Classical Planning
Chapter 10

Mausam

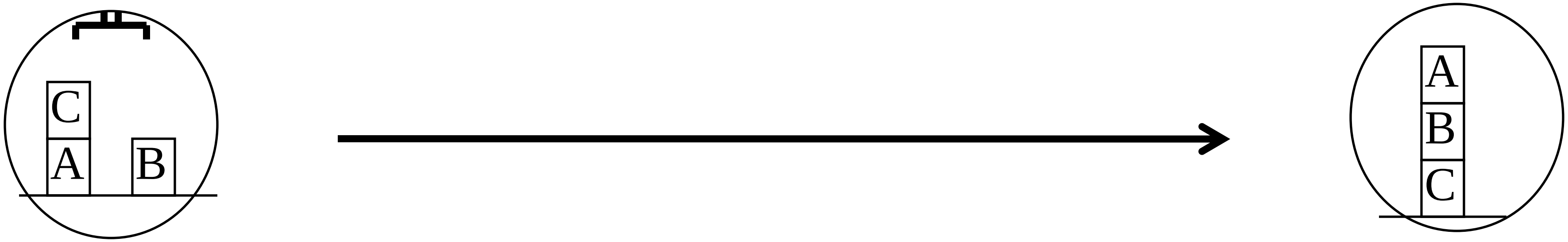
(Based on slides of Dan Weld, Stuart Russell, Marie desJardins)

Planning

- Given
  - a logical description of the **initial situation**,
  - a logical description of the **goal conditions**, and
  - a logical description of a set of **possible actions**,

- find
  - a **sequence of actions** (a **plan of actions**) that brings us from the initial situation to a situation in which the goal conditions hold.

Example: BlocksWorld



Planning Input: State Variables/Propositions

- Types: block --- a, b, c
- (on-table a) (on-table b) (on-table c)
- (clear a) (clear b) (clear c)
- (arm-empty)
- (holding a) (holding b) (holding c)
- (on a b) (on a c) (on b a) (on b c) (on c a) (on c b)

No. of state variables =16

No. of states = 2^{16}

No. of reachable states = ?

- (on-table ?b); clear (?b)
- (arm-empty); holding (?b)
- (on ?b1 ?b2)

Planning Input: Actions

- **pickup a b, pickup a c, …**
- **place a b, place a c, …**
- **pickup-table a, pickup-table b, …**
- **place-table a, place-table b, …**

- **pickup ?b1 ?b2**
- **place ?b1 ?b2**
- **pickup-table ?b**
- **place-table ?b**

Total: 6 + 6 + 3 + 3 = 18 "ground" actions

Total: 4 action schemata

Planning Input: Actions (contd)

- **:action pickup ?b1 ?b2**

  :precondition

  (on ?b1 ?b2)

  (clear ?b1)

  (arm-empty)

  :effect

  (holding ?b1)

  (not (on ?b1 ?b2))

  (clear ?b2)

  (not (arm-empty))

- **:action pickup-table ?b**

  :precondition

  (on-table ?b)

  (clear ?b)

  (arm-empty)

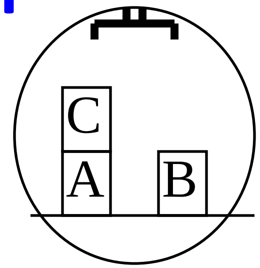
  :effect

  (holding ?b)

  (not (on-table ?b))

  (not (arm-empty))

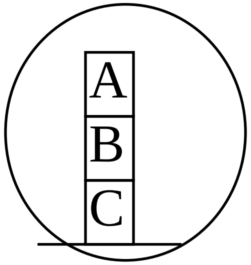
Planning Input: Initial State



- **(on-table a) (on-table b)**
- **(arm-empty)**
- **(clear c) (clear b)**
- **(on c a)**

- **All other propositions false**
  - **not mentioned → false**

Planning Input: Goal



- **(on-table c) AND (on b c) AND (on a b)**
- **Is this a state?**
- **In planning a goal is a set of states**

Planning Input Representation

- Description of initial state of world
  - Set of propositions
- Description of goal: i.e. set of worlds
  - E.g., Logical conjunction
  - Any world satisfying conjunction is a goal
- Description of available actions

Planning vs. Problem-Solving

Basic difference: **Explicit, logic-based representation**

- States/Situations: descriptions of the world by logical formulae
  → agent can explicitly reason about and communicate with the world.
- Goal conditions as logical formulae vs. goal test (black box)
  → agent can reflect on its goals.
- Operators/Actions: Axioms or transformation on formulae in a logical form
  → agent can gain information about the effects of actions by inspecting the operators.

