

# CS7612 Homework 3: Graphplan, SATPlan, Partial-Order Planning, Integer Programming

William Halliburton

Spring 2003

## 1 Problem 1

Initial State	A B C	Goal	C D E
Operator	Preconditions	Add	Delete
O1	A	F	C
O2	A	D	B
O3	B	C	
O4	F	D E G	

### 1.1 Graphplan

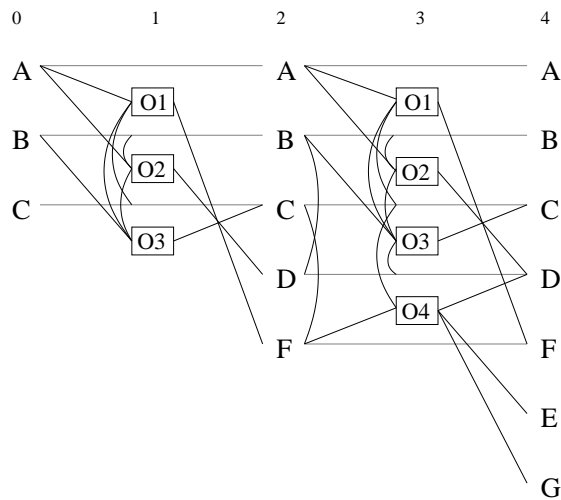


Figure 1: Planning Graph. Shown in the manner used in Daniel Weld's *Recent Advances in AI Planning*. The graph has been expanded two time steps which is sufficient to find a valid plan.

From the Weld paper:

- Two actions instances at level  $i$  are mutex if either
  - *Inconsistent Effects*: the effect of on action the the negation of another action's effect, or
  - *Interference*: one action deletes the precondition of another, or
  - *Competing Needs*: the actions have preconditions that are mutually exclusive at level  $i - 1$ .
- Two propositions at level  $i$  are mutex if all ways of achieving the propositions (*i.e.*, actions at level  $i - 1$ ) are pairwise mutex (*Inconsistent Support*).

**Level 0** begins with the initial state of  $A B C$ .

**Level 1** consists of the possible actions that have preconditions satisfied from level 0. All arcs draw between levels indicate mutex relations. Action  $O1$  and  $O3$  are mutually exclusive (mutex) due to inconsistent effects.  $O1$  and the maintenance action of  $C$  are also mutex for the same reason.  $O2$  is mutex  $O3$  by interference and mutex the maintenance action of  $B$  by inconsistent effects.

**Level 2** consists of the possible effects from the actions in level 1.  $A$  and  $B$  are possible due to the maintenance actions.  $C$  is possible from a maintenance action and action  $O3$ .  $D$  is possible from action  $O2$ , and  $F$  is possible from action  $O1$ .  $B D$  and  $C F$  are mutex by inconsistent support

**Level 3** contains all the actions and mutexes from level 1 with one additional action and two additional mutexes. Action  $O4$  is now possible due to the addition of  $F$  at level 2. Action  $O4$  is mutex the maintenance action of  $C$  by competing needs. Action  $O3$  is mutex the maintenance action of  $D$  by competing needs.

**Level 4** consists of all the possible effects from the actions in level 3.  $A B C D F$  are possible from maintenance actions.  $C$  from action  $O3$ .  $D$  from action  $O2$  or  $O3$ .  $F$  from action  $O1$ .  $E$  and  $G$  from action  $O4$ .

**Graphplan** proceeds by expanding the planning graph to level 2. It then checks to see if effects present at this level can satisfy the goal requirements. In this case  $C$  and  $D$  are present but  $E$  is absent. Graphplan then proceed to expand the graph to level 4. Level 4 contains effects that satisfy the goal conditions, so graphplan begins to search backward to find a valid plan. I will discuss a successful backward search shown in Figure 2, an unsuccessful search would result in dead ends with no actions to choose within the mutex constraints which would then backtrack the search and continue. At level 3

