

SOME EXTENSIONS OF CLASSICAL PLANNING

CHAPTER 12

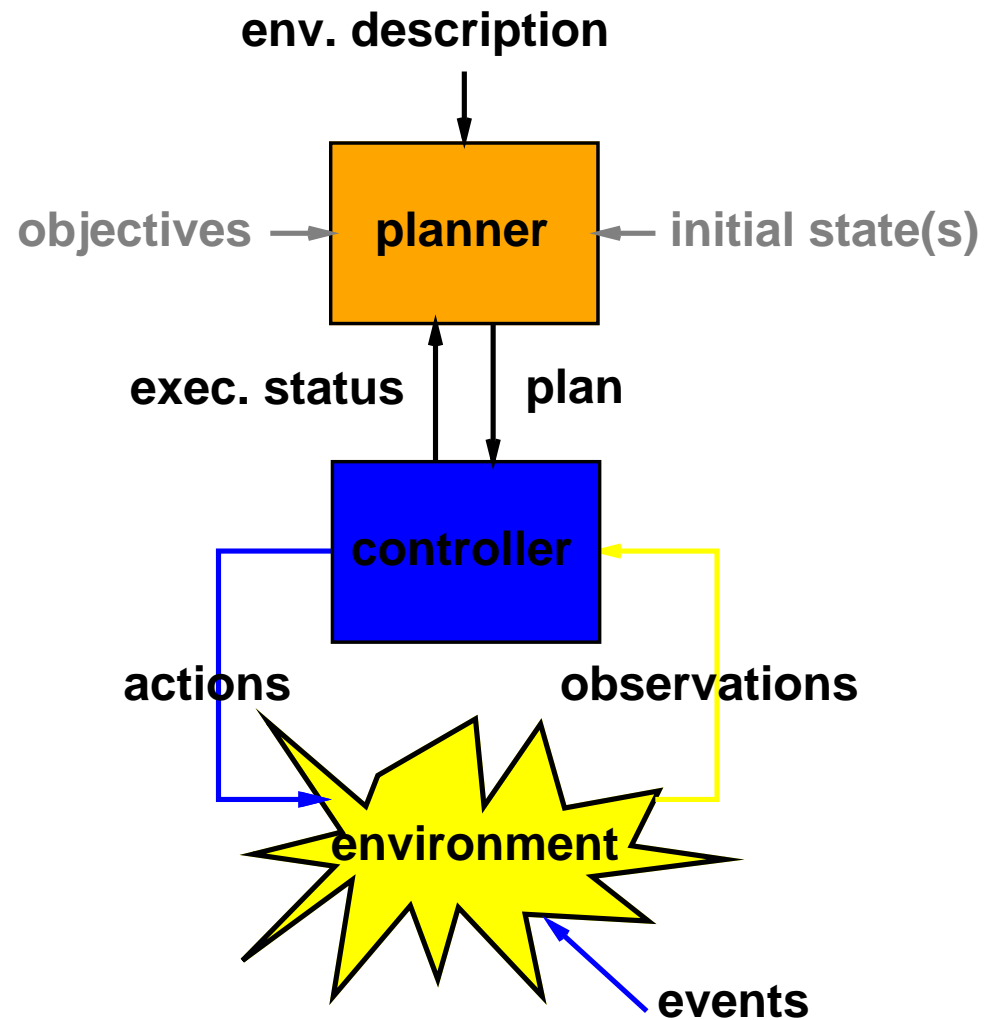
Outline

- ◇ Recall what classical planning is about
- ◇ Time, schedules and resources
- ◇ Nondeterministic planning

Classical planning

A few classical planning assumptions:

- static, single agent
- determinism
- full observability
- implicit time
- reachability goals
- off-line planning



Planning vs scheduling

Two related, often interleaving research areas.

- Planning task:
 - Compute a (e.g., partially ordered) collection of actions
- Scheduling task:
 - Assume the action collection is given (e.g., computed in a previous planning step)
 - Add resources and time to the problem
 - Schedule the actions in such a way that all resource and time constraints are respected
 - Can have cost functions such as minimize total time, minimize total resource consumption...

