Week 7: Relaxed Plan Heuristics and Iterated Width (IW)

COMP90054 – AI Planning for Autonomy

Key concepts

- Relaxed Plan Heuristics (h^{ff})
- Iterated Width (IW)

Relaxed Plan Heuristics ($\boldsymbol{h^{ff}}$)

- h* is the perfect heuristic
- h^+ is the **optimal delete relaxation** heuristic (not easy to compute)
- h^{max} is an approximation of h^+
- h^{add} is an approximation of h^+
- h^{ff} is an approximation of h^+

	Pros	Cons
h^{max}	Admissible	Very small (optimistic)
h^{add}	More informed than h^{max}	Not admissible (pessimistic) over-counting

h^{ff} can reduce over-counting (but it is still inadmissible)

Find h^{ff} based on h^{max} and h^{add}

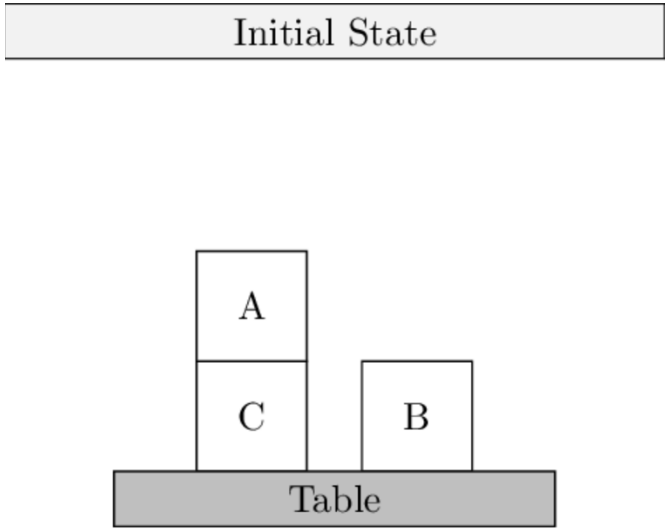
Problem 1: Computing h^{ff}

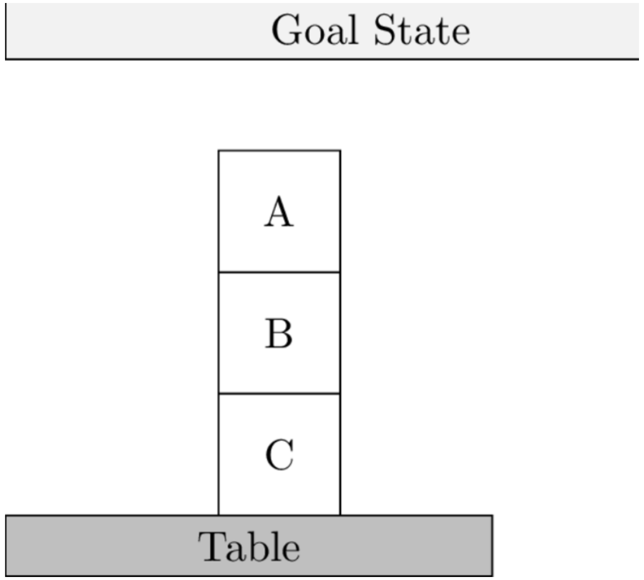
Initial state

I = {on(A, C), onTable(C), onTable(B), clear(A), clear(B), handFree}

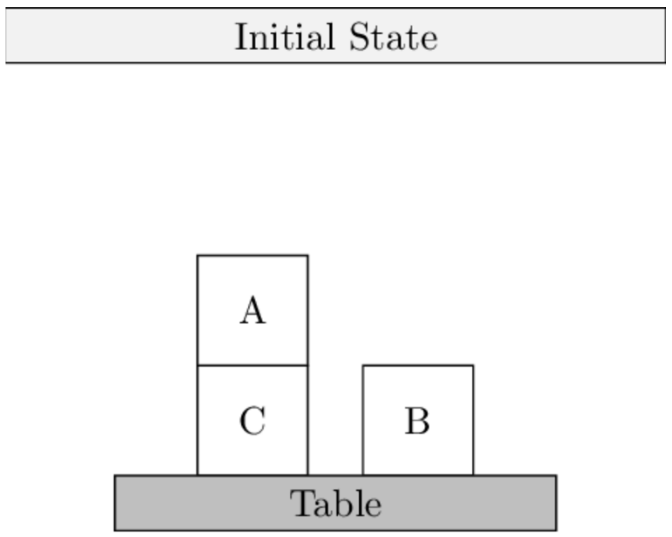
Goal state

G = {on(A,B), on(B,C), onTable(C)}





Problem 1: Computing h^{ff}



1. Find best-supporter function (bs)

2. Relaxed Plan Extraction for state s

Problem 1: Find the best-supporter function for each fact

h^{add} / h^{max}

Iter	c(A)	c(B)	c(C)	hand Free	h(A)	h(B)	h(C)	on(A, A)	on(A,B)	on(A,C)	on(B,A)	on(B,B)	on(B,C)	on(C,A)	on(C,B)	on(C,C)	onT(A)	onT(B)	onT(C)
0	0	0	∞	0	∞	∞	∞	∞	∞	0	∞	∞	∞	∞	∞	∞	∞	0	0
1	0	0	1	0	1	1	∞	∞	∞	0	∞	∞	∞	∞	∞	∞	∞	0	0
2	0	0	1	0	1	1	2	2	2	0	2	2	3 / 2	∞	∞	∞	2	0	0
3	0	0	1	0	1	1	2	2	2	0	2	2	3 / 2	3	3	4 / 3	2	0	0
4	0	0	1	0	1	1	2	2	2	0	2	2	3 / 2	3	3	4 / 3	2	0	0

1. Which actions can we take to make **clear(C)** True?

putdown(C), stack(C, A), stack(C, B), unstack(A, C), unstack(B, C), stack(C, C), unstack(C, C)

/ 7 actions

2. Which action is the best-supporter function of **clear(C)**?

Define Operators

O = {

pickup(x)