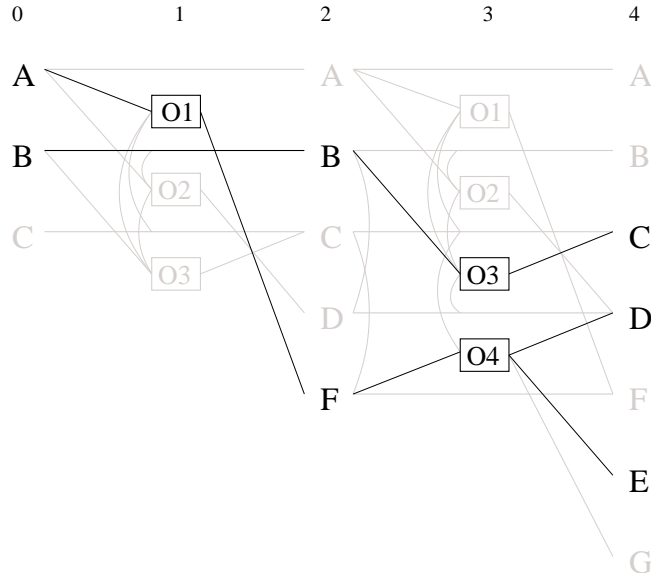


Figure 2: Planning Graph Solution

graphplan chooses  $O_3$  to satisfy  $C$  and  $O_4$  to satisfy  $D$  and  $E$ , this is valid as there is no mutex between  $O_3$  and  $O_4$ . The preconditions for level 3,  $B$  and  $F$ , are now the new goals to achieve. At level 1 graphplan chooses the maintenance action of  $B$  to satisfy  $B$  and action  $O_1$  to satisfy  $F$ . The preconditions from level 1 are satisfied from the initial state so the plan is completed –  $O_1$  at time 1,  $O_3$  and  $O_4$  at time 2.

**Note** that the official graphplan code that we used for our last homework added additional mutex relations that I have not included in my graph. This additional rule places mutex relations between actions and maintenance rules of the effects of the actions. I did not include these mutex relations because the Weld paper does not. For example, at level 1 there would have been a mutex between action  $O_3$  and the maintenance relation of  $C$ .

## 1.2 SAT Plan

The graphplan from Figure 1 is converted into a set of logical statements using several rules.

- The initial state is completely specified at time zero.

$$A_0 \wedge B_0 \wedge C_0 \wedge \neg D_0 \wedge \neg E_0 \wedge \neg F_0 \wedge \neg G_0$$

- The goals hold at the highest level.

$$C_4 \wedge D_4 \wedge E_4$$

- Conflicting actions are mutually exclusive.

$$\begin{array}{ll} \neg(O1_1 \wedge O3_1) & \neg(O2_1 \wedge O3_1) \\ \neg(O1_1 \wedge \text{maintain}C_1) & \neg(O2_1 \wedge \text{maintain}B_1) \\ \neg(O1_3 \wedge O3_3) & \neg(O2_3 \wedge O3_3) \\ \neg(O1_3 \wedge \text{maintain}C_3) & \neg(O2_3 \wedge \text{maintain}B_3) \\ \neg(O3_3 \wedge \text{maintain}D_3) & \neg(O4_3 \wedge \text{maintain}C_3) \end{array}$$

- Actions imply their preconditions.

$$\begin{array}{ll} O1_1 \rightarrow A_0 & \text{maintain}A_1 \rightarrow A_0 \\ O2_1 \rightarrow A_0 & \text{maintain}B_1 \rightarrow B_0 \\ O3_1 \rightarrow B_0 & \text{maintain}C_1 \rightarrow C_0 \\ O1_3 \rightarrow A_2 & O2_3 \rightarrow A_2 \\ O3_3 \rightarrow B_2 & O4_3 \rightarrow F_2 \\ \text{maintain}A_3 \rightarrow A_2 & \text{maintain}B_3 \rightarrow B_2 \\ \text{maintain}C_3 \rightarrow C_2 & \text{maintain}D_3 \rightarrow D_2 \\ \text{maintain}F_3 \rightarrow F_2 & \end{array}$$

- Each fact implies the disjunctions of all actions at the previous level.

$$\begin{array}{ll} A_2 \rightarrow \text{maintain}A_1 & B_2 \rightarrow \text{maintain}B_1 \\ C_2 \rightarrow \text{maintain}C_1 \vee O3_1 & D_2 \rightarrow O2_1 \\ F_2 \rightarrow O1_1 & A_4 \rightarrow \text{maintain}A_3 \\ B_4 \rightarrow \text{maintain}B_3 & C_4 \rightarrow \text{maintain}C_3 \vee O3_3 \\ D_4 \rightarrow \text{maintain}D_3 \vee O2_3 \vee O4_3 & F_4 \rightarrow \text{maintain}F_3 \vee O1_3 \end{array}$$

These rules are then combined into one conjunction which is solved using systematic or stochastic search. Remember that  $P \rightarrow Q \Rightarrow \neg P \vee Q$  and  $\neg(P \wedge Q) \Rightarrow \neg P \vee \neg Q$ .