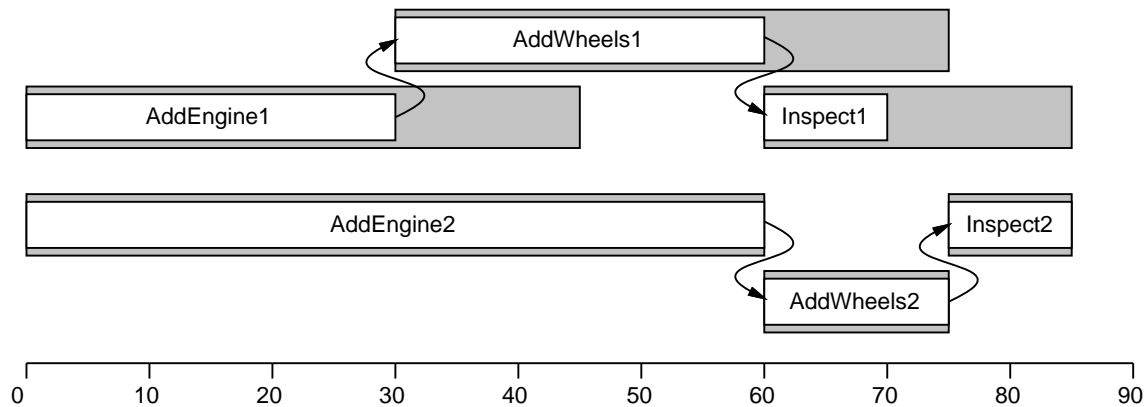
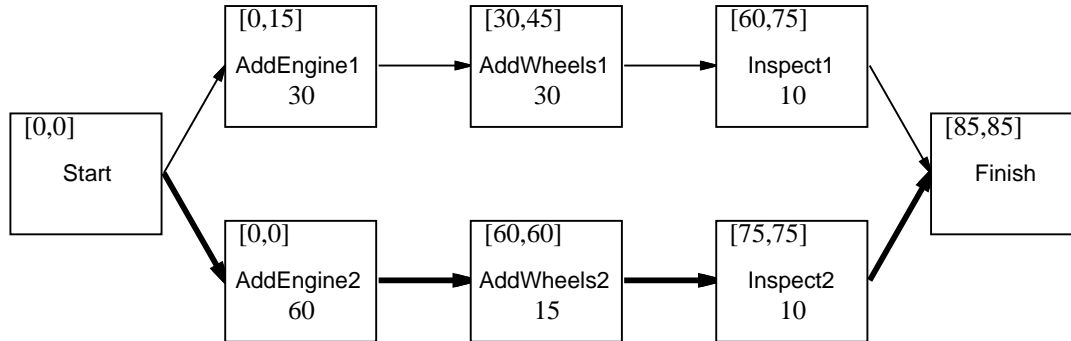
Scheduling and time

- Scheduling task: given a partially ordered solution, determine when each action should begin (i.e., compute a schedule)
- Actions have durations
- A **path** in a partially ordered solution: a totally ordered subset of actions that begins with the dummy action Start and ends with the dummy action Finish (End)
- A **critical path** is a path whose total duration is the longest

A schedule can't be shorter than the duration of a critical path. An optimal schedule has the duration of the critical path (assuming that time is the only resource considered in a scheduling task).

Example

Job Shop Scheduling



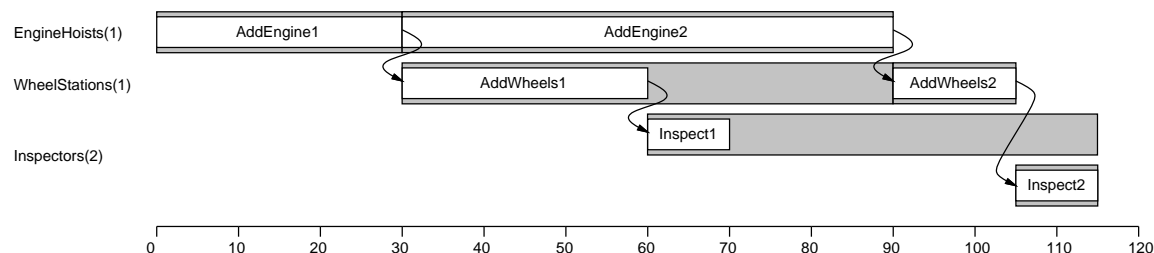
Computing the action starting times

- For each action, determine ES (earliest possible start) and LS (latest possible start)
- Action slack = $LS - ES$
- Schedule (a slightly more precise definition): the ES and LS times for all actions
- Computing ES and LS :
 - $ES(\text{Start}) = 0$
 - $ES(b) = \max_{a \prec b} ES(a) + \text{Duration}(a)$
 - $LS(\text{Finish}) = ES(\text{Finish})$
 - $LS(a) = \min_{a \prec b} LS(b) - \text{Duration}(a)$
- Tractable, polynomial-time computation

Scheduling and resources

So far we only considered time. Now we look at other resources. With resources, finding a minimum-makespan schedule is NP-hard.

