2.11. Repeat Exercise 2.8 on the planning problem in Figure 2.18(b), with $s_1 = \gamma(s_0, \text{unstack(c,a)})$ and $s_2 = \gamma(s_0, \text{pickup(b)})$.

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

(c)
HACERLO CON LANDMARK TAMBIEN

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

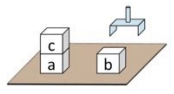
putdown(x)
  pre: holding = x
  eff: loc(x) ← table, holding ← nil

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* ∪ {hand, table, nil}
Blocks = {a, b, c}

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

g = {loc(a) = b, loc(b) = c}

(b) objects, initial state, and goal

s_1 = {top(a) = nil, top(b) = nil, top(c) = nil, holding = a, loc(a) = table, loc(b) = table, loc(c) = hand}

$h^{ADD}(s_1) = ?$ = 6

g = {loc(a) = b, loc(c) = b}

ADD 3+3=6

+

3 loc(a)=b

MIN

stack(a,b) 2+1=3

+ 2+0=2

Holding=a 0

MIN 2

unstack(a,b) ≥2

unstack(c,c) ≥2

top(b)= nil 0 0

3 loc(b)=c

MIN

stack(b,c) 2+1=3

+ 2+0=2

holding=b

MIN 2

unstack(b,a) ≥2

top(b)=nil 0

unstack(b,c) ≥2

1+1=2

pickup(b)

1+0+0=1 +

holding=nil 1

top(b)=nil 0

loc(b)= table 0

pickup(a) 1+0+0=1

+

holding=nil 0

MIN 1

top(a)=nil 0

loc(a) = table 0

putdown(a) ≥0

putdown(b) ≥0

putdown(c) 1+0=1

+

holding= c 0

combinations of unstack(x, y) ≥0

30/68 chapter

$g_1 = loc(a) = b$

landmarks $= \{loc(a) = b\}$

$cola = \{loc(b) = c\}$

$R = \{stack(a,b)\}$

$prec(stack(a,b)) = \{holding = a, top(b) = nil\}$

g=

$h^{SO}(S1) = ?$

landmarks: $\{loc(a) = b\}$

$cola = \{loc(b) = a\}$

$\hat{S}_0$

$loc(a) = table$

$loc(b) = table$

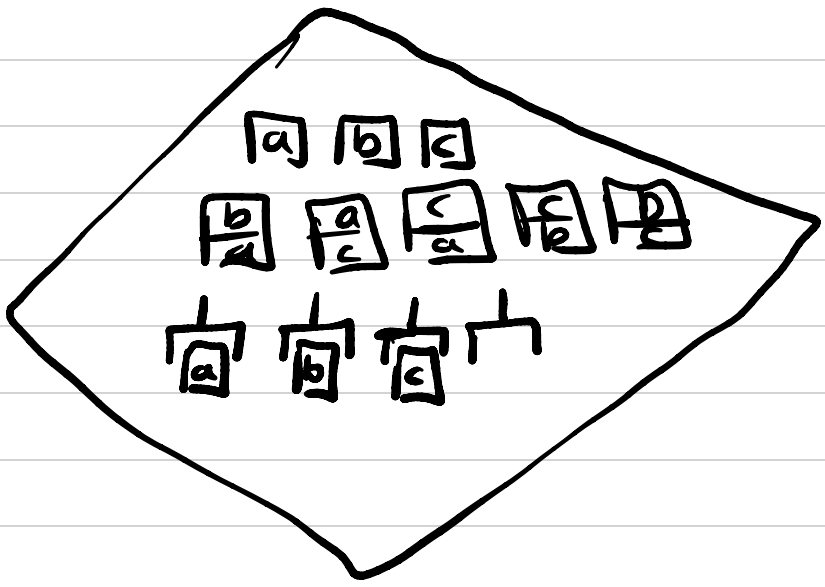
$loc(c) = hand$

$top(a) = nil$

$top(b) = nil$

$top(c) = nil$

holding = c



$\hat{A}_1$

stack(c,a)

stack(c,b)

putdown(c)

