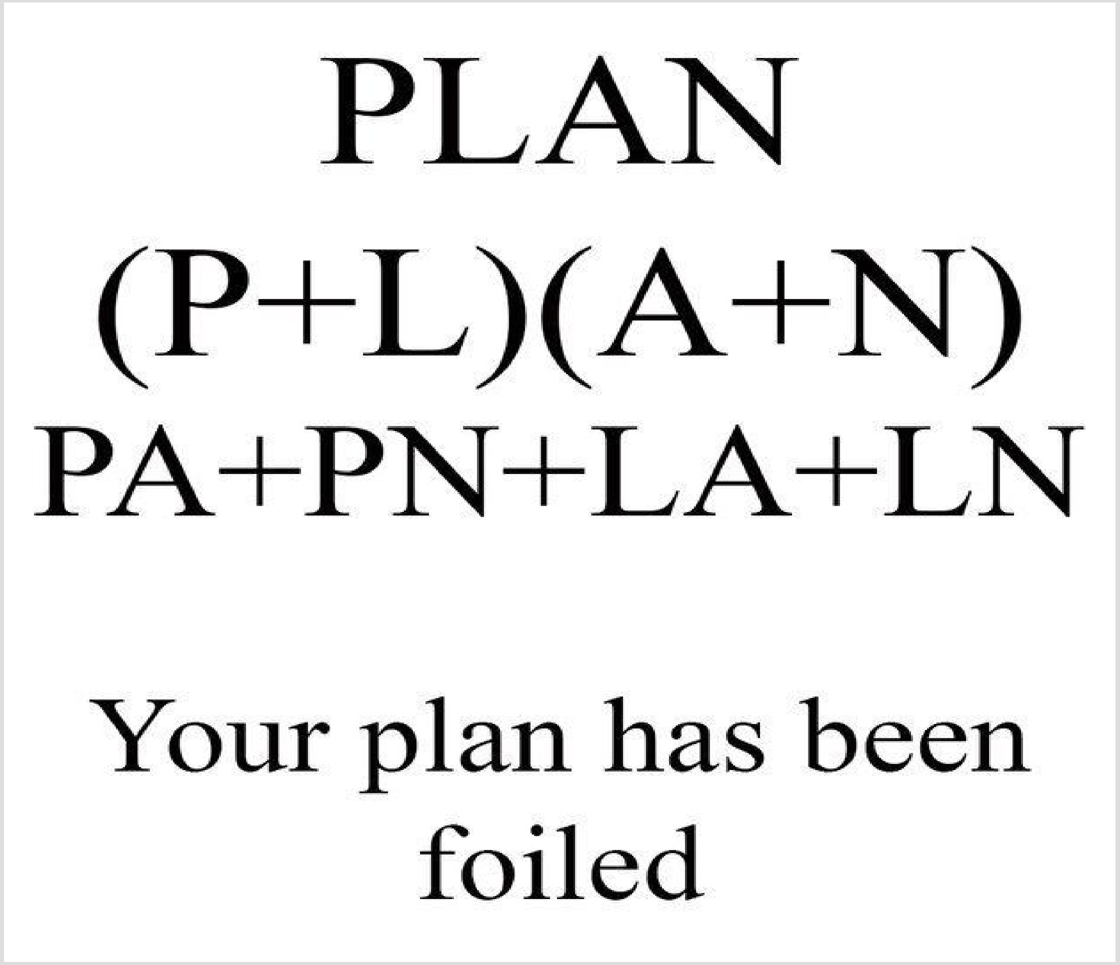
Planning (Ch. 10)



PLAN

(P+L)(A+N)

PA+PN+LA+LN

Your plan has been foiled

Announcements

Writing assignment 5 posted
-Similar to WA 3 if on project

Grade catch-up this week

Graphplan review

Graphplan relaxes planning problems by:

1. Taking multiple actions at a time
2. Not removing any relationships

To make the problem more realistic,
we do keep track of which pairs of relations are impossible together

(However, we do not look at triplets or any higher order subsets)

Mutexes: actions

Mutex Action rules:

1. $x \in Effect(A1) \wedge \neg x \in Effect(A2)$
2. $x \in Pre(A1) \wedge \neg x \in Effect(A2)$
3. $x \in Pre(A1) \wedge \neg x \in Pre(A2)$

Mutex State rules (between pairs):

1. If opposite relations
2. If all actions that lead to this pair are in mutex (above). (For 2 “no change” actions if previously had mutexes)

Mutexes: actions

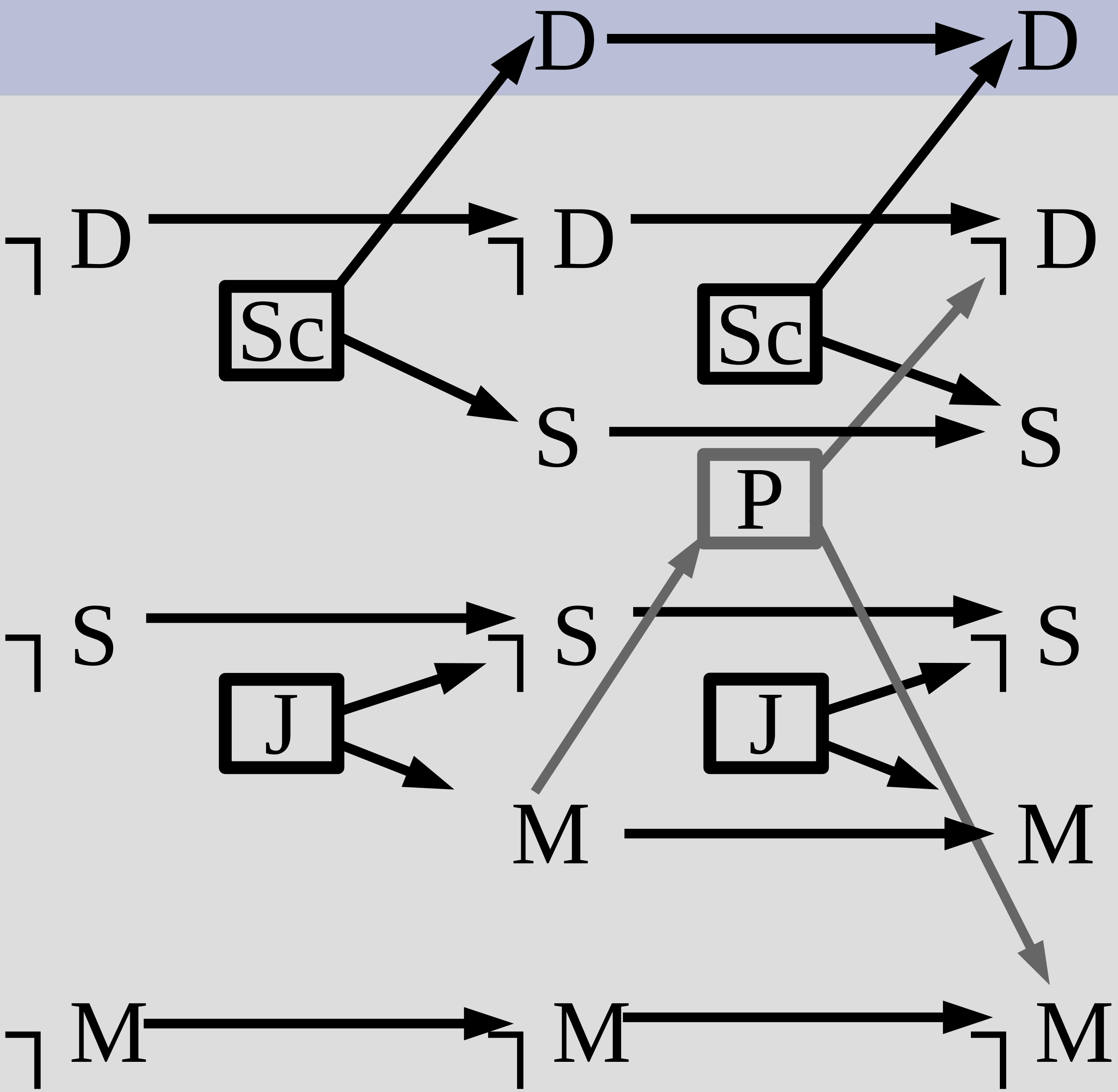
Initial: $\neg Money(me) \wedge \neg Smart(me) \wedge \neg Debt(me)$
Goal 1: $Money(me) \wedge Smart(me) \wedge \neg Debt(me)$
Goal 2: $\neg Money(me) \wedge Smart(me) \wedge \neg Debt(me)$

