

CPS 170: Artificial Intelligence

<http://www.cs.duke.edu/courses/spring09/cps170/>

Planning

Instructor: Vincent Conitzer

Planning

- We studied how to take actions in the world (search)
- We studied how to represent objects, relations, etc. (logic)
- Now we will combine the two!

State of the world (STRIPS language)

- State of the world = conjunction of positive, ground, function-free literals
- At(Home) AND IsAt(Umbrella, Home) AND CanBeCarried(Umbrella) AND IsUmbrella(Umbrella) AND HandEmpty AND Dry
- Not OK as part of the state:
 - NOT(At(Home)) (negative)
 - At(x) (not ground)
 - At(Bedroom(Home)) (uses the function Bedroom)
- Any literal not mentioned is assumed false
 - Other languages make different assumptions, e.g., negative literals part of state, unmentioned literals unknown

An action: TakeObject

- TakeObject(location, x)
- **Preconditions:**
 - HandEmpty
 - CanBeCarried(x)
 - At(location)
 - IsAt(x, location)
- **Effects** (“NOT something” means that that something should be removed from state):
 - Holding(x)
 - NOT(HandEmpty)
 - NOT(IsAt(x, location))

Another action

- WalkWithUmbrella(location1, location2, umbr)
- Preconditions:
 - At(location1)
 - Holding(umbr)
 - IsUmbrella(umbr)
- Effects:
 - At(location2)
 - NOT(At(location1))

Yet another action

- `WalkWithoutUmbrella(location1, location2)`
- **Preconditions:**
 - `At(location1)`
- **Effects:**
 - `At(location2)`
 - `NOT(At(location1))`
 - `NOT(Dry)`

A goal and a plan

- Goal: At(Work) AND Dry
- Recall initial state:
 - At(Home) AND IsAt(Umbrella, Home) AND CanBeCarried(Umbrella) AND IsUmbrella(Umbrella) AND HandEmpty AND Dry
- TakeObject(Home, Umbrella)
 - At(Home) AND CanBeCarried(Umbrella) AND IsUmbrella(Umbrella) AND Dry AND Holding(Umbrella)
- WalkWithUmbrella(Home, Work, Umbrella)
 - At(Work) AND CanBeCarried(Umbrella) AND IsUmbrella(Umbrella) AND Dry AND Holding(Umbrella)

Planning to write a paper

- Suppose your goal is to be a co-author on an AI paper with both theorems and experiments, within a year

LearnAbout(x,y)

Preconditions: HasTimeForStudy(x)

Effects: Knows(x,y),
NOT(HasTimeForStudy(x))

HaveNewIdea(x)

Preconditions: Knows(x,AI),
Creative(x)

Effects: Idea, Contributed(x)

FindExistingOpenProblem(x)

Preconditions: Knows(x,AI)

Effects: Idea

ProveTheorems(x)

Preconditions: Knows(x,AI),
Knows(x,Math), Idea

Effect: Theorems, Contributed(x)

PerformExperiments(x)

Preconditions: Knows(x,AI),
Knows(x,Coding), Idea

Effect: Experiments, Contributed(x)

WritePaper(x)

Preconditions: Knows(x,AI),
Knows(x,Writing), Idea,
Theorems, Experiments

Effect: Paper, Contributed(x)

Goal: Paper AND Contributed(You)

Name a few things that are missing/unrealistic...

Some start states

Start1: HasTimeForStudy(You) AND Knows(You,Math) AND Knows(You,Coding) AND Knows(You,Writing)

Start2: HasTimeForStudy(You) AND Creative(You) AND Knows(Advisor,AI) AND Knows(Advisor,Math) AND Knows(Advisor,Coding) AND Knows(Advisor,Writing)
(Good luck with that plan...)

Start3: Knows(You,AI) AND Knows(You,Coding) AND Knows(OfficeMate,Math) AND HasTimeForStudy(OfficeMate) AND Knows(Advisor,AI) AND Knows(Advisor,Writing)

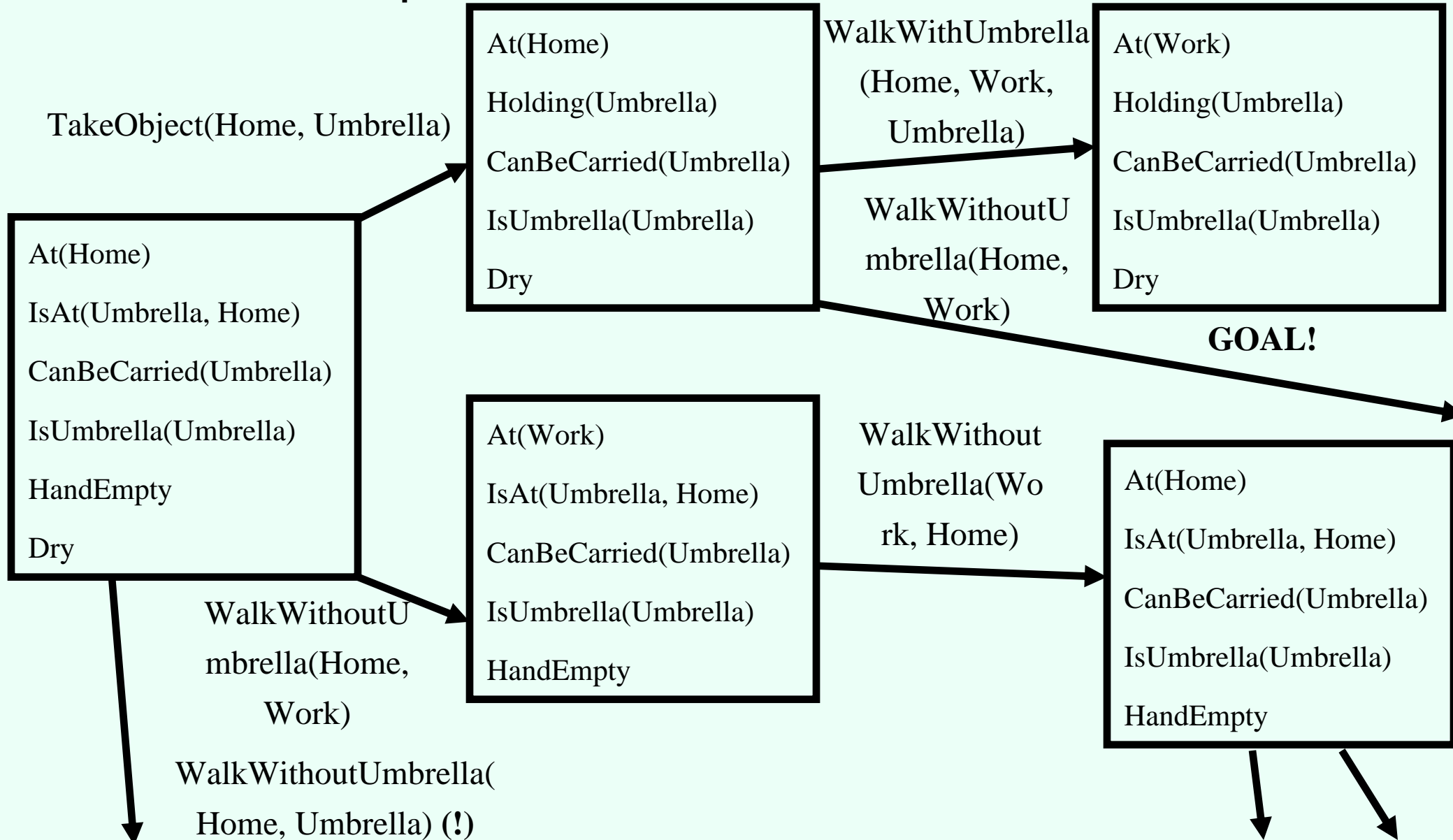
Start4: HasTimeForStudy(You) AND Knows(Advisor,AI) AND Knows(Advisor,Math) AND Knows(Advisor,Coding) AND Knows(Advisor,Writing)

