Dipartimento di elettronica e informazione
Politecnico di Milano

Artificial Intelligence 2010-11

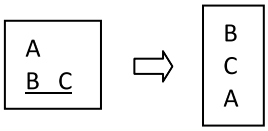
11. Plan-Space Planning

11.1 Partially ordered plans

Sussman's anomaly (see Section 10.4) is due to the fact that different goals in a goal list are *not independent*: by this we mean that the actions required to achieve them interfere with each other. Consider again the planning problem

S_0 = {On(A,B), On(B,F), On(C,F), Clear(A), Clear(C), Clear(F)}

G_0 = On(B,C) $\wedge$ On(C,A)



Now suppose that we try to achieve the two goals independently of each other. The problem

S_0 = {On(A,B), On(B,F), On(C,F), Clear(A), Clear(C), Clear(F)}

G_1 = On(B,C)

can be easily solved using a STRIPS-like planner, giving the plan

P_1 = move(A,B,F); move(B,F,C)

Also the problem

S_0 = {On(A,B), On(B,F), On(C,F), Clear(A), Clear(C), Clear(F)}

G_2 = On(C,A)

can be easily solved, giving the plan

P_2 = move(C,F,A)

Now let us remark that the optimal plan to achieve G_0 can be obtained by suitably 'merging' the two plans:

P_0 = $\overbrace{\text{move(A,B,F);}\ \underbrace{\text{move(C,F,A)}}_{P_2}\text{; move(B,F,C)}}^{P_1}$

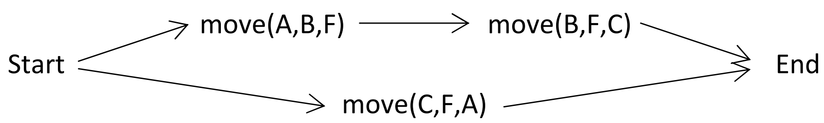
Why do we have to insert action move(C,F,A) right in the middle of plan P_1 and not, for example, at the beginning or at the end of P_1? The reason is that the three actions interfere with each other in various ways: action move(C,F,A) cannot be executed

- before move(A,B,F), because it would destroy a precondition of the latter (i.e., Clear(A));
- after move(B,F,C), because the latter would destroy one of its preconditions (i.e., Clear(C)).

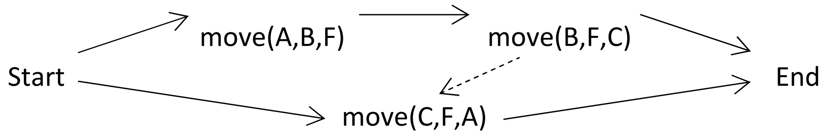
However, we can try to reformulate our planning problem as follows. Given the goal list

G_0 = On(B,C) $\wedge$ On(C,A)

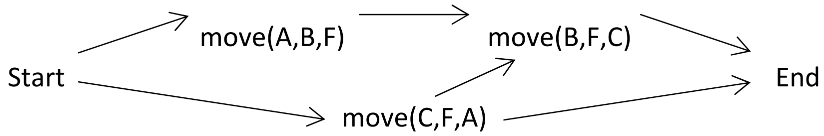
we can build *two* plans, on the assumption that the two goals are independent. What we achieve is a *partially ordered plan*:



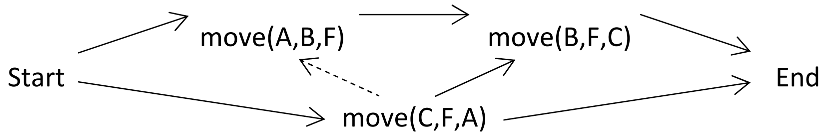
Now, if the two goals were really independent, any total ordering of the actions contained in the partially ordered plan would achieve the final goal list. This sometimes happens, but in general there will be interferences between certain actions. We can, however, insert further ordering relationships into out partially ordered plan, so that such interferences are dealt with. For example, we can discover that action move(B,F,C) *threats* action move(C,F,A), because it destroys a precondition of the latter, Clear(C).