Classical Planning

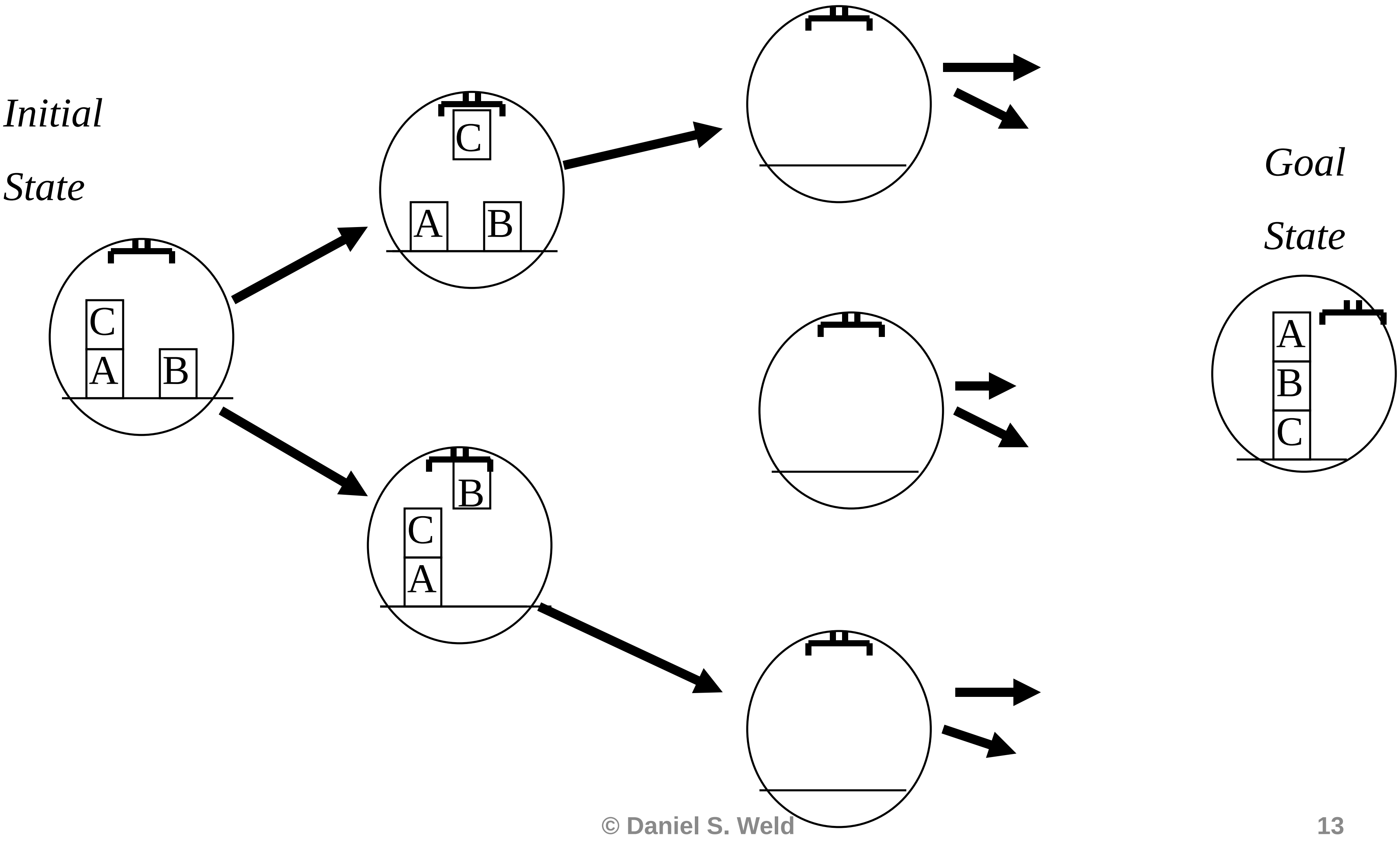
- Simplifying assumptions
  - Atomic time
  - Agent is omniscient (no sensing necessary).
  - Agent is sole cause of change
  - Actions have deterministic effects

- STRIPS representation
  - World = set of true propositions (conjunction)
  - Actions:
    - Precondition: (conjunction of *positive* literals, no functions)
    - Effects (conjunction of literals, no functions)
  - Goal = conjunction of *positive* literals

  - Is Blocks World in STRIPS?

  - Goals = conjunctions (Rich ^ Famous)

Forward World-Space Search



Forward State-Space Search

- Initial state: set of positive ground literals (CWA: literals not appearing are false)
- Actions:
  – applicable if preconditions satisfied
  – add positive effect literals
  – remove negative effect literals
- Goal test: checks whether state satisfies goal
- Step cost: typically 1

Complexity of Planning

- Size of Search Space
  - Size of the world state space
- Size of World state space
  - exponential in problem representation
- What to do?
  - Informative heuristic that can be computed in polynomial time!

Heuristics for State-Space Search

- Count number of false goal propositions in current state

  Admissible?

  NO

- Subgoal independence assumption:
  - Cost of solving conjunction is sum of cost of solving each subgoal independently
  - Optimistic: ignores negative interactions
  - Pessimistic: ignores redundancy

  - Admissible? No
  - Can you make this admissible?

Heuristics for State Space Search (contd)