$\hat{S}_1$

top(a) = c

top(b) = c

loc(c) = a

loc(b) = b

loc(c) = table

holding = nil

$\hat{S}_0$

$\hat{A}_2$

pickup(a)

pickup(b)

pickup(c) *

unstack(c,a) *

unstack(a,b) *

$\hat{S}_2$

loc(a) = hand

loc(b) = hand

holding = a

holding = b

$\hat{S}_1$

$\hat{A}_3$

0

0 *

preconditions of actions h

$N = \{stack(a,b)\}$

prec (stack(a,b)) = {top(b) = nil, holding = a}

nuevos_landmarks = {(holding = a)}

cda = {loc(b) = c, holding = a}

2.15. Let P be a planning problem in which the action templates and initial state are as shown in Figure 2.16, and the goal is $g = \{\mathsf{loc(c1)} = \mathsf{loc2}\}$. In the Run-Lazy-Lookahead algorithm, suppose the call to Lookahead(P) returns the following solution plan:

$$\pi = \{\mathsf{take(r1,loc1,c1)}, \mathsf{move(r1,loc1,loc2)}, \mathsf{put(r1,loc2,c1)}\}.$$

(a) Suppose that after the actor has performed take(r1,loc1,c1) and move(r1,loc1,loc2), monitoring reveals that c1 fell off of the robot and is still back at loc1. Tell what will happen, step by step. Assume that Lookahead(P) will always return the best solution for P.

(b) Repeat part (a) using Run-Lookahead.

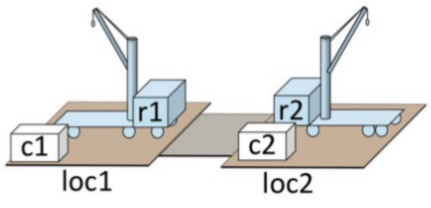
(c) Suppose that after the actor has performed take(r1,loc1,c1), monitoring reveals that r1's wheels have stopped working, hence r1 cannot move from loc1. What should the actor do to recover? How would you modify Run-Lazy-Lookahead, Run-Lookahead, and Run-Concurrent-Lookahead to accomplish this?

take(r, l, c)
  pre: loc(r) = l, pos(c) = l,
    cargo(r) = nil
  eff: cargo(r) = c, pos(c) ← r

put(r, l, c)
  pre: loc(r) = l, pos(c) = r
  eff: cargo(r) ← nil, pos(c) ← l

move(r, l, m)
  pre: loc(r) = l
  eff: loc(r) ← m

(a) action templates



$s_0 = \{\mathsf{loc(r1)} = \mathsf{loc1}, \mathsf{loc(r2)} = \mathsf{loc2},$
$\mathsf{cargo(r1)} = \mathsf{nil}, \mathsf{cargo(r2)} = \mathsf{nil},$
$\mathsf{pos(c1)} = \mathsf{loc1}, \mathsf{pos(c2)} = \mathsf{loc2}\}$

$g = \{\mathsf{pos(c1)} = \mathsf{loc2}, \mathsf{pos(c2)} = \mathsf{loc2}\}$

(b) initial state and goal

$h^{FF}(S2)$

$s_2 = \gamma(s_0, a_2) = \{\text{loc(c1) = d1, loc(r1) = d2, cargo(r1) = nil}\}$

1step: construct a relaxed solution

Atoms in $\hat{S_0} = S_2$ | Actions in A_1 | Atoms in $\hat{S_1}$ | Actions in A_2: | Atoms in $\hat{S_2}$:

loc(c1)=d1
loc(r1)=d2
cargo(r1)=nil

move(r1,d2,d1)
move(r1,d2,d3)

loc(r1)=d1
loc(r1)=d3
loc(c1)=d1
loc(r1)=d2
cargo(r1)=nil

ano $\hat{S_0}$

move(r1,d1,d2)
move(r1,d1,d3)
move(r1,d3,d1)
move(r1,d3,d2)
move(r1,d2,d1)
move(r1,d2,d3)
load(r1,c1,d1)

ano $\hat{S_1}$

loc(r1)=d2
loc(r1)=d3
loc(r1)=d1
loc(c1)=d1
cargo(r1)=nil
cargo(r1)=c1
loc(c1)=r1

