Conditional Planning

Section 12.4

Outline

- Fully observable environments
- Partially observable environments
- Conditional POP

Uncertainty

- The agent might not know what the initial state is
- The agent might not know the outcome of its actions
- The plans will have branches rather than being straight line plans, includes *conditional steps*

  $$\textbf{if } < test > \textbf{ then } plan_A \textbf{ else } plan_B$$

- *Full observability*: The agent knows what state it currently is, does not have to execute an *observation action*
  Simply get plans ready for all possible contingencies

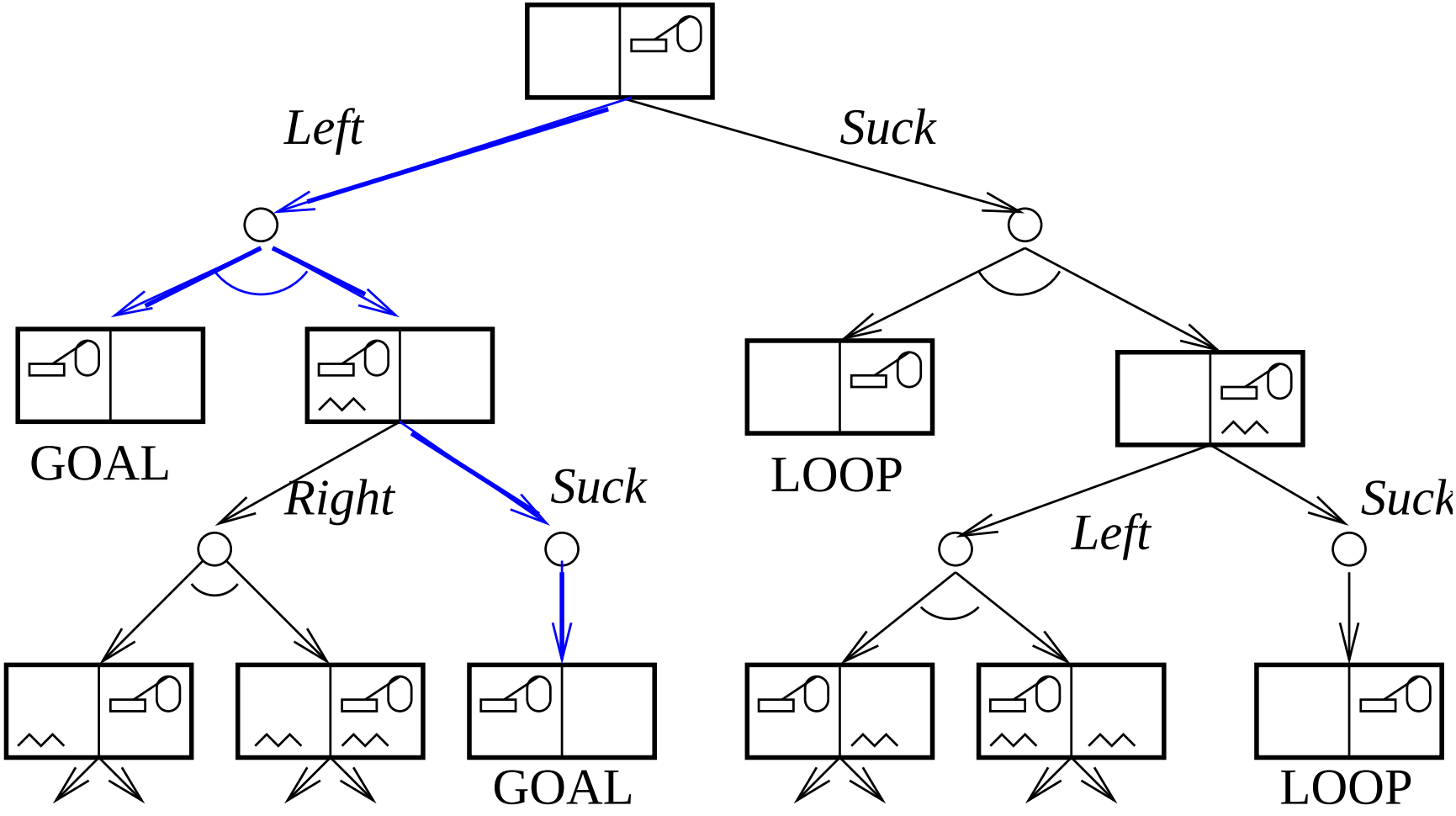
Modeling uncertainty

- Actions sometimes fail $\rightarrow$ disjunctive effects
- Example: moving left sometimes fails
  Action(*Left*,PRECOND: *AtR*,EFFECT: *AtL* $\vee$ *AtR*)
- *Conditional effects*: effects are conditioned on secondary preconditions
  Action(*Suck*, PRECOND: ;,
  EFFECT: (**when** *AtL: CleanL*) $\wedge$ (**when** *AtR: CleanR*))
- Actions may have both disjunctive and conditional effects:
  Moving sometimes dumps dirt on the destination square only when that square is clean
  Action(*Left*, PRECOND: *AtR*;,
