Planning

to build control algorithms that enable an agent to synthesize a course of action that will achieve its goals.

Situation Calculus

Although it is possible to code a planning problem within the situation calculus and to obtain a plan (situation) through answer extraction, most work in the field has looked at specialized planning algorithms as the most efficient way to proceed. This will be the topic of this section.

Strips Operators

1)PC- Precondition list
2)D - delete list
3)A - Add List

Each a set of literals

```
move(X, Y, Z)
PC: On(X, Y) ^ Clear(X) ^ Clear(Z)
D: Clear(Z), On(X,Y)
A: On(X,Z), clear(Y), Clear(F1)
```


Planning

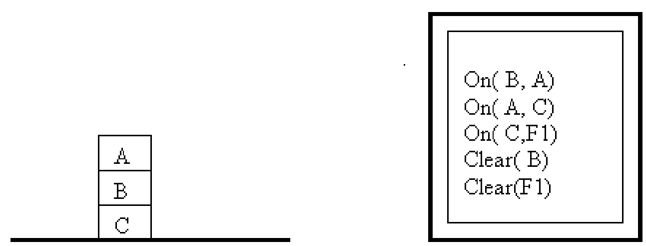
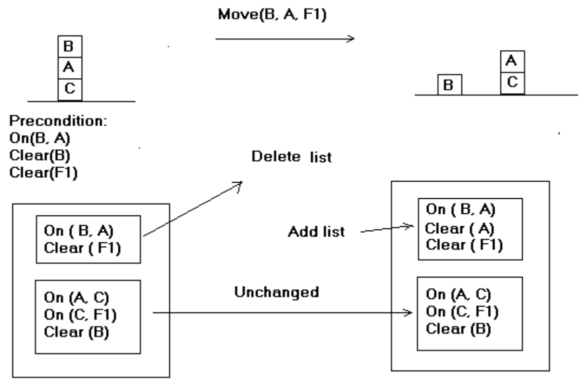


Figure 22.1

F1 = Table from (Nilsson 1998) The state description on the right of the picture above is wrong. What should it be?

Example



A Strips Operator

Example (cont)

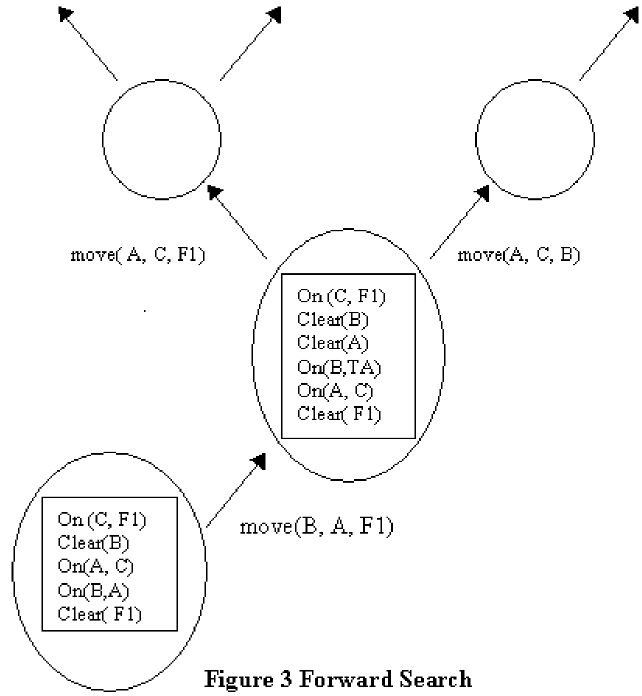


Figure 3 Forward Search

Planning (Cont)

Given a goal formula G, we want to find a state S such that $S \models G$.

Heuristics are needed to do this effectively.

Recursive Strips

STRIPS(γ)

The algorithm given below allows the goal to contain existentially quantified variables in addition to ground literals. Also, the state description may contain arbitrary universally quantified formulas in addition to ground literals . But it may be simpler to to think of both states and goals as being conjunctions of ground literals.

Recursive Strips(cont)

STRIPS(γ).

We assume we have a Global Data Structure **S** which is a set of literals. It is intitially set to the literals true in the intitial state.