We'll use these as examples...

Forward state-space search

(progression planning)

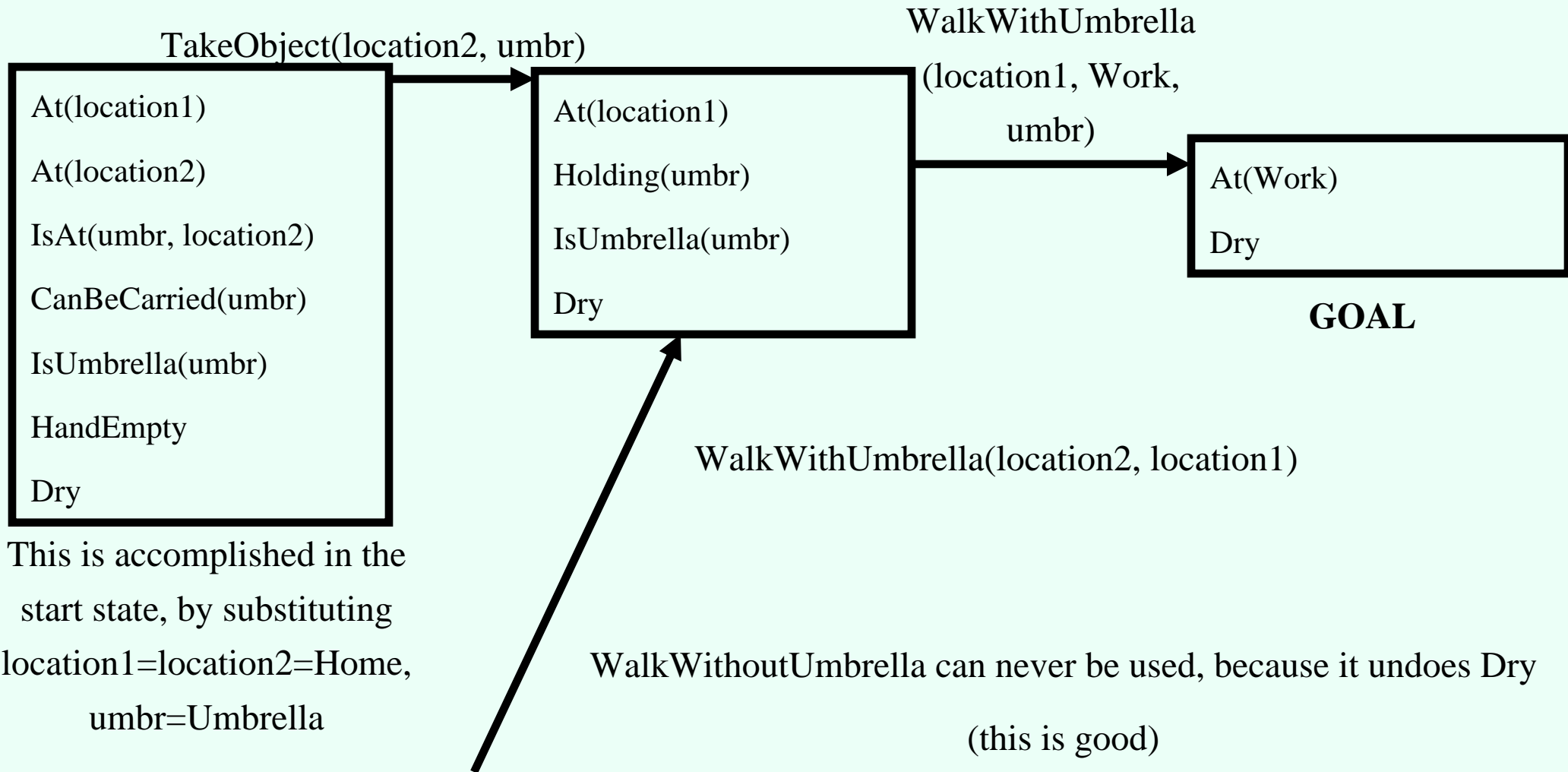
- Successors: all states that can be reached with an action whose preconditions are satisfied in current state



Backward state-space search

(regression planning)

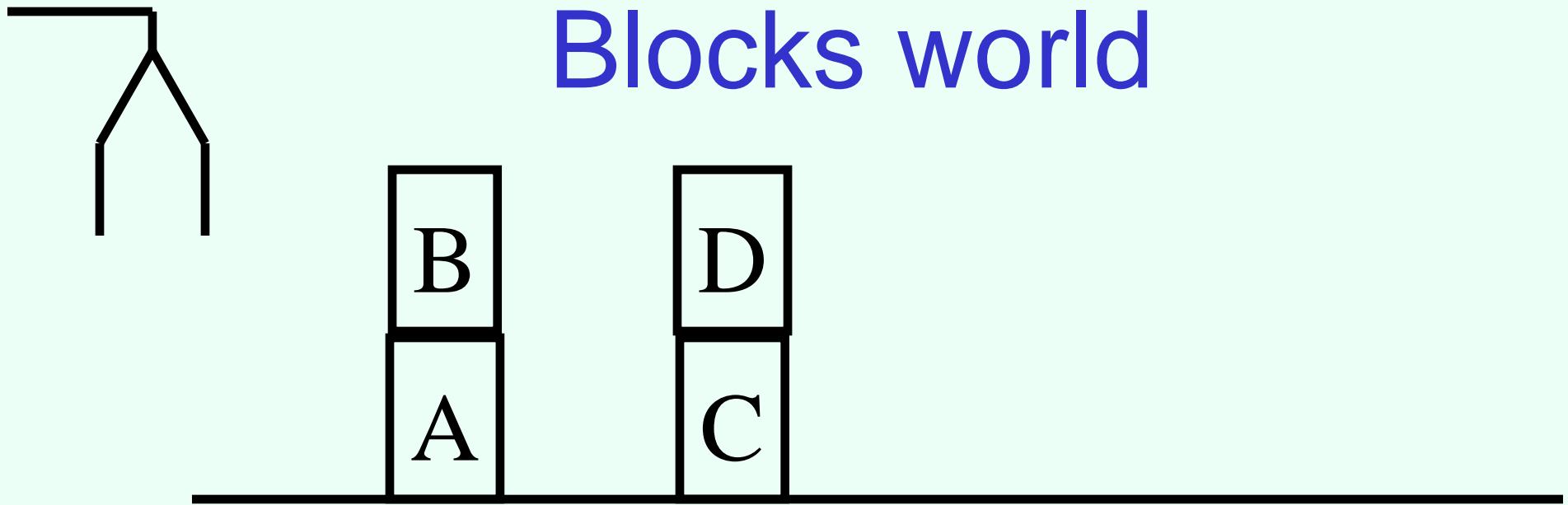
- Predecessors: for every action that accomplishes one of the literals (and does not undo another literal), remove that literal and add all the preconditions