  EFFECT: *AtL* $\vee$ (*AtL* $\wedge$ **when** *CleanL:* $\neg$ *CleanL*))

The vacuum world example

- Double Murphy world
  - the vacuum cleaner sometimes deposits dirt when it moves to a clean destination square
  - sometimes deposits dirt if SUCK is applied to a clean square
- The agent is playing a game against nature

Perform and-or search



The plan

In the “double-Murphy” vacuum world, the plan is:

[
  Left,
   if *AtL* $\wedge$ *CleanL* $\wedge$ *CleanR*
     then [ ]
     else *Suck*
]

And-or Search Algorithm

function AND-OR-GRAPH-SEARCH (*problem*)
returns *a conditional plan*, or *failure*

OR-SEARCH(INITIAL-STATE[*problem*], *problem*, [])

function OR-SEARCH (*state, problem, path*)
returns *a conditional plan*, or *failure*

if GOAL-TEST[*problem*](*state*) **then return** the empty plan
if *state* is on *path* **then return** *failure*
for each *action, state-set* in SUCCESSORS [*problem*](*state*) **do**
plan $\leftarrow$ AND-SEARCH (*state, problem*, [*state* | *path*])
if *plan* $\neq$ *failure* **then return** [*action* | *plan*]
return *failure*

And-or Search Algorithm

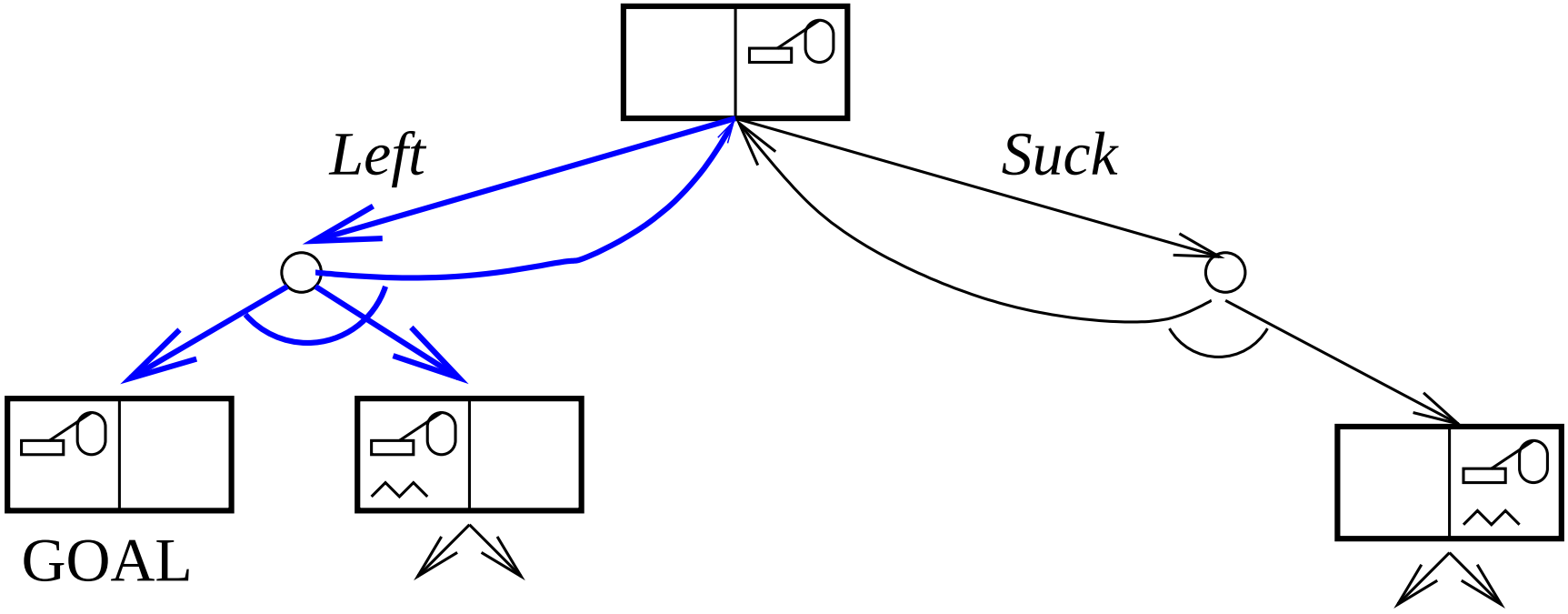
function AND-SEARCH (*state-set, problem, path*)
returns *a conditional plan*, or *failure*

