Universität des Saarlandes
Fachrichtung 6.2 – Informatik
J. Siekmann, S. Autexier, K. Fischer, E. Melis, C. Zinn
M. Schiller, E. Schulz, R.Vollmann

10th Theoretical Assignment in Artificial Intelligence (WS 2006/2007)
Solutions

Exercise 10.1
Consider the following planning problem, where A, B, C and D are constants supposed to represent persons, and x, y and z are variables:

$Op($Action: $Start$,
Effect: $ambitious(A) \wedge lazy(C) \wedge lazy(D) \wedge collaborate(A, B)$
$\wedge collaborate(B, C) \wedge collaborate(C, D))$

$Op($Action: $Finish$,
Precond: $collaborate(A, D) \wedge leader(A))$

$Op($Action: $Meeting(x, y, z)$,
Precond: $collaborate(x, y) \wedge collaborate(y, z)$
Effect: $collaborate(x, z))$

$Op($Action: $FindGroupLeader(x, y, z)$,
Precond: $ambitious(x) \wedge lazy(y) \wedge lazy(z)$,
Effect: $leader(x) \wedge \neg collaborate(x, y) \wedge \neg collaborate(x, z))$

Generate a partial order plan with the POP algorithm! For this exercise, assume that the POP algorithm first tries to satisfy $collaborate(A, D)$! Give a description for every step of the planning process with new *casual links* (including the preconditions for which the causal link was added). Moreover, describe any new operator you want to add as well as the resulting *threats* (that is, steps that might delete a precondition protected by a causal link) and the ordering constraints included to resolve the threats (by ordering the threats to come before or after the causal link). You need not give a complete diagramm of the plan for every step. Give a diagram of the final plan, inclusive causal link, as well as their preconditions and ordering constraints.

Solution:

Credits go to Shady Elbassuoni for the solution.

We represent iterations of the POP algorithm (slide 6 in the lecture about POP planning) as a table. We start with the initial plan containing the actions $A := \{START, FINISH\}$ *and the agenda* $\{(collaborate(A, D), FINISH), (leader(A, D), FINISH)\}$.

Selected pair (Q, A_{need}) *from agenda*	*Selected action* A_{add}	*Variable bindings*	*New goal set*
(collaborate(A,D), FINISH)	*MEETING*(x_1, y_1, z_1)	$\{x_1 \rightarrow A, y_1 \rightarrow D\}$	agenda' := $\{(collaborate(A, y_1), MEETING_1), (collaborate(y_1, D), MEETING_1), (leader(A, D), FINISH)\}$
(collaborate(A,y_1*), MEETING*$_1$*)*	*START*	$\{y \rightarrow B\}$	agenda' := $\{(B, D), MEETING_1), (leader(A, D), FINISH)\}$
(collaborate(B,D), MEETING$_1$*)*	*MEETING*(x_2, y_2, z_2)	$\{x_2 \rightarrow B, z_2 \rightarrow D\}$	agenda' := $\{(collaborate(B, y_2, MEETING_2), (collaborate(y_2, D), MEETING_2), (leader(A, D), FINISH)\}$
(collaborate(B,y_2*), MEETING*$_2$*)*	*START*	$\{y \rightarrow C\}$	agenda' := $\{(collaborate(C, D), MEETING_2), (leader(A, D), FINISH)\}$
(collaborate(C,D), MEETING$_2$*)*	*START*		agenda' := $\{(leader(A, D), FINISH)\}$
(leader(A), FINISH)	*FINDGROUPLEADER*(x_3, y_3, z_3)	$\{x_3 \rightarrow A\}$	agenda' := {}

Figure 1: The intial plan

The intial plan shown in figure 1 contains only the two actions START and FINISH. The set of open preconditions contains the preconditions of FINISH, Collaborate(A,D) and Leader(A).

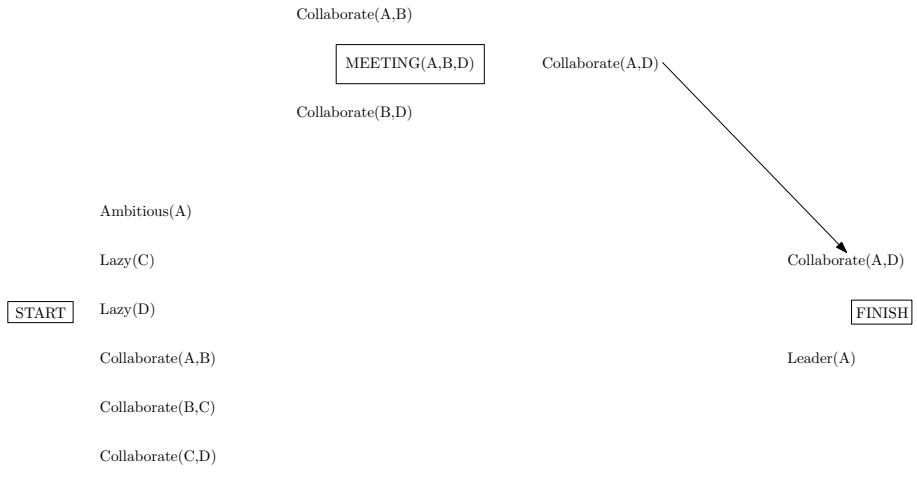


Figure 2: The second phase

The planner picks one of the open preconditions, namely the Collaborate(A,D) to fullfill. The action MEETING(A,B,D) is the only action with an effect of Collaborate(A,D) and so it is added to the

plan. Its preconditions, Collaborate(A,B) and Collaborate(B,D) are also added to the set of open preconditions. Collaborate(A,D) is then removed from the set of open preconditions. This is shown in figure 2.

