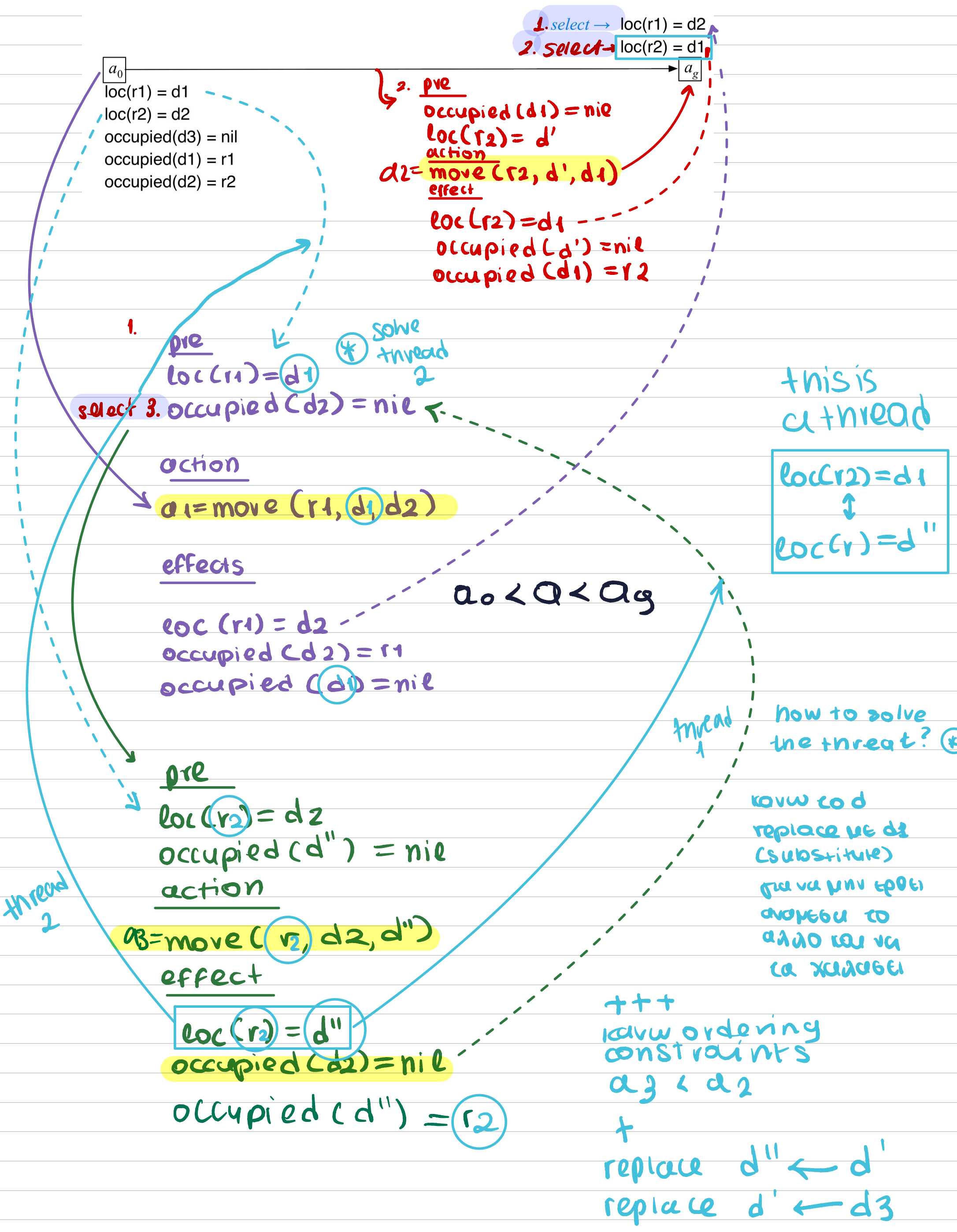
PSP Algorithm



a0

loc(r1) = d1
loc(r2) = d2
occupied(d3) = nil
occupied(d1) = r1
occupied(d2) = r2

* → open goals

① 2 open goals
② 3 open goals
③ 4 open goals
④ 5 open goals, 1 threat
⑤ 4 open goals, 2 threats
⑥ 4 open goals, 1 threat
⑦ 3 open goals, 1 threat
⑧ 3 open goals, 0 threats
⑨ 2 open goals
⑩ 1 open goal
⑪ no open goals, no threats