Action($School(x)$,
Precondition: ,
Effect: $Debt(x) \wedge Smart(x)$)

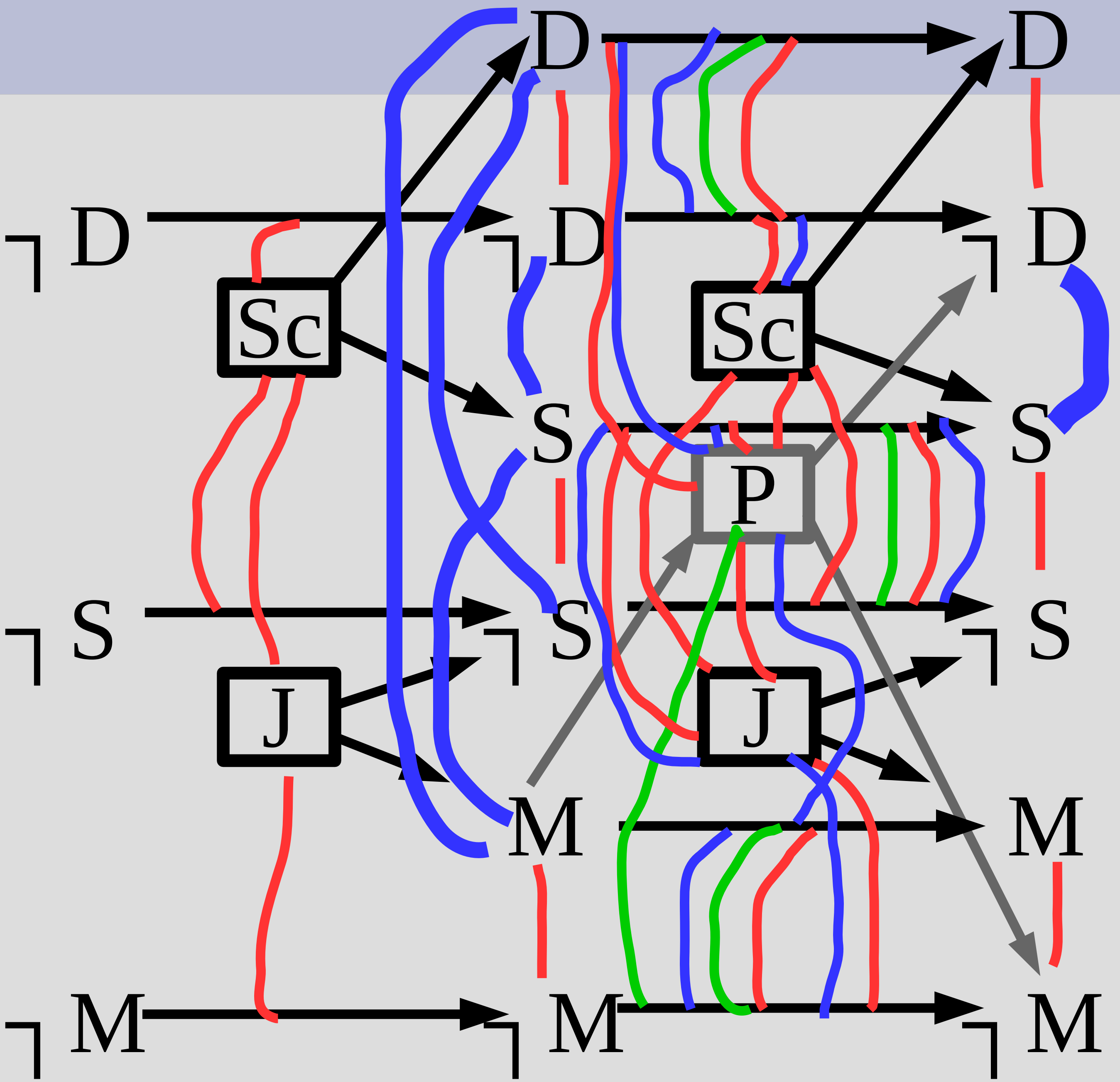
Action($Job(x)$,
Precondition: ,
Effect: $Money(x) \wedge \neg Smart(x)$)

Action($Pay(x)$,
Precondition: $Money(x)$,
Effect: $\neg Money(x) \wedge \neg Debt(x)$)

Mutexes: actions



Mutexes: actions



Non-trivial action mutexes:
(SC, P),
(J, P),
(SC, J),
(P,D&M),
(SC,┐ D&┐ S),
(J,┐ M&S)

GraphPlan: states

Let's consider this problem:

Initial: $Clean \wedge Garbage \wedge Quiet$

Goal: $Food \wedge \neg Garbage \wedge Present$

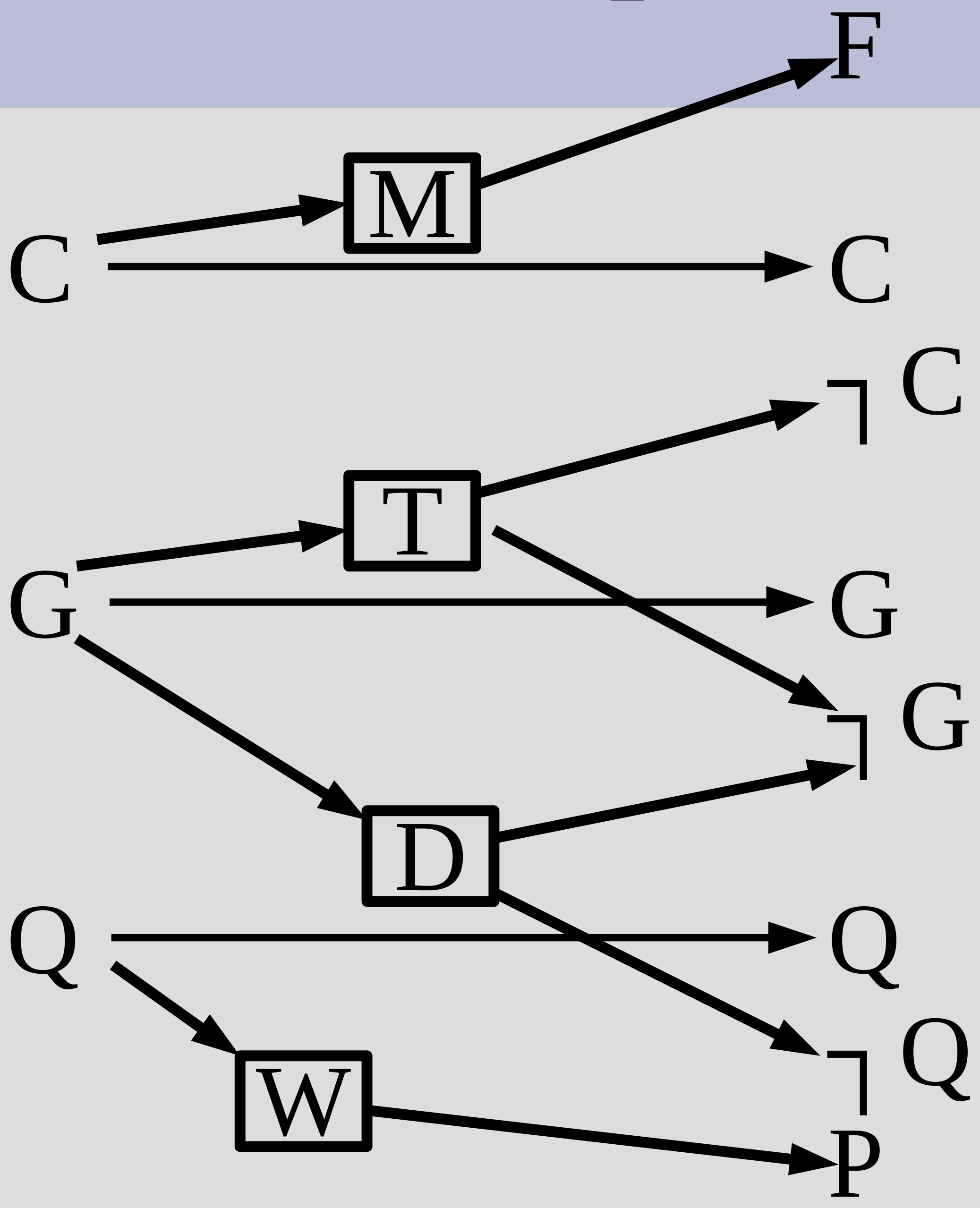
Action: ($MakeFood$,
Precondition: $Clean$,
Effects: $Food$)