- Prec: onTable(x), clear(x), handFree
- Add: holding(x)
- Del: onTable(x), clear(x), handFree

unstack(x, y)

- Prec: on(x, y), clear(x), handFree
- Add: holding(x), clear(y)
- Del: on(x, y), clear(x), handFree

putdown(x)

- Prec: holding(x)
- Add: clear(x), onTable(x), handFree
- Del: holding(x)

stack(x, y)

- Prec: holding(x), clear(y)
- Add: clear(x), on(x,y), handFree
- Del: clear(y), holding(x)

}

Problem 1: Find the best-supporter function for each fact

h^{add} / h^{max}

Iter	c(A)	c(B)	c(C)	hand Free	h(A)	h(B)	h(C)	on(A, A)	on(A,B)	on(A,C)	on(B,A)	on(B,B)	on(B,C)	on(C,A)	on(C,B)	on(C,C)	onT(A)	onT(B)	onT(C)
0	0	0	∞	0	∞	∞	∞	∞	∞	0	∞	∞	∞	∞	∞	∞	∞	0	0
1	0	0	1	0	1	1	∞	∞	∞	0	∞	∞	∞	∞	∞	∞	∞	0	0
2	0	0	1	0	1	1	2	2	2	0	2	2	3 / 2	∞	∞	∞	2	0	0
3	0	0	1	0	1	1	2	2	2	0	2	2	3 / 2	3	3	4 / 3	2	0	0
4	0	0	1	0	1	1	2	2	2	0	2	2	3 / 2	3	3	4 / 3	2	0	0

unstack(x, y)
- Prec: on(x, y), clear(x), handFree
- Add: holding(x), clear(y)
- Del: on(x, y), clear(x), handFree

putdown(x)
- Prec: holding(x)
- Add: clear(x), onTable(x), handFree
- Del: holding(x)

stack(x, y)
- Prec: holding(x), clear(y)
- Add: clear(x), on(x,y), handFree
- Del: clear(y), holding(x)

2. Which action is the best-supporter function of **clear(C)**?

putdown(C), stack(C, A), stack(C, B), unstack(A, C), unstack(B, C), stack(C, C), unstack(C, C)

Use h^{add} := action cost + sum (preconditions)

Use h^{add}

putdown(C) = 1 + hold(C) = 1 + 2 = 3
stack(C, A) = 1 + hold(C) + clear(A) = 1 + 2 + 0 = 3
stack(C, B) = 1 + hold(C) + clear(B) = 1 + 2 + 0 = 3
stack(C, C) = 1 + hold(C) + clear(C) = 1 + 2 + 1 = 4
unstack(A, C) = 1 + on(A, C) + clear(A) + handFree = 1 + 0 + 0 + 0 = 1
unstack(B, C) = 1 + on(B, C) + clear(B) + handFree = 1 + 3 + 0 + 0 = 4
unstack(C, C) = 1 + on(C, C) + clear(C) + handFree = 1 + 4 + 1 + 0 = 6