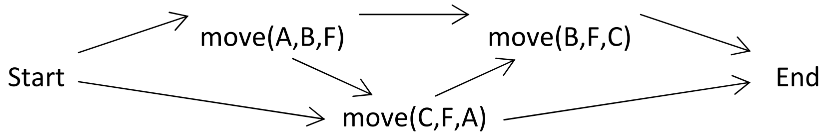
Therefore we add a new order relationship, which runs opposite to the threat:



Moreover, we can discover that action move(C,F,A) threats action move(A,B,F), because it destroys a precondition of the latter, Clear(A):



Therefore we add a new order relationship, which runs opposite to the threat:



Now the plan does not contain any threat. Indeed, the plan has become totally ordered, and thus only one sequence of action can realize it:

move(A,B,F); move(C,F,A); move(B,F,C)

In other cases, after solving all threats we will still have a partially ordered plan; in such cases, any total ordering of the actions that satisfies the partial order will be an acceptable realization of the plan.

The method we have sketched has an important difference with respect to what we have seen so far. STRIPS-like planning tried to find a solution by searching a *space of states*. But the operations we have described above are not carried out in a space of states: rather, they are carried out in a *space of plans*. In other words, the objects being processed are not states, but partial plans. Plan formation of this kind is called *plan-space planning* (PSP)

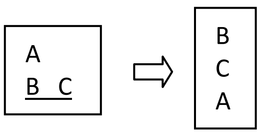
11.2 Plan-space planning: an example

We shall now outline a PSP method. Instead of working on problem states, we shall work on *partial plans*, which will be *refined* until a complete plan is achieved. In general, the complete plan will be partially ordered, even if it can be totally ordered as a limiting case (like in the previous example).

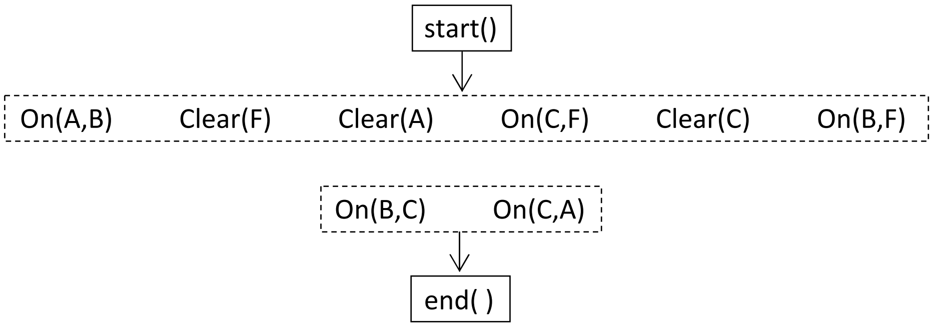
Again, we start from the by now well known Sussman's problem,

S_0 = {On(A,B), On(B,F), On(C,F), Clear(A), Clear(C), Clear(F)}

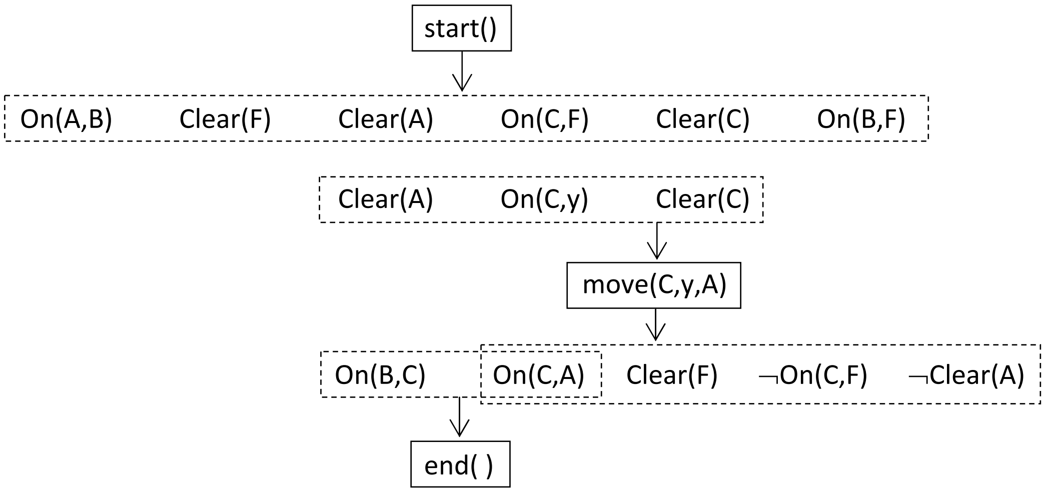
G_0 = On(B,C) $\wedge$ On(C,A)