Action: ($Takeout$,
Precondition: $Garbage$,
Effects: $\neg Garbage \wedge \neg Clean$)

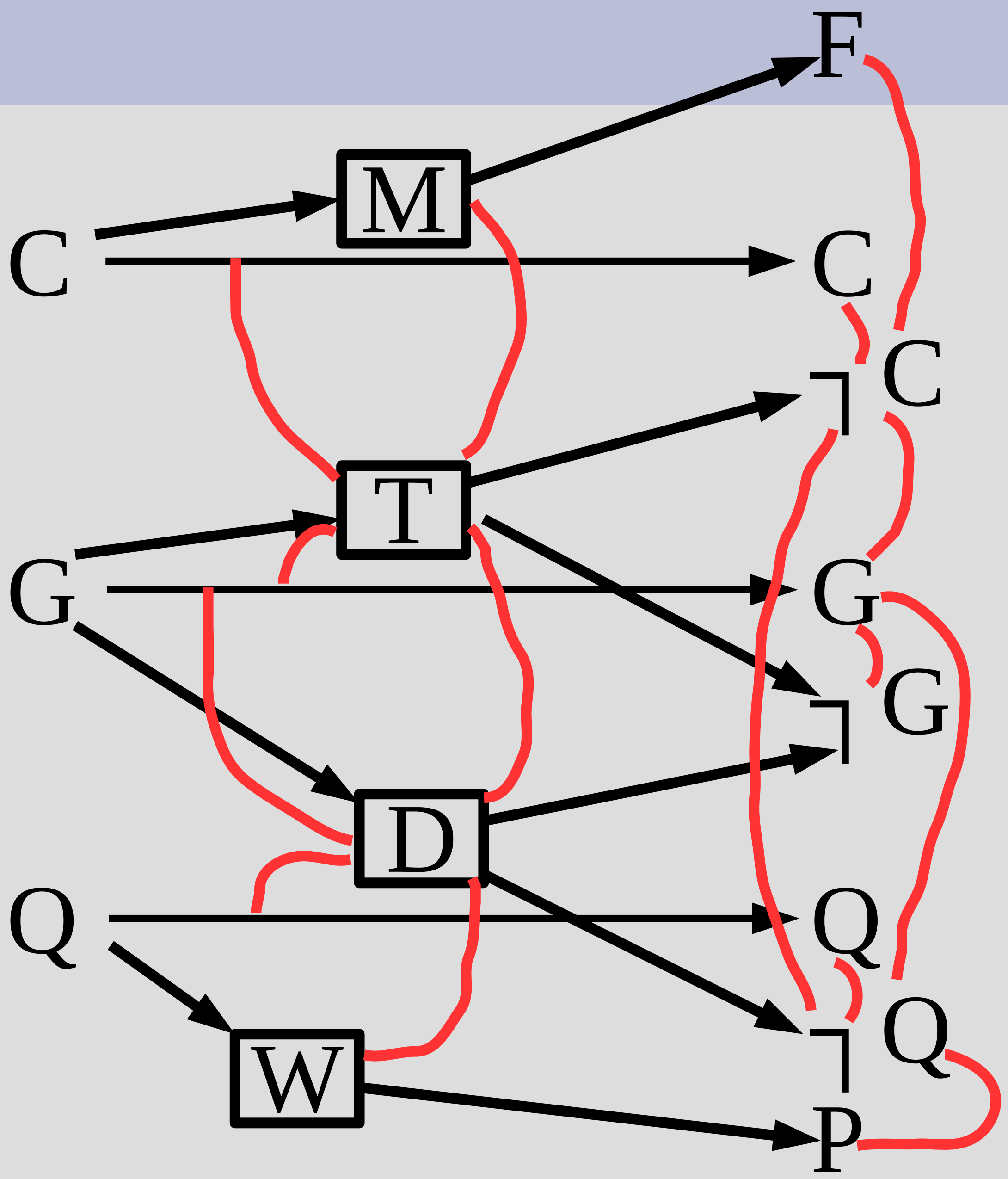
Action: ($Wrap$,
Precondition: $Quiet$,
Effects: $Present$)

Action: ($Dolly$,
Precondition: $Garbage$,
Effects: $\neg Garbage \wedge \neg Quiet$)

GraphPlan: states



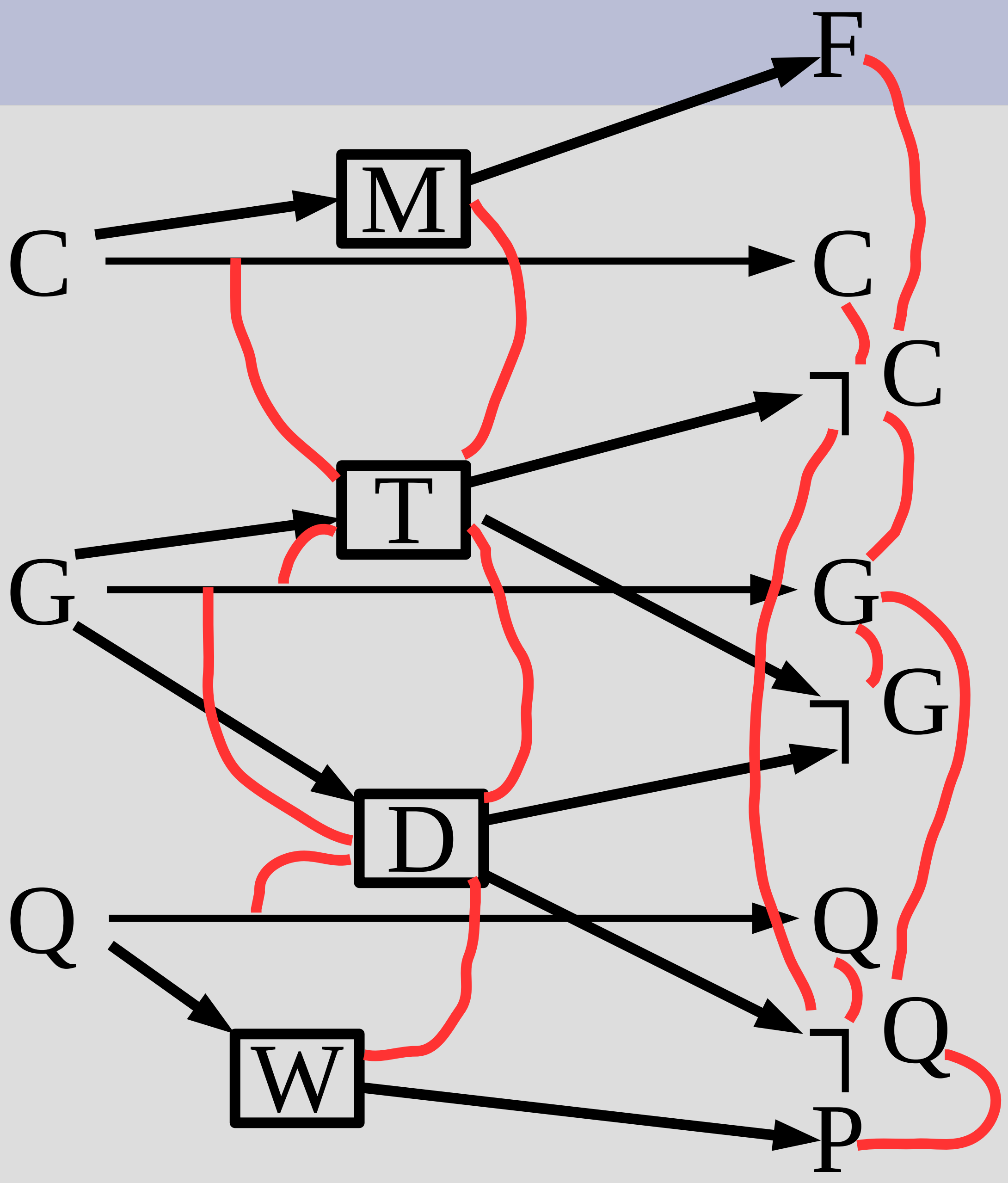
Mutexes



Possible state pairs:

F, C	C, Q
~~F, ┐C~~	C, ┐Q
F, G	C, P
F, ┐G	~~┐C, G~~
F, Q	┐C, ┐G
F, ┐Q	┐C, Q
F, P	~~┐C, ┐Q~~
~~C, ┐C~~	┐C, P
C, G	... (more)
C, ┐G	

Mutexes

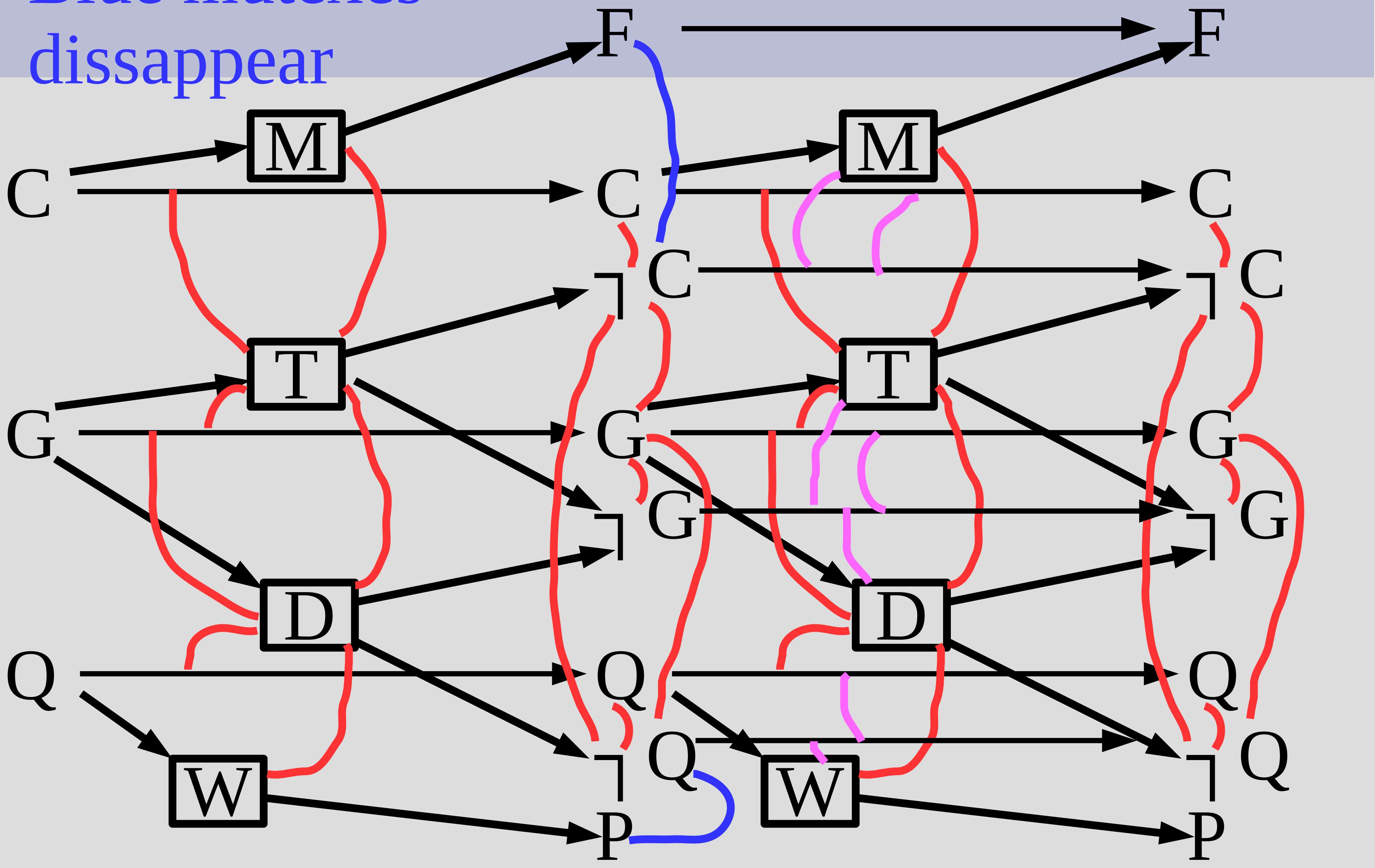


Make
one
more
level
here!

Mutexes

Blue mutexes dissappear

Pink = new mutex



GraphPlan as heuristic

GraphPlan is optimistic, so if any pair of goal states are in mutex, the goal is impossible

3 basic ways to use GraphPlan as heuristic:
(1) Maximum level of all goals
(2) Sum of level of all goals (not admissible)
(3) Level where no pair of goals is in mutex

(1) and (2) do not require any mutexes, but are less accurate (quick 'n' dirty)

GraphPlan as heuristic

For heuristics (1) and (2), we relax as such:

1. Multiple actions per step, so can only take fewer steps to reach same result
2. Never remove any states, so the number of possible states only increases

This is a valid simplification of the problem, but it is often too simplistic directly