best-supporter function of clear(C) when using h^{add} ← unstack(A,C) min ←

Problem 1: Find the best-supporter function for each fact

h^{add} / h^{max}

Iter	c(A)	c(B)	c(C)	hand Free	h(A)	h(B)	h(C)	on(A, A)	on(A,B)	on(A,C)	on(B,A)	on(B,B)	on(B,C)	on(C,A)	on(C,B)	on(C,C)	onT(A)	onT(B)	onT(C)
0	0	0	∞	0	∞	∞	∞	∞	∞	0	∞	∞	∞	∞	∞	∞	∞	0	0
1	0	0	1	0	1	1	∞	∞	∞	0	∞	∞	∞	∞	∞	∞	∞	0	0
2	0	0	1	0	1	1	2	2	2	0	2	2	3 / 2	∞	∞	∞	2	0	0
3	0	0	1	0	1	1	2	2	2	0	2	2	3 / 2	3	3	4 / 3	2	0	0
4	0	0	1	0	1	1	2	2	2	0	2	2	3 / 2	3	3	4 / 3	2	0	0

unstack(x, y)
- Prec: on(x, y), clear(x), handFree
- Add: holding(x), clear(y)
- Del: on(x, y), clear(x), handFree

putdown(x)
- Prec: holding(x)
- Add: clear(x), onTable(x), handFree
- Del: holding(x)

stack(x, y)
- Prec: holding(x), clear(y)
- Add: clear(x), on(x,y), handFree
- Del: clear(y), holding(x)

2. Which action is the best-supporter function of **clear(C)**?

putdown(C), stack(C, A), stack(C, B), unstack(A, C), unstack(B, C), stack(C, C), unstack(C, C)

Use h^{max} := action cost + max (preconditions)

best-supporter function of clear(C) when using h^{max} ← unstack(A, C) min ←

Use h^{max}

putdown(C) = 1 + hold(C) = 1 + 2 = 3
stack(C, A) = 1 + max(hold(C), clear(A)) = 1 + max(2, 0) = 3
stack(C, B) = 1 + max(hold(C), clear(B)) = 1 + max(2, 0) = 3
stack(C, C) = 1 + max(hold(C), clear(C)) = 1 + max(2, 1) = 3
unstack(A, C) = 1 + max(on(A, C), clear(A), handFree) = 1 + max(0, 0, 0) = 1
unstack(B, C) = 1 + max(on(B, C), clear(B), handFree) = 1 + max(2, 0, 0) = 3
unstack(C, C) = 1 + max(on(C, C), clear(C), handFree) = 1 + max(3, 1, 0) = 4

Problem 1: Find the best-supporter function for each fact

h^{add} / h^{max}

Iter	c(A)	c(B)	c(C)	hand Free	h(A)	h(B)	h(C)	on(A, A)	on(A,B)	on(A,C)	on(B,A)	on(B,B)	on(B,C)	on(C,A)	on(C,B)	on(C,C)	onT(A)	onT(B)	onT(C)
0	0	0	∞	0	∞	∞	∞	∞	∞	0	∞	∞	∞	∞	∞	∞	∞	0	0
1	0	0	1	0	1	1	∞	∞	∞	0	∞	∞	∞	∞	∞	∞	∞	0	0
2	0	0	1	0	1	1	2	2	2	0	2	2	3 / 2	∞	∞	∞	2	0	0
3	0	0	1	0	1	1	2	2	2	0	2	2	3 / 2	3	3	4 / 3	2	0	0
4	0	0	1	0	1	1	2	2	2	0	2	2	3 / 2	3	3	4 / 3	2	0	0

pickup(x)
- Prec: onTable(x), clear(x), handFree
- Add: holding(x)
- Del: onTable(x), clear(x), handFree

unstack(x, y)
- Prec: on(x, y), clear(x), handFree
- Add: holding(x), clear(y)
- Del: on(x, y), clear(x), handFree

putdown(x)
- Prec: holding(x)
- Add: clear(x), onTable(x), handFree
- Del: holding(x)

stack(x, y)
- Prec: holding(x), clear(y)
- Add: clear(x), on(x,y), handFree
- Del: clear(y), holding(x)

2. Which action is the best-supporter function of **on(B, C)**?

stack(B, C)

stack (B,C)

