EECS 496: Sequential Decision Making

Soumya Ray

sray@case.edu

Office: Olin 516

Office hours : T 4-5:30 or by appointment

Recap

- With Kalman filters, the state is modeled as a set of ____ _____.
- The transition/sensor distributions for a new state is obtained as ____ functions of the old state with ____ ____.
- In continuous state scenarios, we are often interested in _____ prediction problems. These are where we _____.
- What is filtering? Smoothing? “Prediction”? Most likely path?
- In filtering, the state distribution at (t+1) is a function of the __ __, the __ ___ and the _____.
- If the sensor and transition distributions are __ __, then the ____ of filtering is ___ ___.
- KFs make strong assumptions, which can be relaxed by integrating them with _____. In this case each _____ has a ____ ____ distribution over its ____. Further flexibility could be added through ____ ____ which have ____ ____ distributions in the ____.

Today

- Part 2: Automated Planning (Ch 10 R&N)

Automated Planning

- Consider a situation where an agent has to carry out a sequence of actions to achieve a goal

- Suppose the agent starts off with detailed, *structured* knowledge of the world
  - Could we take advantage of this?

The Planning Problem

- Given:
  - An initial state of the world, described as a set of logical facts
  - A set of goal conditions, described as a set of logical facts
  - A set of actions, also described in logic

- Find a sequence of actions that will move the world from the initial state to the final state
  - This sequence is called a *plan*
  - Often also try to optimize some criteria

“Classical” Planning

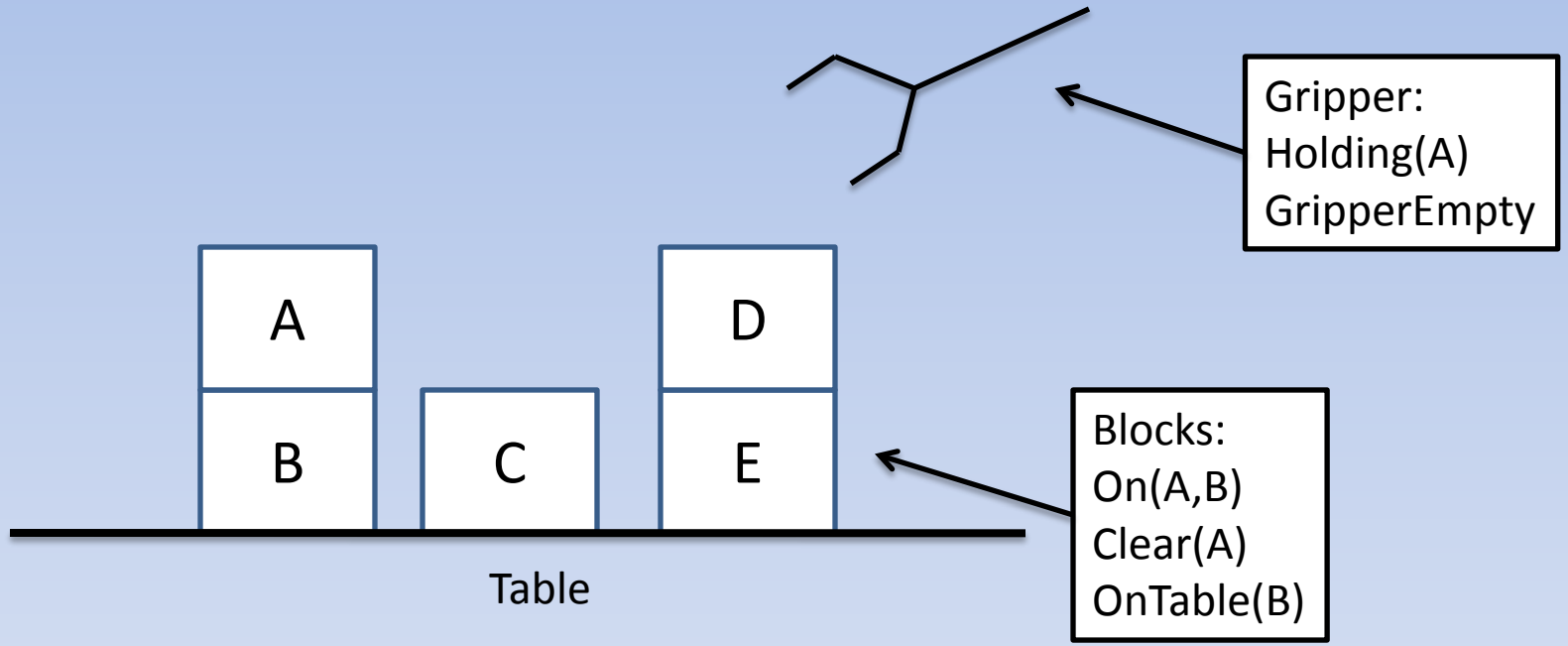
- We’ll study planning algorithms designed to work when the world is:
  - Static
  - Deterministic
  - Fully observable
  - Discrete
  - Actions are instantaneous
- These restrictions can be relaxed (more or less)

Makespan

- Typically, in planning, we are interested in minimizing the *duration* of the plan
  - Equivalent to the number of actions in the current setup
  - This is called the *makespan*

- Nonclassical planning allows arbitrary plan metrics to be minimized

Example: Blocks World



Task: Starting with initial configuration of blocks,
produce a desired goal configuration by moving blocks around.