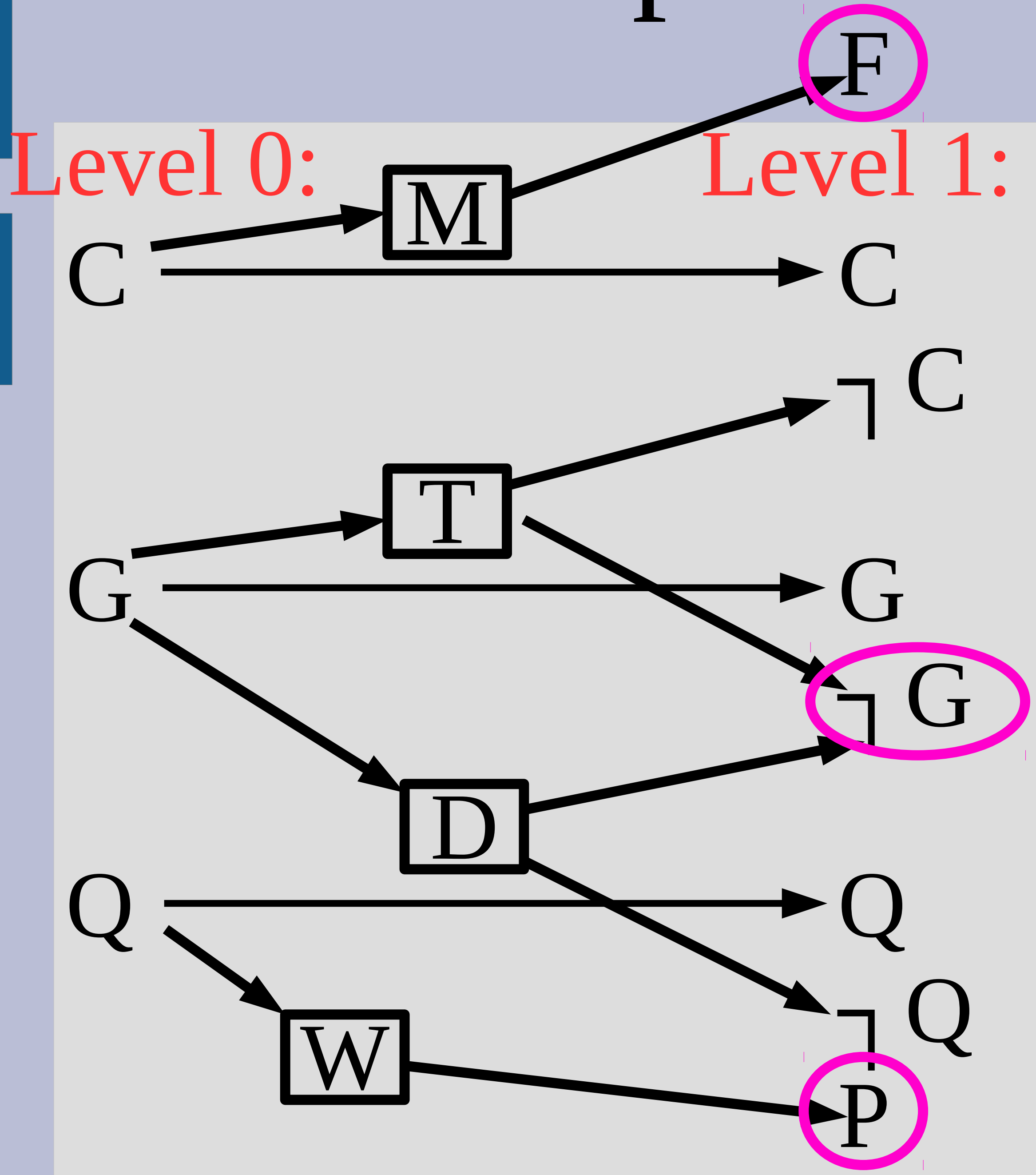
GraphPlan as heuristic

Heuristic (1) directly uses this relaxation and finds the first time when all 3 goals appear at a state level

(2) tries to sum the levels of each individual first appearance, which is not admissible (but works well if they are independent parts)

Our problem: goal={Food, ¬ Garbage, Present}
First appearance: F=1, ¬ G=1, P=1

GraphPlan: states



Heuristic (1):
Max(1,1,1) = 1

Heuristic (2):
1+1+1=3

GraphPlan as heuristic

Often the problem is too trivial with just those two simplifications

So we add in mutexes to keep track of invalid pairs of states/actions

This is still a simplification, as only impossible state/action pairs in the original problem are in mutex in the relaxation

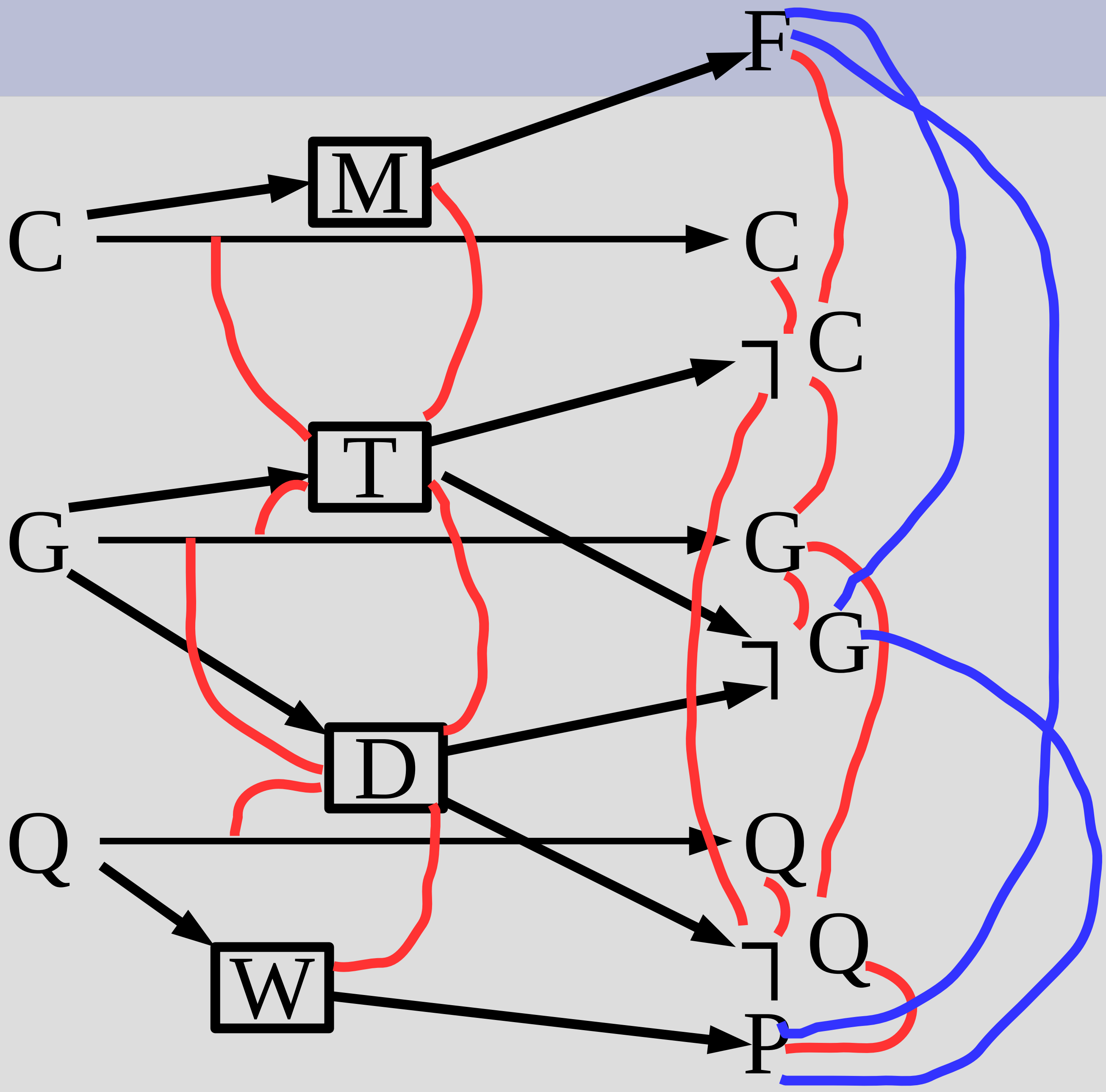
GraphPlan as heuristic

Heuristic (3) looks to find the first time none of the goal pairs are in mutex

For our problem, the goal states are:
(Food, ¬ Garbage, Present)

So all pairs that need to have no mutex:
(F, ¬ G), (F, P), (¬ G, P)

Mutexes



None of the pairs are in mutex at level 1

This is our heuristic estimate

Finding a solution

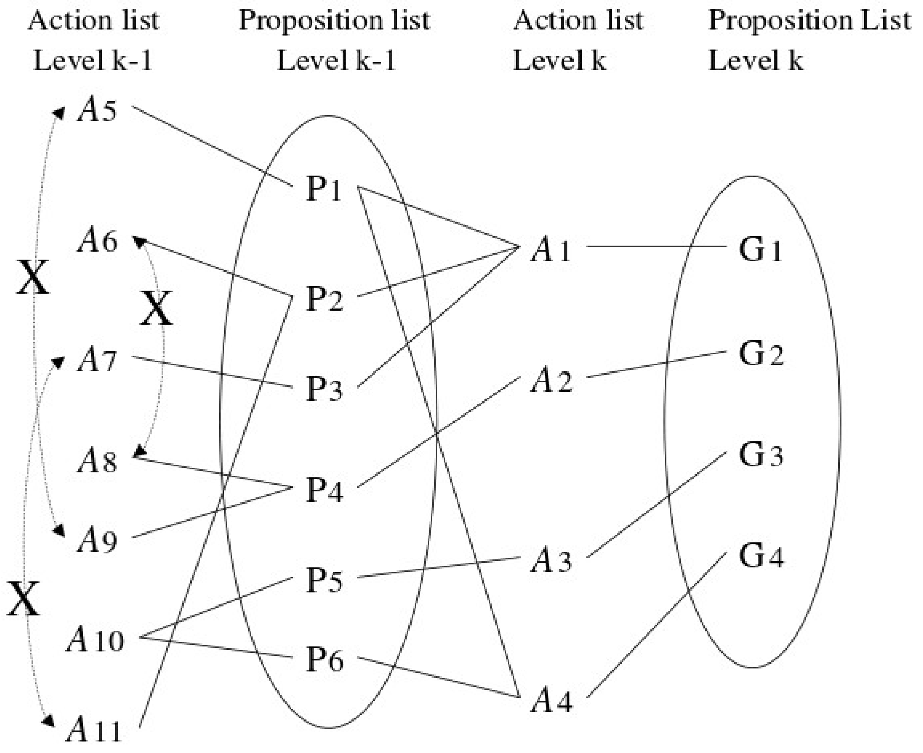
GraphPlan can also be used to find a solution:
(1) Converting to a CSP
(2) Backwards search