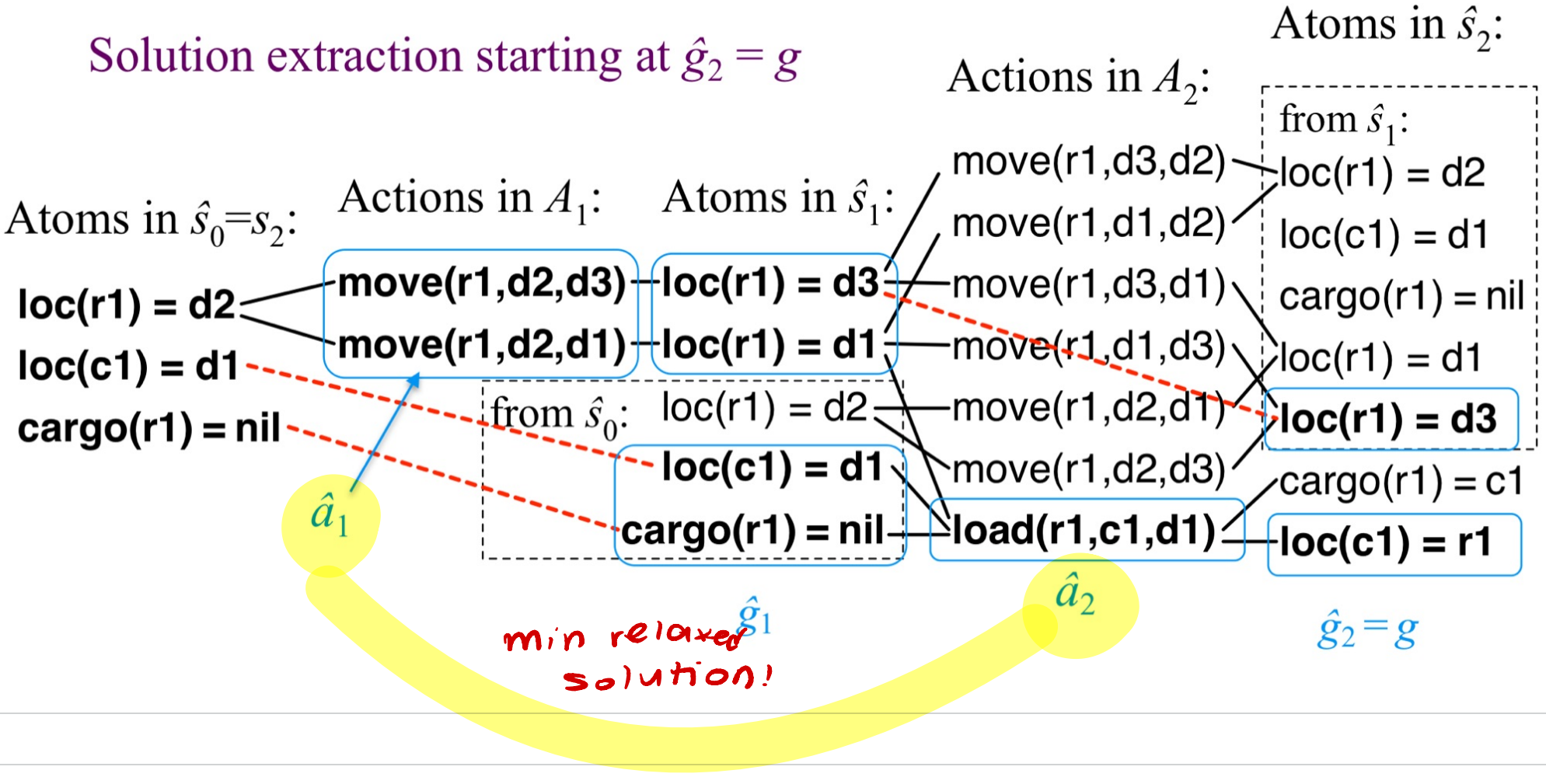
$\hat{S_2}$ r-satisfies g
so <A1,A2> is a relaxed solution

$g = \{loc(r1)=d3, loc(c1)=r1\}$

2 step: extract minimal reluxed solution

*highlighted: παω πιθω να βρω ει action προκαλεσε το atom

etc...