Heuristics for state-space search

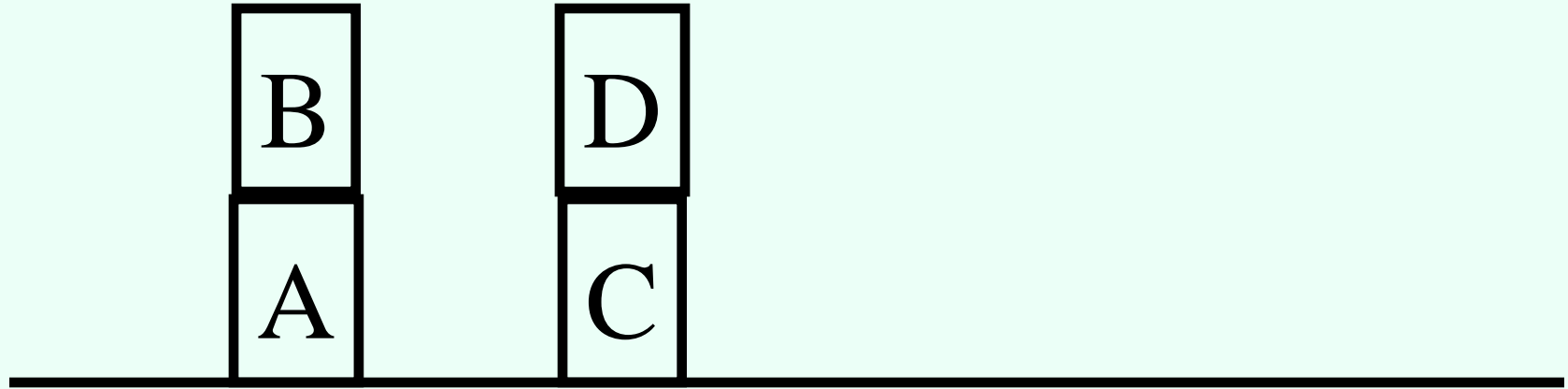
- Cost of a plan: (usually) number of actions
- *Heuristic 1*: plan for each subgoal (literal) separately, sum costs of plans
 - Does this ever underestimate? Overestimate?
- *Heuristic 2*: solve a relaxed planning problem in which actions never delete literals (**empty-delete-list** heuristic)
 - Does this ever underestimate? Overestimate?
 - Very effective, even though requires solution to (easy) planning problem
- Progression planners with empty-delete-list heuristic perform well

Blocks world



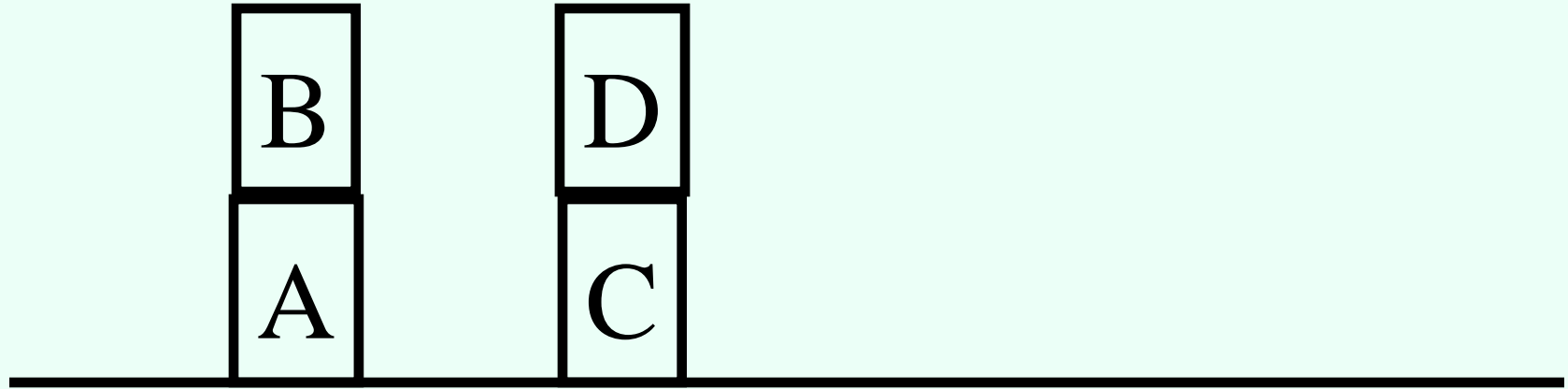
- $\text{On}(B, A)$, $\text{On}(A, \text{Table})$, $\text{On}(D, C)$, $\text{On}(C, \text{Table})$, $\text{Clear}(B)$, $\text{Clear}(D)$

Blocks world: Move action



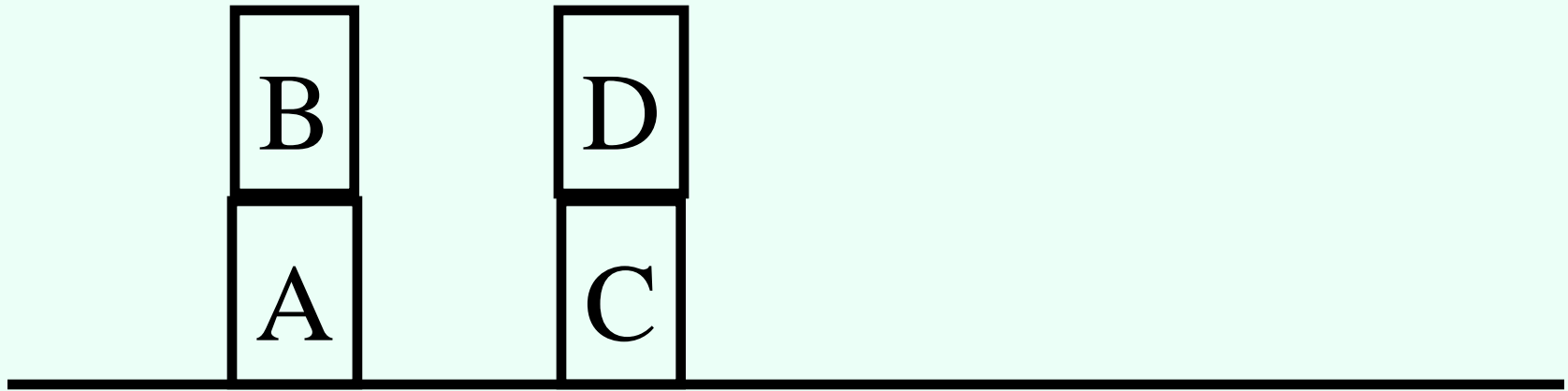
- $\text{Move}(x,y,z)$
- Preconditions:
 - $\text{On}(x,y), \text{Clear}(x), \text{Clear}(z)$
- Effects:
 - $\text{On}(x,z), \text{Clear}(y), \text{NOT}(\text{On}(x,y)), \text{NOT}(\text{Clear}(z))$