Both of these ways can be run once GraphPlan has all goal pairs not in mutex (or converges)

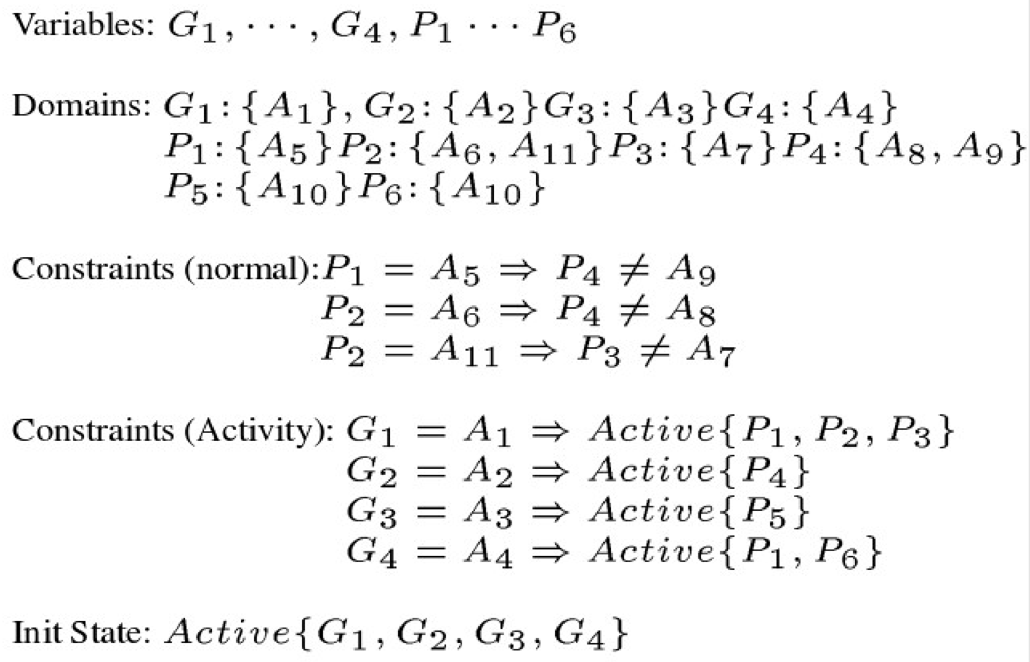
Additionally, you might need to extend it out a few more levels further to find a solution (as GraphPlan underestimates)

GraphPlan as CSP

Variables = states, Domains = actions out of
Constraints = mutexes & preconditions



(a) Planning Graph



Variables: $G_1, \cdots, G_4, P_1 \cdots P_6$

Domains: $G_1: \{A_1\}, G_2: \{A_2\} G_3: \{A_3\} G_4: \{A_4\}$
$P_1: \{A_5\} P_2: \{A_6, A_{11}\} P_3: \{A_7\} P_4: \{A_8, A_9\}$
$P_5: \{A_{10}\} P_6: \{A_{10}\}$

Constraints (normal): $P_1 = A_5 \Rightarrow P_4 \neq A_9$
$P_2 = A_6 \Rightarrow P_4 \neq A_8$
$P_2 = A_{11} \Rightarrow P_3 \neq A_7$

Constraints (Activity): $G_1 = A_1 \Rightarrow Active\{P_1, P_2, P_3\}$
$G_2 = A_2 \Rightarrow Active\{P_4\}$
$G_3 = A_3 \Rightarrow Active\{P_5\}$
$G_4 = A_4 \Rightarrow Active\{P_1, P_6\}$

Init State: $Active\{G_1, G_2, G_3, G_4\}$

(b) DCSP

from Do & Kambhampati

Finding a solution

For backward search, attempt to find arrows back to the initial state(without conflict/mutex)

This backwards search is similar to backward chaining in first-order logic (depth first search)

If this fails to find a solution, mark this level and all the goals not satisfied as: (level, goals)

(level, goals) stops changing, no solution

Graph Plan

Remember this...

Initial: $\neg Money(me) \wedge \neg Smart(me) \wedge \neg Debt(me)$

Goal: $\neg Money(me) \wedge Smart(me) \wedge \neg Debt(me)$

Action($School(x)$,
Precondition: ,
Effect: $Debt(x) \wedge Smart(x)$)

Action($Job(x)$,
Precondition: ,
Effect: $Money(x) \wedge \neg Smart(x)$)

Action($Pay(x)$,
Precondition: $Money(x)$,
Effect: $\neg Money(x) \wedge \neg Debt(x)$)

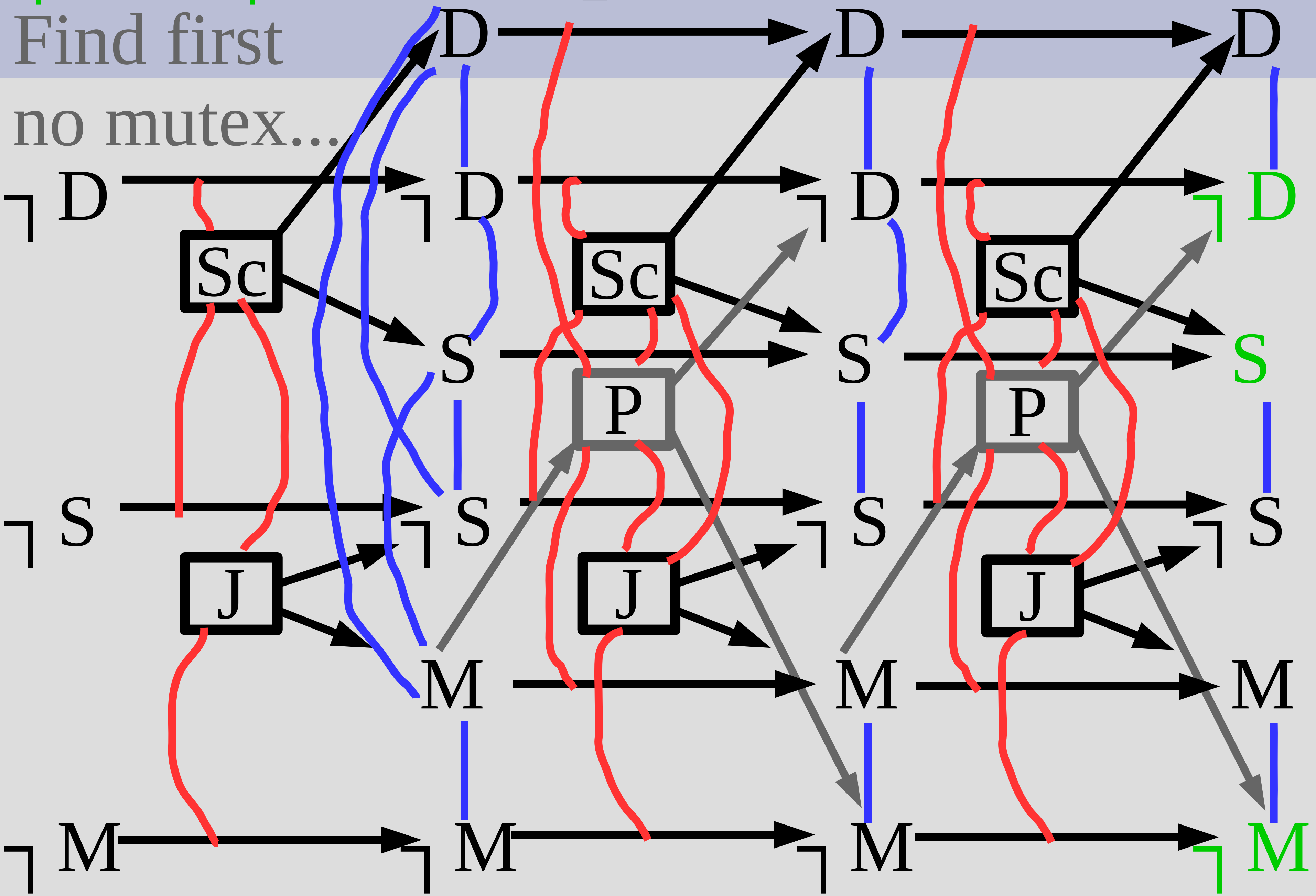
Graph Plan

Ask:

¬D^S^¬M

Find first

no mutex...