a1 = move(r1, d1, d2)
pre: loc(r1) = d1, occupied(d2) = nil
eff: loc(r1) = d2, occupied(d1) = nil, occupied(d2) = r1

a2 = move(r2, d3, d1)
pre: loc(r2) = d3, occupied(d1) = nil
eff: loc(r2) = d1, occupied(d3) = nil, occupied(d1) = r2

a3 = move(r, d2, d'3)
pre: occupied(d'3) = nil, loc(r2) = d2
eff: loc(r2) = d'3, occupied(d2) = nil, occupied(d'3) = r

ag
1. loc(r1) = d2
2. loc(r2) = d1

μ: d → d1
μ: r → r2
d'3 → d3
d''3 → d1
a3 < a2 (ordering constraint)
r ≠ r1 (constraint)

threat 1
threat 2.

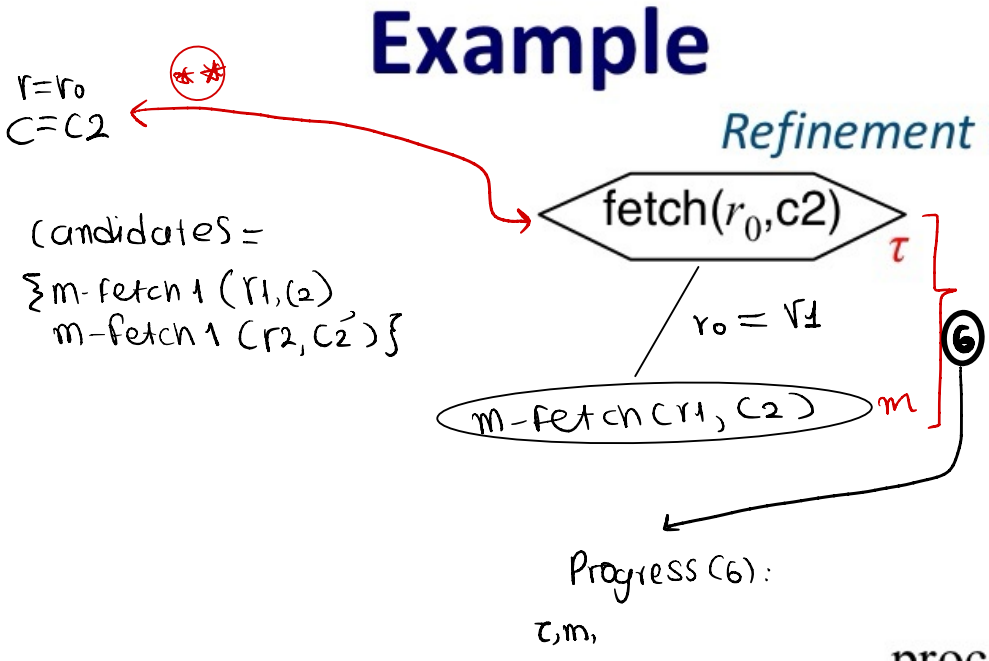
1. 2. 3. 4.

Acting RAE

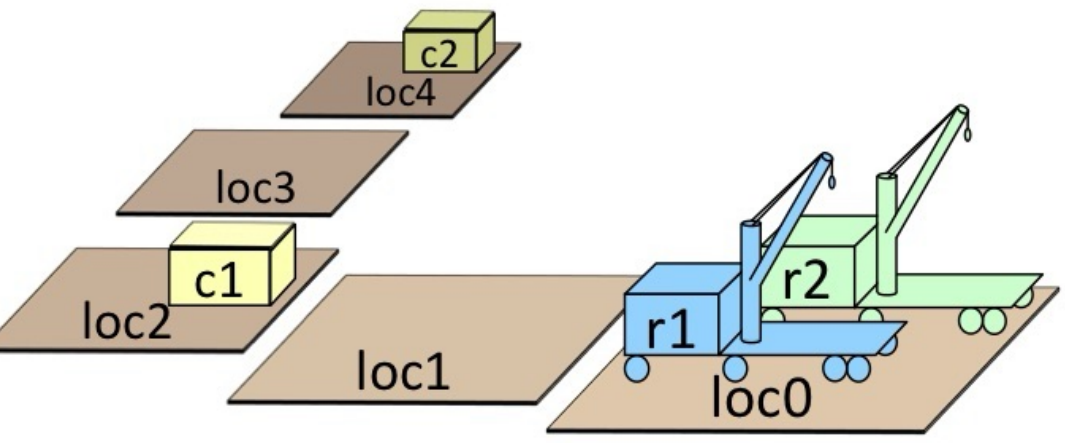
```
m-fetch1(r,c)
    task:   fetch(r,c)
    pre:    pos(c) = unknown
    body:
       if ∃l (view(l) = F) then
             move-to(r,l)
             perceive(r,l)
             if pos(c) = l then
                    take(r,c,l)
             else fetch(r,c)
       else fail
```

```
m-fetch2(r,c)
    task:   fetch(r,c)
    pre:    pos(c) ≠ unknown
    body:
          if loc(r) = pos(c) then
                take(r,c,pos(c))
          else do
                move-to(r,pos(c))
                take(r,c,pos(c))
```