Blocks world: MoveToTable action



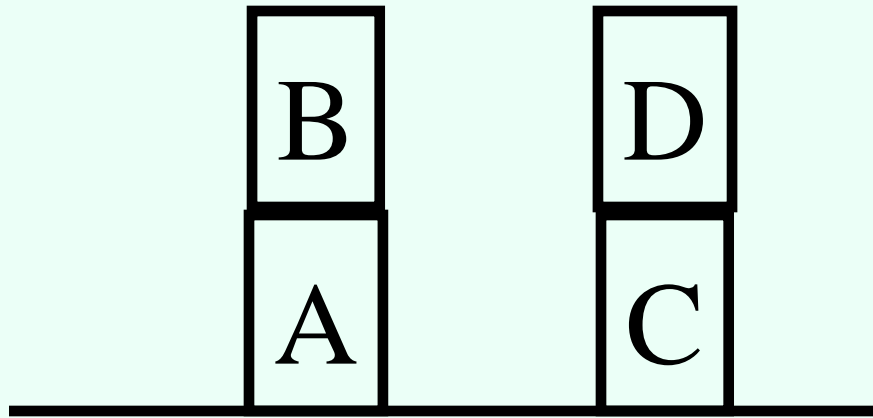
- MoveToTable(x,y)
- Preconditions:
 - On(x,y), Clear(x)
- Effects:
 - On(x,Table), Clear(y), NOT(On(x,y))

Blocks world example

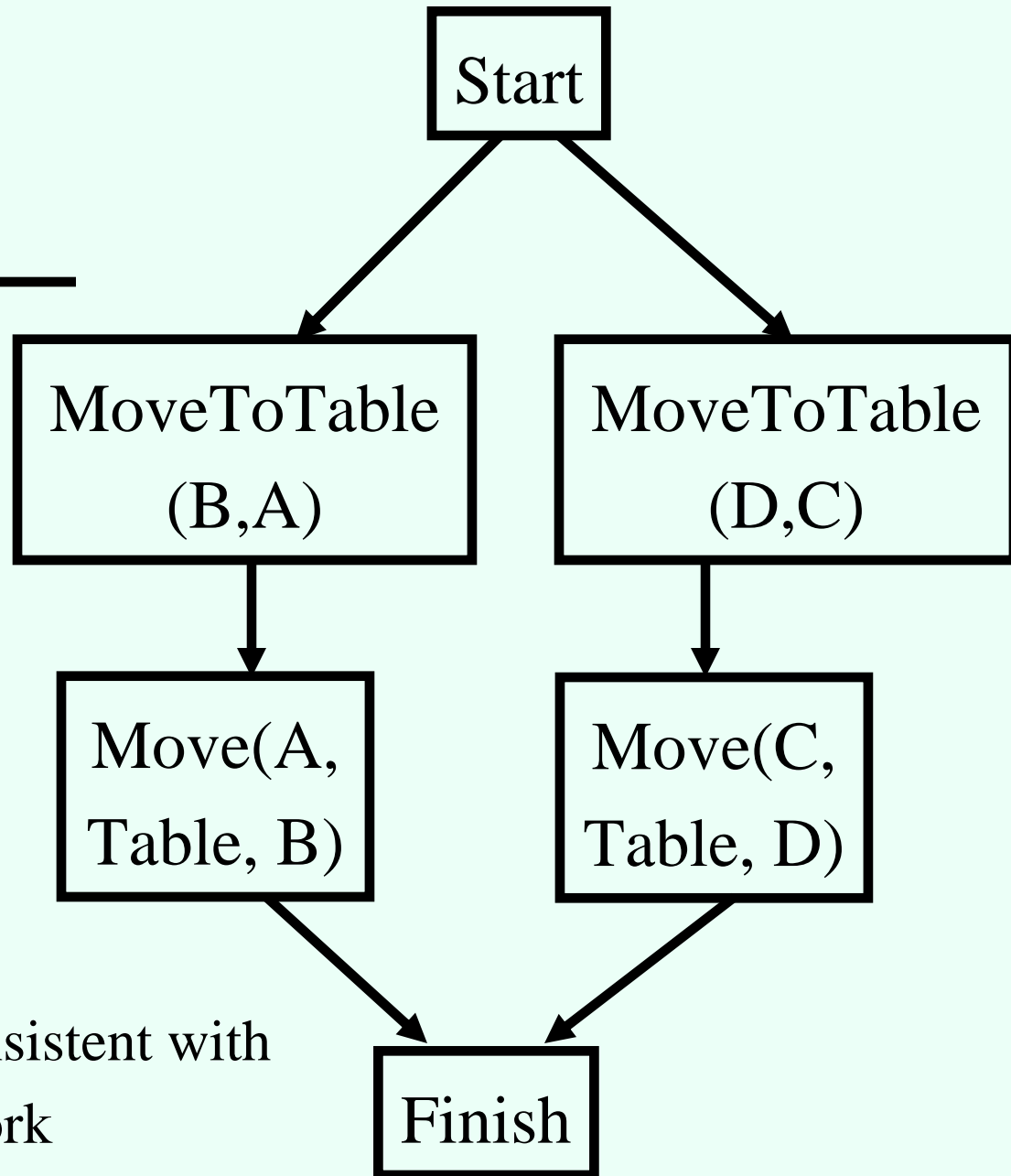


- Goal: $\text{On}(A,B)$ AND $\text{Clear}(A)$ AND $\text{On}(C,D)$ AND $\text{Clear}(C)$
- A plan: $\text{MoveToTable}(B, A)$, $\text{MoveToTable}(D, C)$, $\text{Move}(C, \text{Table}, D)$, $\text{Move}(A, \text{Table}, B)$
- Really two separate problem instances