for each s_i **in** *state-set* **do**
  $plan_i \leftarrow$ OR-SEARCH(S_i, *problem, path*)
  if *plan* $=$ *failure* **then return** *failure*
return
[if s_1
  then $plan_1$
  else if s_2
    then $plan_2$
    else . . . if s_{n-1}
      then $plan_{n-1}$
      else $plan_n$**]**

Triple Murhpy vacuum world

- The vacuum cleaner sometimes deposits dirt when it moves to a clean destination square
- It sometimes deposits dirt if *suck* is applied to a clean square
- + move sometimes fails

First level of the search



Triple Murphy vacuum world

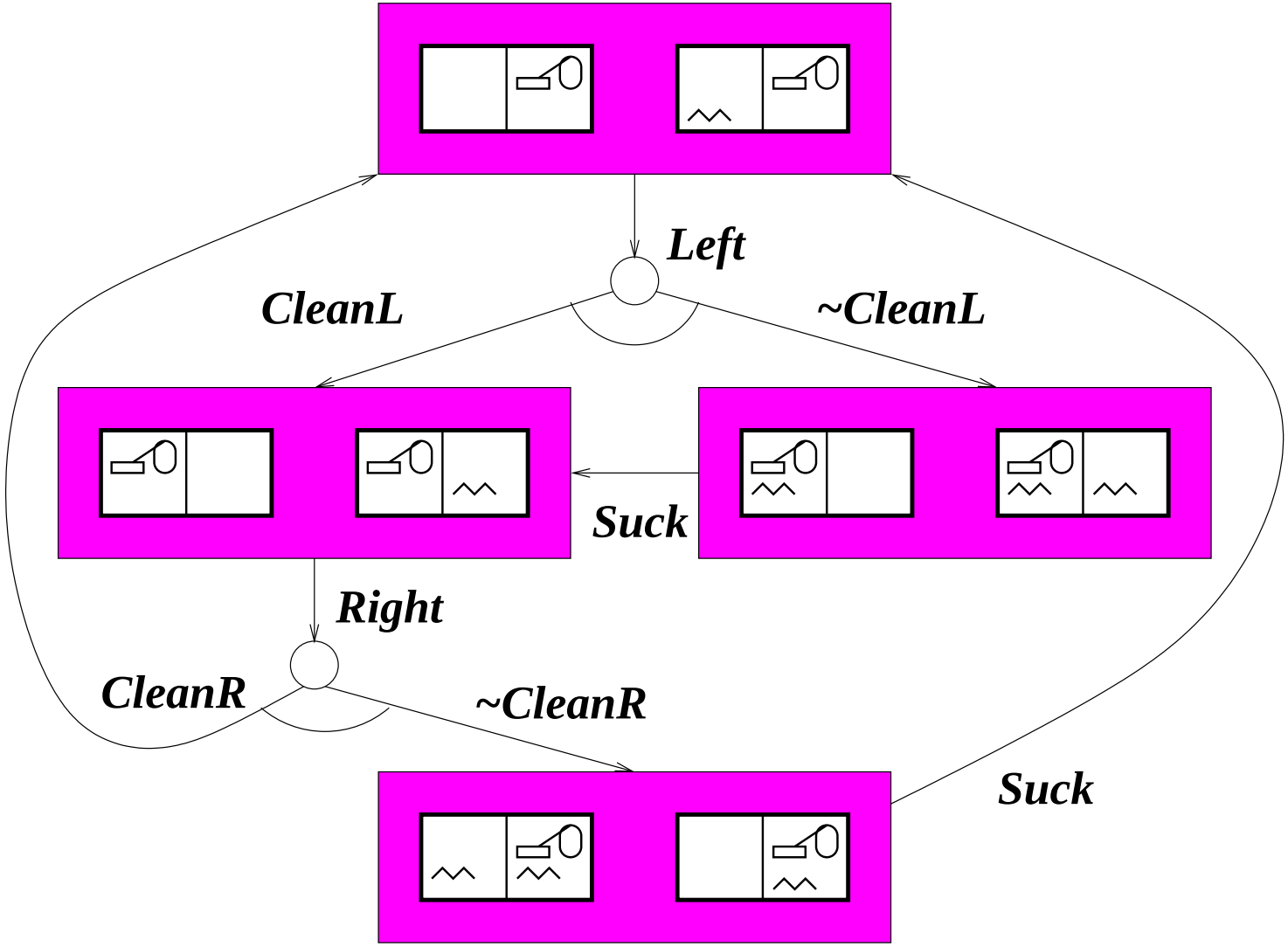
- No acyclic solutions
- A cyclic solution is to try going left until it works. Use a *label*.