Example



- Container locations unknown
- Partially observable
  - Robot only sees current location



procedure RAE:

loop:

1. for every new external task or event τ do
2. choose a method instance *m* for τ
3. create a refinement stack for τ, *m*
   add the stack to *Agenda*
4. for each stack σ in *Agenda*
   call Progress(σ)
   if σ is finished then remove it

2.9. Figure 2.17 shows a partial plan for the variable-swapping problem in Exercise 2.8.

(a) How many threats are there? What are they? What are their resolvers?

(b) Can PSP generate this plan? If so, describe an execution trace that will produce it. If no, explain why not.

(c) In PSP's search space, how many immediate successors does this partial plan have?

(d) How many solution plans can PSP produce from this partial plan?

(e) How many of the preceding solution plans are minimal?

(f) Trace the operation of PSP if we start it with the plan in Figure 2.17. Follow whichever of PSP's execution traces finds the shortest plan.

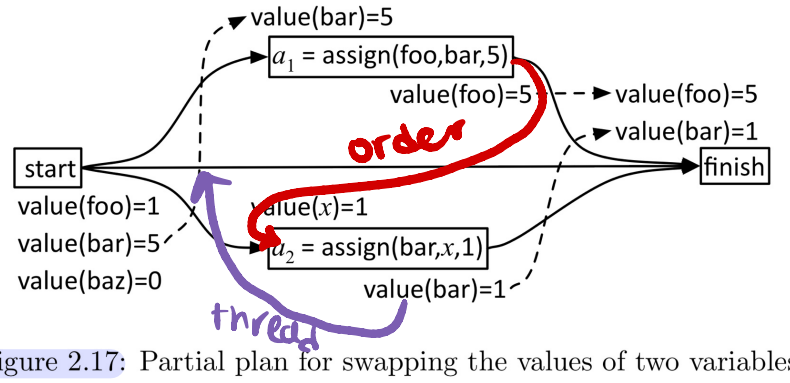


Figure 2.17: Partial plan for swapping the values of two variables.

2.8. Here is a state-variable version of the problem of swapping the values of two variables. The set of objects is $B = \mathit{Variables} \cup \mathit{Numbers}$, where $\mathit{Variables} = \{\mathsf{foo}, \mathsf{bar}, \mathsf{baz}\}$, and $\mathit{Numbers} = \{0, 1, 2, 3, 4, 5\}$. There is one action template:

$$\begin{aligned} &\mathsf{assign}(x_1, x_2, n)\\ &\text{pre: } \mathsf{value}(x_2) = n\\ &\text{eff: } \mathsf{value}(x_1) \leftarrow n \end{aligned}$$

where $\text{Range}(x_1) = \text{Range}(x_2) = \mathit{Variables}$, and $\text{Range}(n) = \mathit{Numbers}$. The initial state and goal are

$$\begin{aligned} s_0 &= \{\mathsf{value}(\mathsf{foo}) = 1, \mathsf{value}(\mathsf{bar}) = 5, \mathsf{value}(\mathsf{baz}) = 0\};\\ g &= \{\mathsf{value}(\mathsf{foo}) = 5, \mathsf{value}(\mathsf{bar}) = 1\}. \end{aligned}$$

At s_0, suppose GBFS is trying to choose between the actions $a_1 = \mathsf{assign}(\mathsf{baz},\mathsf{foo},1)$ and $a_2 = \mathsf{assign}(\mathsf{foo},\mathsf{bar},5)$. Let $s_1 = \gamma(s_0, a_1)$ and $s_2 = \gamma(s_0, a_2)$. Compute each pair of heuristic values below, and state whether or not they will produce the best choice.