- Delete all preconditions from actions, solve easy relaxed problem, use length

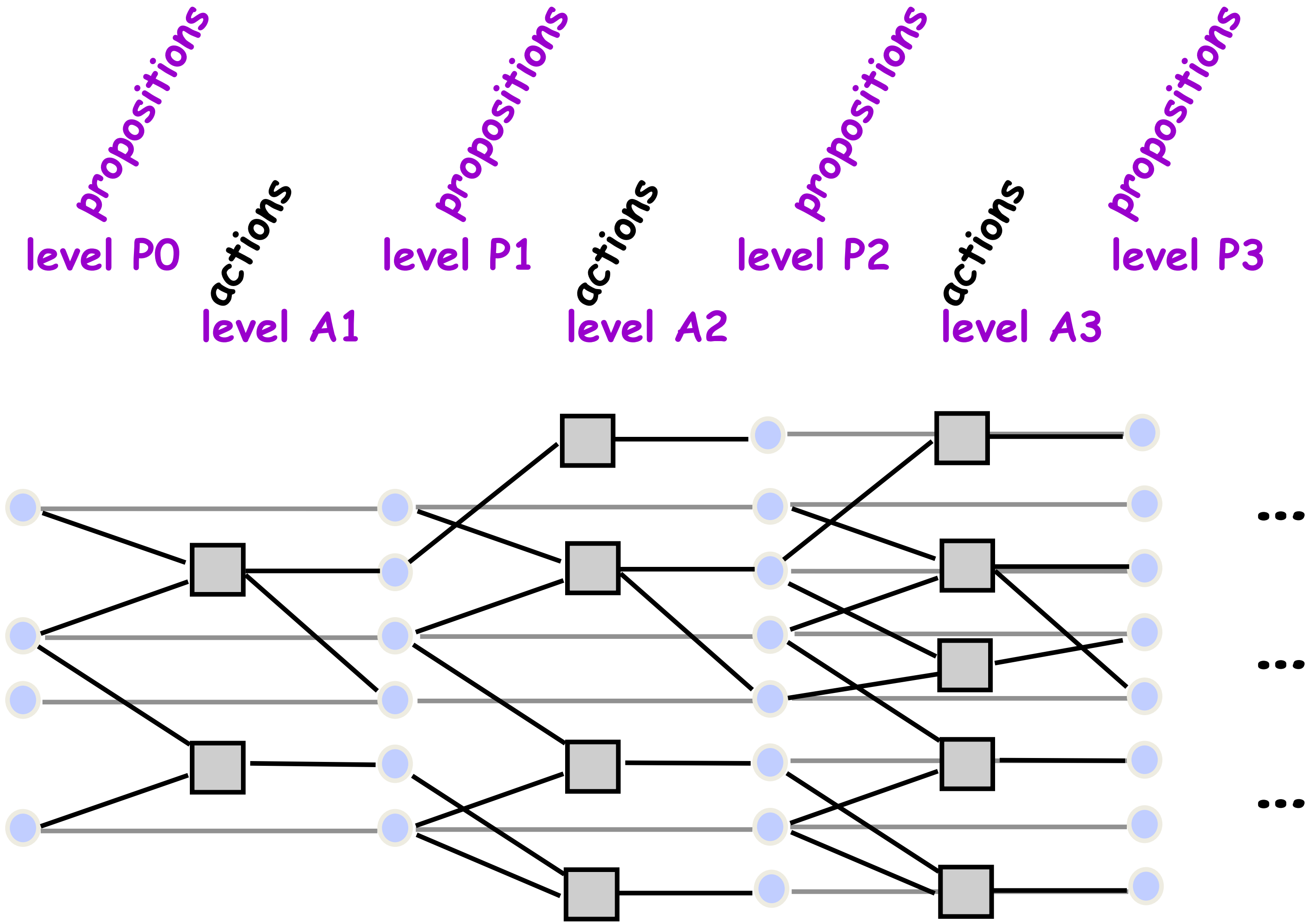
  Admissible?

  YES

Planning Graph: Basic idea

- Construct a planning graph: encodes constraints on possible plans
- Use this planning graph to compute an informative heuristic (Forward A*)
- Planning graph can be built for each problem in polynomial time

The Planning Graph



Note: a few noops missing for clarity

Planning Graphs

- Planning graphs consists of a seq of levels that correspond to time steps in the plan.
  - Level 0 is the initial state.
  - Each level consists of a set of literals and a set of actions that represent what *might be* possible at that step in the plan
  - *Might be* is the key to efficiency
  - Records only a restricted subset of possible negative interactions among actions.

Planning Graphs

- Each level consists of
- *Literals* = all those that *could* be true at that time step, depending upon the actions executed at preceding time steps.
- *Actions* = all those actions that *could* have their preconditions satisfied at that time step, depending on which of the literals actually hold.

PG Example

Init(Have(Cake))

Goal(Have(Cake) $\wedge$ Eaten(Cake))

Action(Eat(Cake),
  PRECOND: Have(Cake)
  EFFECT: $\neg$Have(Cake) $\wedge$ Eaten(Cake))

Action(Bake(Cake),
  PRECOND: $\neg$ Have(Cake)
  EFFECT: Have(Cake))

PG Example

S_0 A_0 S_1

Have(Cake)

$\neg$*Eaten(Cake)*

Create level 0 from initial problem state.