1 **repeat** The main loop of STRIPS is iterative and continues until a state description is produced that satisfies the goal, γ. The termination test in step 9 produces a substitution, σ (possibly empty), such that some conjuncts(possibly none) of $\gamma\sigma$ appear in S. There can be several substitutions tried in performing the test, so the test is a possible backtracking point.

2 $g \leftarrow$ an element of $\gamma\sigma$ such that $S \not\models g$. Another selection and therefore a backtracking point. In "means-ends-analysis" terms, g is regarded as a "difference" that must be "reduced" to achieve the goal.

Recursive Strips(cont)

3 $f \leftarrow$ a STRIPS rule whose add list contains a literal λ, that unifies with g with mgu θ. Since there may be several such rules. This is another backtracking point. f is an operator that is "relevant" to reducing the difference.

4 $f' \leftarrow f\theta$ The instance of $f$using substitution θ. Note that f' is not necessrily a ground instance, and therefore its preconditon may contain variables.

5 $p \leftarrow$ precondition formula of f' (instantiated with the substitution θ).

Recursive STRIPS(cont)

6 STRIPS(p). A recursive call to produce a state description that satisfies the subgoal. This call will typically change S.

7 $f'' \leftarrow$ a ground instance of f' applicable in S.

8 $S \leftarrow$ result of applying f'' to S. Note that S always consists of a conjunction of ground literals.

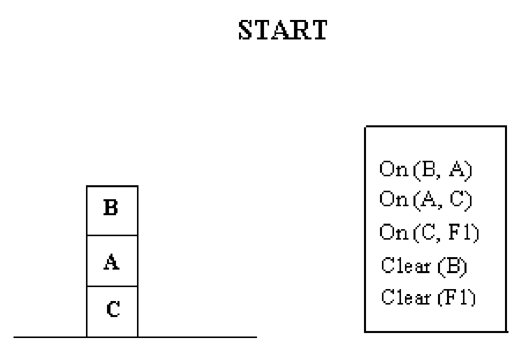
9 **Until** $S \models \gamma$.

from (Nilsson 1998)

Illustration

Consider the following example (Nilsson 1998) to illustrate recursive STRIPS:

The following is the START state:



And the following is the goal state:

$\gamma = On(A, F1) \wedge On(B, F1) \wedge On(C, B)$

Illustration

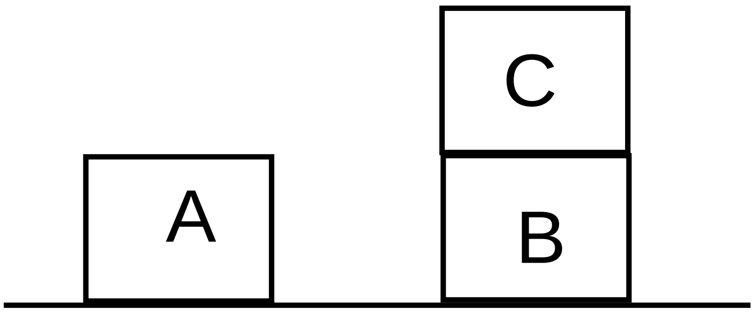


Illustration (cont)

select On(A, F1)as g
select move(A,x, F1)
call Strips (recursive call 1) to achieve
$Clear(A) \wedge Clear(F1) \wedge On(A, x)$
call produces substitution {x/C}

Now

$S \models Clear(F1) \wedge On(A, C)$

but *not* $\models Clear(A)$

Illustration (cont)

So select Clear(A) as g, Select move(y, A, v) to achieve g.

Call STRIPS recursively (Recursive call #2) to achieve the preconditions of move(y,A,v)

PC: $Clear(y) \wedge Clear(v) \wedge On(y, A)$

step 9 produces {y/B}, {v/F1}
now Recursive call #2 succeeds
apply
move (B,A,F1)
(deleting from S the delete list of the action and adding the elements of the add list)

The Plan

Now S is as follows:

```
On(B,F1)
On(A,C)
On(C,F1)
Clear(A)
Clear(B)
Clear(F1)
```

The Plan (continued)

```
now recursive call #1 succeeds
at step 9
S |=  Clear(A)/\ Clear(F1)/\On(A,x)
with {x/C}
```