A partial-order plan

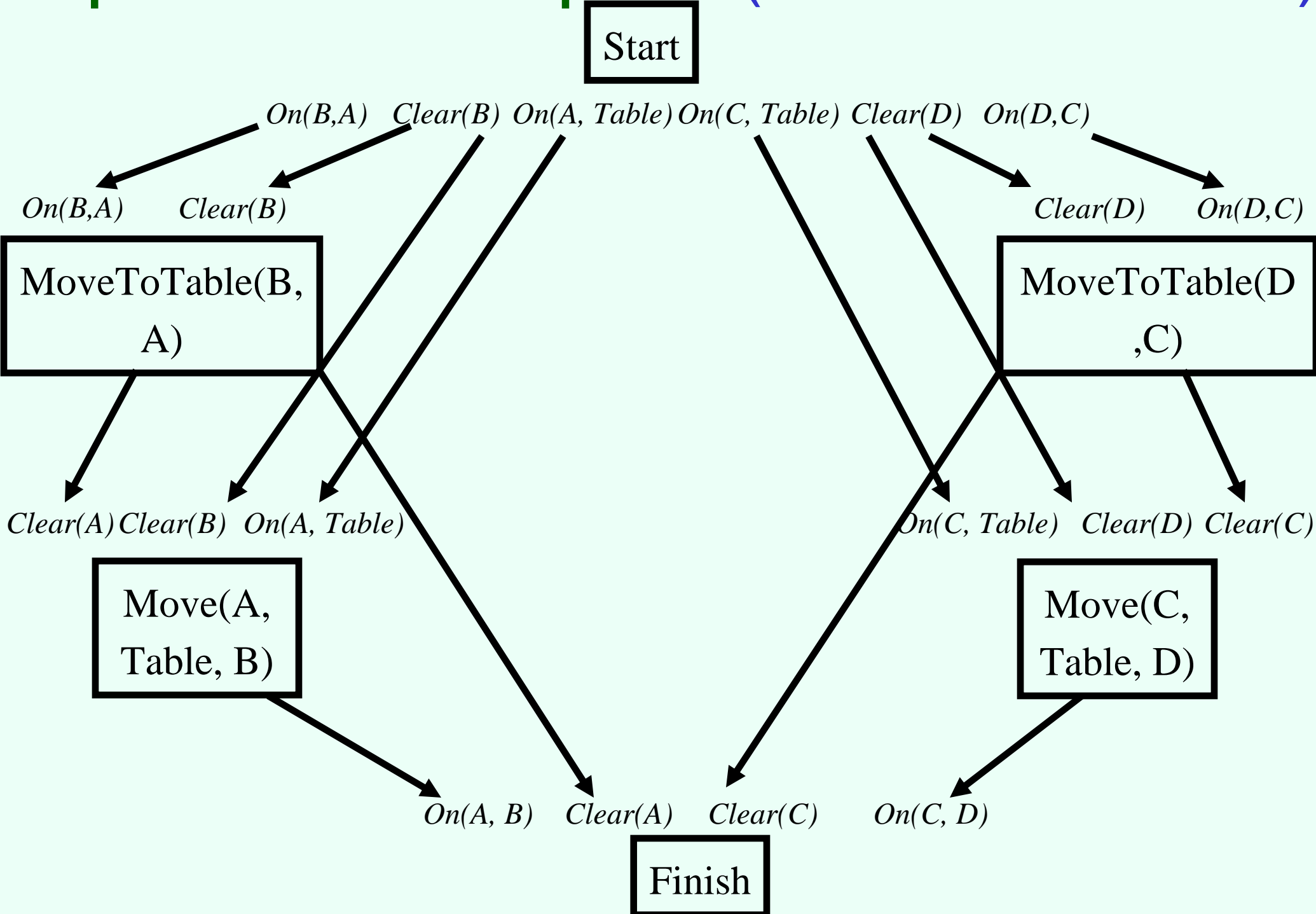


Goal: On(A,B) AND
Clear(A) AND
On(C,D) AND
Clear(C)

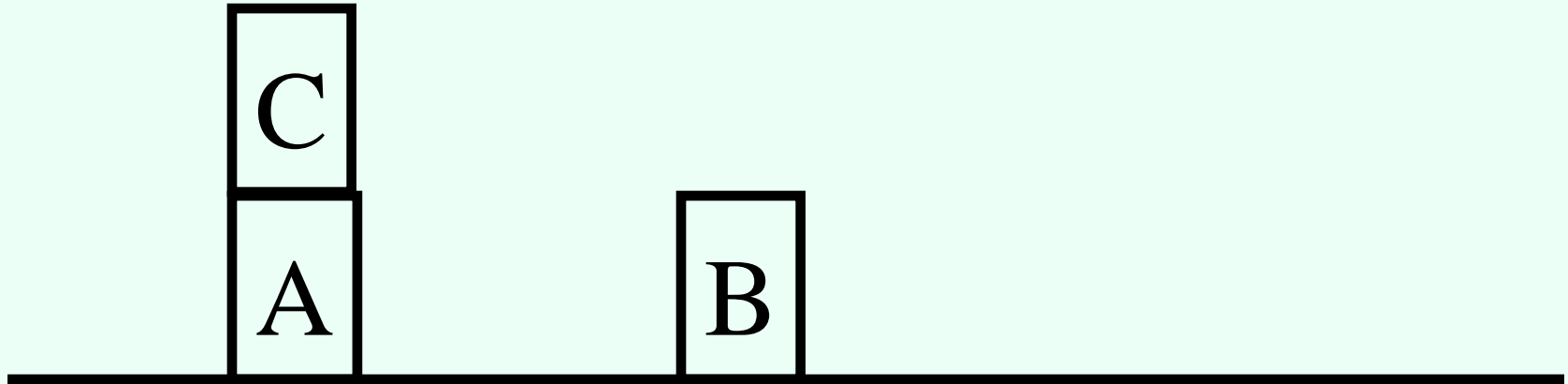


Any total order on the actions consistent with this partial order will work

A partial-order plan (with more detail)