**SAT Instance**  $(A_0 \wedge B_0 \wedge C_0 \wedge \neg D_0 \wedge \neg E_0 \wedge \neg F_0 \wedge \neg G_0) \wedge (C_4 \wedge D_4 \wedge E_4) \wedge \neg(O1_1 \wedge O3_1) \wedge \neg(O2_1 \wedge O3_1) \wedge \neg(O1_1 \wedge \text{maintain}C_1) \wedge \neg(O2_1 \wedge \text{maintain}B_1) \wedge \neg(O1_3 \wedge O3_3) \wedge \neg(O2_3 \wedge O3_3) \wedge \neg(O1_3 \wedge \text{maintain}C_3) \wedge \neg(O2_3 \wedge \text{maintain}B_3) \wedge \neg(O3_3 \wedge \text{maintain}D_3) \wedge \neg(O4_3 \wedge \text{maintain}C_3) \wedge (O1_1 \rightarrow A_0) \wedge (O2_1 \rightarrow A_0) \wedge (O3_1 \rightarrow B_0) \wedge (\text{maintain}A_1 \rightarrow A_0) \wedge (\text{maintain}B_1 \rightarrow B_0) \wedge (\text{maintain}C_1 \rightarrow C_0) \wedge (O1_3 \rightarrow A_2) \wedge (O2_3 \rightarrow A_2) \wedge (O3_3 \rightarrow B_2) \wedge (O4_3 \rightarrow F_2) \wedge (\text{maintain}A_3 \rightarrow A_2) \wedge (\text{maintain}B_3 \rightarrow B_2) \wedge (\text{maintain}C_3 \rightarrow C_2) \wedge (\text{maintain}D_3 \rightarrow D_2) \wedge (\text{maintain}F_3 \rightarrow F_2) \wedge (A_2 \rightarrow \text{maintain}A_1) \wedge (B_2 \rightarrow \text{maintain}B_1) \wedge (C_2 \rightarrow \text{maintain}C_1 \vee O3_1) \wedge (D_2 \rightarrow O2_1) \wedge (F_2 \rightarrow O1_1) \wedge (A_4 \rightarrow \text{maintain}A_3) \wedge (B_4 \rightarrow \text{maintain}B_3) \wedge (C_4 \rightarrow \text{maintain}C_3 \vee O3_3) \wedge (D_4 \rightarrow \text{maintain}D_3 \vee O2_3 \vee O4_3) \wedge (F_4 \rightarrow \text{maintain}F_3 \vee O1_3)$

The resulting solution is  $A_0 \wedge B_0 \wedge C_0 \wedge \neg D_0 \wedge \neg E_0 \wedge \neg F_0 \wedge \neg G_0 \wedge O_{1_1} \wedge \text{maintain}B_1 \wedge B_2 \wedge F_2 \wedge O_{3_3} \wedge O_{4_3} \wedge C_4 \wedge D_4 \wedge E_4 \wedge \neg O_{3_1} \wedge \neg \text{maintain}C_1 \wedge \neg O_{2_1} \wedge \neg D_2 \wedge \neg C_2 \wedge \neg \text{maintain}C_3 \wedge \neg \text{maintain}D_3 \wedge \neg O_{1_3} \wedge \neg O_{2_3}$ . All other values are a do not care. This solution to the conjunction can easily be converted back into the planning problem by selecting the actions that are true, in this case  $O_{1_1}$ ,  $O_{3_3}$ , and  $O_{4_3}$ .

### 1.3 Partial Order Plan

The following diagrams will show one way that a partial order planner would find a solution the planning problem. I will only display a perfect selection sequence, but a real planner may choose incorrectly resulting in backtracking. Each square is an operator. The letters above the squares are preconditions and the letters below are effects. Single width arrow lines are ordering constraints, and double width arrow lines are causal links. I have adopted this display method from Russell and Norvig's *Artificial Intelligence - A Modern Approach*. Notice that not all ordering constraints are shown in the graph as causal links are assumed to be accompanied by a similar ordering constraint and ordering constraints that can be inferred from those shown are not displayed. For example every operator has an ordering constraint of  $Start \prec Operator \prec Goal$  but these are not displayed if they can be inferred.