The Plan Continued

```
apply move(A,C, F1)
(deleting from S the delete list of the action
and adding the elements of the add list)

So now S is as follows:

On(B, F1)
On(A,F1)
On(C,F1)
Clear(A)
Clear(B)
Clear(C)
Clear(F1)

S |= On(A, F1)/\On(B, F1)

but
 S not |= On(C,B)

So STRIPS(On(C, B))
```


The Plan Continued

```
Recursive call
#3 then
apply move(C, F1, B) to finally achieve
            the goal.

So the resulting plan is:

1) move(B,A,F1)
2) move(A,C,F1)
3) move(C,F1,B)
```


Problem

Consider the Following Example:

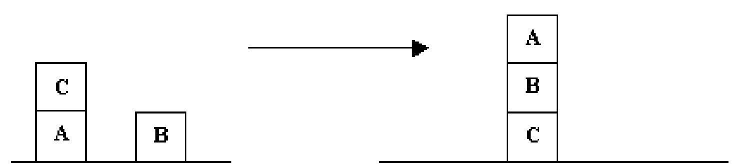


Figure 7 Sussman Anomaly
Goal:
On(A, B) ^ On(B, C)

from (Nilsson 1998)

Can you see why this simple example creates a problem for a STRIPS style planner?

Partial Order Planning

The Sussman Anomaly motivated the development of Partial Order Planners.

STRIPS operators perform state-space search. That is they search through the space of states that are possible solutions to the problem. An alternative – taken by Partial Order Planning – is to search through a space of plans. But now we need a new representation for plans – one in which plans are not completely specified.

Partial Order Planning

Search in the space of Plans by using the following operators to alter plans – making them more specific.

1. add steps
2. reorder steps
3. change partially ordered into fully ordered
4. instantiating variables

Partial Plans

Plans now are incompletely specified. They are not fully ordered. Look at the partial order plan given below and how it can be instantiated into 6 fully ordered plans.

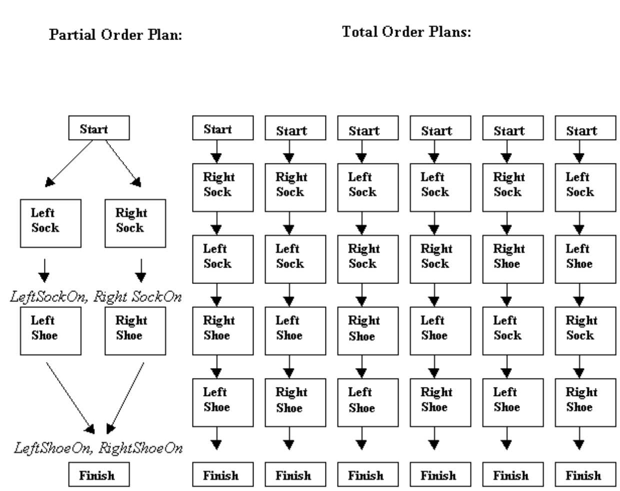


Figure: A partial-order plan for putting on shoes and socks (including preconditions on steps). And the six possible linearizations of the plan.

from (Russell and Norvig 1995)

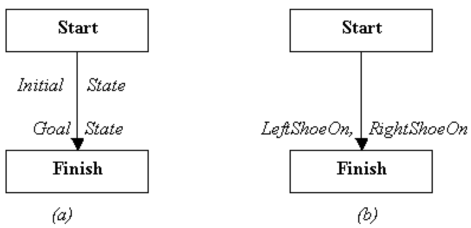
Partial Plans

The plan just specifies that the left show must be put on after the left sock and the right show must be put on after the right sock.

Partial Order Planning

Partial order plans are begun with the simplest plan. They contain a plan consisting of a "dummy" start action which has no prerequisites and has the effect of creating the initial state and a dummy finish action whose prerequisites are the goal state and which does not have any effects.

The rest of the planning process fills in the details.