Problem 1: Find the best-supporter function for each fact

h^{add} / h^{max}

Iter	c(A)	c(B)	c(C)	hand Free	h(A)	h(B)	h(C)	on(A, A)	on(A,B)	on(A,C)	on(B,A)	on(B,B)	on(B,C)	on(C,A)	on(C,B)	on(C,C)	onT(A)	onT(B)	onT(C)
0	0	0	∞	0	∞	∞	∞	∞	∞	0	∞	∞	∞	∞	∞	∞	∞	0	0
1	0	0	1	0	1	1	∞	∞	∞	0	∞	∞	∞	∞	∞	∞	∞	0	0
2	0	0	1	0	1	1	2	2	2	0	2	2	3 / 2	∞	∞	∞	2	0	0
3	0	0	1	0	1	1	2	2	2	0	2	2	3 / 2	3	3	4 / 3	2	0	0
4	0	0	1	0	1	1	2	2	2	0	2	2	3 / 2	3	3	4 / 3	2	0	0

Use h^{add} / h^{max} for the best-supporter function

	clear(a)	clear(b)	clear(c)	handempty()	holding(a)	holding(b)	holding(c)	on(a,a)	on(a,b)	on(a,c)	on(b,a)	on(b,b)	on(b,c)	on(c,a)	on(c,b)	on(c,c)	ontable(a)	ontable(b)	ontable(c)
0	NA	NA	(unstack a c)	NA	(unstack a c)	(pick-up b)	(pick-up c)	(stack a a)	(stack a b)	NA	(stack b a)	(stack b b)	(stack b c)	(stack c a)	(stack c b)	(stack c c)	(put-down a)	NA	NA

→ $h^{add} = h^{max}$ (only for this example)

Problem 1: Relaxed Plan Extraction

I = {on(A, C), onTable(C), onTable(B), clear(A), clear(B), handFree}

G = {on(A,B), on(B,C), onTable(C)}

	clear(a)	clear(b)	clear(c)	handempty()	holding(a)	holding(b)	holding(c)	on(a,a)	on(a,b)	on(a,c)	on(b,a)	on(b,b)	on(b,c)	on(c,a)	on(c,b)	on(c,c)	ontable(a)	ontable(b)	ontable(c)
0	NA	NA	(unstack a c)	NA	(unstack a c)	(pick-up b)	(pick-up c)	(stack a a)	(stack a b)	NA	(stack b a)	(stack b b)	(stack b c)	(stack c a)	(stack c b)	(stack c c)	(put-down a)	NA	NA

Relaxed Plan Extraction for state s and best-supporter function bs

1 $Open := G \setminus s$; $Closed := \emptyset$; $RPlan := \emptyset$
2 **while** $Open \neq \emptyset$ **do**:
3 select $g \in Open$
4 $Open := Open \setminus \{g\}$; $Closed := Closed \cup \{g\}$;
5 $RPlan := RPlan \cup \{bs(g)\}$; $Open := Open \cup (pre_{bs(g)} \setminus (s \cup Closed))$
6 **endwhile**
7 **return** $RPlan$

s = I

line 1: Open = G \ I = { on(A,B), on(B,C) }

Open = {on(A,B), on(B,C)}
Closed = {}
RPlan = {}

Problem 1: Relaxed Plan Extraction

I = {on(A, C), onTable(C), onTable(B), clear(A), clear(B), handFree}

X

G = {on(A,B), on(B,C), onTable(C)}

pickup(x)
- Prec: onTable(x), clear(x), handFree
- Add: holding(x)
- Del: onTable(x), clear(x), handFree

unstack(x, y)
- Prec: on(x, y), clear(x), handFree
- Add: holding(x), clear(y)
- Del: on(x, y), clear(x), handFree

putdown(x)
- Prec: holding(x)
- Add: clear(x), onTable(x), handFree
- Del: holding(x)

stack(x, y)
- Prec: holding(x), clear(y)
- Add: clear(x), on(x,y), handFree
- Del: clear(y), holding(x)

	clear(a)	clear(b)	clear(c)	handempty()	holding(a)	holding(b)	holding(c)	on(a,a)	on(a,b)	on(a,c)	on(b,a)	on(b,b)	on(b,c)	on(c,a)	on(c,b)	on(c,c)	ontable(a)	ontable(b)	ontable(c)
0	NA	NA	(unstack a c)	NA	(unstack a c)	(pick-up b)	(pick-up c)	(stack a a)	(stack a b)	NA	(stack b a)	(stack b b)	(stack b c)	(stack c a)	(stack c b)	(stack c c)	(put-down a)	NA	NA