Graph Expansion

Proposition level 0

initial conditions

Action level i

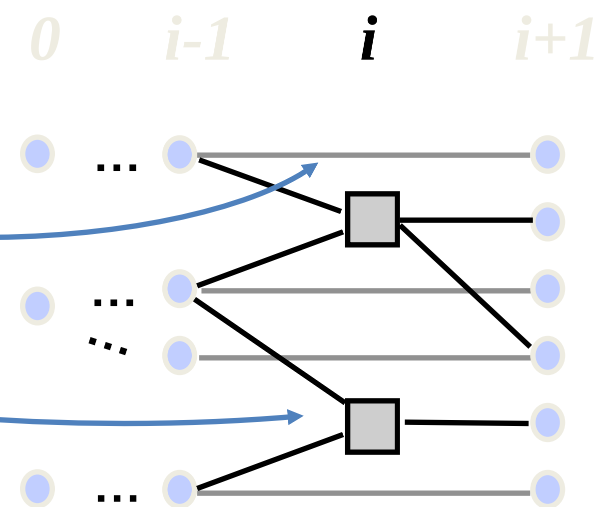
no-op for each proposition at level i-1

action for each operator instance whose

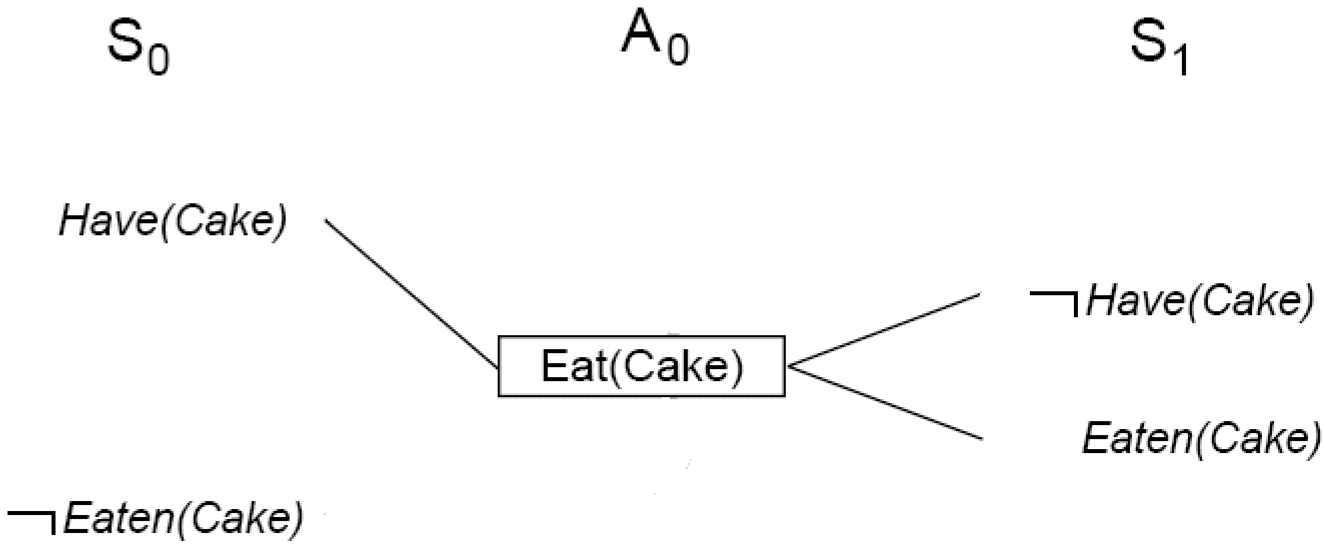
preconditions exist at level i-1

Proposition level i

effects of each no-op and action at level i



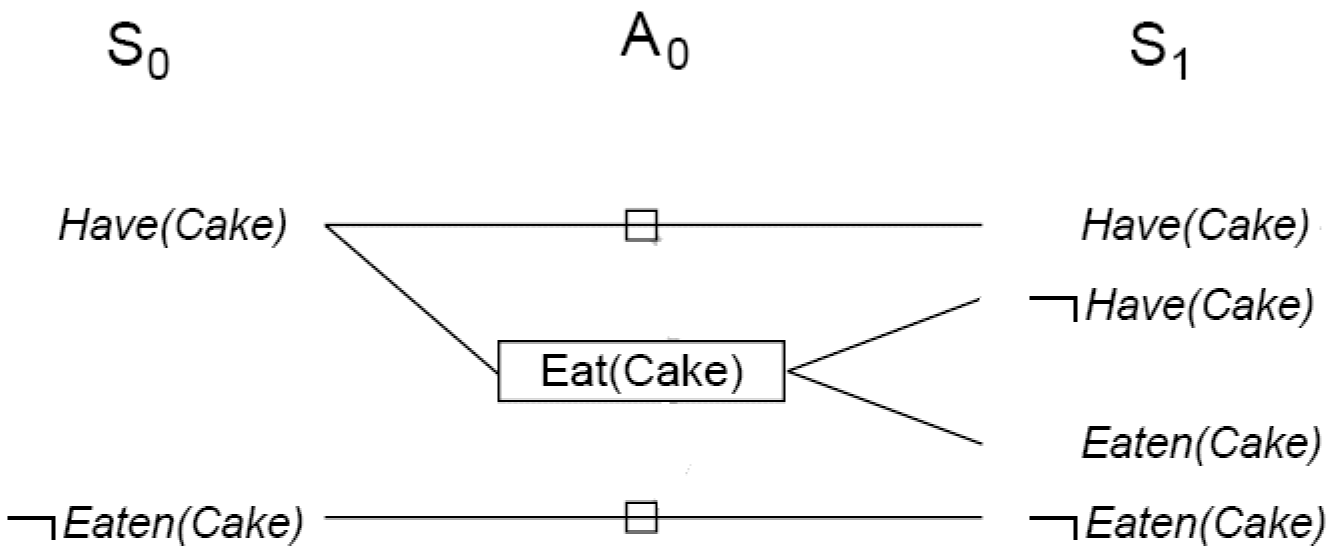
PG Example



Add all applicable actions.

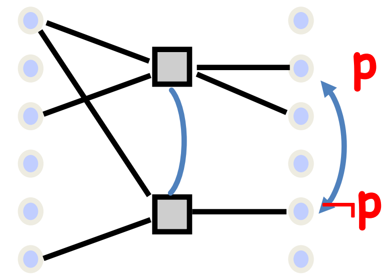
Add all effects to the next state.

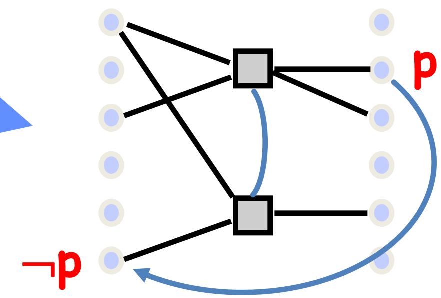
PG Example



Add *persistence actions* (inaction = no-ops) to map all literals in state S_i to state S_{i+1}.