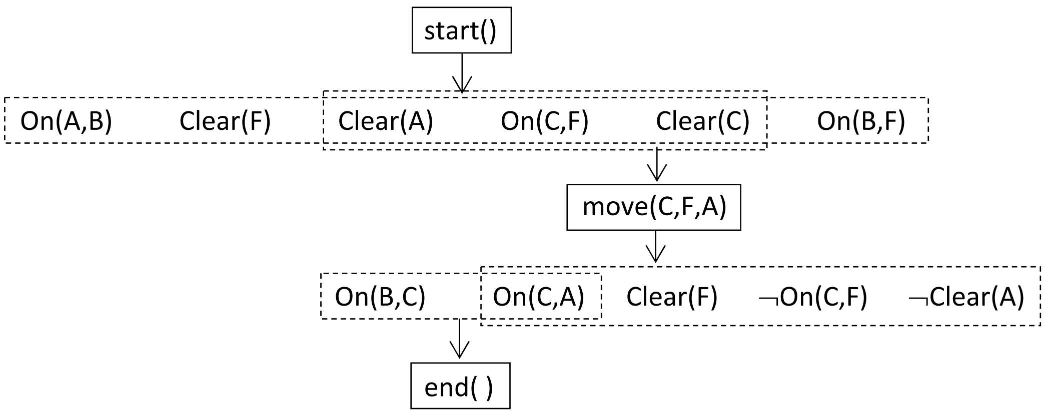
PSP starts from an initial partial plan, which contains exactly two *pseudo-actions*, start() and end(). By definition, action start() has no preconditions, and produces the initial state as effect; action end() has the goal list as preconditions, and produces a *success* as effect. Therefore, the initial partial plan can be represented as:



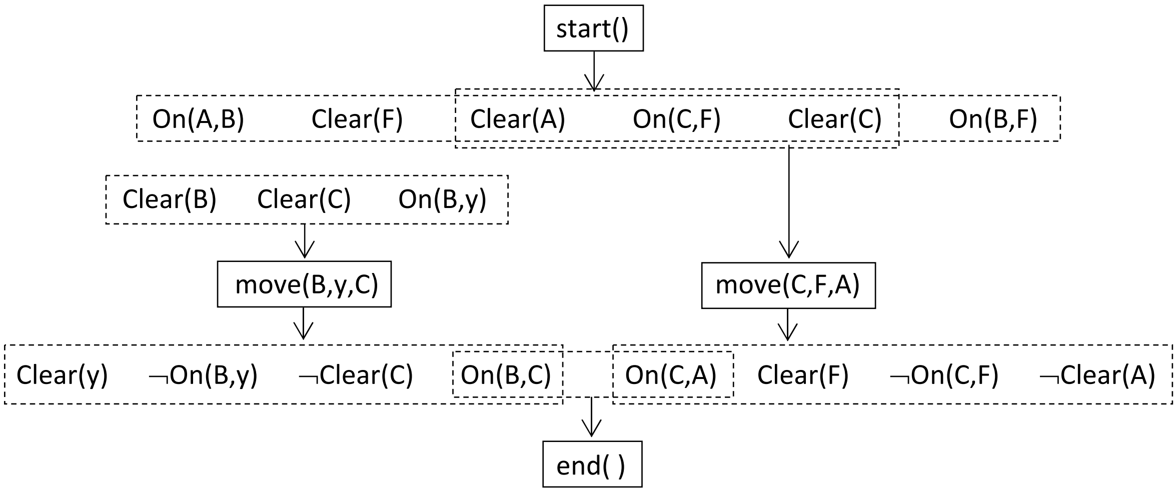
In our graphical conventions, continuous boxes contain actions; dashed boxes contain sets of facts; and arrows connect sets of facts to actions (preconditions), or actions to sets of facts (effects). Now we use a STRIPS-like planning approach to refine this partial plan. Working on goal On(C,A) will produce the following partial plan:



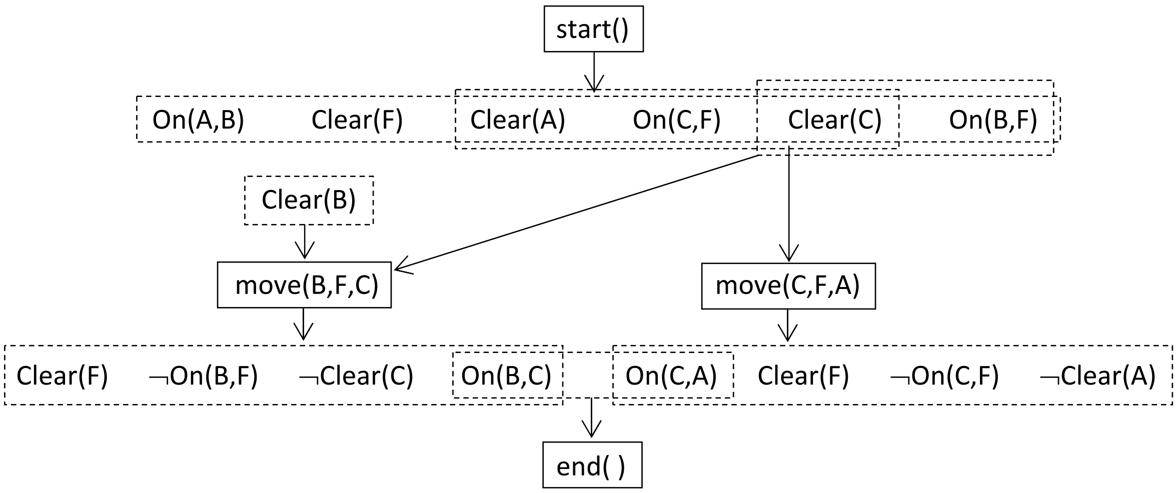
Reasoning as in STRIPS, we can refine this plan into the new partial plan (with y = F):



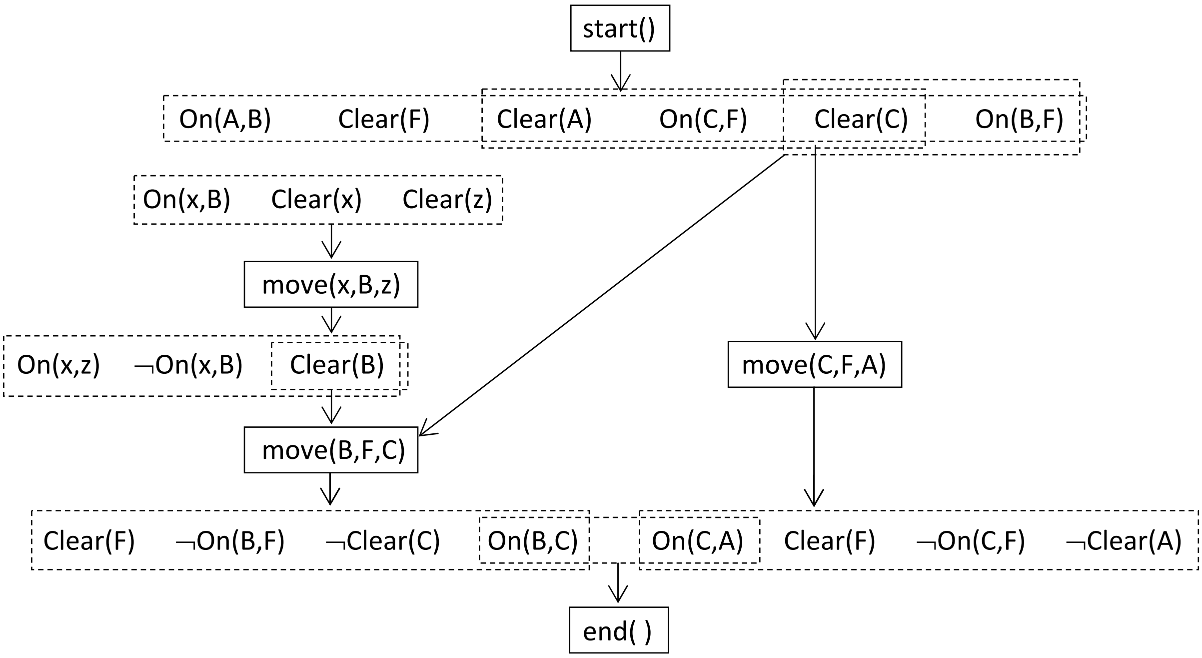
Now we can refine the plan working on the other goal:



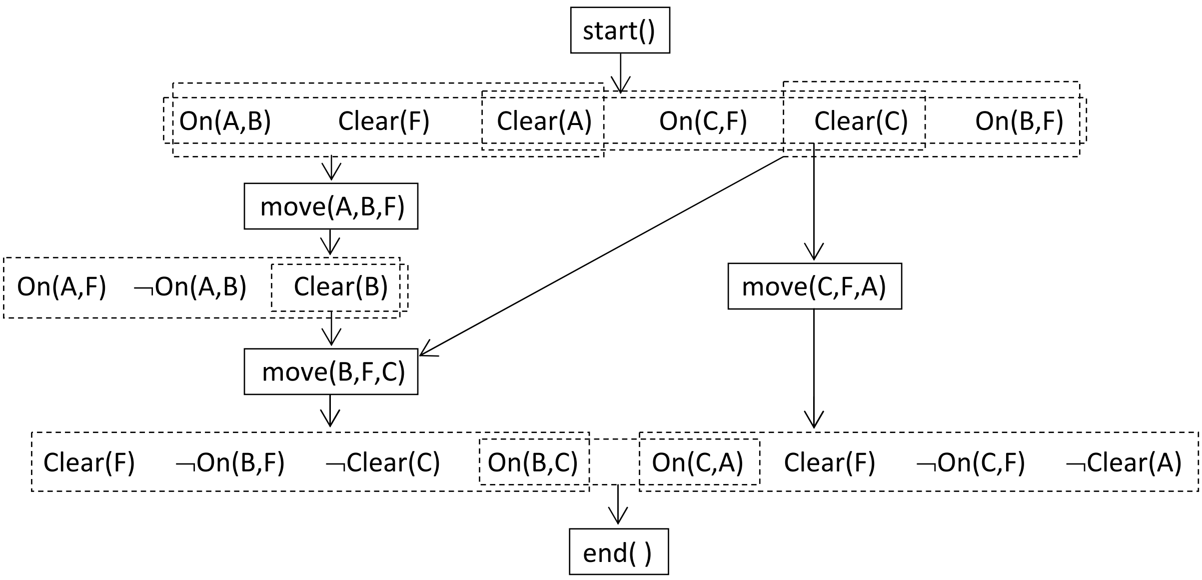
Next (y = F):



Next (y = F):