$h^{FF}(S_1)$

$\hat{S}_0 = S_1$ | A_1 | $\hat{S}_1$

loc(c1) = d1
loc(r1) = d1
cargo(r1) = nil

move(r1, d1, d2)
move(r1, d1, d3)
load(r1, c1, d1)

loc(c1) = d1
loc(r1) = d1
cargo(r1) = nil
loc(r1) = d2
loc(r1) = d3
loc(c1) = r1
cargo(r1) = c1

$\hat{a}_1$ is a minimal set of actions

$\langle \hat{a}_1 \rangle$ minimal relaxed solution

2 actions → 1 + 1 = 2 → $h^{FF}(S_1) = 2$

Landmark heuristic

queue = { loc(c1) = r1 }

landmarks = ∅

R = { load(r1, c1, d1), load(r1, c1, d2), load(r1, c1, d3) }

$\hat{S}_0$ | A_1 | $\hat{S}_1 = \hat{S}_2$

loc(c1) = d1
loc(r1) = d3
cargo(r1) = nil

move(r1, d3, d2)
move(r1, d3, d1)

loc(c1) = d1
loc(r1) = d3
cargo(r1) = nil
loc(r1) = d2
loc(r1) = d1

N = { τα actions από το R τα οποία είναι r-applicable is $\hat{S}_k$ }

N = { load(r1, c1, d1) }

Preconds = { preconds αυτού ΠΛΗΝ αυτά που είναι ήδη στο S_0 }

= { loc(r1) = d1 }

Φ = { loc(r1) = d1 }

queue = { loc(r1) = d1 }

y_i = { loc(c1) = r1 }

Landmarks = { loc(c1) = r1, loc(r1) = d1 }

R = { move(r1, d2, d1), move(r1, d2, d3) }

2.7. Figure 2.16 shows a planning problem involving two robots whose actions are controlled by a single actor.

(c) Compute the values of $h^{\text{add}}(s_0)$ and $h^{\max}(s_0)$.

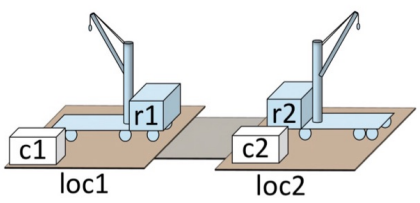
(e) Compute the value of $h^{\text{FF}}(s_0)$.

take(r, l, c)
  pre: loc$(r) = l$, pos$(c) = l$, cargo$(r) =$ nil
  eff: cargo$(r) = c$, pos$(c) \leftarrow r$

put(r, l, c)
  pre: loc$(r) = l$, pos$(c) = r$
  eff: cargo$(r) \leftarrow$ nil, pos$(c) \leftarrow l$

move(r, l, m)
  pre: loc$(r) = l$
  eff: loc$(r) \leftarrow m$

(a) action templates



$s_0 = \{\text{loc(r1)} = \text{loc1}, \text{loc(r2)} = \text{loc2}, \text{cargo(r1)} = \text{nil}, \text{cargo(r2)} = \text{nil}, \text{pos(c1)} = \text{loc1}, \text{pos(c2)} = \text{loc2}\}$

$g = \{\text{pos(c1)} = \text{loc2}, \text{pos(c2)} = \text{loc2}\}$

(b) initial state and goal

$h^{add}(s_0)$

ADD

pos(c1) = loc2

put(r1, loc2, c1) — pre: loc(r1) = loc2 (MIN) → move(r1, loc1, loc2) 1+0=1 (max) → loc(r1) = loc1, true in s_0 ✓

put(r2, loc2, c1) — pre: loc(r2) = loc2 (true in s_0), pos(c1) = r2

pos(c2) = loc2

put(r2, loc2, c2) — pre: true in s_0, loc(r2) = loc2, pos(c2) = r2 (>0)

put(r1, loc2, c2) — pre: >0, loc(r1) = loc2, pos(c2) = r2 (>0)

pos(c1) = r1 (MIN)