- Resources can be:
 - Reusable. E.g., a screwdriver or a hoist
 - Consumable. E.g., fuel
- Resources add new constraints to the scheduling task. E.g., if there is only one hoist to lift engines, two actions that install engines cannot overlap.
- Example: one engine hoist, one wheel station, two inspectors.



Resource aggregation

- Resources can be modeled in classical planning but this can be highly inefficient.
- Example: model 9 car inspectors
 - Classical planning: create 9 constant objects of type inspector.
 - Scheduling: create a new resource and set the available amount to 9.
- Attempt to schedule 10 inspect actions in parallel
 - Classical planning: generate 10! unsuccessful combinations
 - Scheduling: simply note that the amount of this resource is less than required

Nondeterministic planning

- Sources of uncertainty:
 - Incomplete information
 - Incorrect information
 - Dynamic changes in the world
- Modeling uncertainty:
 - Actions have multiple possible effects – pure nondeterminism or probabilistic effects
 - Partial observability – belief states
 - Exogenous events

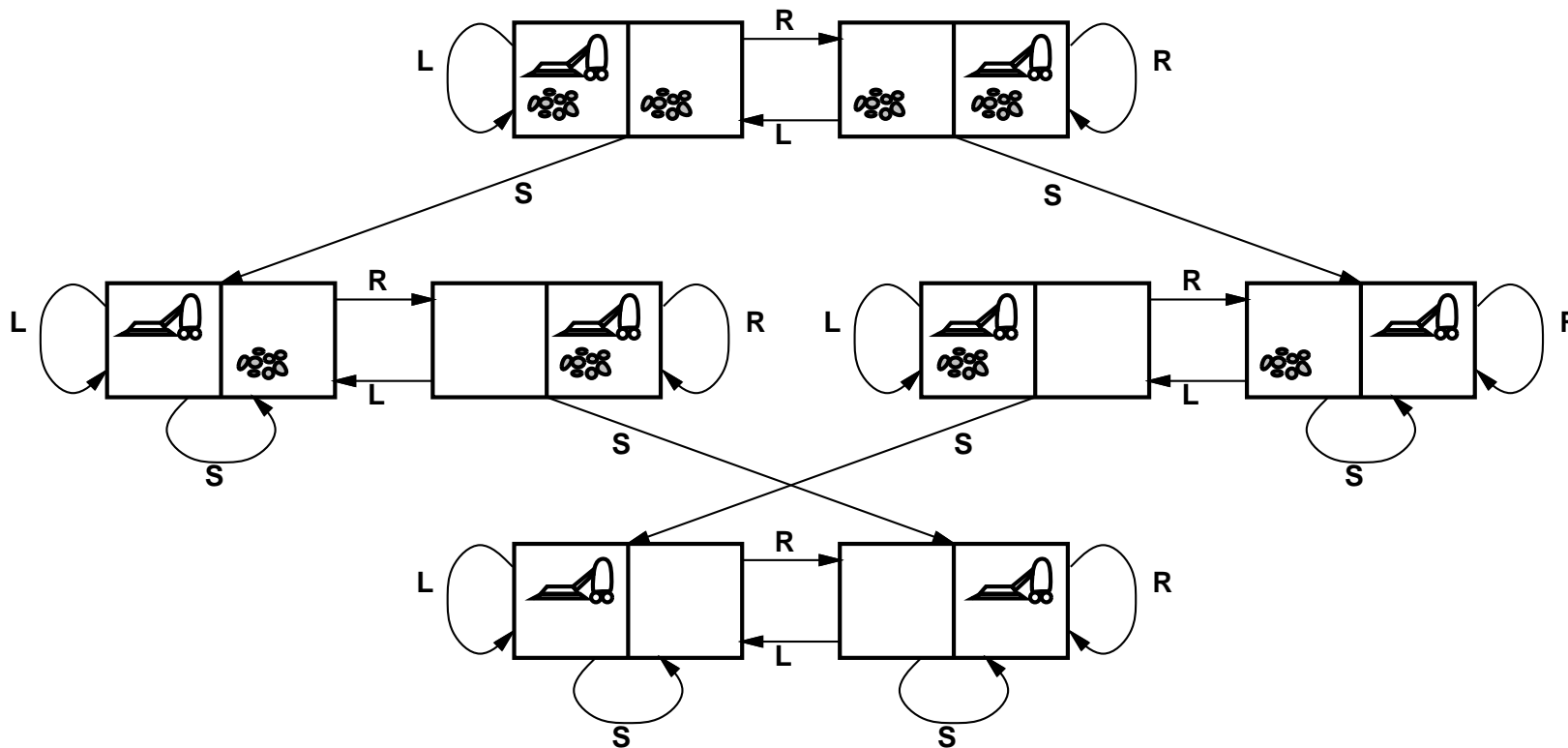
Techniques for nondeterministic planning