Relaxed Plan Extraction for state s and best-supporter function bs

1 $Open := G \setminus s$; $Closed := \emptyset$; $RPlan := \emptyset$
2 **while** $\boxed{Open \neq \emptyset}$ **do**
3 select $g \in Open$
4 $Open := Open \setminus \{g\}$; $Closed := Closed \cup \{g\}$;
5 $RPlan := RPlan \cup \{bs(g)\}$; $Open := Open \cup (pre_{bs(g)} \setminus (s \cup Closed))$
6 **endwhile**
7 **return** $RPlan$

Open = {~~on(A,B)~~, on(B,C)} ∪ {holding(A)}
Closed = {} ∪ {on(A,B)}
RPlan = {} ∪ {stack(A,B)}

Iteration 1:
- Select *g* from Open (line 3) : g = on(A,B)
- Put *g* into Closed (line 4)
- Get *bs(g)* and add *bs(g)* into RPlan (line 5) bs(on(A,B)) = stack(A,B)
- Get *preconditions* of bs(g) and update Open list if necessary (line 5)

$pre_{stack(A,B)}$ = {holding(A), clear(B)}

$pre_{stack(A,B)} \setminus (I \cup Closed)$ = {holding(A)}

Problem 1: Relaxed Plan Extraction

I = {on(A, C), onTable(C), onTable(B), clear(A), clear(B), handFree}

G = {on(A,B), on(B,C), onTable(C)}

pickup(x)
- Prec: onTable(x), clear(x), handFree
- Add: holding(x)
- Del: onTable(x), clear(x), handFree

unstack(x, y)
- Prec: on(x, y), clear(x), handFree
- Add: holding(x), clear(y)
- Del: on(x, y), clear(x), handFree

putdown(x)
- Prec: holding(x)
- Add: clear(x), onTable(x), handFree
- Del: holding(x)

stack(x, y)
- Prec: holding(x), clear(y)
- Add: clear(x), on(x,y), handFree
- Del: clear(y), holding(x)

	clear(a)	clear(b)	clear(c)	handempty()	holding(a)	holding(b)	holding(c)	on(a,a)	on(a,b)	on(a,c)	on(b,a)	on(b,b)	on(b,c)	on(c,a)	on(c,b)	on(c,c)	ontable(a)	ontable(b)	ontable(c)
0	NA	NA	(unstack a c)	NA	(unstack a c)	(pick-up b)	(pick-up c)	(stack a a)	(stack a b)	NA	(stack b a)	(stack b b)	(stack b c)	(stack c a)	(stack c b)	(stack c c)	(put-down a)	NA	NA

Relaxed Plan Extraction for state s and best-supporter function bs

$Open := G \setminus s$; $Closed := \emptyset$; $RPlan := \emptyset$
while $Open \neq \emptyset$ **do**:
  select $g \in Open$
  $Open := Open \setminus \{g\}$; $Closed := Closed \cup \{g\}$;
  $RPlan := RPlan \cup \{bs(g)\}$; $Open := Open \cup (pre_{bs(g)} \setminus (s \cup Closed))$
endwhile
return $RPlan$

Open = {~~on(B,C)~~, holding(A)} ∪ {holding(B), clear(C)}
Closed = {on(A, B)} ∪ {on(B,C)}
RPlan = {stack(A, B)} ∪ {stack(B, C)}

Iteration 2:
- Select *g* from Open g = on(B,C)
- Put *g* into Closed
- Get *bs(g)* and add *bs(g)* into RPlan bs(on(B,C)) = stack(B,C)
- Get *preconditions* of bs(g) and update Open list if necessary

$pre_{stack(B,C)} = \{holding(B), clear(C)\}$

$pre_{stack(B,C)} \setminus (I \cup Closed) = \{holding(B), clear(C)\}$

Problem 1: Relaxed Plan Extraction

I = {on(A, C), onTable(C), onTable(B), clear(A), clear(B), handFree}

G = {on(A,B), on(B,C), onTable(C)}

pickup(x)
- Prec: onTable(x), clear(x), handFree
- Add: holding(x)
- Del: onTable(x), clear(x), handFree

putdown(x)
- Prec: holding(x)
- Add: clear(x), onTable(x), handFree
- Del: holding(x)

unstack(x, y)
- Prec: on(x, y), clear(x), handFree
- Add: holding(x), clear(y)
- Del: on(x, y), clear(x), handFree

stack(x, y)
- Prec: holding(x), clear(y)
- Add: clear(x), on(x,y), handFree
- Del: clear(y), holding(x)

	clear(a)	clear(b)	clear(c)	handempty()	holding(a)	holding(b)	holding(c)	on(a,a)	on(a,b)	on(a,c)	on(b,a)	on(b,b)	on(b,c)	on(c,a)	on(c,b)	on(c,c)	ontable(a)	ontable(b)	ontable(c)
0	NA	NA	(unstack a c)	NA	(unstack a c)	(pick-up b)	(pick-up c)	(stack a a)	(stack a b)	NA	(stack b a)	(stack b b)	(stack b c)	(stack c a)	(stack c b)	(stack c c)	(put-down a)	NA	NA