1+0=1

take(r1, loc1, c1) — cargo(r1) = nil (true in s_0); loc(r1) = loc1 (true in s_0); pos(c1, loc1) (true in s_0)

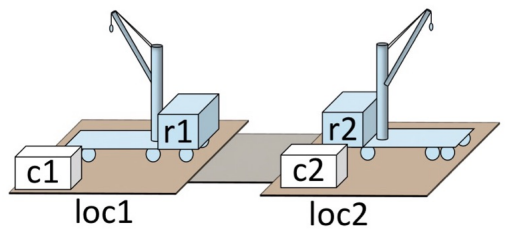
take(r1, loc2, c1) ~~crossed out~~

take(r, l, c)
  pre: loc$(r) = l$, pos$(c) = l$,
       cargo(r) = nil
  eff: cargo$(r) = c$, pos$(c) \leftarrow r$

put(r, l, c)
  pre: loc$(r) = l$, pos$(c) = r$
  eff: cargo$(r) \leftarrow$ nil, pos$(c) \leftarrow l$

move(r, l, m)
  pre: loc$(r) = l$
  eff: loc$(r) \leftarrow m$

(a) action templates



$s_0 = \{$loc(r1) = loc1, loc(r2) = loc2,
cargo(r1) = nil, cargo(r2) = nil,
pos(c1) = loc1, pos(c2) = loc2$\}$

$g = \{$pos(c1) = loc2, pos(c2) = loc2$\}$

(b) initial state and goal

$h^{FF}(S_0)$?

$\hat{S_0} = S_0$ A1 $\hat{S_1}$ $\hat{A_2}$ $\hat{S_2}$

loc(r1)=l1
loc(r2)=l2
cargo(r1)=nil
cargo(r2)=nil
pos(c1)=l1
pos(c2)=l2 ✓

move(r1, l1, l2)
move(r1, l2, l1)
take(r1, l1, c1)
take(r2, l2, c2)

loc(r1)= l2
loc(r2)=l1
cargo(r1)=c1
cargo(r2)=c2
pos(c1)=r1
pos(c2)=r2
cargo(r1)=nil
cargo(r2)=nil
pos(c1)=l1
pos(c2)=l2