- **Conformant (aka sensorless) planning:** build a plan that will work in all possible circumstances. No sensing has to be done during plan execution.
- **Contingency (aka conditional) planning:** plans are trees rather than single sequences. Each branch covers one possible situation.
- **Execution monitoring and replanning:** execute the current plan as long as it is possible. When a conflict occurs (e.g., preconditions of an action do not hold), re-plan from the current state.
- **Continuous planning:** general model that combines an ever-changing goal formulation, dynamic changes in the world, planning, monitoring, execution...

In this lecture we talk about contingency planning.

Running example

A few versions of the **Vacuum world** domain
State space in the deterministic version:



Double Murphy vacuum world

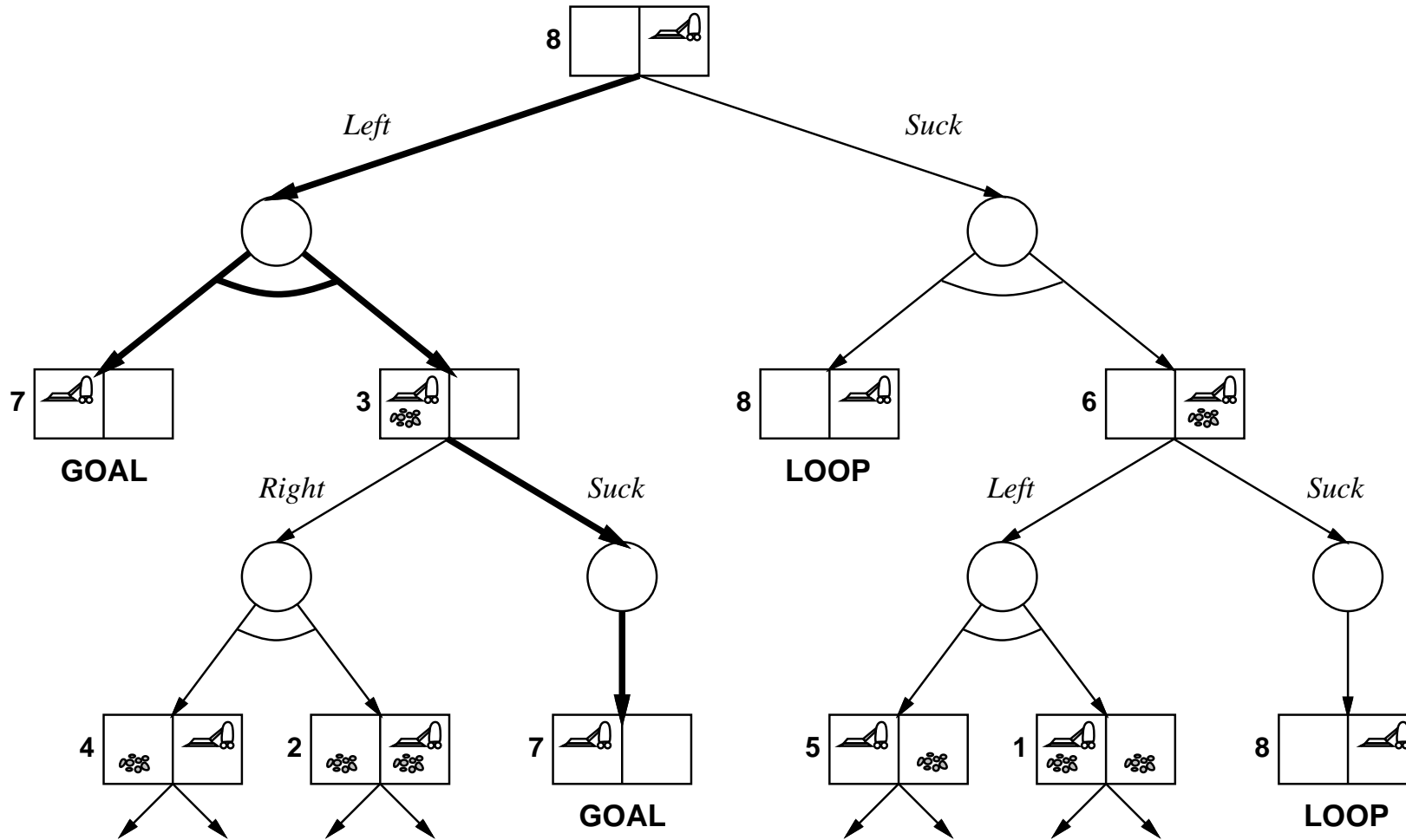
Nondeterministic version of vacuum world:

- When the suck action is applied in a clean room, the vacuum might or might not dump some dirt
- When the destination of a move action is a clear room, the vacuum might or might not dump some dirt in the destination room
- In effect, the previous picture gets more transitions
- Goal: have both rooms clean and have the vacuum in the leftmost room

Contingency planning with full observability

- Actions have non-deterministic (disjunctive) effects
- Thus, resulting state is not known before applying an action
- The planning agent, which can decide what action to apply, is one player
- The nature (opponent), decides the outcomes of actions
- The search space is an AND-OR graph (see example next)
- Each OR node models a (fully known) state
- AND nodes are just branching points, with no states attached
- Solution: tree with all leaves goal states

Example

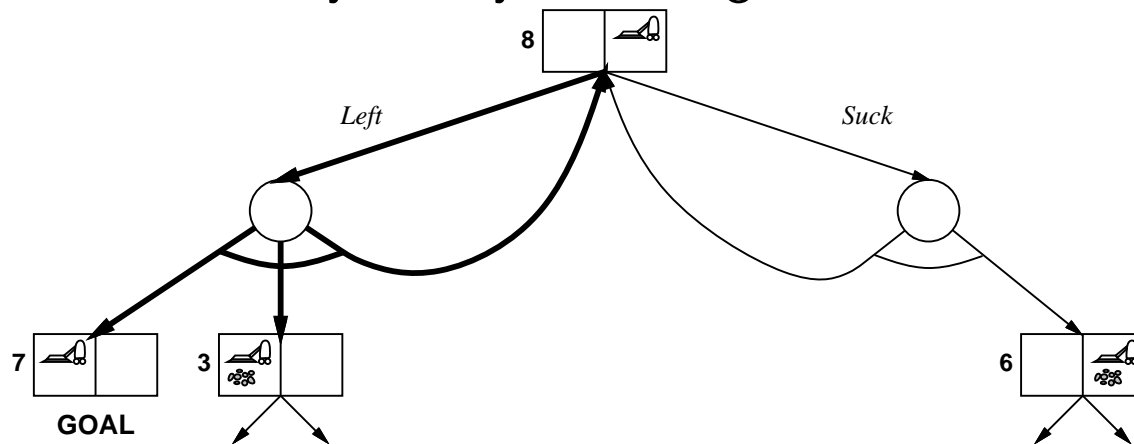


Bold lines show a solution.

Cyclic vs acyclic solutions

- The solution on the previous slide has no cycles
- This guarantees the plan can be executed in a finite number of steps
- In some cases, cyclic solutions exist, but acyclic solutions don't
- Loops can potentially be infinite
- Example: triple Murphy vacuum world = double Murphy plus: "some move actions might have no effects at all"

Below, the Left action may or may not change the location of the vacuum.