Mutual Exclusion



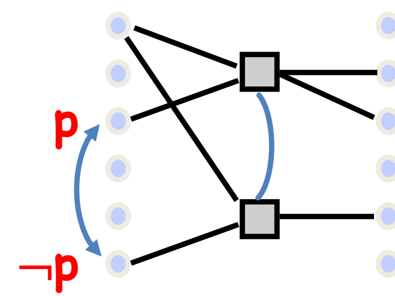


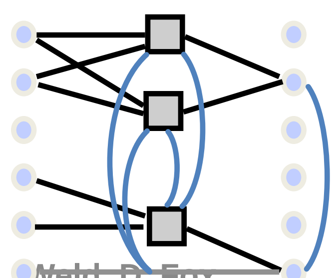
Two actions are mutex if

- one clobbers the other's effects or preconditions
- they have mutex preconditions

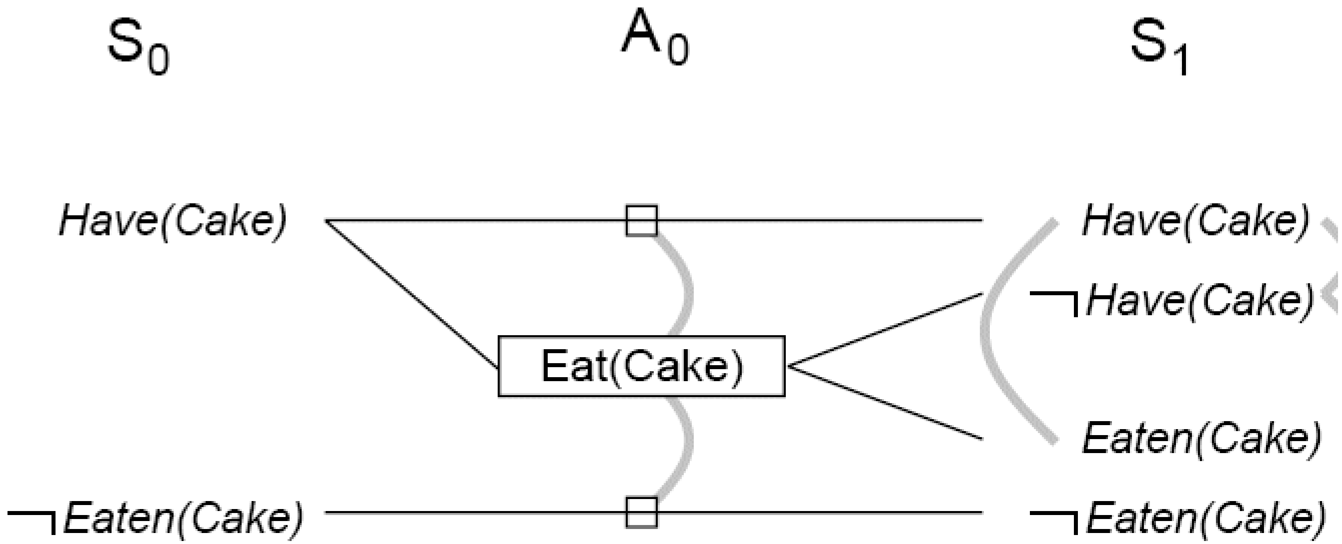
Two proposition are mutex if

- one is the negation of the other
- all ways of achieving them are mutex



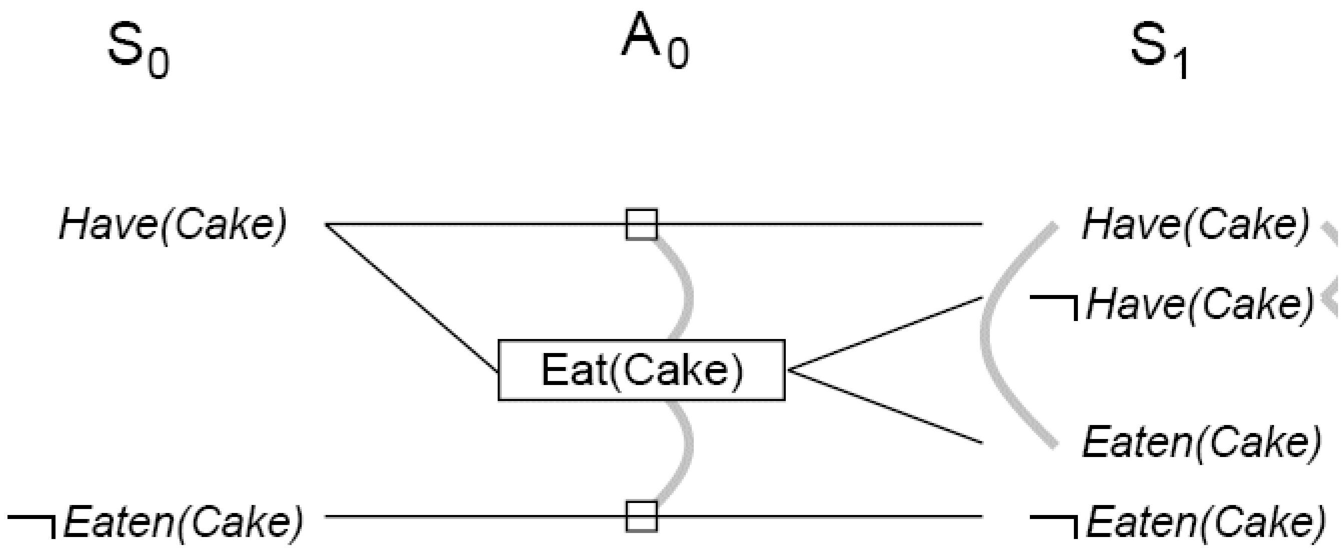


PG Example



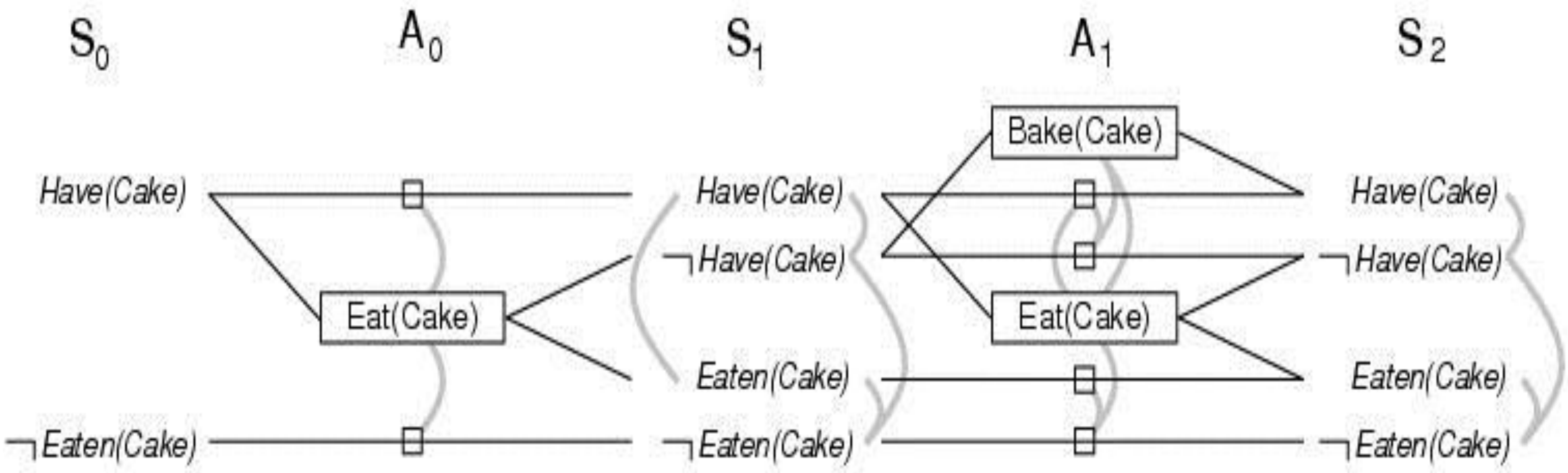
Identify *mutual exclusions* between actions and literals based on potential conflicts.