Extended Syntax of Planning

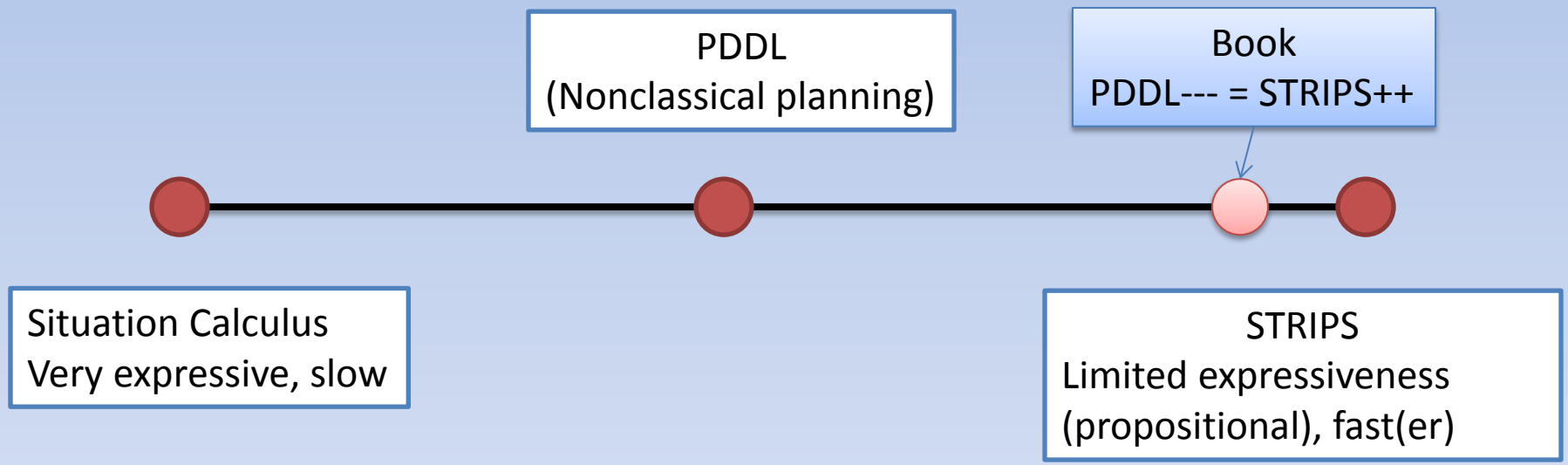
- Holding(A)
  - A “predicate” that can take multiple “arguments” and evaluates to true or false
  - Internally map to a proposition like Holding_A
- Holding(x)
  - This predicate takes a *variable* argument
  - When planning, the inference algorithm will try *all possible values for x* that are applicable (later)
  - Map to Holding_A, Holding_B,…

Extended Syntax of Planning

- The fragment of FOL that we just saw is called STRIPS ("Stanford Research Institute Problem Solver")

- States, actions and goals will be represented in this language
  - Then we'll see planning algorithms (which are inference algorithms in disguise) that find plans in this language

Extended Syntax of Planning



Representation/computation/optimality tradeoff---universal in AI

Representing States in STRIPS

- States in STRIPS are conjunctions of unnegated ground literals
  - All conditions that hold in that state
  - *Block(A)*, *Block(B)*, *On(A,B)*, *On(B, Table)*, *GripperEmpty*
  - Technical condition: also need to be "function free"
    - A function is an FOL construction that maps one object to another, e.g. Mother(X)=Y
  - The "Closed World Assumption" is used

No variables

Closed World Assumption (CWA)

- Anything that is not explicitly listed as part of a state definition is false
  - No “unknown” variables

Representing Goals in STRIPS

- Goals are conjunctions of unnegated, ground (function-free) literals
- Goals may not fully determine a state of the world
  - In this case, the goal is any state where these literals hold
- Example: $On(A,E) \wedge On(B,D)$

Representing Actions in STRIPS

- Want to represent an action of picking up a block from the table

Pickup_from_Table(x)