Contingency planning with partial observability

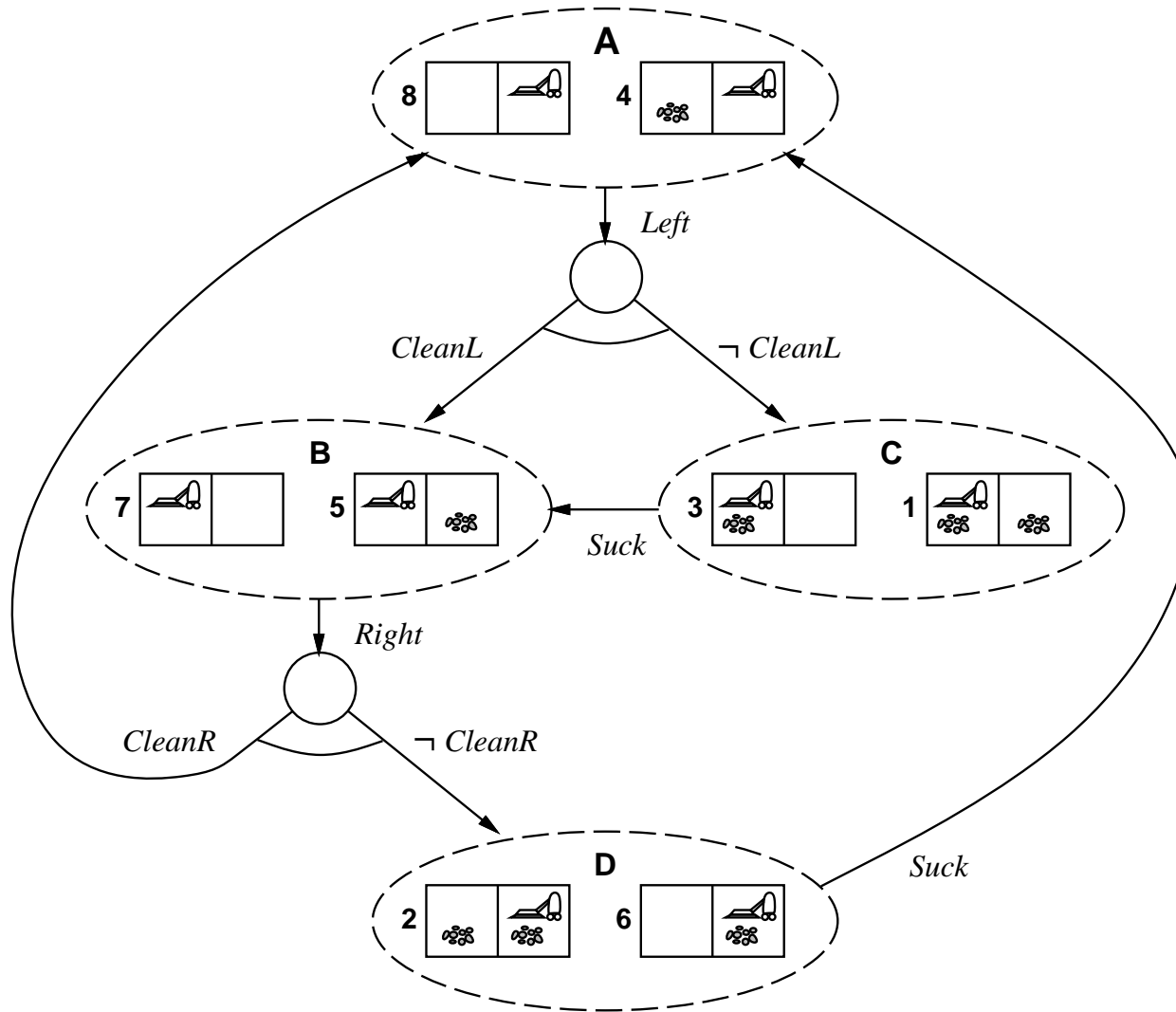
- Current state is not fully determined (partial observability)
- **Belief state**: collection of states in which the system might be now
- Search space: AND-OR graph
- Each OR node models a belief state
- Size blow-up: for $|S|$ states, there are $2^{|S|}$ belief states
- Goal belief state: all contained states must be goal states

Example

Alternate double Murphy:

- Partial observability: The agent can sense whether the current room is clean, but it does not know the status of the other room
- Nondeterminism: When moving from a clean room, dirt might be dumped before leaving
- Goal as before: have both rooms clean and have the vacuum in the leftmost room. Is there a solution?

Example



Full observability vs partial observability

- AND-OR tree in both cases
- Nodes:
 - Single states in the case of full observability
 - Belief states in the case of partial observability
- Challenges in partial observability
 - There can be many more belief states than actual states
 - Represent belief states compactly
 - * Naive, explicit enumeration of contained states
 - * With logical statements that capture what is common in all contained states (e.g., binary decision diagrams BDDs)
 - * With knowledge propositions. These last two options trade expressivity for compactness.