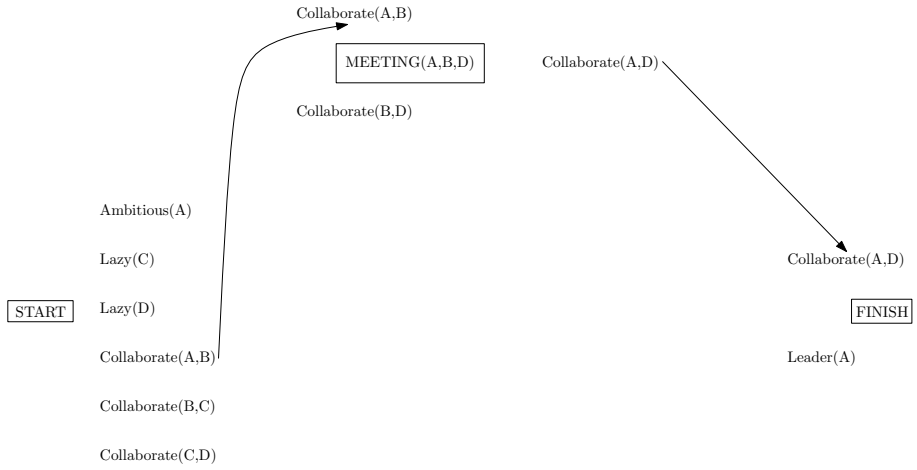


Figure 3: The third phase

The Collaborate(A,B) precondition is fullfilled by the START action and so a casual link from START to MEETING(A,B,D) is added as shown in figure 3. Collaborate is now removed from the set of open preconditions.

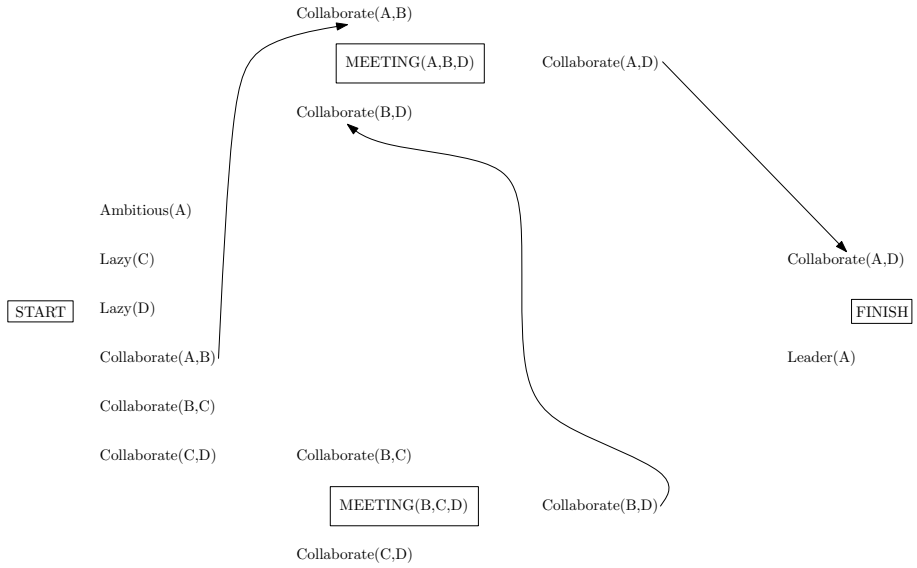


Figure 4: The fourth phase

In order to fullfill the Collaborate(B,D) precondition, the action MEETING(B,C,D) must be added to the plan. Its preconditions are added to the set of open preconditions as shown in figure 4.

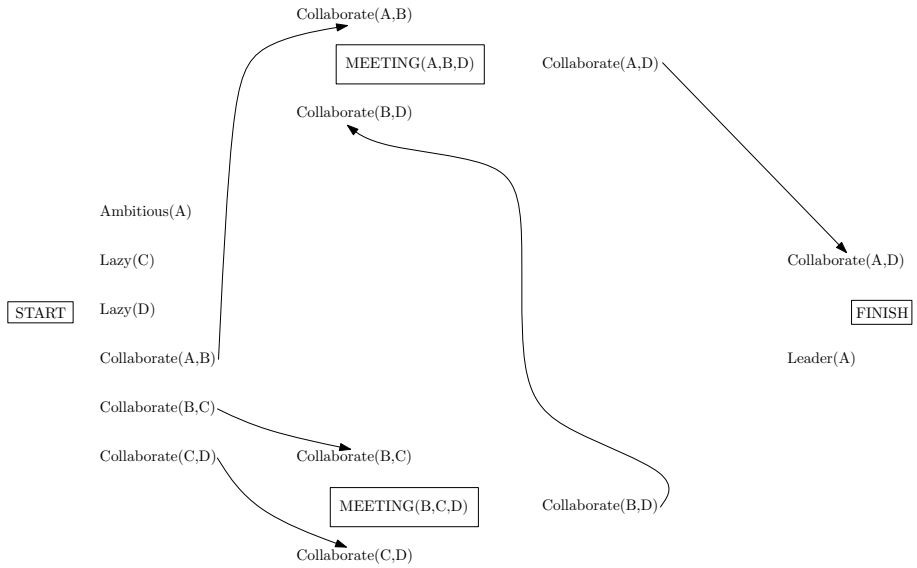


Figure 5: The fifth phase

Collaborate(B,C) and Collaborate(C,D) are both effects of the START action, and thus can be removed from the set of open preconditions after adding the necessary casual links as shown in figure 5.