[L_1: *Left*, **if** *atR* **then** L_1 **else if** *CleanL* **then** [] **else** *Suck*]

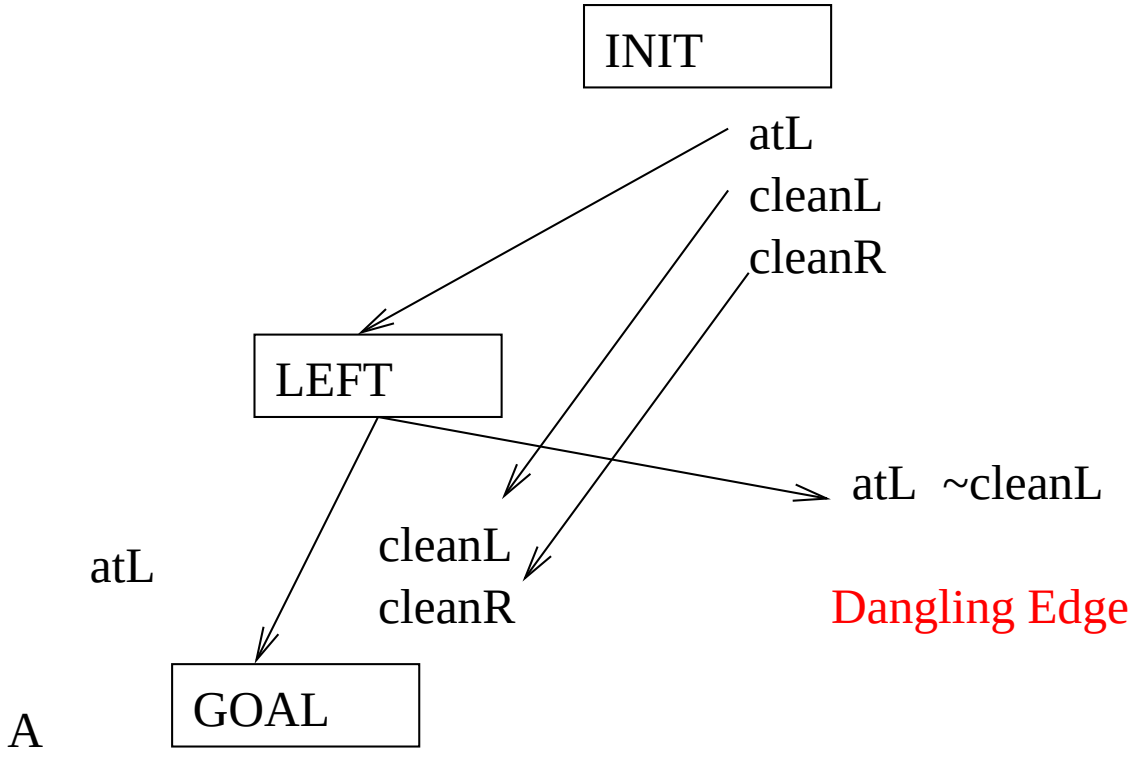
Partially observable environments

- The agent knows only a certain amount of the actual state (e.g., local sensing only, does not know about the other squares)
  - *Automatic sensing*: at every time step the agent gets all the available percepts
