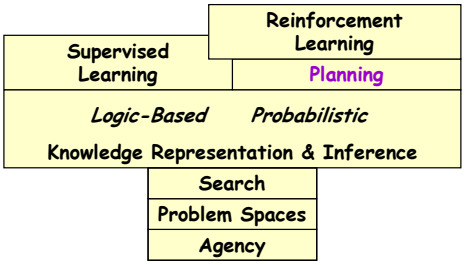
Planning II

CSE 573

Logistics

- Reading for Wed
  - Ch 18 thru 18.3
- Office Hours
  - No Office Hour Today

573 Topics



Planning

Input:

- *Description of initial state of world*
  - E.g., Set of propositions:
  - **((block a) (block b) (block c) (on-table a) (on-table b) (clear a) (clear b) (clear c) (arm-empty))**
- *Description of goal: i.e. set of worlds or…*
  - E.g., Logical conjunction
  - Any world satisfying conjunction is a goal
  - **(and (on a b) (on b c)))**
- *Description of available actions*

Output:

- *Sequence of Actions*

STRIPS Representation

- ***Simplifying assumptions***
  - Atomic time
  - Agent is omniscient (no sensing necessary).
  - Agent is sole cause of change
  - Actions have deterministic effects
- ***STRIPS representation***
  - World = set of true propositions
  - Actions:
    - Precondition: (conjunction of literals)
    - Effects (conjunction of literals)

Action Schemata

- Instead of defining:
  pickup-A and **pickup-B** and …
- Define a schema:

(:operator **pick-up**
***:parameters* ((block ?ob1))**
:precondition (and (clear ?ob1)
(on-table ?ob1)
(arm-empty))
:effect (and (not (clear ?ob1))
(not (on-table ?ob1))
(not (arm-empty))
(holding ?ob1)))

Note: strips doesn't allow derived effects: you must be complete!

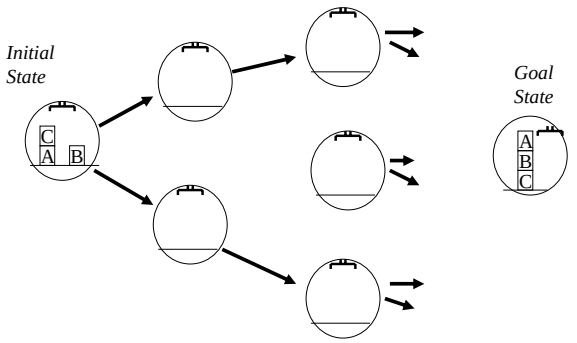
Planning Outline

- The planning problem
- Representation
- Compilation to SAT
- Searching world states
  - Regression
  - Heuristics
- Graphplan
- Reachability analysis & heuristics
- Planning under uncertainty

Planning as Search

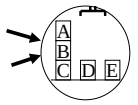
- Nodes
  - World states
- Arcs
  - Action executions
- Initial State
  - The state satisfying the complete description of the initial conds
- Goal State
  - Any state satisfying the goal propositions

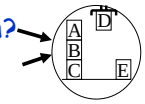
Forward-Chaining World-Space Search



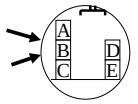
Backward-Chaining Search Thru Space of Partial World-States

- Problem: Many possible goal states are equally acceptable.
- From which one does one search?

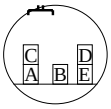




* * *



Initial State is completely defined

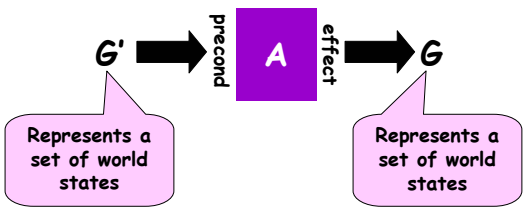


Planning as Search - Backward

- Nodes
  - A conjunctive goal ("set of goals")
- Arcs
  - Regression of goal through action defn
- Initial State
  - The goal of the planning problem
- Goal State
  - A set of goals ⊆ the planning problem's initial description

Regression

- Regressing a goal, G, thru an action, A
- Yields the weakest precondition G'
  - Such that: if G' is true before A is executed
  - G is guaranteed to be true afterwards