Cake example

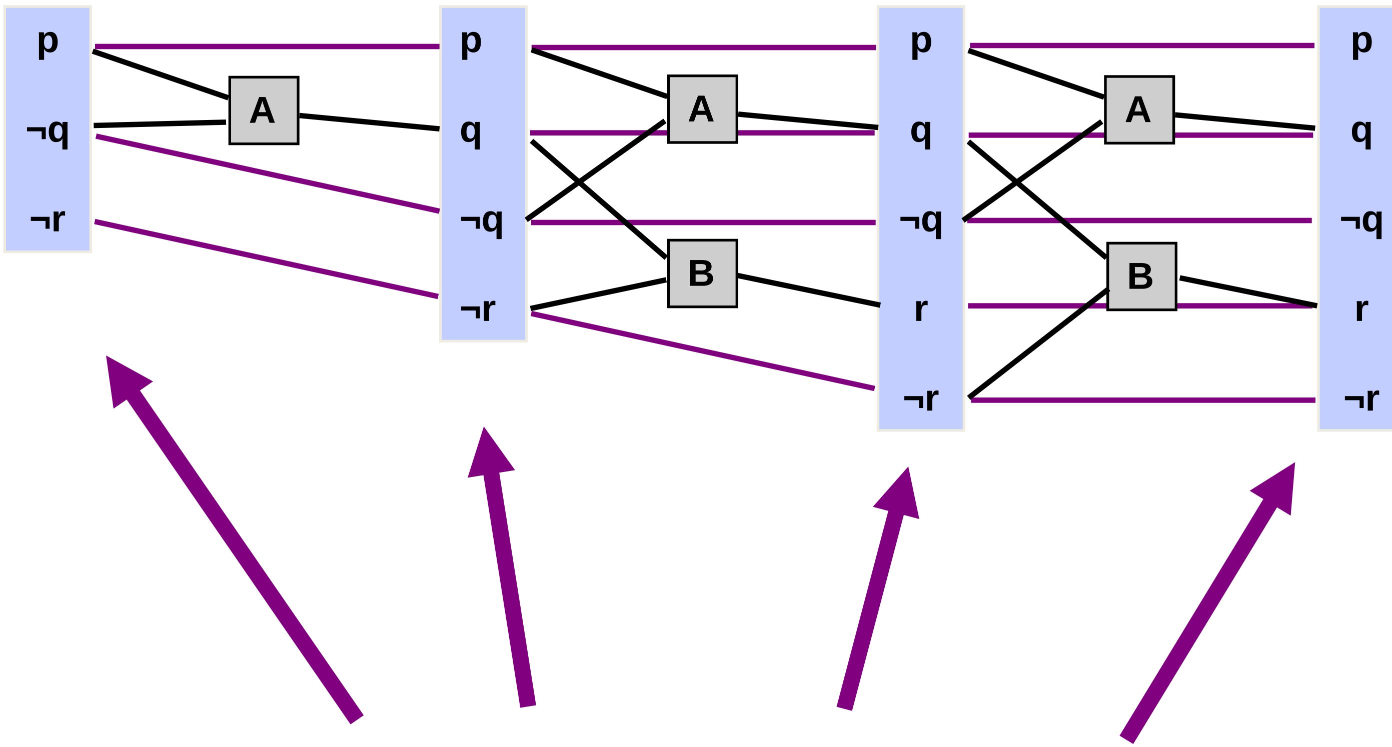


- Level S_1 contains all literals that could result from picking any subset of actions in A_0
  - Conflicts between literals that can not occur together (as a consequence of the selection action) are represented by mutex links.
