),
  - Inflated (Spare), Intact (Spare), and Off (Spare)
- Goal:
  - On (x) and Inflated (x)
- Plan:
  - Remove(Tire1), Put-On (Spare)

Possible Contingencies

- The tire may just not be inflated.
- We would like a plan:
  - If Intact (Tire1) Then Inflate (Tire1)
  - Else Remove (Tire1) and Put-On (Spare)
- There is uncertainty in the state: is the tire intact?
- Mark precondition Intact as *unknown* and check.
- Three-valued logic: True, False, Unknown.

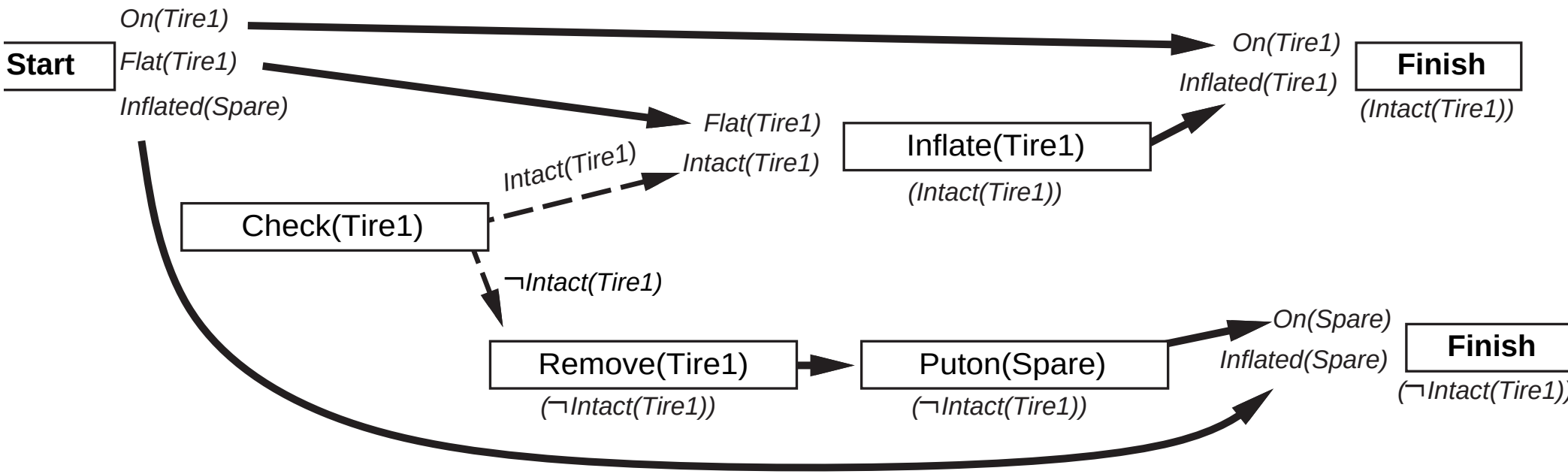
The Flat Tire Operators - Revised

```
Operator REMOVE (x)
p: On (x)

a: Off (x) and
   ClearHub (x)
d: On (x)
```

```
Operator PUT-ON (x)
p: Off (x) and
   ClearHub (x)
a: On (x)

d: ClearHub (x) and
   Off (x)
```

```
Operator INFLATE (x)
p: UNK (Intact (x))
   Flat (x)
a: Inflated (x)

d: Flat (x)
```

```
Operator CHECKTIRE (x)
p: Tire (x)

a: KnowsWhether
        (Intact (x))
```


On(Tire1)
Start
Flat(Tire1)
Inflated(Spare)
On(Tire1)
Inflated(Tire1)
Finish
(Intact(Tire1))
Flat(Tire1)
Intact(Tire1)
Intact(Tire1)
Inflate(Tire1)
(Intact(Tire1))
Check(Tire1)
¬Intact(Tire1)
On(Spare)
Inflated(Spare)
Finish
(¬Intact(Tire1))
Remove(Tire1)
(¬Intact(Tire1))
Puton(Spare)
(¬Intact(Tire1))

Game Playing

- Game playing - another example of planning and execution
- Discussion

Robot Problem Solving

- The planning algorithm consists of finding the complete sequence of actions to a goal state from a given state.
  - Discussion!
  - Enumerate (all) the complications of planning for robot problem solving.
  - We will write them all on the board.

Summary - Planning - Action Selection

- Planning: selecting one sequence of actions (operators) that transform (apply to) an initial state to a final state where the goal statement is true.
- Planning is complex: multiple actions, interacting goals, search.
- Planning and execution: models are incomplete and incorrect.
- Robot problem solving: reactive, real-time, some planning.