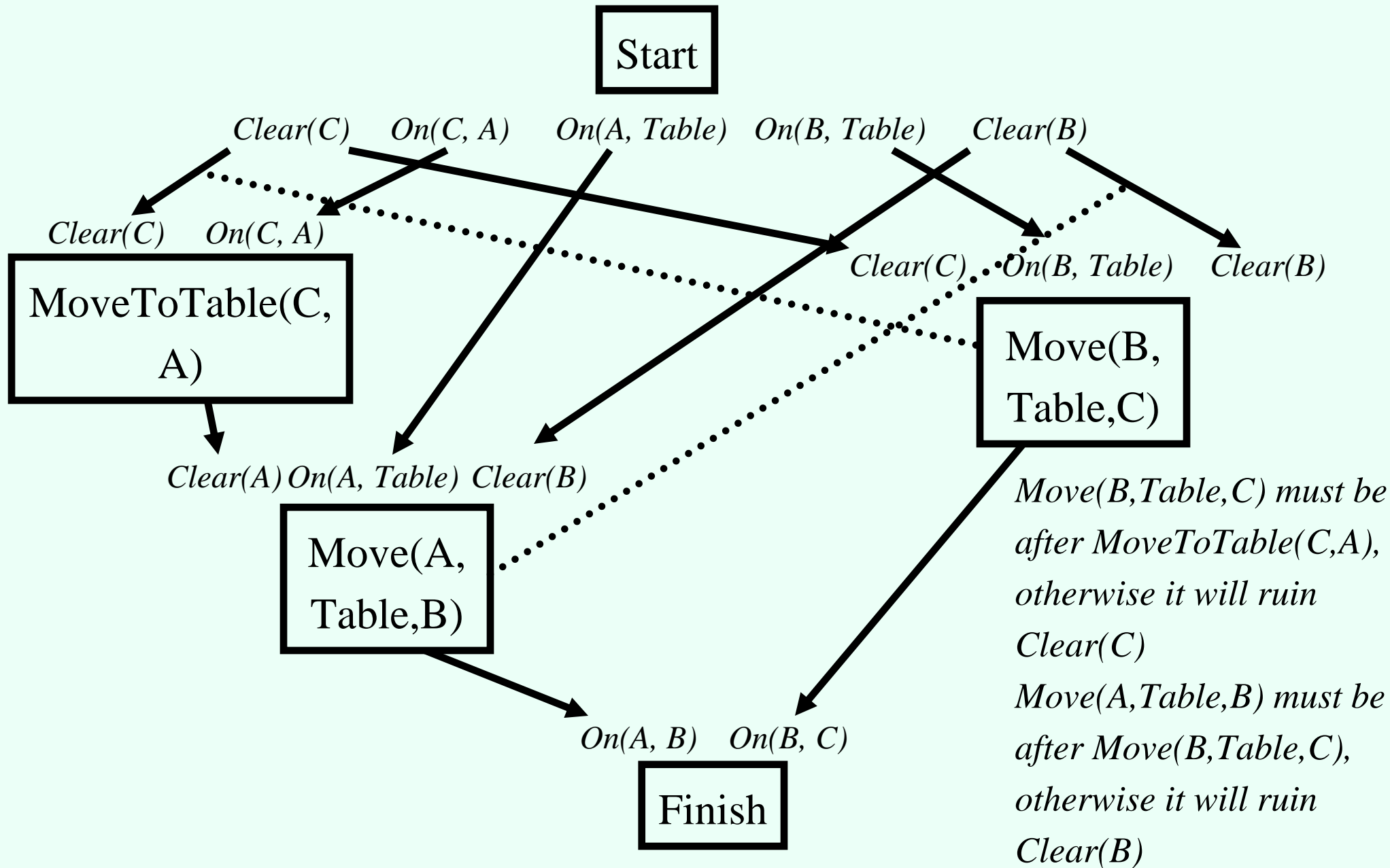


Not everything decomposes into multiple problems: Sussman Anomaly

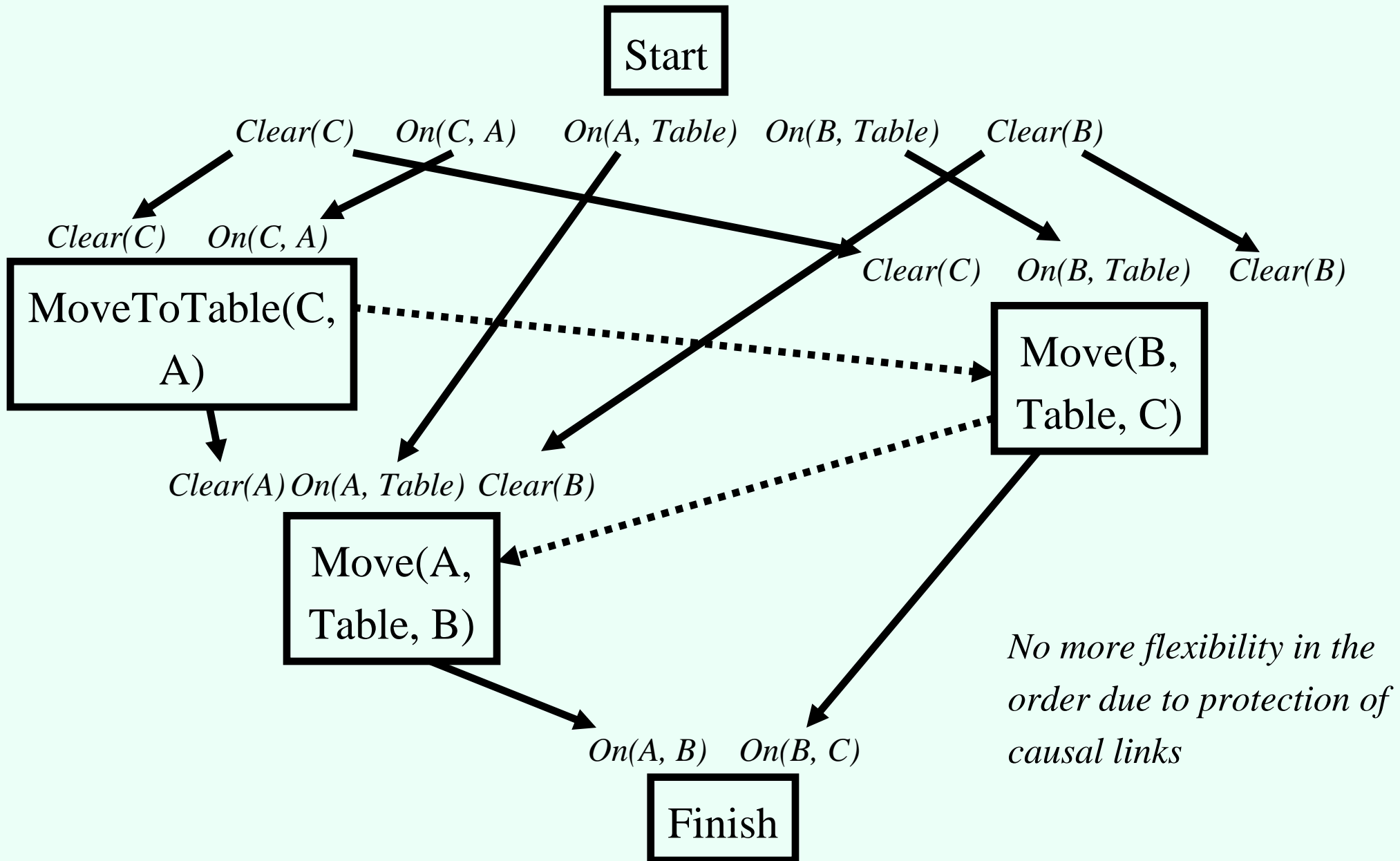


- Goal: $\text{On}(A,B)$ AND $\text{On}(B,C)$
- Focusing on one of these two individually first does not work
- Optimal plan: $\text{MoveToTable}(C,A)$,
 $\text{Move}(B,\text{Table},C)$, $\text{Move}(A,\text{Table},B)$