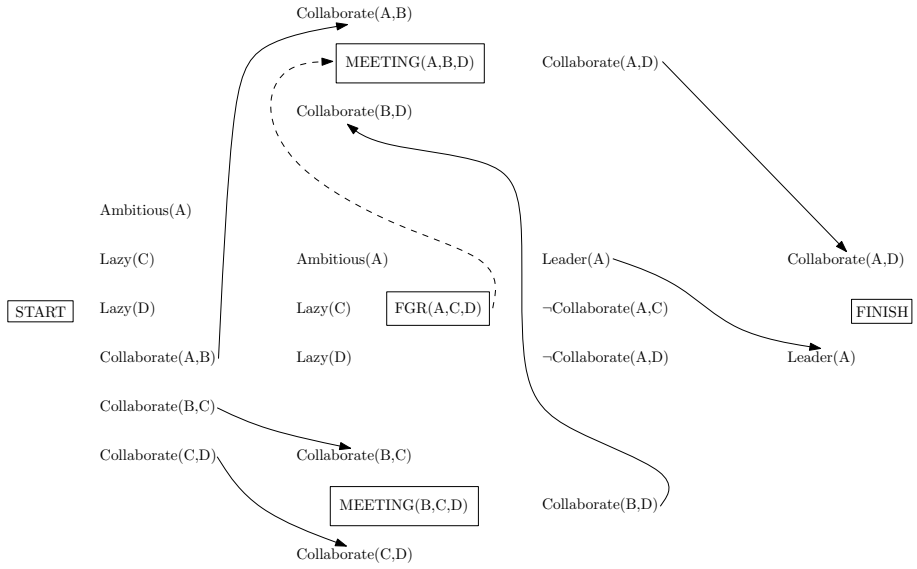


Figure 6: The sixth phase

The only remaining precondition in the set of open preconditions is Leader(A) which can only be fullfilled by adding the action FINDGROUPLEADER(A,C,D). The preconditions of this added action are then added to the set of open preconditions. Since FINDGROUPLEADER(A,C,D) has ¬ Collaborate(A,D) as an effect, it will threaten the casual link between MEETING(A,B,D) and FINISH (fullfilling the precondition Collaborate(A,D)) and thus FINDGROUPLEADER has to occur somewhere outside the protection area of such a link which would mean it has to occur either before MEETING(A,B,D) or after FINISH. Since nothing can occur after the FINISH action, it thus has to take place before MEETING(A,B,D) which is presented by the dashed link in figure 6.
The only 3 open preconditions left are achieved by the START action. Figure 7 gives the final plan.

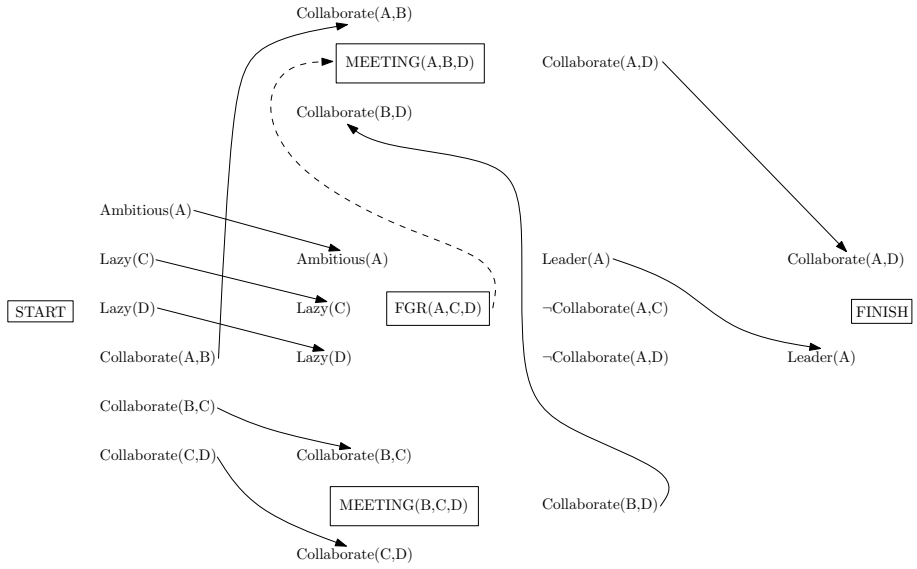


Figure 7: The final plan

Exercise 10.2
Consider a shopping problem, where the shopping agent can use the following operators:

$Op($Action:$Start$,
Effect:$At(Home) \wedge Sells(HWS, Drill) \wedge Sells(SM, Milk) \wedge Sells(SM, Banana))$

$Op($Action:$Finish$,
Precond:$Have(Drill) \wedge Have(Milk) \wedge Have(Banana))$

$Op($Action:$Go(there)$,
Precond:$At(here)$
Effect:$At(there) \wedge \neg At(here))$

$Op($Action:$Buy(x)$,
Precond:$At(store) \wedge Sells(store, x)$,
Effect:$Have(x))$

Here, SM stands for *Supermarkt* and HWS for *Hardware store*. We want to include money, at least in a simple way.

(a) Let CC denote a credit card that the agent can use to buy any object. Modify the description of Buy so that the agent has to have its credit card in order to buy anything.

(b) Write a *PickUp*-Operator that enables the agent to $Have$ an object if it is portable and at the same location as the agent and owned by the agent. (Hint: you need to change the vocabulary of the domain.)

(c) Whenever the agent changes his location the credit card has to change its location as well if the agent has it with him.

i. Give a plausible specification of Go, such that it has $Have$ as precondition and appropriate changes of At as effects.

ii. Why is that not a correct way to update the location of objects that the agent has with him? (5 P)

iii. How would you handle this problem in this context? Explain.

iv. What should be done to enable a complete correct specification of Go?