(a) $h^{\text{add}}(s_1)$ and $h^{\text{add}}(s_2)$.

(b) $h^{\max}(s_1)$ and $h^{\max}(s_2)$.

(c) $h^{\text{FF}}(s_1)$ and $h^{\text{FF}}(s_2)$.

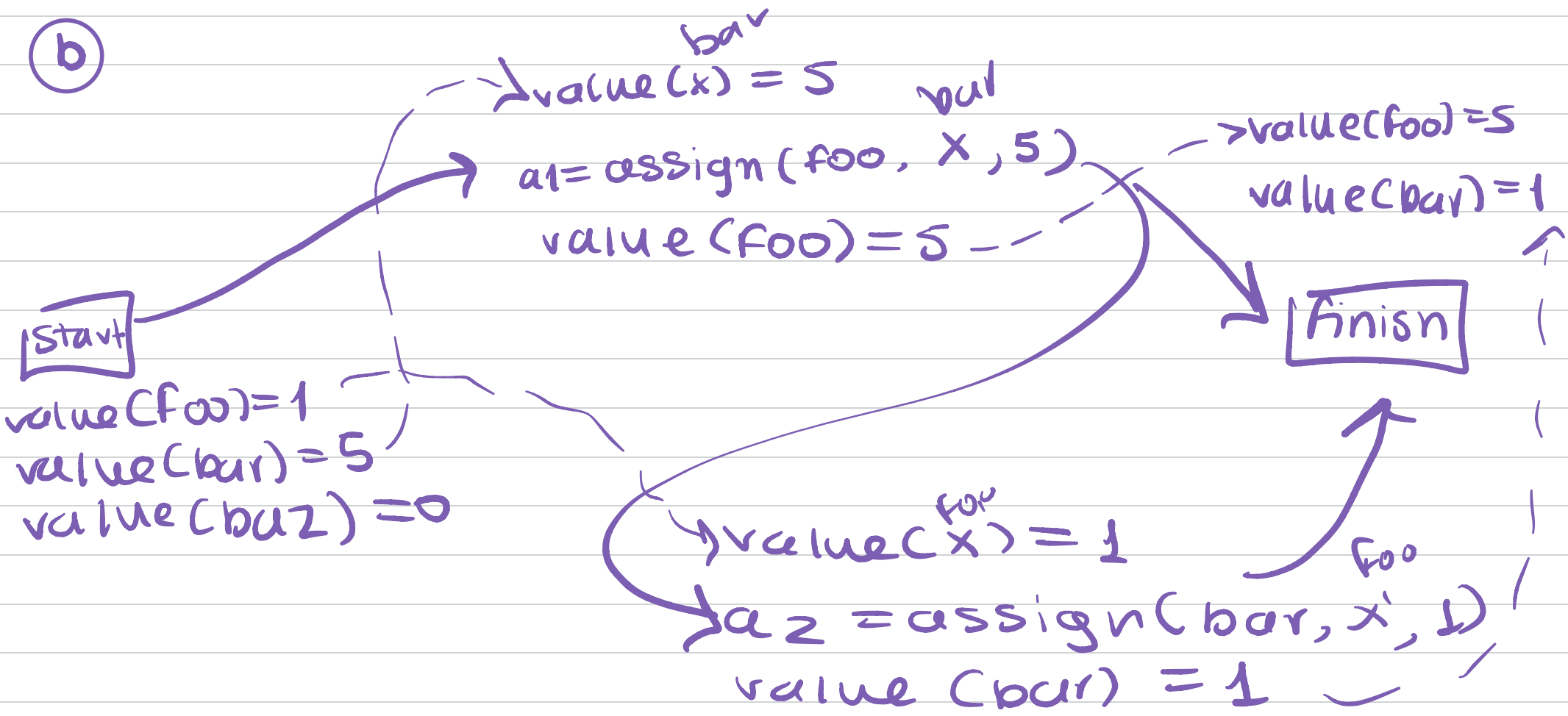
(a) Threat is when a2 sets value(bar)=1,
it can come between start & a1,
so then a1 cannot be executed
because it preconditions are
value(bar) =5

so a2 threatens a1's
preconditions value(bar)=5

1 thread.