  - S1 defines multiple states and the mutex links are the constraints that define this set of states.

Cake example



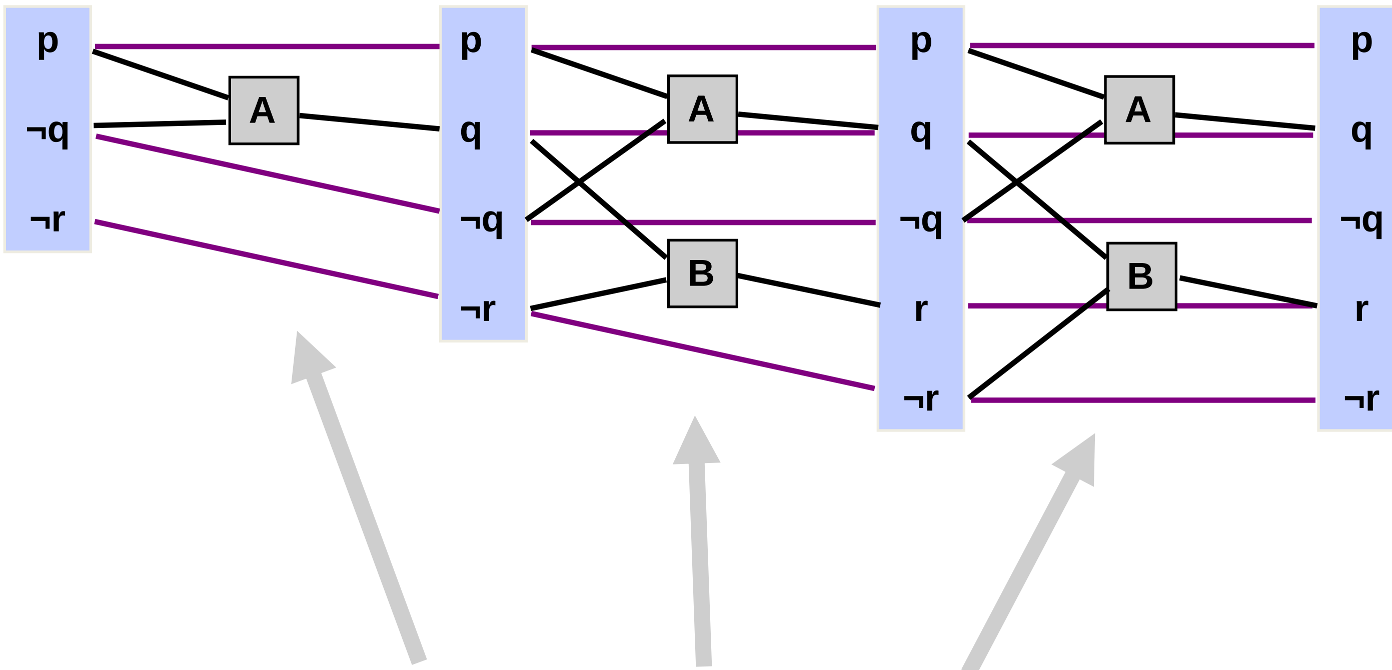
Observation 1



Propositions monotonically increase

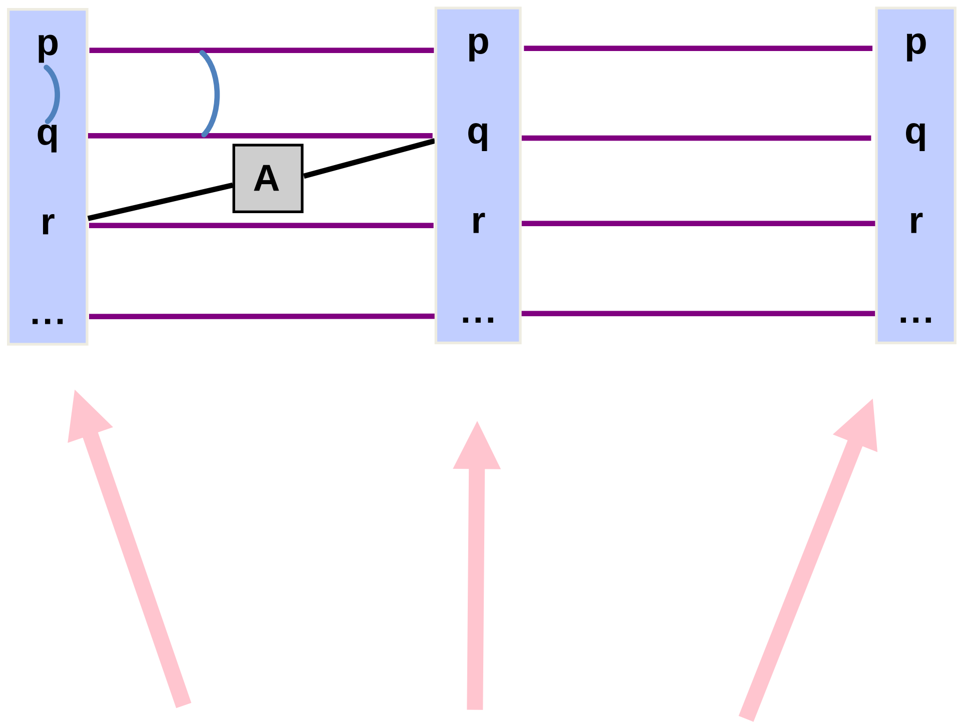
(always carried forward by no-ops)