Solution:

Credits go to Shady Elbassuoni for the solution.

(a)
```
Op(Action: Buy(x)
    Precond: At(Store),Sells(Store,x),Have(CC)
    Effect: Have(x))
```

(b) *We introduce the following new predicates:*

- `Portable(x)` *is true, when* `x` *is portable.*
- `At(x,y)` *is true, when* `x` *is at location* `y`.
- `Owns(x,y)` *is true, when* `x` *owns* `y`.

```
Op(Action: PickUp(x)
   Precond: Portable(x), Owns(Agent,x), At(Agent,y)
   Effect: Have(x))
```

(c) i.

```
Op(Action: Go(x)
   Precond: At(Agent,y), Have(obj)
   Effect: At(Agent,x), not At(Agent,y), At(obj,x), not At(obj,y)
```

ii. *This method is not correct because the agent can only have one object with it. It also cannot go anywhere without an object.*

iii. *One should notice that the* `Go`*-Operator is managing the objects' locations incorrectly in cases where the agent possesses more than one object. In this case one has to decide which of the objects the agent possesses changes the location.*
In addition, the `Go`*-Operator is not applicable when one possesses nothing.*

iv. *When the agent moves, all the objects with him should move as well.We could make use of the universal quantification in the definition of* `Go` *as in the following example:*

```
forall(obj)(if Have(obj) then not At(obj,y)
                and At(obj,x))
```

Exercise 10.3

Consider Johnny, who is unemployed and not married. He does not suffer from depression. He can take the following actions:

- Johnny can go look for a job, if he is unemployed, free from depression, and not married. The result is that he will not be unemployed, but suffer from depression.
- Johnny can go to a therapist if he is depressed. Afterwards, he will not be depressed any more.
- Johnny can make a proposal to Mary if he is not unemployed. In this case he will be married.

Assume that Johnny's goal is to be married and not depressive.

1. Formalise these statements as a planning problem!
2. Construct a planning graph (until it *levels off*)! Indicate all mutex links (and what kind of mutex links these are – *inconsistent effects*, *interference*, *competing needs* or *inconsistent supports*)!
3. How can you use the planning graph to check if there is a solution to the planning problem?
4. Use the *GRAPHPLAN* algorithm to extract the solution (you can use the algorithm on p.298 ff. in Russell/Norvig, or the algorithm presented on the lecture slides)! Indicate each step!
5. Use partial order planning to generate a partial order plan. What differences to *GRAPHPLAN* do you notice?
6. Assume there is one more thing Johnny can do – if he is crazy, he can kidnap Mary and force her to marry him. This will result in marriage, but requires Johnny to be crazy (assume that initially he is not). Does this extra operator change the way graph planning looks for a solution? Can it make a difference for the way the POP algorithm searches for a solution?

Solution:

1. Credits go to Johannes Tran for the solution.

Initial conditions: (and (unemployed) (not (married)) (not (depressive)))

Goal: (and (married) (not (depressive)))

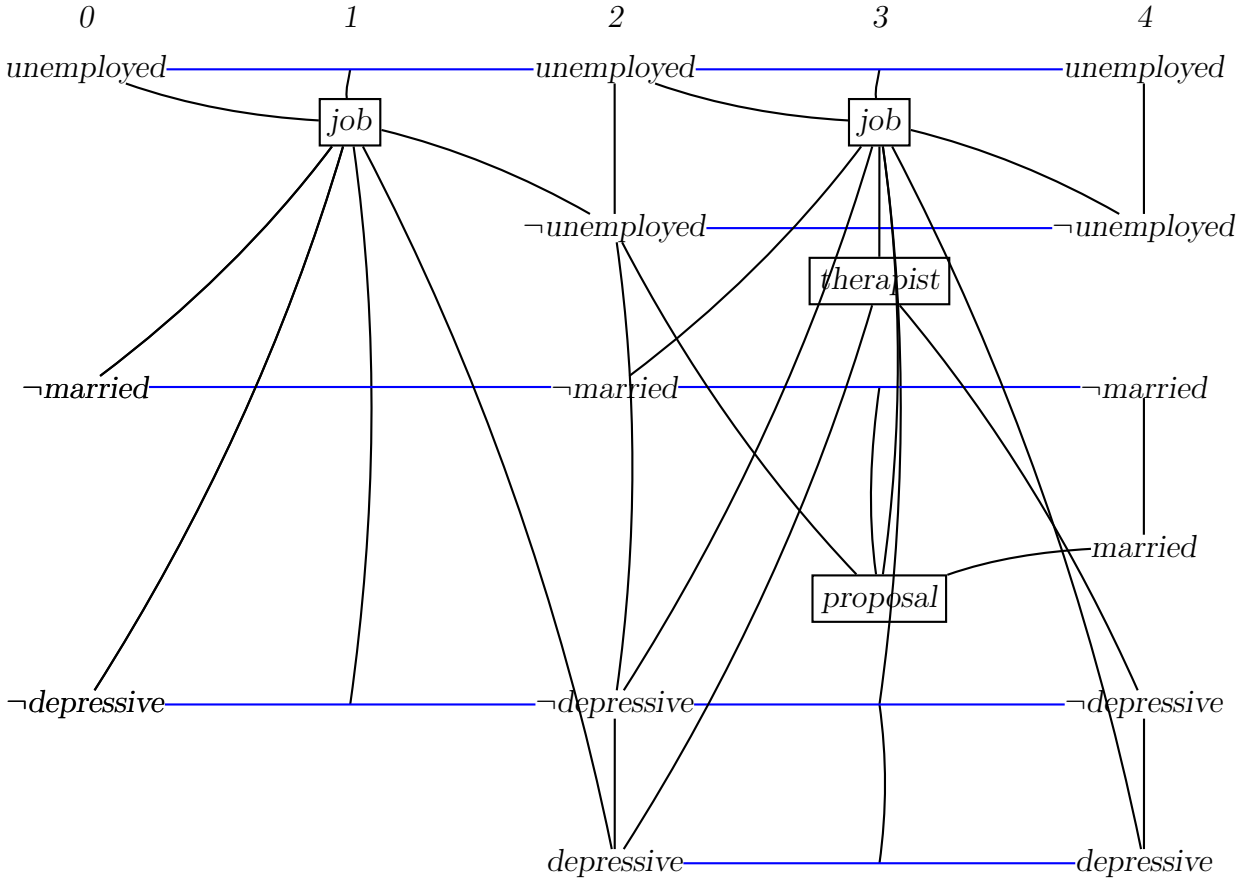
Actions:

job: precondition (and (unemployed) (not (depressive)) (not (married)))
result (and (not (unemployed)) (depressive))

therapist: precondition (depressive)
result (not (depressive))

proposal: precondition (not (unemployed))
result (married)

2. Credits go to Johannes Tran for the solution.



Mutex links:

Inconsistent effects: (job, No-op(unemployed)), (job, No-op(¬depressive)), (job,therapist), (therapist, No-op(depressive)), (proposal, No-op(¬married)), (job, proposal)

Interference: (job, No-op(unemployed)), (job, No-op(¬depressive)), (job, proposal), (therapist, No-op(depressive))

Competing needs: (therapist, job), (proposal, job), (No-op(depressive), No-op(¬depressive)), (No-op(unemployed), No-op(¬unemployed)), (job, No-op(unemployed)), (therapist, No-op(depressive))

Inconsistent supports: (married, ¬married), (depressive, ¬depressive), (unemployed,

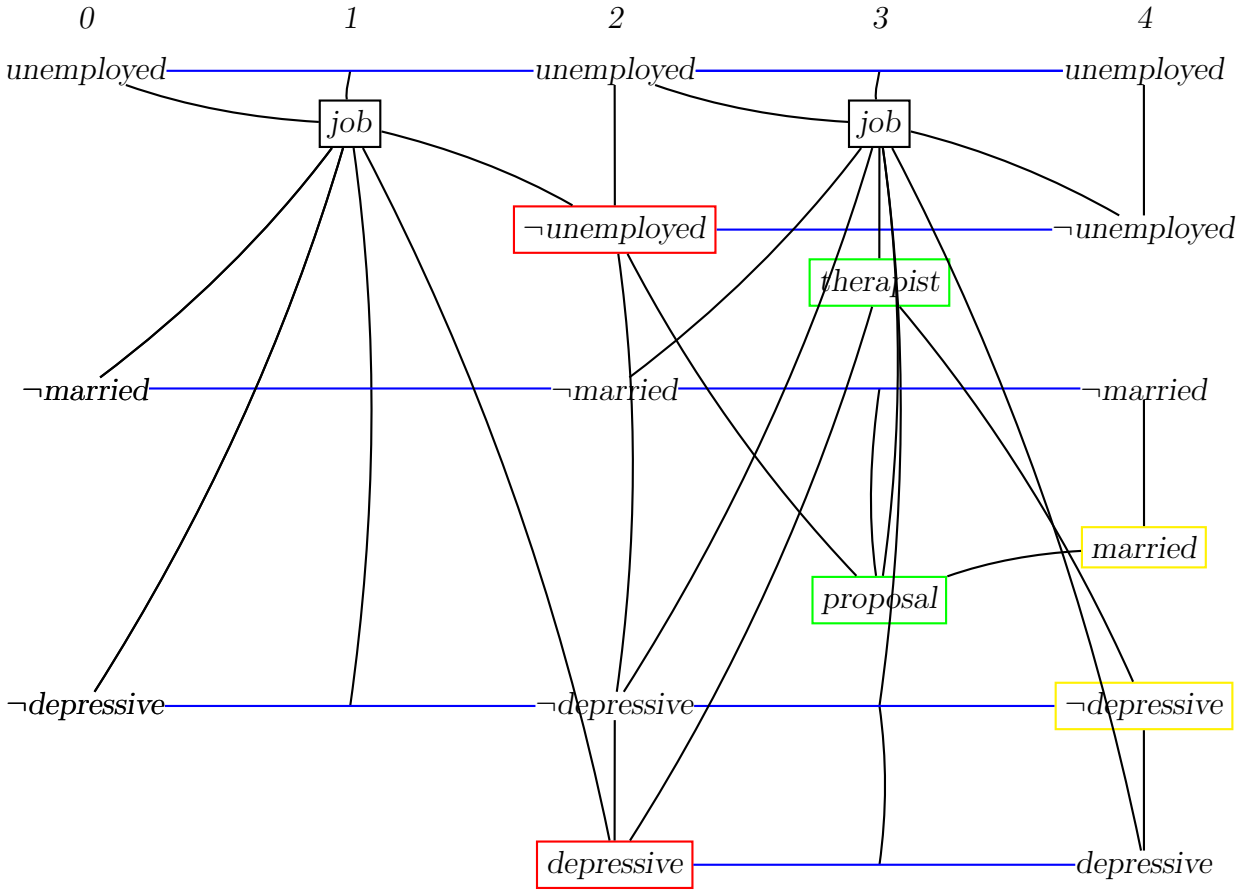
¬unemployed), (¬unemployed, ¬depressive).