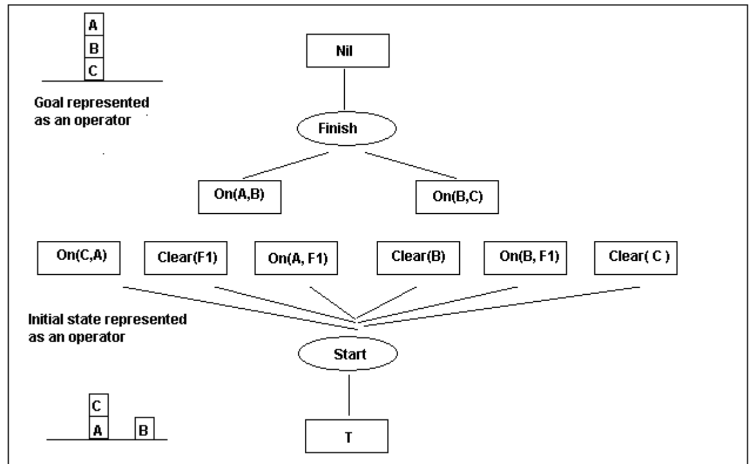


(a) Problems are defined by partial plans containing only Start and Finish steps. The Initial state is entered as the effects of the Start step, and the goal state is the precondition of the Finish step. Ordering constraints are shown as arrows between Boxes.
(b) The initial plan for the shoes-and-socks problem.

Figure11.4

Sussman Anomaly

The following is the initial plan for the Sussman Anomaly:



Graphical Representations fo finish and start Rules

from (Nilsson 1998)

Sussman Anomaly

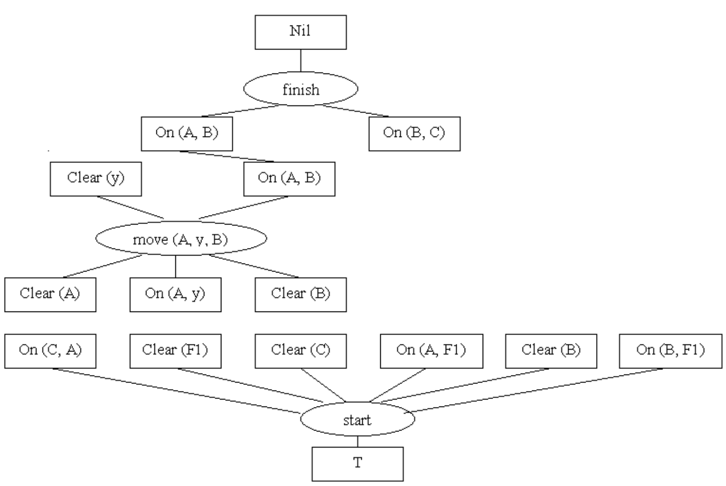
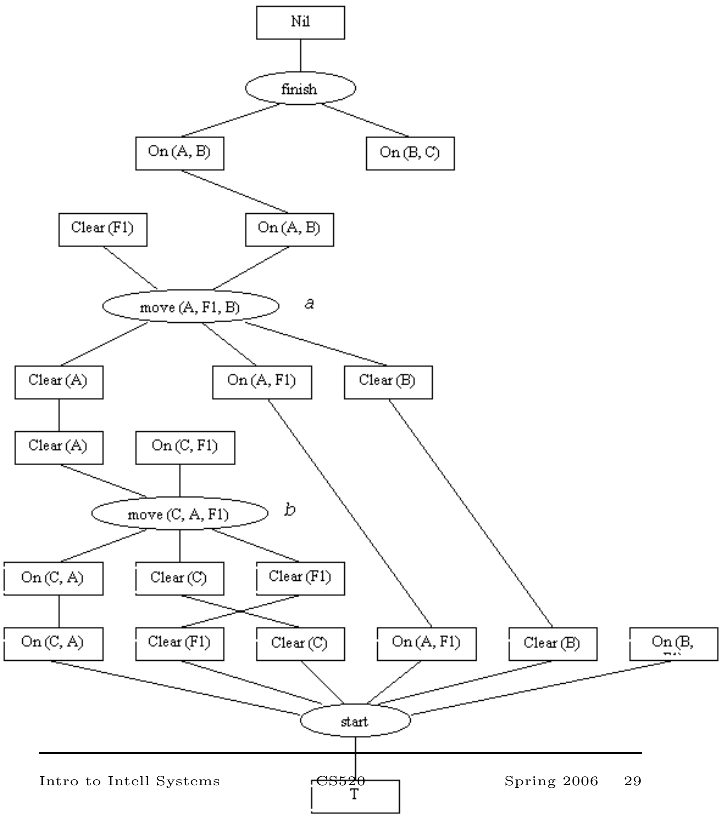