See Figure 3

**Step 0** The plan is created with only the start and goal states.

**Step 1** The planner selects for a subgoal the precondition  $D$  from the goal state. It then chooses the  $O_4$  operator to satisfy this subgoal. It places an ordering constrain  $Start \prec O_4 \prec Goal$ , a causal link  $O_4 \xrightarrow{D} Goal$  and ordering constrain  $O_4 \prec Goal$  (from now on I will omit the mention of the ordering constraint that accompanies the causal link - it is now implied).

**Step 2** The planner selects for a subgoal the precondition  $E$  from the goal state. It then chooses the  $O_4$  operator to satisfy this subgoal. It adds a causal link  $O_4 \xrightarrow{E} Goal$ .

**Step 3** The planner selects for a subgoal the precondition  $F$  from operator  $O_4$ . It then chooses the  $O_1$  operator to satisfy this subgoal. It adds an ordering constraint  $Start \prec O_1 \prec Goal$  and a causal link  $O_1 \xrightarrow{F} O_4$

**Step 4** The planner selects for a subgoal the precondition  $A$  from operator  $O_1$ . It then chooses the start state to satisfy this subgoal. It adds a causal link  $Start \xrightarrow{A} O_1$ .

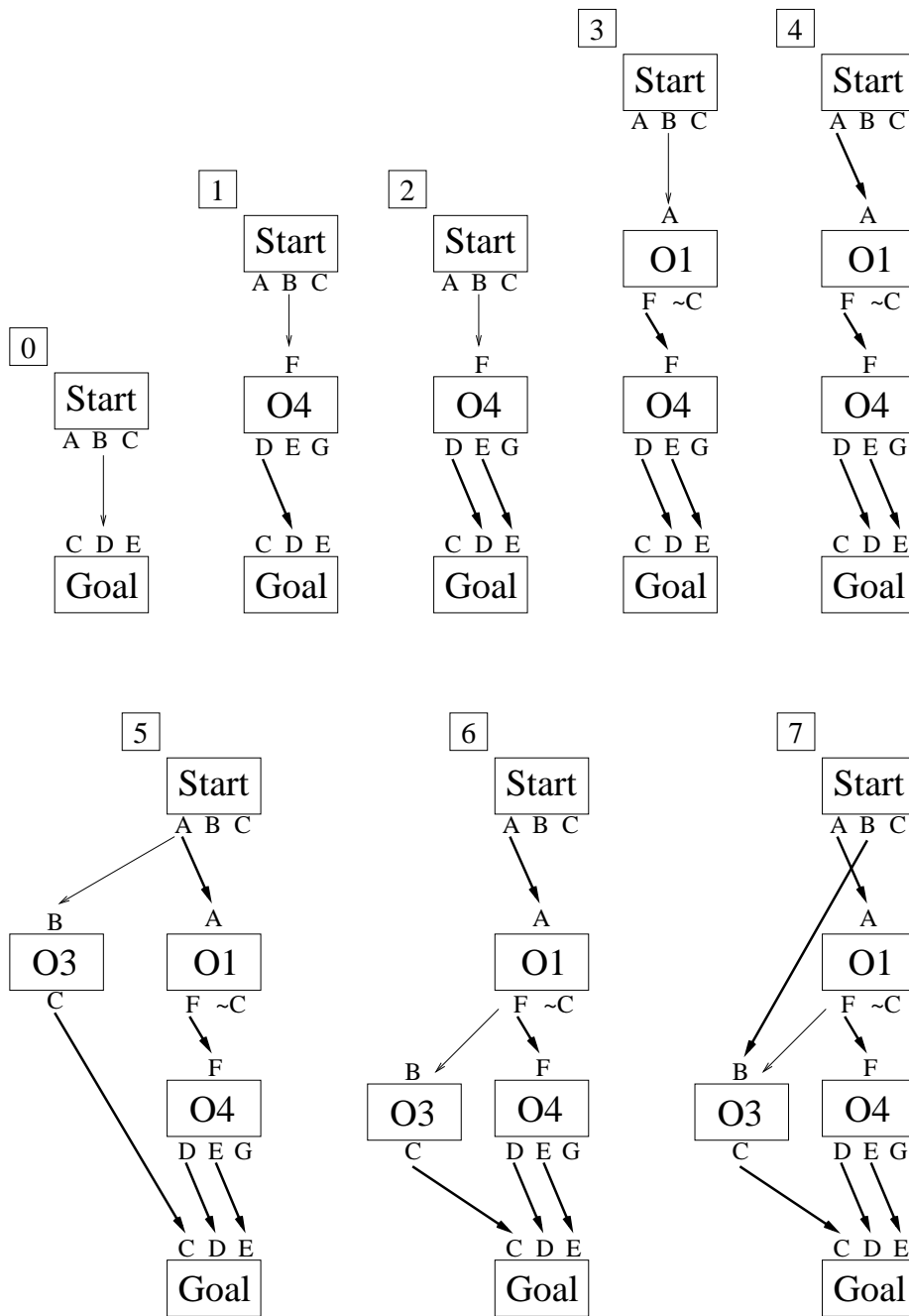


Figure 3: Steps taken by a partial order planner with perfect selection.

**Step 5** The planner selects for a subgoal the precondition  $C$  from operator goal state. It then chooses the  $O3$  operator to satisfy this subgoal. It places an ordering constrain  $Start \prec O3 \prec Goal$  and a causal link  $O3 \xrightarrow{C} Goal$ .

**Step 6** There is a threat between  $O3$  and  $O1$ . The planner chooses to promote  $O1$  by adding an ordering constraint  $O1 \prec O3$ .

**Step 7** The planner selects for a subgoal the precondition  $B$  from operator  $O3$ . It then chooses the start state to satisfy this subgoal. It adds a causal link  $Start \xrightarrow{B} O3$ .