Regression Example

(and (clear C)
(on-table C)
(arm-empty)
(on A B))

(and (holding C)
(on A B))

pick-up :*parameters* ((block ?ob1))
:*precondition* (and (clear ?ob1)
(on-table ?ob1)
(arm-empty))
:*effect* (and (not (clear ?ob1))
(not (on-table ?ob1))
(not (arm-empty))
(holding ?ob1)))

Disjunction
∀ preconditions

Conditional Effects

```
move-briefcase (?loc ?new)
    :prec   (and (at briefcase ?loc) (location ?new)
                 (not (= ?loc ?new)))
    :effect (and (at briefcase ?new) (not (at briefcase ?loc))
                 (when (in paycheck briefcase)
                       (and (at paycheck ?new)
                            (not (at paycheck ?loc))))
                 (when (in keys briefcase)
                       (and (at keys ?new)
                            (not (at keys ?loc)))))
```


Regressing Conditional Effects

(and (at briefcase bank)
(in keys briefcase)
(not (in paycheck briefcase))
(at paycheck bank))

(and (at keys home)
(at paycheck bank))

bank home

```
move-briefcase (?loc ?new)
    :prec   (and (at briefcase ?loc)
                 (not (= ?loc ?new)))
    :effect (and (at briefcase ?new) (not (at briefcase ?loc))
                 (when (in paycheck briefcase)
                       (and (at paycheck ?new)
                            (not (at paycheck ?loc))))
                 (when (in keys briefcase)
                       (and (at keys ?new)
                            (not (at keys ?loc)))))
```


Heuristics

- Raise issue
- Defer until…

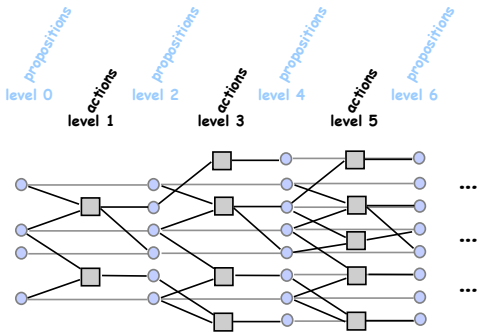
Planning Outline

- The planning problem
- Representation
- Compilation to SAT
- Searching world states
  - Regression
  - Heuristics
- Graphplan
- Reachability analysis & heuristics
- Planning under uncertainty

Graphplan

- Phase 1 - Graph Expansion
  - **Necessary (insufficient) conditions for plan existence**
  - **Local consistency of plan-as-CSP**
- Phase 2 - Solution Extraction
  - **Variables**
    - **action execution at a time point**
  - **Constraints**
    - **goals, subgoals achieved**
    - **no side-effects between actions**

The Plan Graph

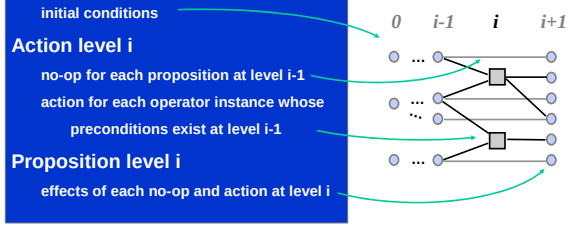


Graph Expansion

Proposition level 0
- initial conditions

Action level i
- no-op for each proposition at level i-1
- action for each operator instance whose preconditions exist at level i-1

Proposition level i
- effects of each no-op and action at level i



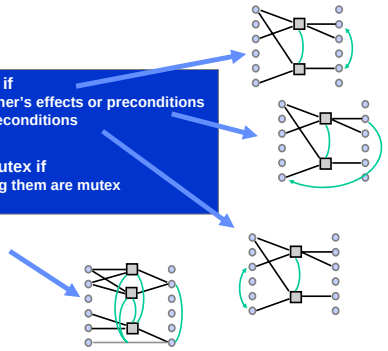
Constructing the planning graph…

- Initial proposition layer
  - Just the initial conditions
- Action layer i
  - If all of an action's preconds are in i-1
    - Then add action to layer I
  - Nop actions have P as precond and effect
- Proposition layer i+1
  - For each action at layer i
  - Add all its effects at layer i+1