1 possible resolver is to add a
precedence contraint that
a2 con only happen after a1
so a1 < a2

(b)



1. 2 open goals
2. 2 open goals

resolver
x = bar

x' = foo

Yes it can!

(c) successors:

- resolver of threat of value(boo)=1 ono

casual link ono start to a_2

→ order: $a_1 < a_2$

- open goal value(x) = 1 in a_2

resolves

→ casual support

can start from

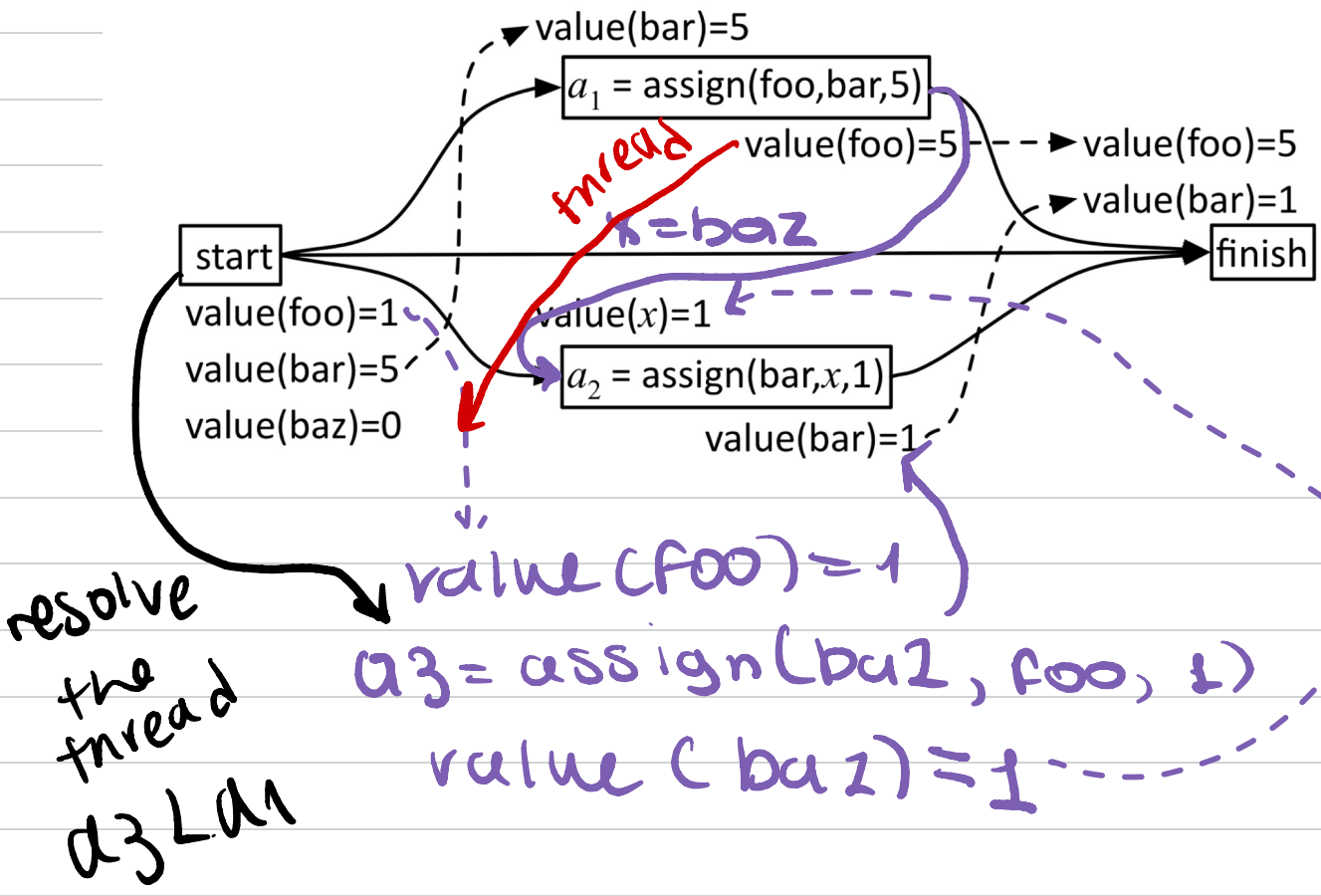
value(foo)=1 to value(x)=1

→ create action:

assign (x, y, 1)