  - *Active sensing*: percepts are obtained only by executing specific sensory actions
- *Belief state*: The **set** of possible states that the agent can be in
- "Alternate double Murphy world": dirt can sometimes be left behind when the agent leaves a clean square

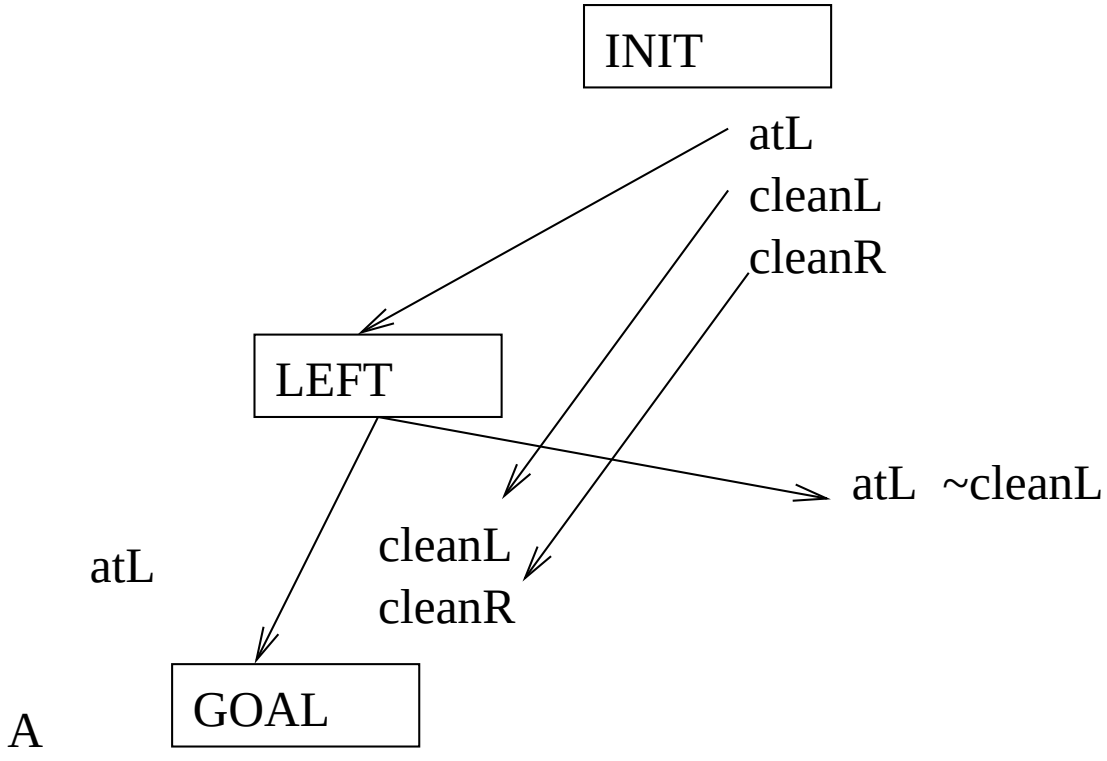
Part of the search



Conditional POP (CNLP algorithm)



Conditional POP (CNLP algorithm)