Mutual Exclusion

Two actions are mutex if
- one clobbers the other's effects or preconditions
- they have mutex preconditions

Two proposition are mutex if
- all ways of achieving them are mutex



Mutual Exclusion

- Actions A,B ***exclusive (at a level)*** if
  - A deletes B's precond, or
  - B deletes A's precond, or
  - A & B have inconsistent preconds
- Propositions P,Q ***inconsistent (at a level)*** if
  - all ways to achieve P
  - exclude all ways to achieve Q

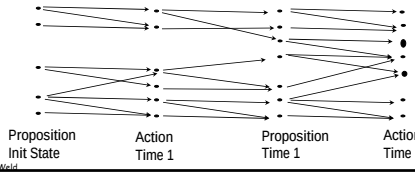
Graphplan

- Create level 0 in planning graph
- Loop
  - If goal $\subseteq$ contents of highest level (nonmutex)
  - Then search graph for solution
    - If find a solution then return and terminate
  - Else extend graph one more level

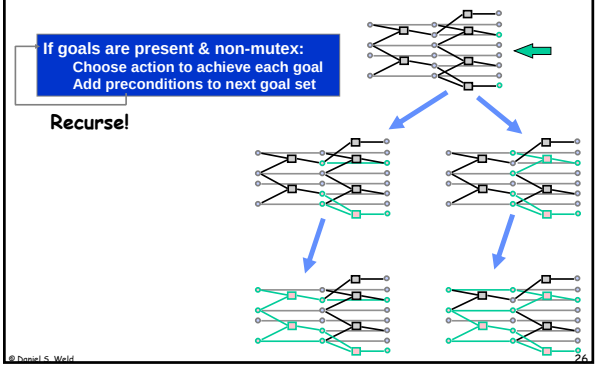
A kind of double search: forward direction checks necessary (but insufficient) conditions for a solution, … Backward search verifies…

Searching for a Solution

- For each goal G at time t
  For each action A making G true @t
  - Select A unless mutex with previously chosen action
  - If no actions work, backup to last G (breadth first search)
- Recurse on
  preconditions of actions selected, t-1



Searching for a solution



Dinner Date

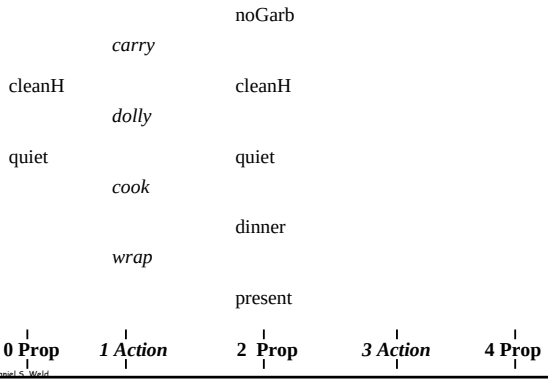
Initial Conditions: (:and (cleanHands) (quiet))

Goal: (:and (noGarbage) (dinner) (present))

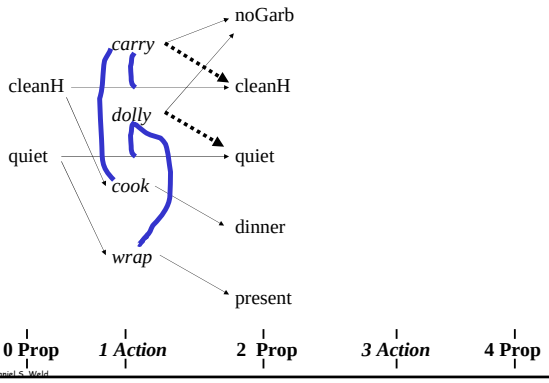
Actions:

(:operator **carry** *:precondition*
:effect (:and (noGarbage) (:not (cleanHands)))
(:operator **dolly** *:precondition*
:effect (:and (noGarbage) (:not (quiet)))
(:operator **cook** *:precondition* (cleanHands)
:effect (dinner))
(:operator **wrap** *:precondition* (quiet)
:effect (present))