3. Credits go to Johannes Tran for the solution.

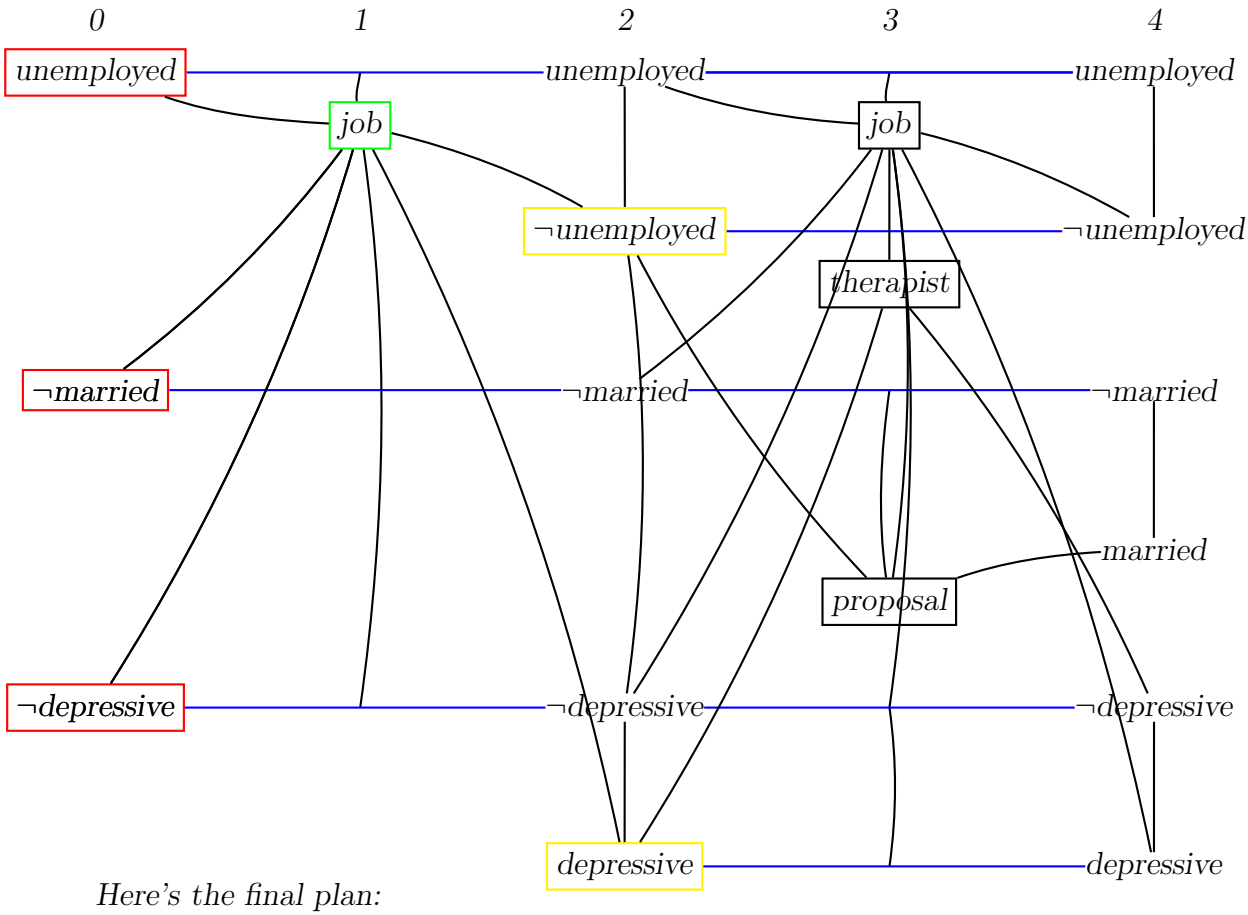
   Check, that all goal-literals occur at the last level and that they are not mutex with each other.