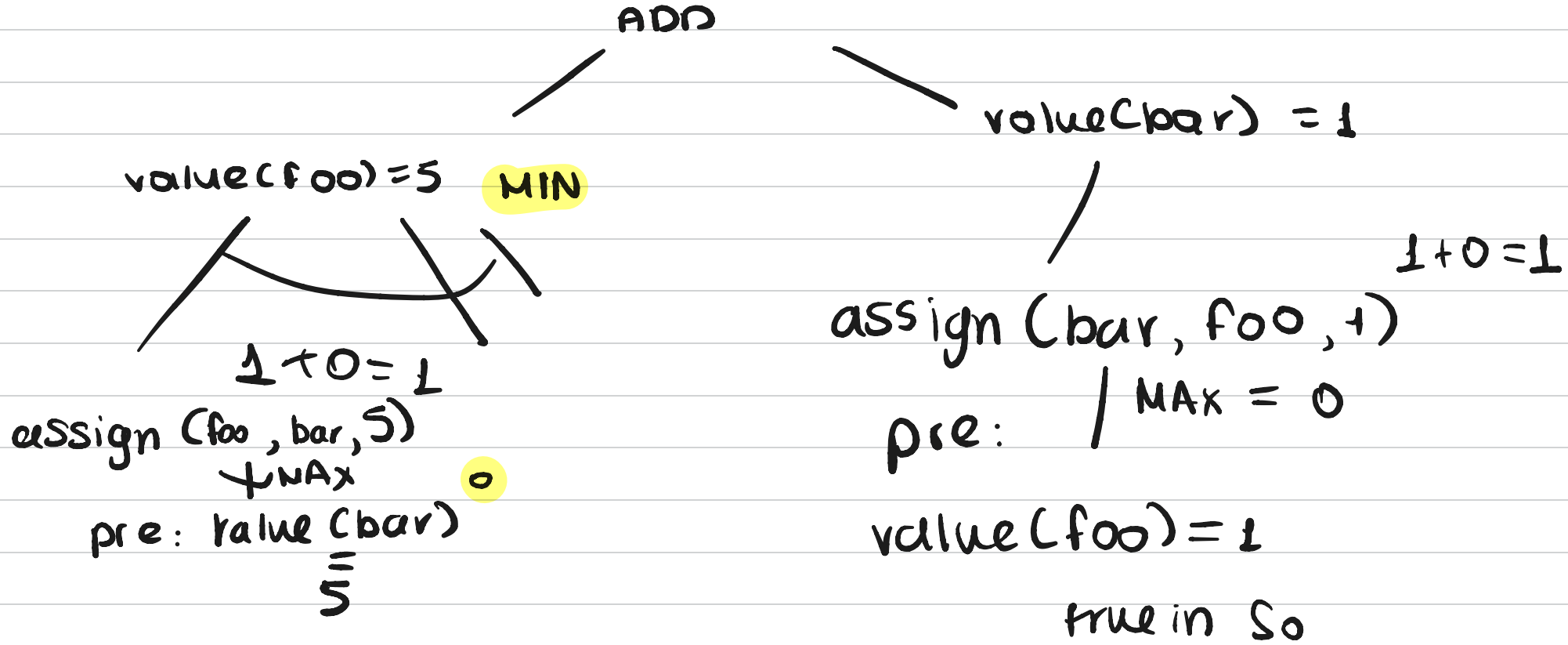
$\hat{S_0}$

put(r1, l2, c1)
put(r2, l1, c2)
move(r1, l1, l2)
move(r1, l2, l1)
take(r1, l1, c1)
take(r2, l2, c2)

loc(r1)=l2
loc(r2)=l1
cargo(r1)=c1
cargo(r2)=c2
pos(c1)=r1
pos(c2)=r2
cargo(r1)=nil
cargo(r2)=nil
pos(c1)=l1
pos(c2)=l2
pos(c1)=l2 ✓
pos(c2)=l1

Actions = 3 $\Rightarrow h^{FF}(S_0) = 3$

n^add (s1)


ADD
value(foo)=5
MIN
1+0=1
assign(foo, bar, 5)
↓MAX
0
pre: value(bar) = 5
value(bar) = 1
1+0=1
assign(bar, foo, 1)
pre: | MAX = 0
value(foo) = 1
true in So

$s_1 = f(s_0, unstack(c,a))$

$= \{ loc(c) = hand,$ ✓

$top(a) = nil,$ ✓

$holding = c,$ ✓

$top(b) = nil$ ✓

$loc(a) = table$ ✓

$loc(b) = table$ ✓

ADD

$loc(a) = b$ $\qquad$ $loc(b) = c$

1

$stack(a,b)$ (>0)

$\max(>0, 0)$

pre:

$holding = a$ $\qquad$ $top(b) = nil$

true in s_0 $\qquad$ 0

$g_i = \{pos(c1) = loc2\}$

queue = $\{pos(c1) = loc2\}$

Landmark = ϕ

Landmark = $\{pos(c1) = loc2\}$

R = { put(r1, loc2, c1)
put(r2, loc2, c1) }

$\hat{S_0}$ A1 $\hat{S_1}$ A2 $\hat{S_2}$

$s_1 = \{top(a)=nil, top(b)=nil, top(c)=nil,$
holding=a,
loc(a)=table, loc(b)=table, loc(c)=hand$\}$

$n^{add}(s_1)$? $g = \{loc(a)=b, loc(c)=b\}$