**Finished** The planner has now reached a solution.  $O1$  followed by  $O3$  and  $O4$  together or in any order.

## 1.4 Integer Programming

The graphplan from Figure 1 is converted into a set of constraints using several rules. The definition of these constraints are found in *On the Use of Integer Programming Models in AI Planning* by Vossen, Ball, Lotem, and Nau. From Figure 1 levels 0 and 1 are period 1, levels 2 and 3 are period 2, and level 4 is period 3.

$$X_{e,i} = \begin{cases} 1 & \text{if effect } e \text{ is true in period } i, \\ 0 & \text{otherwise.} \end{cases}$$

$$Y_{a,i} = \begin{cases} 1 & \text{if action } a \text{ is carried out in period } i, \\ 0 & \text{otherwise.} \end{cases}$$

$Y_{e,i}$  signifies the maintenance action for effect  $e$  during period  $i$ .

### Constraints

- Initial

$$\begin{aligned} X_{A,1} &= X_{B,1} = X_{C,1} = 1 \\ X_{D,1} &= X_{E,1} = X_{F,1} = X_{G,1} = 0 \end{aligned}$$

- Goal

$$X_{C,3} = X_{D,3} = X_{E,3} = 1$$

- Precondition

$$\begin{array}{lll} Y_{O1,1} \leq X_{A,1} & Y_{O2,1} \leq X_{A,1} & Y_{O3,1} \leq X_{B,1} \\ Y_{A,1} \leq X_{A,1} & Y_{B,1} \leq X_{B,1} & Y_{C,1} \leq X_{C,1} \\ Y_{O1,2} \leq X_{A,2} & Y_{O2,2} \leq X_{A,2} & Y_{O3,2} \leq X_{B,2} \\ Y_{O4,2} \leq X_{F,2} & Y_{A,2} \leq X_{A,2} & Y_{B,2} \leq X_{B,2} \\ Y_{C,2} \leq X_{C,2} & Y_{D,2} \leq X_{D,2} & Y_{F,2} \leq X_{F,2} \end{array}$$

- Backward Chaining

$$\begin{array}{ll}
 X_{A,2} \leq Y_{A,1} & X_{B,2} \leq Y_{B,1} \\
 X_{C,2} \leq Y_{C,1} + Y_{O3,1} & X_{D,2} \leq Y_{O2,1} \\
 X_{F,2} \leq Y_{O1,1} & X_{A,3} \leq Y_{A,2} \\
 X_{B,3} \leq Y_{B,2} & X_{C,3} \leq Y_{C,2} + Y_{O3,2} \\
 X_{D,3} \leq Y_{D,2} + Y_{O4,2} + Y_{O2,2} & X_{F,3} \leq Y_{F,2} + Y_{O1,2} \\
 X_{E,3} \leq Y_{E,2} & X_{G,3} \leq Y_{G,2}
 \end{array}$$