Figure: The Next Plan Structure

Sussman Anomaly

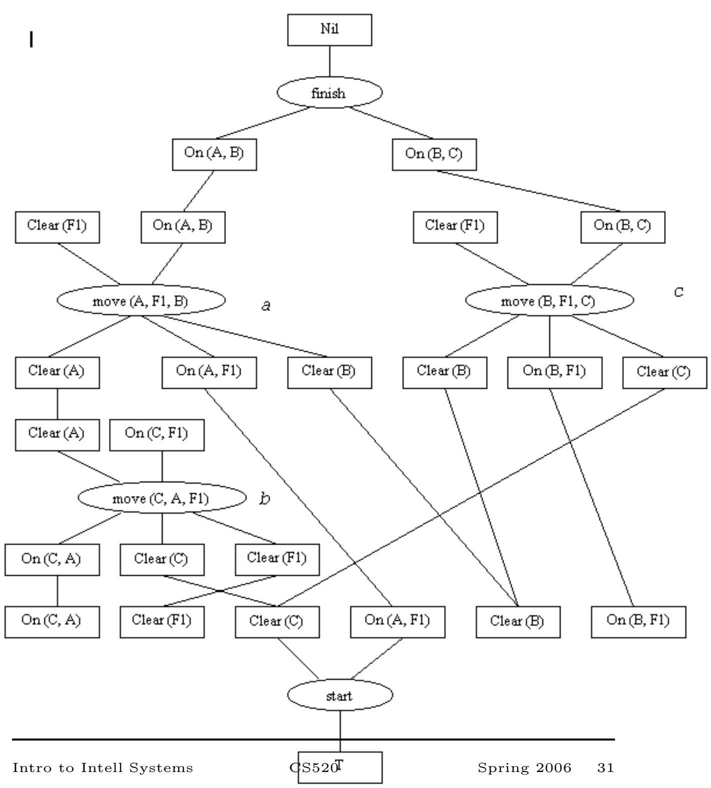
Sussman Anomaly



Additional Points

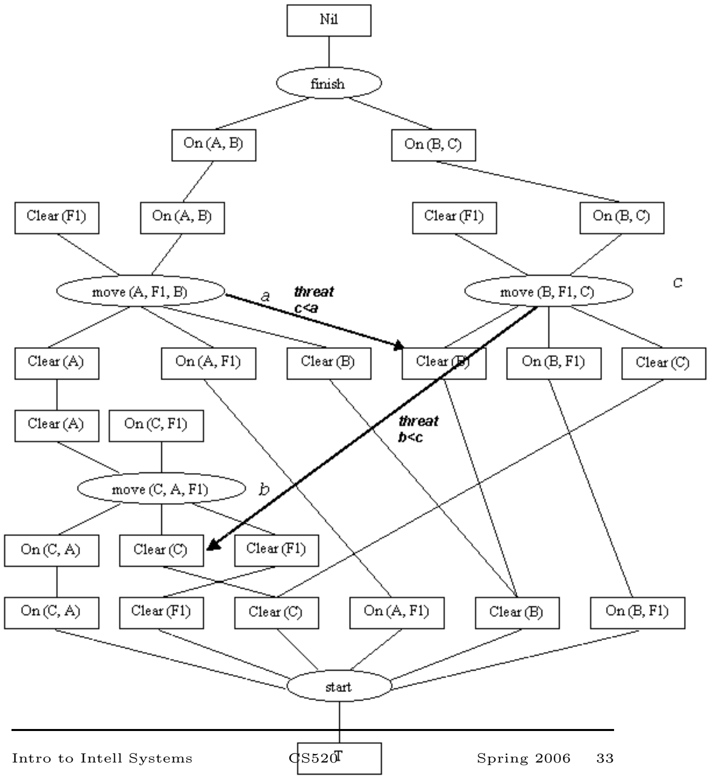
Note that the since b achieves a prerequisite of a, it must be the case that $b < a$ (b occurs before a).

Additional Points



Additional Points

Additional Points



Recent Work

- medic Planner tested in Lisp
- Graph Plan (Blum and Furst)
- Compilation to SAT
- Much work done at University of Washington `http://www.cs.washington.edu`
- Pedagogical Graph plan in Lisp
- Survey Article by Dan Weld
- Applied to Nasa's Deep Space One