Duplicate the goal and label it

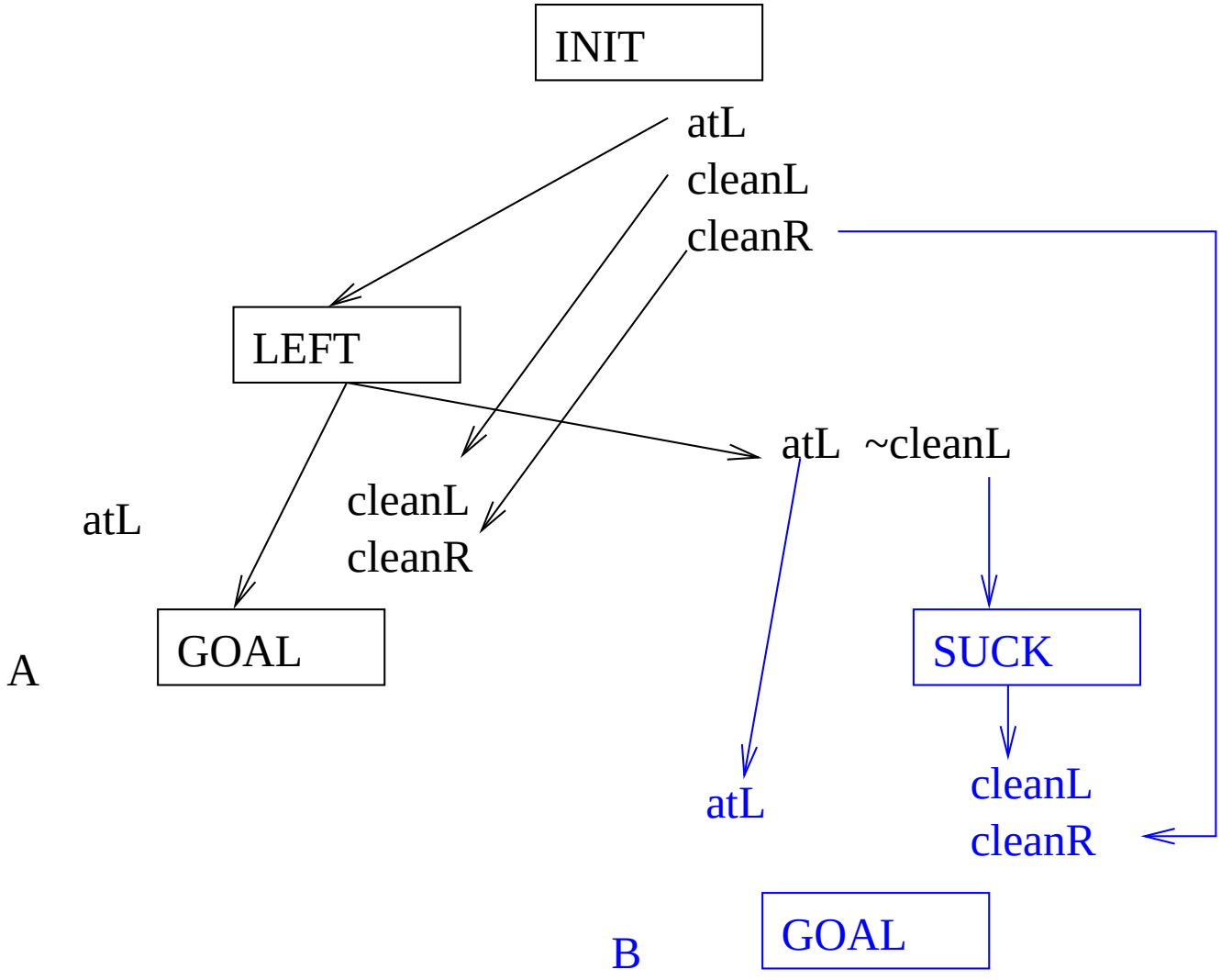
atL

cleanL
cleanR

B

GOAL

Conditional POP (CNLP algorithm)



Comments

- Classical planning is NP
- Conditional planning is harder than NP
- Had to go back to state space search
- Many problems are intractable