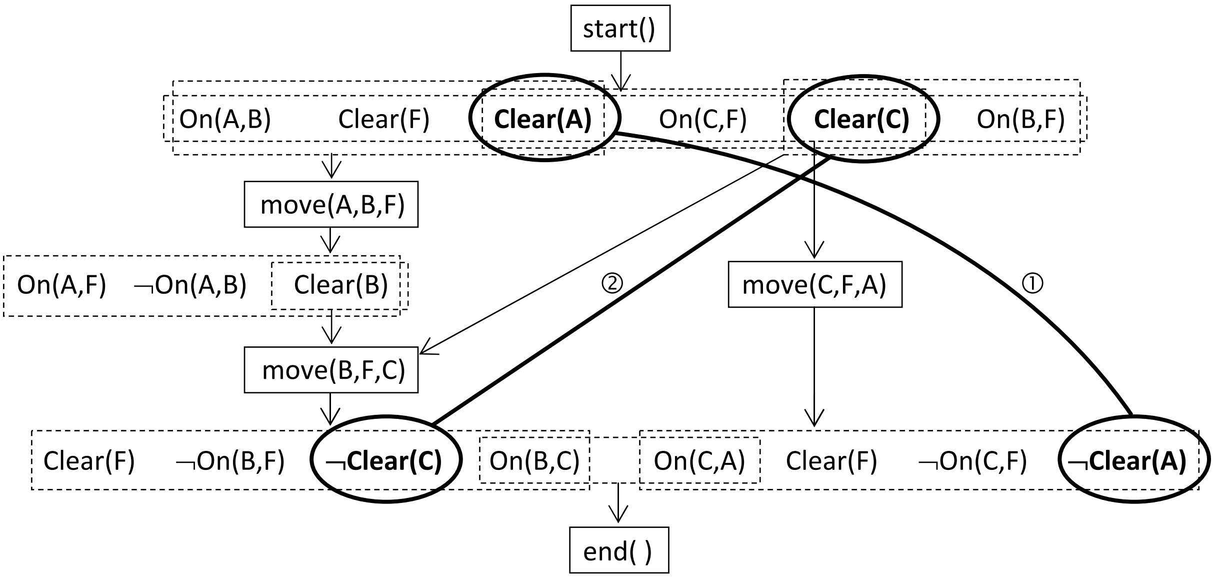
Next (x = A, z = F):



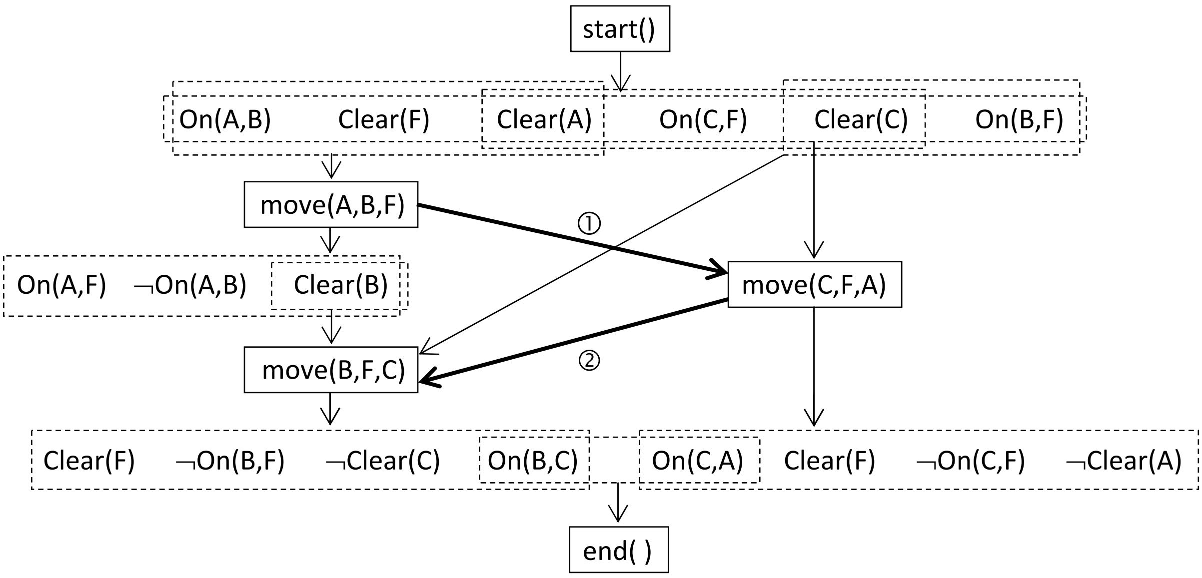
Now all the goals have been achieved, and we have a partially ordered plan. However, before linearising it we have to discover possible threats. Considering that action a_1 threats action a_2 if an effect of a_1 contradicts a precondition of a_2, we find two threats:

①: action move(C,F,A) threats action move(A,B,F)

②: action move(B,F,C) threats action move(C,F,A)



The two threats can be xx by introducing two additional precedence constraints:



No further threats are present. Therefore, we can linearise the partial plan, satisfying the precedence constraints. The only linearization is

move(A,B,F); move(C,F,A); move(B,F,C)

which is the optimal plan for our initial problem.