3 successors

(d)



a3, a1, a2

$a_3 < a_1 < a_2$

(e) 1 minimal

2.12. Repeat Exercise 2.9 using the planning problem in Figure 2.18(b) and the partial plan shown in Figure 2.19.

pickup(x)
  pre: loc(x) = table, top(x) = nil, holding = nil
  eff: loc(x) ← hand, holding ← x

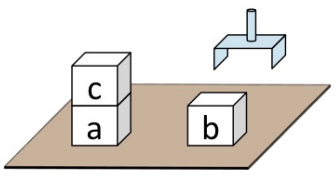
putdown(x)
  pre: holding = x
  eff: loc(x) ← table, holding ← nil

unstack(x, y)
  pre: loc(x) = y, top(x) = nil, holding = nil
  eff: loc(x) ← hand, top(y) ← nil, holding ← x

stack(x, y)
  pre: holding = x, top(y) = nil
  eff: loc(x) ← y, top(y) ← x, holding ← nil

Range(x) = Range(y) = *Blocks*

(a) action templates



$Objects = Blocks \cup \{\text{hand, table, nil}\}$
$Blocks = \{\text{a, b, c}\}$

$s_0 = \{\text{top(a)} = \text{c, top(b)} = \text{nil,}$
$\text{top(c)} = \text{nil, holding} = \text{nil,}$
$\text{loc(a)} = \text{table, loc(b)} = \text{table,}$
$\text{loc(c)} = \text{a}\}$

$g = \{\text{loc(a)} = \text{b, loc(b)} = \text{c}\}$

(b) objects, initial state, and goal

Figure 2.18: Blocks-world planning domain, and a planning problem.

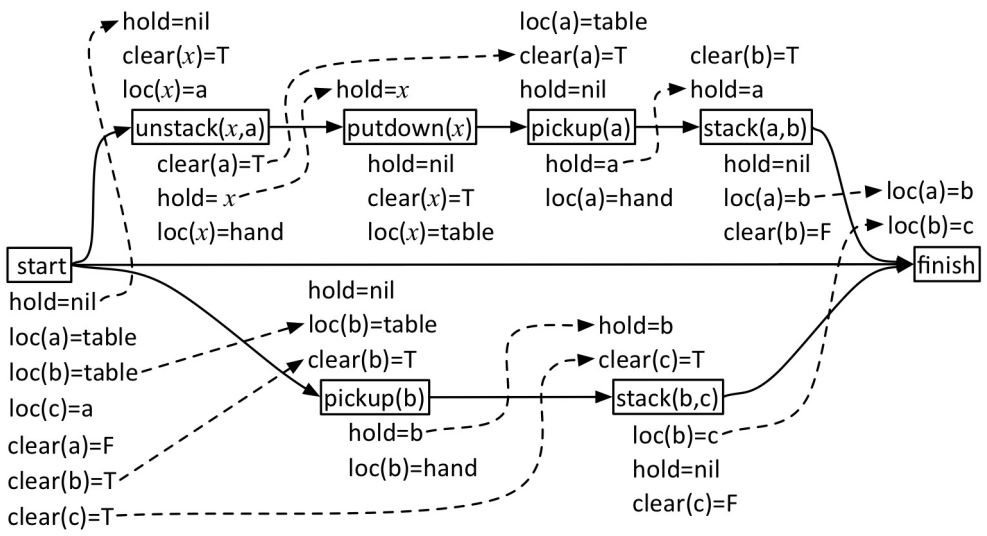


Figure 2.19: Partial plan for Exercise 2.12.

reviewing the examples of the slides on timelines (consistency, security, causal support), learning how temporal actions and methods are modeled through assertions and restrictions, as well as learning to apply the PC algorithm to triangular networks, where I recommend you give yourself cases with different ranges of values to practice (the algorithm is not very complicated either). I'm not going to upload a video of this because with the slides there is more than enough to understand how it goes. If anyone has questions, write me an email.

1.
2.
3.

3. PC algorithm to triangular networks