Observation 2



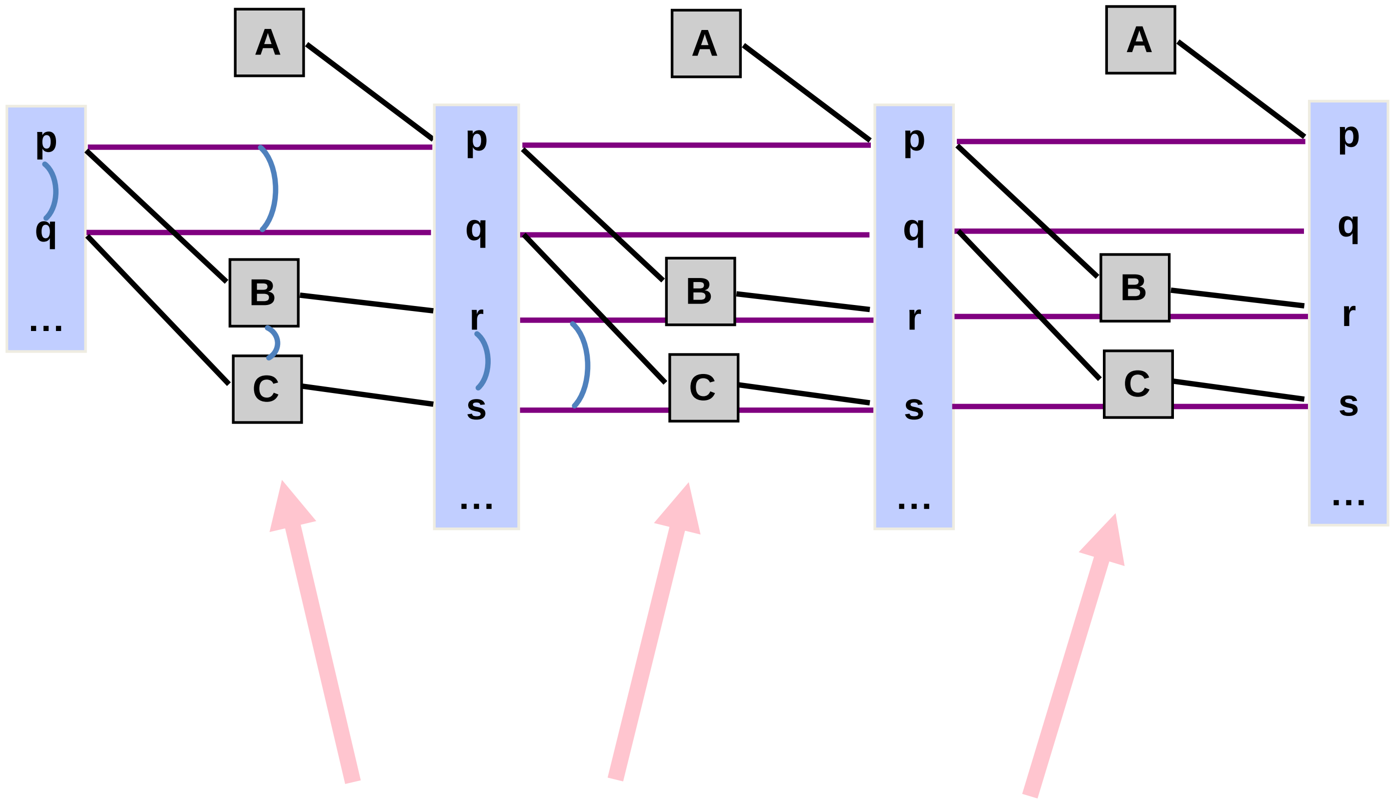
Actions monotonically increase

Observation 3



Proposition mutex relationships monotonically decrease

Observation 4



Action mutex relationships monotonically decrease

Observation 5

Planning Graph 'levels off'.

- After some time k all levels are identical
- Because it's a finite space, the set of literals never decreases and mutexes don't reappear.

Properties of Planning Graph

- If goal is absent from last level
  - Goal cannot be achieved!
- If there exists a path to goal

  goal is present in the last level

- If goal is present in last level

  there may not exist any path still

Heuristics based on Planning Graph

- Construct planning graph starting from s
- h(s) = level at which goal appears non-mutex
  - Admissible?
  - YES

- Relaxed Planning Graph Heuristic
  - Remove negative preconditions build plan. graph
  - Use heuristic as above
  - Admissible? YES
  - More informative? NO
  - Speed: FASTER

Popular Application

Planning Summary

- Problem solving algorithms that operate on explicit propositional representations of states and actions.
- Make use of domain-independent heuristics.
- STRIPS: restrictive propositional language
- Heuristic search
  - forward (progression)
  - backward (regression) search [didn’t cover]
- Local search FF [didn’t cover]