Ask:

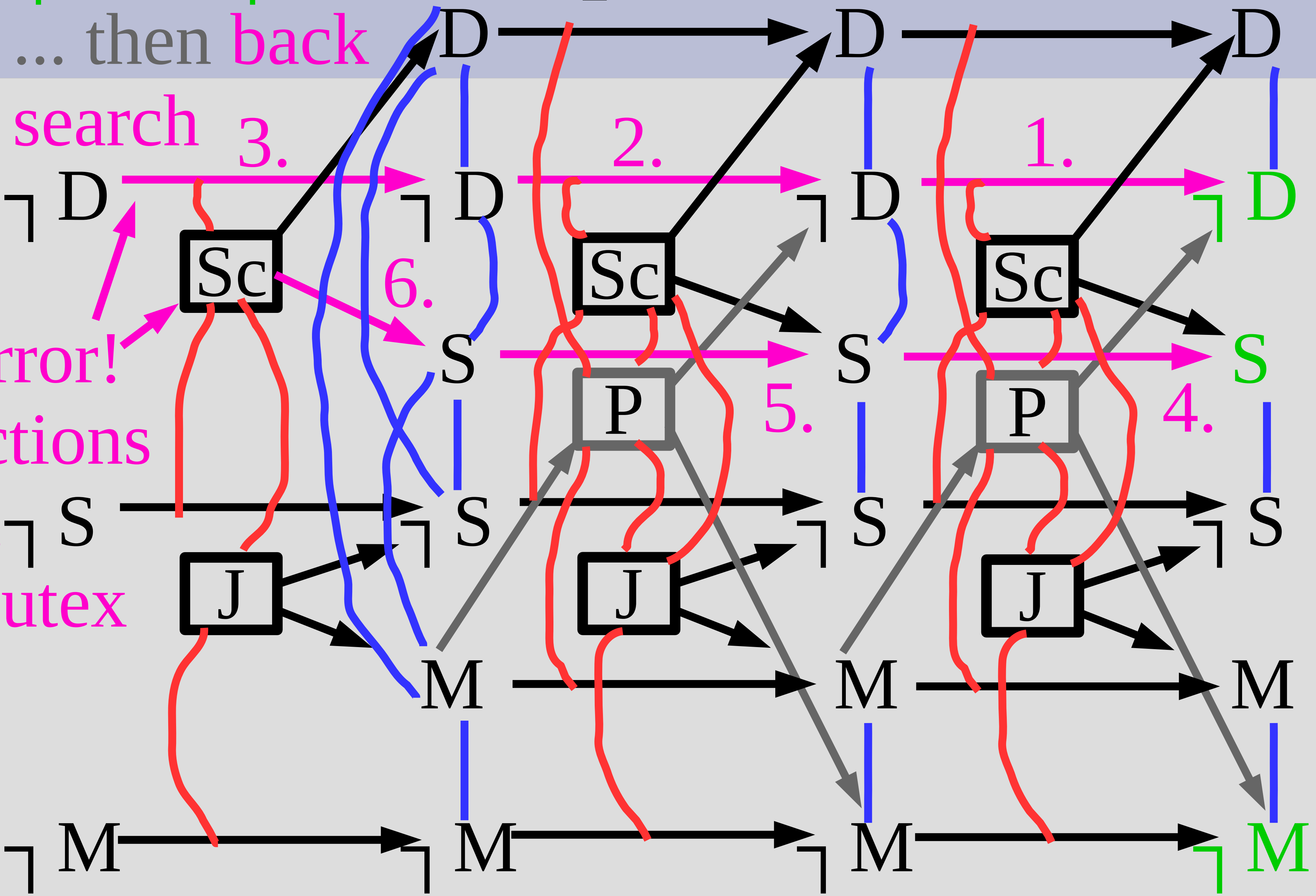
¬ D^S^¬ M

... then back

Graph Plan

search

Error! actions in mutex



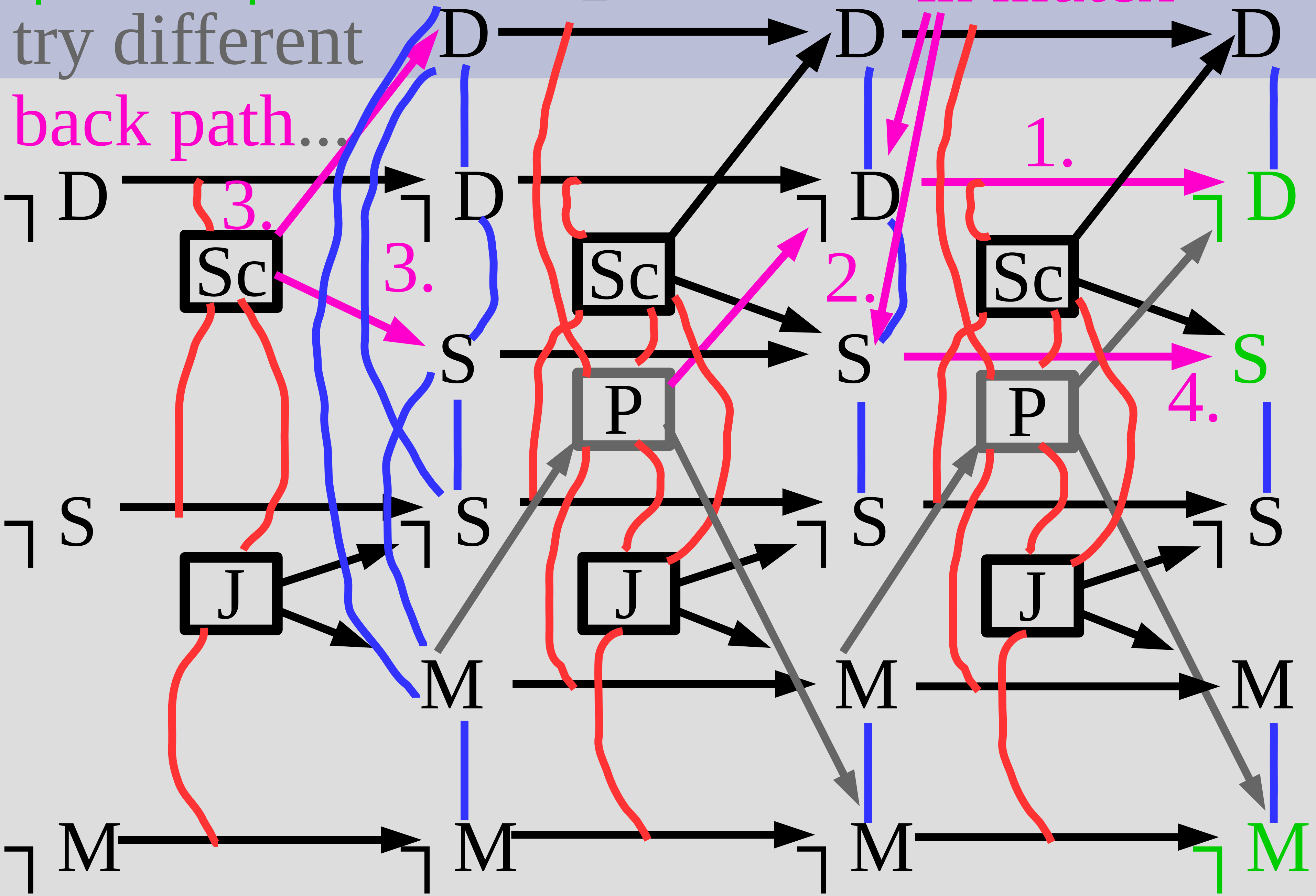
Graph Plan

Ask:

┐D^S^┐M

try different back path...