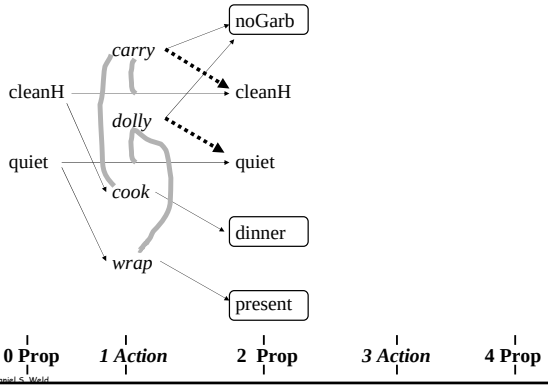
Planning Graph



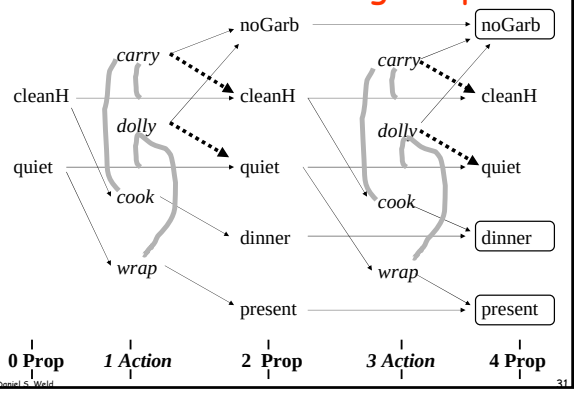
Are there any exclusions?



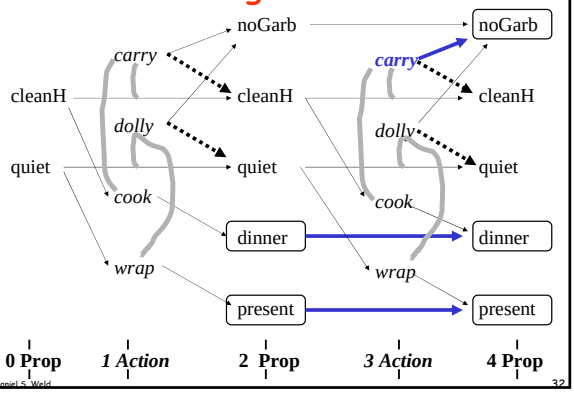
Do we have a solution?



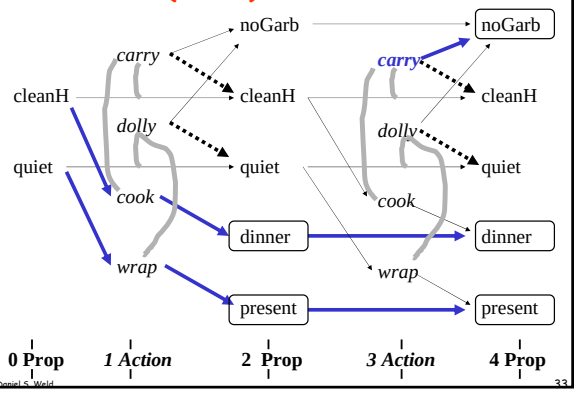
Extend the Planning Graph



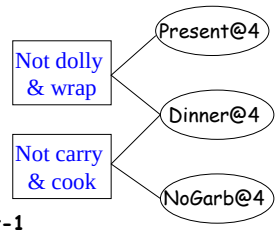
Searching Backwards



One (of 4) Possibilities



Graphplan Solution Extraction as a Constraint Network



Vars = Goals@t

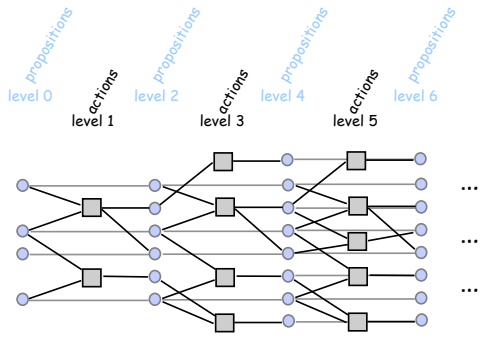
Domain = Actions@t-1

Constraints =

Review: Arc- & Path-Consistency

- Connect with graph construction
- What is k?

Blackbox SAT Encoding



Heuristics for Regression Planning