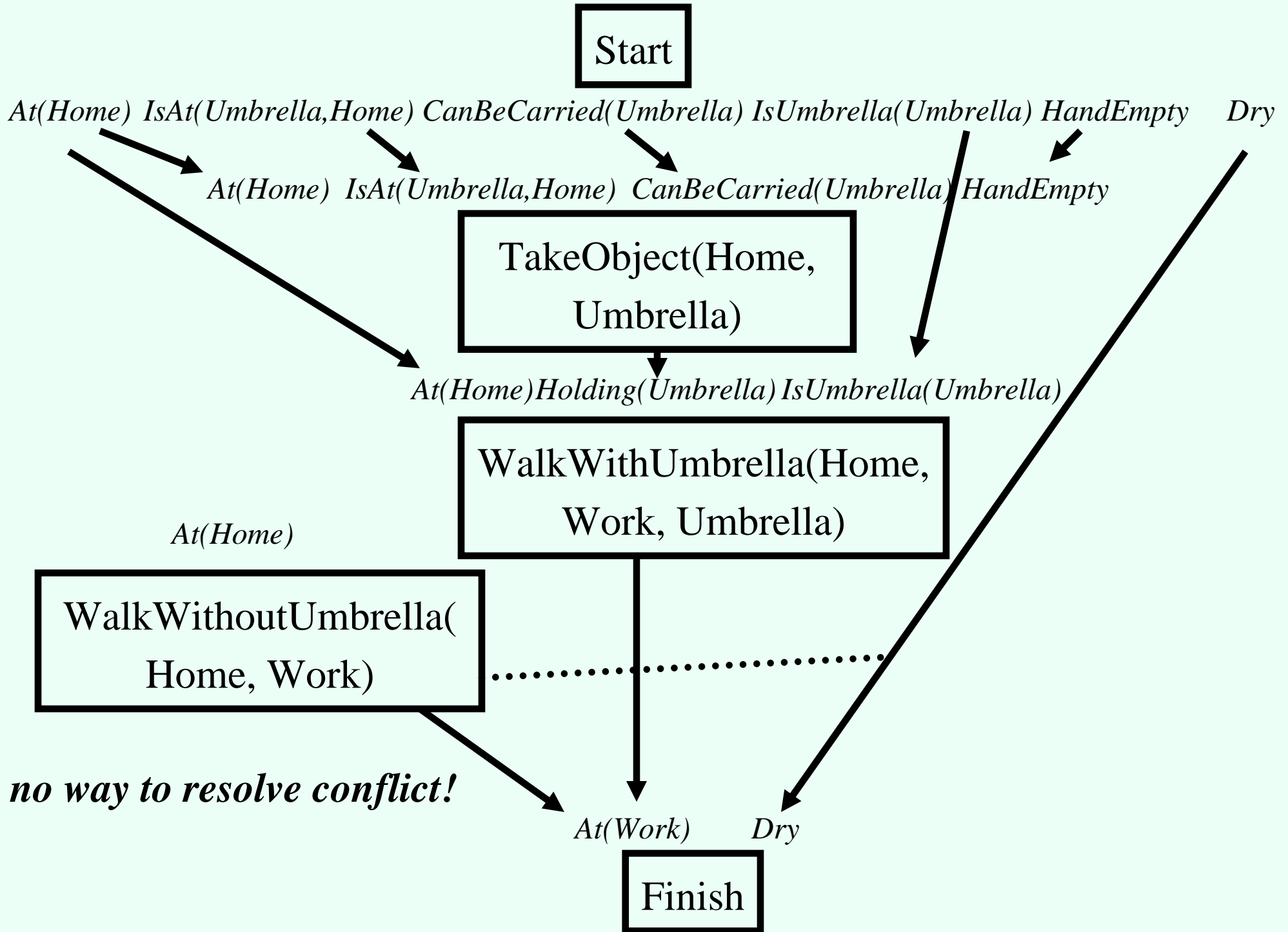
An incorrect partial order plan for the Sussman Anomaly