- Exclusiveness

$$\begin{array}{l}
 Y_{O1,1} + Y_{O3,1} \leq 1 \\
 Y_{O2,1} + Y_{O3,1} \leq 1 \\
 Y_{O1,2} + Y_{O3,2} \leq 1 \\
 Y_{O2,2} + Y_{O3,2} \leq 1
 \end{array}$$

**Solution** The solution is found by feeding the previous constraints into a linear function solver.

## 2 Problem 2

### 2.1 Conformant Plan

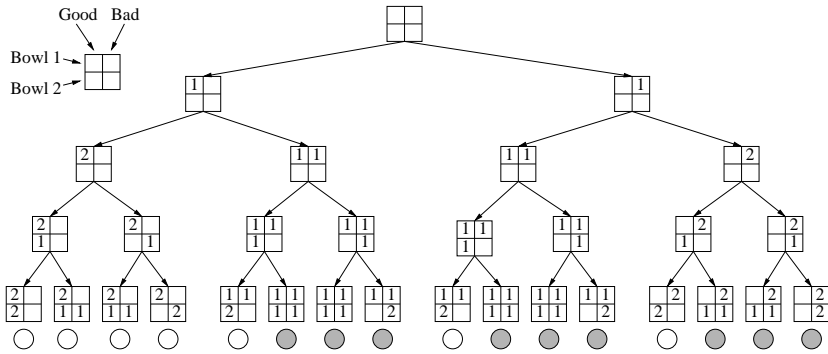


Figure 4: Conformant Plan. The first two levels are breaking an egg into bowl 1. The last two levels are breaking an egg into bowl 2.

The conformant plan that maximizes the probability of having a bowl with two good eggs with 5 or fewer actions is the simple plan of cracking two eggs into each bowl. The reasoning for this plan is that there is no need for cracking less or more than two eggs into a bowl as this will not allow for a solution. There is no way to choose actions based on past sensing so there is no reason to smell the bowls. There is no reason to dump a bowl into the trash because you are

equally likely to dump out a good egg as a bad. There is no reason to dump a bowl into another bowl as you might as well crack another egg into the bowl. You are more likely to succeed if you use fill both bowls as opposed to only one of the bowls. Under these constraints and reasons one of the best plans is shown in Figure 4. The probability of succeeding is shown as  $\frac{7}{16}$  or 0.4375.

## 2.2 Conditional Plan

The following conditional plan has a probability of success of 50%. This percentage is easily calculated from the last statement. There is a  $\frac{1}{4}$  chance that both bowl 1 and bowl 2 are OK resulting in a win. The second condition of has a  $\frac{1}{4}$  chance that bowl 1 is OK and bowl 2 is bad so the chance that emptying and egg into bowl 1 resulting in a win is  $\frac{1}{4} * \frac{1}{2} = \frac{1}{8}$ . The same logic holds for the third conditional. The fourth conditional is the probability that both initial eggs are bad or  $\frac{1}{4}$ . This results in a total probability for winning of  $\frac{2}{4}$  or 0.5.

```
(empty egg bowl1)
(empty egg bowl2)
(smell bowl1)
(smell bowl2)
(if (and (ok bowl1) (ok bowl2)) (dump bowl1 bowl2)
    (else (if (ok bowl1)) (empty egg bowl1)
          (else (if (ok bowl2)) (empty egg bowl2))
          (else fail))))
```

The reasoning used to justify this plan as the maximum probability is that the sensing of each bowl is necessary for each initial egg in order to not waste an action on a bad egg. Dumping eggs into each other without sensing is also without merit as you may contaminate a good egg with a bad. Dumping bowls into the trash is not useful due the plan length limit - for example, empty then smell then dump would only leave time to empty two more eggs. Sensing of all bowls early is very useful because the conditional can have as many clauses as necessary to incorporate all the information. Conditionals must have sensing and sensing must have eggs so the conditionals must come after the smelling. You may argue that you could empty one bowl then smell then start conditionally branching but the only actions left are to empty into the smelled bowl or the empty bowl — if you empty into the smelled bowl you are wasting an action that could be incorporated into a conditional later — if you empty into the empty bowl you are just following the shown plan. Under these constraints the previous plan is a plan with the best outcome. This is only one of several possible plans of this type as it is possible to empty then smell then empty then smell.