4. Credits go to Johannes Tran for the solution.

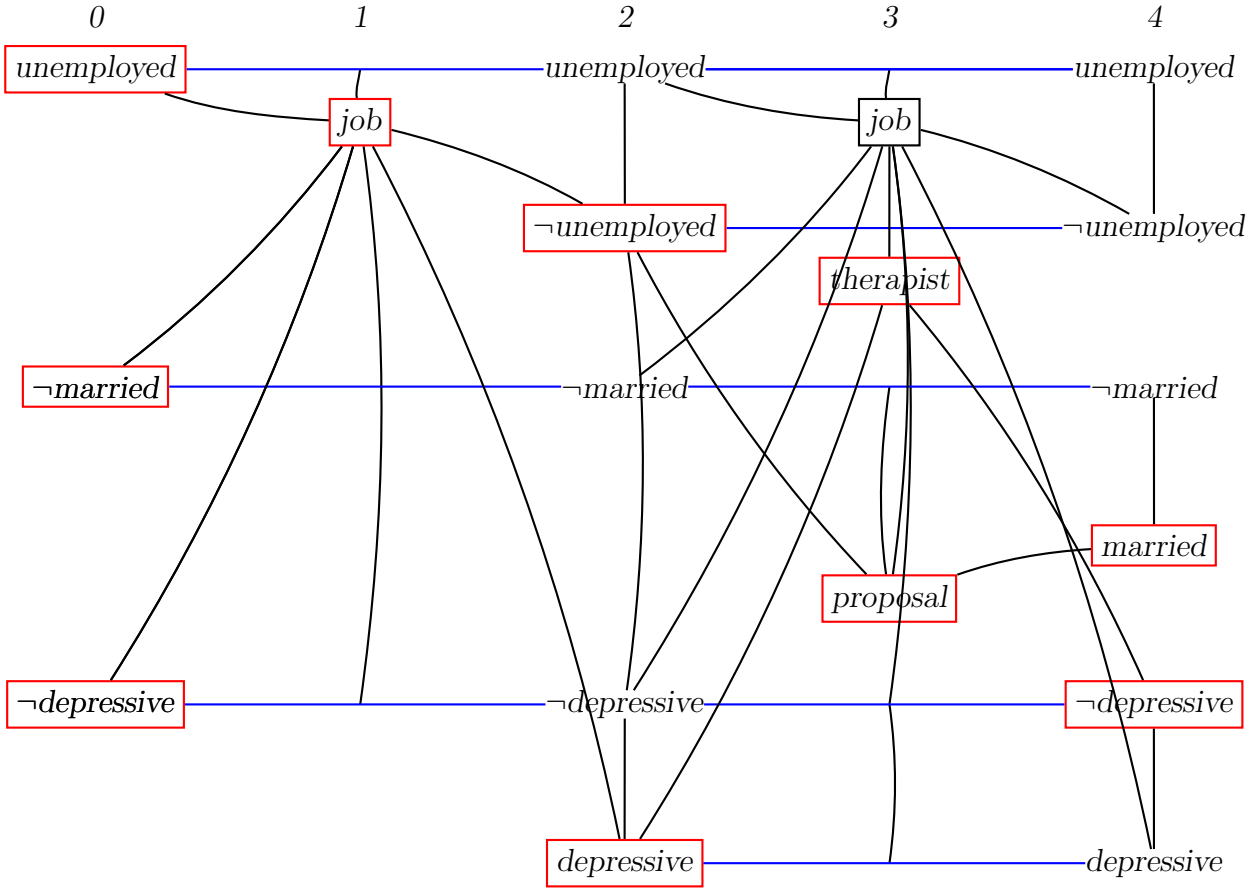
   Step 1: for each goal at level 4 (yellow), choose an action at level 3 (green) that achieves it. The preconditions of the chosen actions become the new set to be considered [red]



   Step 2: Same as Step 1 with the new set (yellow), the actions (green), that achieve it. The preconditions of that action is considered (red). We are at level 0 and we have got the plan.



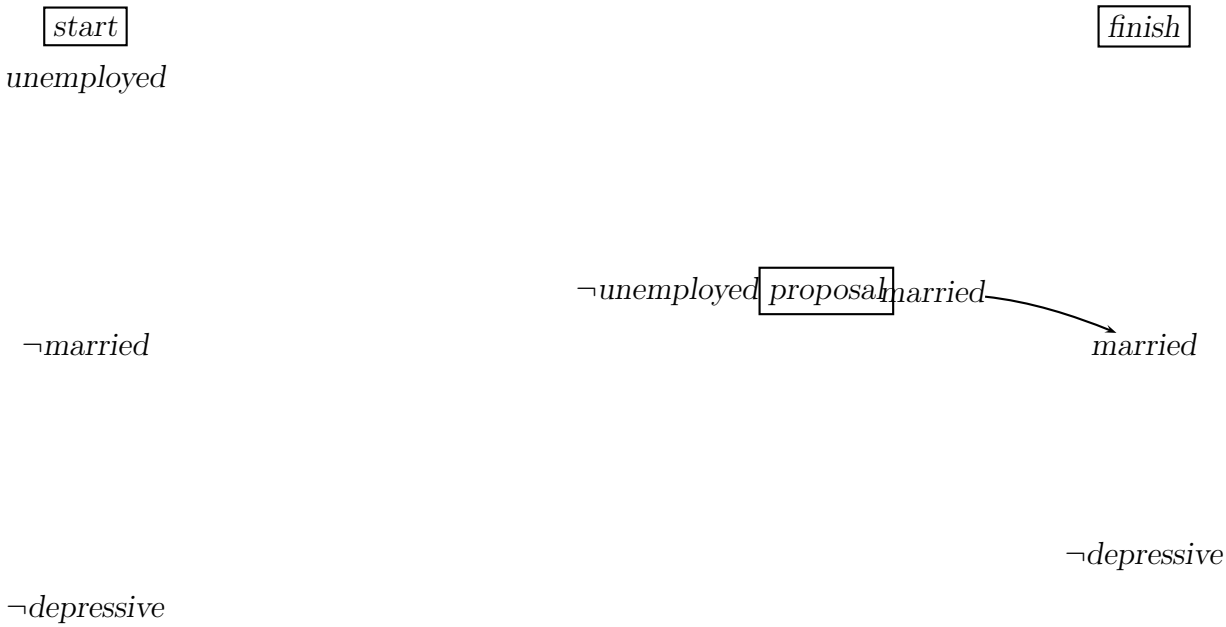
Here's the final plan:



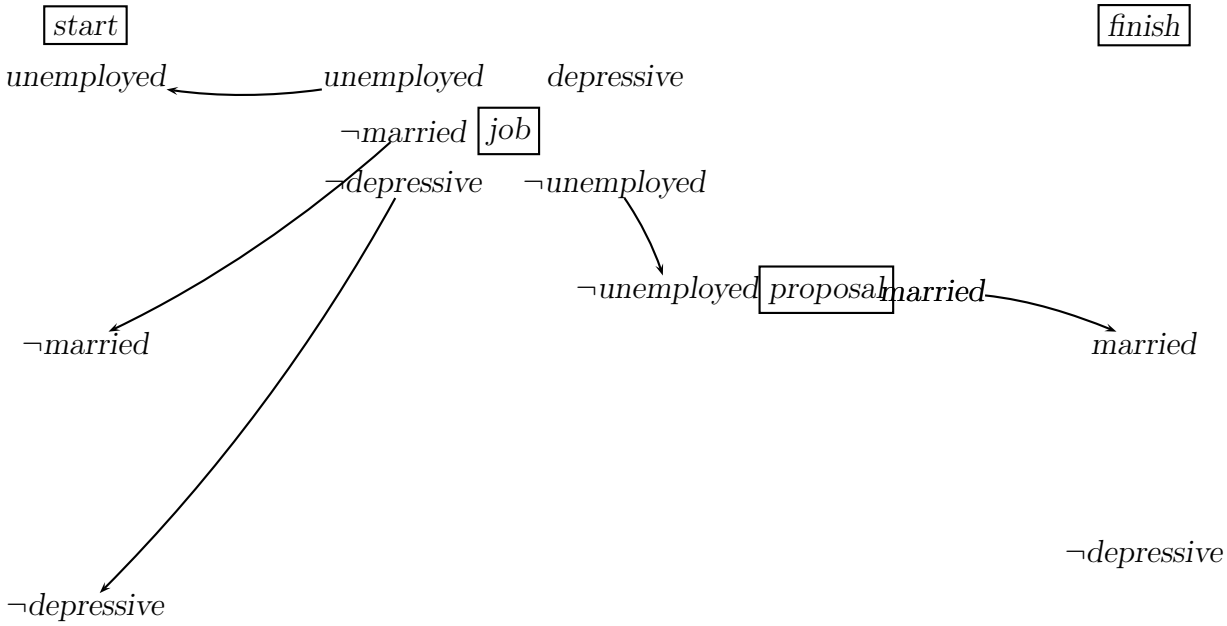
5. *POP: (see 10.1 for the discription of the steps). Credits go to Johannes Tran for the solution.*

Step 1:

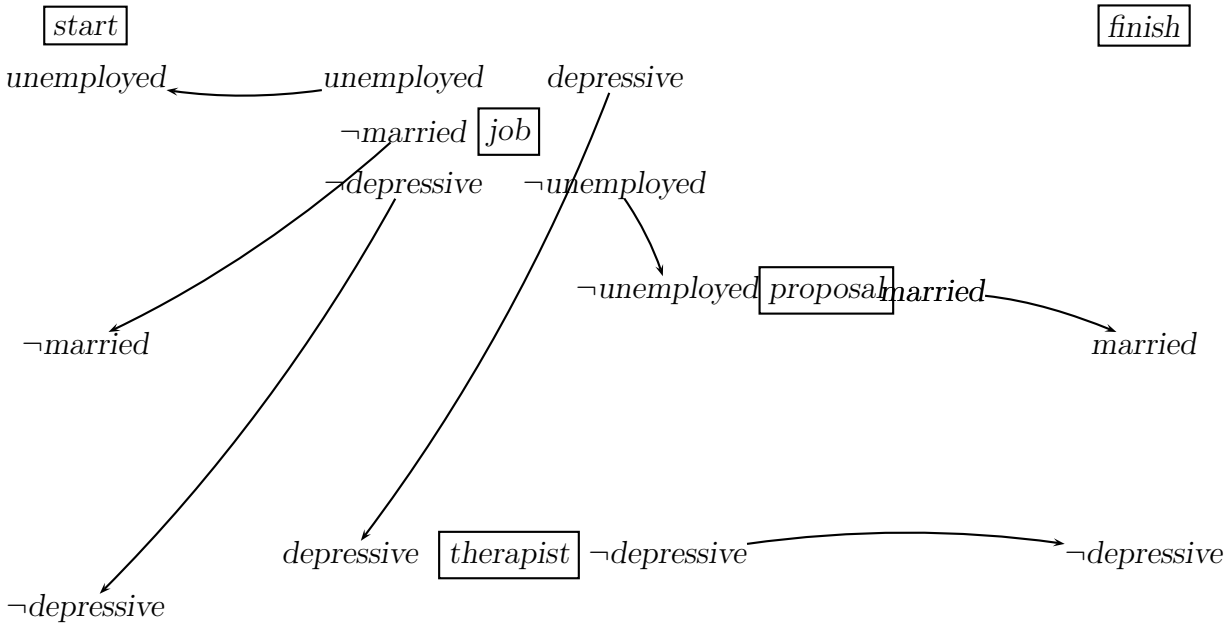
Step 2:



Step 3:



Step 4:



We notice that one difference between planning graphs and POP is that during the construction of the planning graph, the operators are applied in forward direction directly to the description of the states of the planning problem, whereas POP first looks at the goals (and then the subgoals) of the problem.

6. *The new operator does not have an effect on the planning graph, because with the given operators and the given starting state, there will never be a situation where* `crazy` *holds. Because graphplan only considers states that can be reached from the start state, the new operator will never be applicable in this problem. Therefore, the "unnecessary" operator does not change the way Graphplan looks at the problem. On the other hand, POP considers the preconditions of the* `finish` *operator first, which means that the new operator can be applied to satisfy* `married`*, which is one of the goals. Only later the POP algorithm will come to the point where the preconditions of the new operator cannot be fulfilled, and needs to backtrack.*

Exercise 10.4
Show the following: When a literal does not appear in the final level of the planning graph, it cannot be achieved.

Solution:

Proof by contradiction. Assume that the literal l is achievable. Then there exists at least one (totally ordered) sequence of actions $A_0, ...A_n$ that achieves the literal l from the preconditions of the problem. If this is the case, then – by construction of the graph – the first level contains the application of A_0, and the effects of the application. Then – again by construction – the second level contains A_1 and the respective effects, and so on (until A_n)[1]. But this means that the literal l is in the graph – a contradiction.

---

[1]Note that A_n need not necessarily appear on level n for the first time, but could also appear earlier.