Corrected partial order plan for the Sussman Anomaly



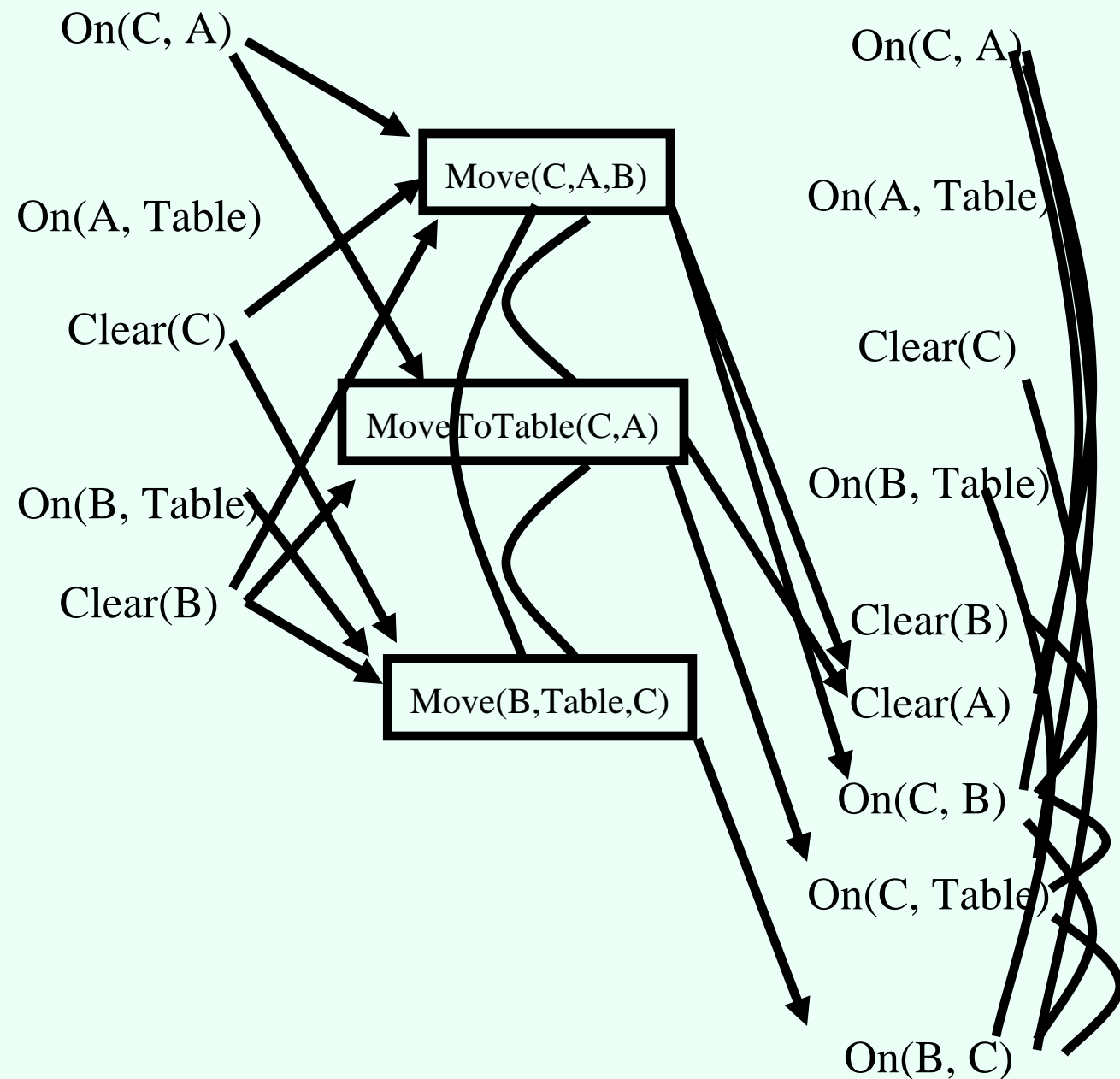
Searching for a partial-order plan



Searching for partial-order plans

- Somewhat similar to constraint satisfaction
- Search state = partially completed partial order plan
 - Not to be confused with states of the world
 - Contains actions, ordering constraints on actions, causal links, some open preconditions
- Search works as follows:
 - Choose one open precondition p ,
 - Consider all actions that achieve p (including ones already in the plan),
 - For each such action, consider each way of resolving conflicts using ordering constraints
- Why do we need to consider only one open precondition (instead of all)? Is this true for backward state-space search?
- Tricky to resolve conflicts if we leave variables unbound
 - E.g., if we use `WalkWithUmbrella(location1, Work, umbr)` without specifying what `location1` or `umbr` is

Planning graphs



- Each level has literals that “could be true” at that level
- **Mutex (mutual exclusion)** relations indicate incompatible actions/literals

... continued on board

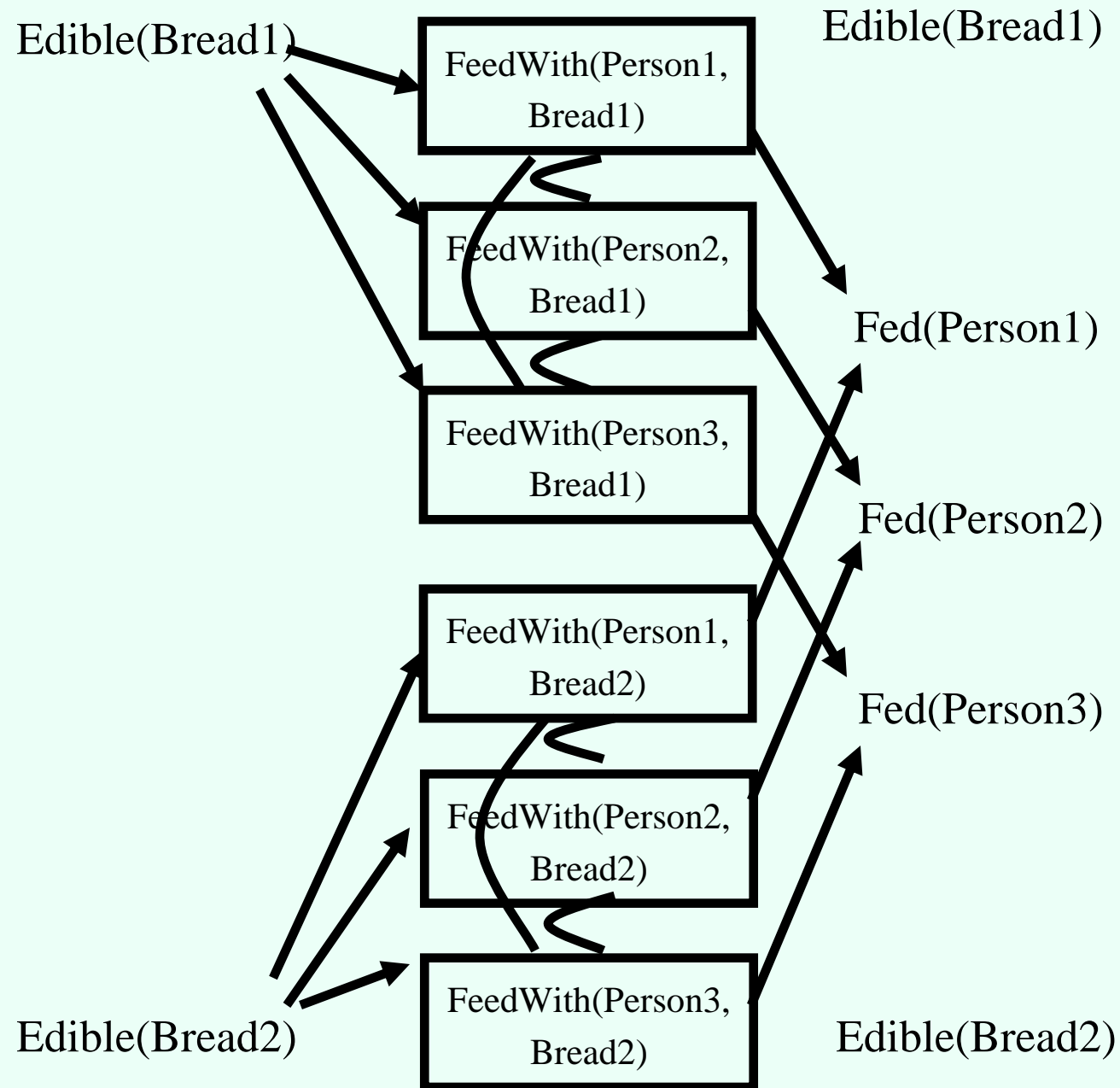
Reasons for mutex relations...

- ... between actions:
 - **Inconsistent effects**: one action negates effect of the other
 - **Interference**: one action negates precondition of the other
 - **Competing needs**: the actions have preconditions that are mutex
- ... between literals:
 - **Inconsistent support**: any pair of actions that can achieve these literals is mutex

A problematic case for planning graphs

- **FeedWith(x, y)**
 - Preconditions: **Edible(y)**
 - Effects: **NOT(Edible(y)), Fed(x)**
- **Start: Edible(Bread1), Edible(Bread2)**
- **Goal: Fed(Person1), Fed(Person2), Fed(Person3)**

Planning graph for feeding



- Any **two** of these could simultaneously be true at time 1, so no mutex relations
- Really need **3-way** mutex relations, but experimentally this is computationally not worthwhile

Uses of planning graphs

- If the goal literals do not all appear at a level (or have mutex relations) then we know we need another level
 - Converse does not hold
- Useful **heuristic**: first time that all goal literals appear at a level, with no mutex relations
- **Graphplan** algorithm: once all goal literals appear, try to extract solution from graph
 - Can use CSP techniques by labeling each action as “in the plan” or “out of the plan”
 - In case of failure, generate another level

Example

- Fast-Forward planner...
 - <http://members.deri.at/~joergh/ff.html>
- ... with towers of Hanoi example...
 - <http://www.tempastic.org/vhpop/>
- ... in course directory:
- `./ff -o hanoi-domain.pddl -f hanoi-3.pddl`
- Btw., why is towers of Hanoi solvable with **any** number of disks?