Error states in mutex



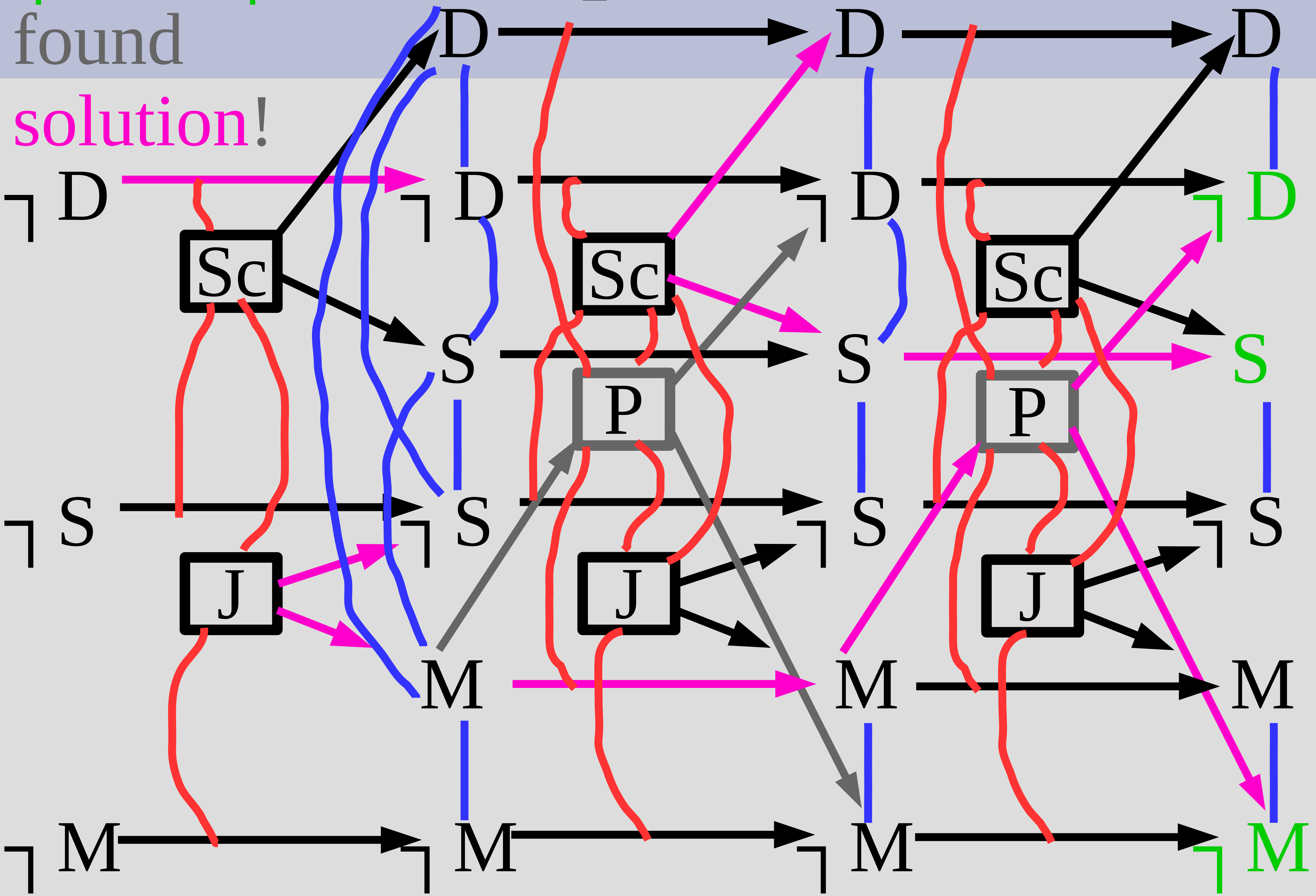
Graph Plan

Ask:

¬D^S^¬M

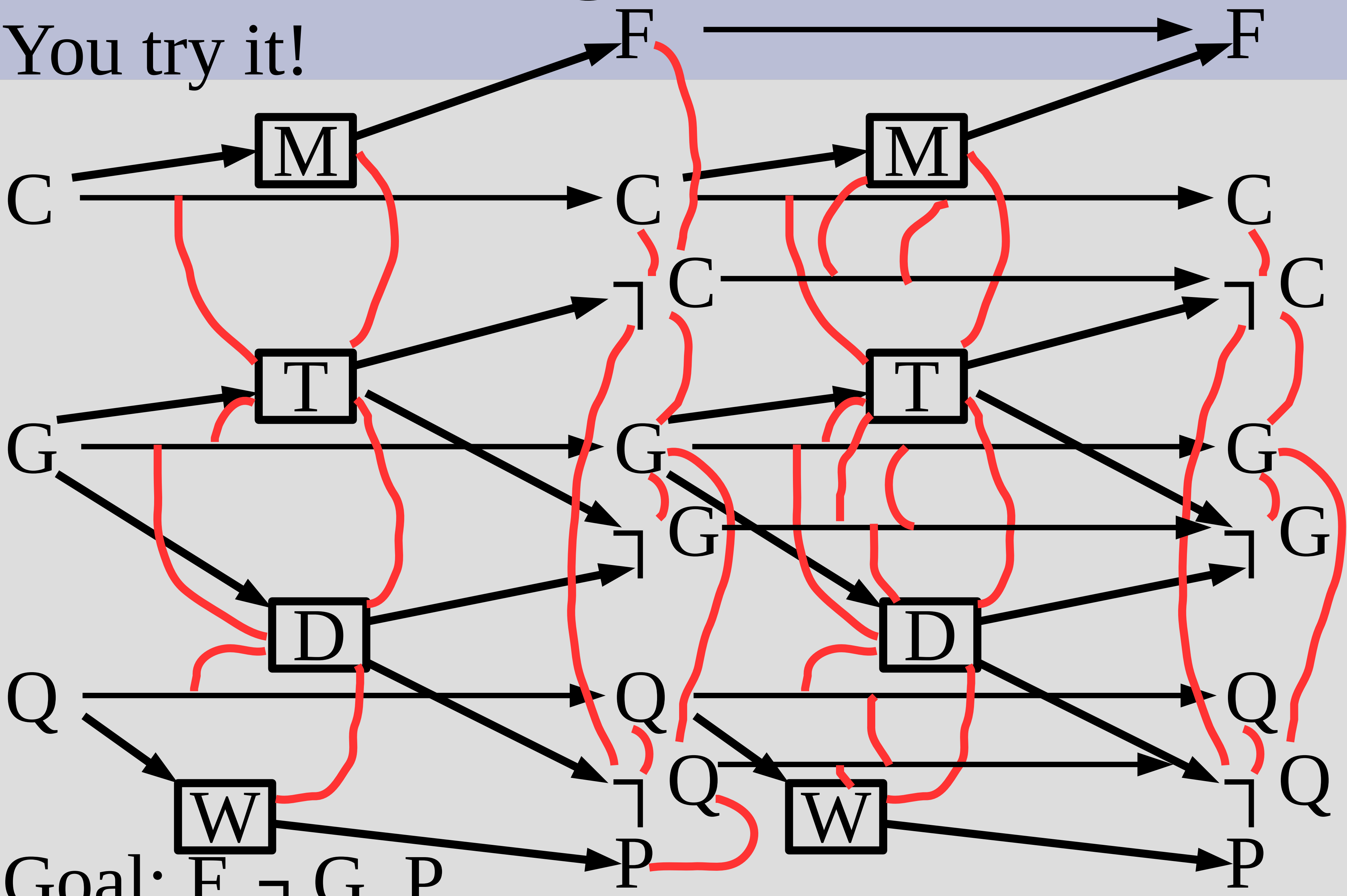
found

solution!



Finding a solution

You try it!



Goal: F, ¬ G, P

Finding a solution

Formally, the algorithm is:

```
graph = initial
noGoods = empty table (hash)
for level = 0 to infinity
    if all goal pairs not in mutex
        solution = DFS with noGoods
        if success, return paths
    if graph & noGoods converged, return fail
    graaph = expand graph
```

GraphPlan

GraphPlan can be computed in $O(n(a+l)^2)$,
where n = levels before convergence
a = number of actions
l = number of relations/literals/states
(square is due to needing to check all pairs)

The original planning problem is PSPACE, which is known to be harder than NP