Preconditions: *Block(x), GripperEmpty, Clear(x), On(x,Table)*

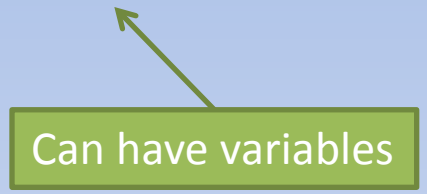
Add List: *Holding(x)*

Delete List: *GripperEmpty, On(x,Table)*

"Applicability": action can be used at a state iff its preconditions are satisfied

Representing Actions in STRIPS

- An "action schema" represents a non-ground action using three parts:
  - The action name and argument list
  - The preconditions: a conjunction of unnegated non-ground (function-free) literals. Any variables in this list are parameters to the action.
  - The effects: a conjunction of function-free literals describing how the state changes.



Add and Delete Lists

- Often, the unnegated literals in the action effects are collected into an “ADD” list, and the negated literals are collected into a “DELETE” list
  - Idea: Starting with initial state, to get result of applying action, add the literals in ADD list and delete the literals in the DELETE list

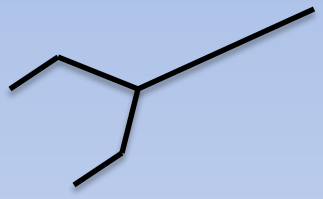
The STRIPS assumption

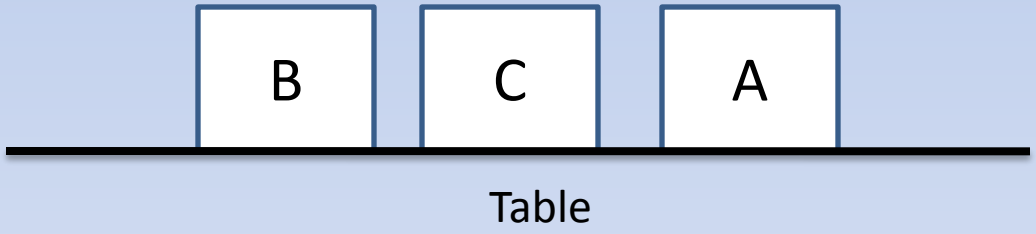
- Every possible effect of actions are listed
  - i.e., if a literal does not appear in the effects list, it is unchanged in the resulting state
  - Together with CWA, solves the “frame problem”
    - In general logic, a literal whose truth value is unspecified becomes “unknown”

Restrictions in STRIPS

- States are described by unnegated ground (function-free) literals
- CWA + STRIPS assumption
- Ground conjunctive goals
- Conjunctive effects of actions

Example: Blocks World





Example

- Init($On(A,Table) \wedge On(B,Table) \wedge On(C,Table) \wedge Block(A) \wedge Block(B) \wedge Block(C) \wedge Clear(A) \wedge Clear(B) \wedge Clear(C) \wedge GripperEmpty$)
- Goal($On(A,B)$)
- Action($MoveToTable(b,x)$,
  - Preconditions($On(b,x) \wedge Clear(b) \wedge Block(b) \wedge Block(x) \wedge GripperEmpty$)
  - AddEffects($On(b,Table) \wedge Clear(x)$)
  - DelEffects($On(b,x)$)

Planning Algorithms

- Given a STRIPS representation of a classical planning problem, how do we solve it?
  - Since the world is static, deterministic, fully observable, we could use search
  - Remember that in this case, the search algorithm is actually performing logical inference

Kinds of Search for Planning

- Search algorithms for classical planning fall into two categories
  - "State space planners": States of the search problem are states of the world; search operators are actions of the world
  - "Plan space planners": States of the search problem are partial plans; search operators are modifications to the current partial plan

Forward State-Space Search

- "Progression" planning

- Setup:
  - States=world states (in STRIPS)
  - Initial state=given
  - Operators=*applicable* actions (in STRIPS)
  - Goal test=given (in STRIPS)
  - Operator costs=unit (minimize number of actions)

Forward State-Space Search

- We could apply any search algorithm, e.g. A*

- The key differences are:
  - Only applicable actions need to be explored at a state
  - Getting the next state is done through the STRIPS specification of states and actions
  - Heuristics are based on planning ideas

Search Heuristics

- From any state, want to estimate the number of actions to search termination admissibly

- Two possibilities:
  - Relax the planning problem
  - Consider subproblems

Relaxed Plans

- There are different ways to arrive at a less constrained planning problem

- One way is to remove all DELETE effects from STRIPS actions
  - This is admissible (why?)
  - To estimate this cost, need to run an internal planning loop; but this is usually very fast

Subproblems

- The goal is a conjunction of literals
- We can generate subproblems by just considering a single literal at a time
  - "Subgoal Independence" (admissible)
- Combine with the max function