Relaxed Plan Extraction for state s and best-supporter function bs

$Open := G \setminus s$; $Closed := \emptyset$; $RPlan := \emptyset$
while $Open \neq \emptyset$ **do**:
  select $g \in Open$
  $Open := Open \setminus \{g\}$; $Closed := Closed \cup \{g\}$;
  $RPlan := RPlan \cup \{bs(g)\}$; $Open := Open \cup (pre_{bs(g)} \setminus (s \cup Closed))$
endwhile
return $RPlan$

Open = {~~holding(A)~~, holding(B), clear(C)}
Closed = {on(A, B), on(B, C)} ∪ {holding (A)}
RPlan = {stack(A, B), stack(B, C)} ∪ {unstack (A,C)}

Iteration 3:
- Select *g* from Open   g = holding (A)
- Put *g* into Closed
- Get *bs(g)* and add *bs(g)* into RPlan   bs(holding(A)) = unstack (A,C)
- Get *preconditions* of bs(g) and update Open list if necessary

pre unstack(A,C) = { on (A,C), clear (A), handFree }

pre unstack (A,C) \ (I ∪ Closed) = { }

Problem 1: Relaxed Plan Extraction

pickup(x)
- Prec: onTable(x), clear(x), handFree
- Add: holding(x)
- Del: onTable(x), clear(x), handFree

putdown(x)
- Prec: holding(x)
- Add: clear(x), onTable(x), handFree
- Del: holding(x)

unstack(x, y)
- Prec: on(x, y), clear(x), handFree
- Add: holding(x), clear(y)
- Del: on(x, y), clear(x), handFree

stack(x, y)
- Prec: holding(x), clear(y)
- Add: clear(x), on(x,y), handFree
- Del: clear(y), holding(x)

I = {on(A, C), onTable(C), onTable(B), clear(A), clear(B), handFree}

G = {on(A,B), on(B,C), onTable(C)}

	clear(a)	clear(b)	clear(c)	handempty()	holding(a)	holding(b)	holding(c)	on(a,a)	on(a,b)	on(a,c)	on(b,a)	on(b,b)	on(b,c)	on(c,a)	on(c,b)	on(c,c)	ontable(a)	ontable(b)	ontable(c)
0	NA	NA	(unstack a c)	NA	(unstack a c)	(pick-up b)	(pick-up c)	(stack a a)	(stack a b)	NA	(stack b a)	(stack b b)	(stack b c)	(stack c a)	(stack c b)	(stack c c)	(put-down a)	NA	NA

Relaxed Plan Extraction for state s and best-supporter function bs

$Open := G \setminus s$; $Closed := \emptyset$; $RPlan := \emptyset$
while $Open \neq \emptyset$ **do**:
  select $g \in Open$
  $Open := Open \setminus \{g\}$; $Closed := Closed \cup \{g\}$;
  $RPlan := RPlan \cup \{bs(g)\}$; $Open := Open \cup (pre_{bs(g)} \setminus (s \cup Closed))$
endwhile
return $RPlan$

Open = {~~holding(B)~~, clear(C)}
Closed = {on(A, B), on(B, C), holding(A)} ∪ {holding (B)}
RPlan = {stack(A, B), stack(B, C), unstack(A, C)}
∪ {pick-up(B)}

Iteration 4:
- Select *g* from Open   g = holding(B)
- Put *g* into Closed
- Get *bs(g)* and add *bs(g)* into RPlan   bs(holding (B)) = pick-up (B)
- Get *preconditions* of bs(g) and update Open list if necessary

pre pick-up(B) = {onTable (B), clear(B), handFree}
pre pick-up (B) \ (I ∪ Closed) = { }

Problem 1: Relaxed Plan Extraction

I = {on(A, C), onTable(C), onTable(B), clear(A), clear(B), handFree}

G = {on(A,B), on(B,C), onTable(C)}

pickup(x)
- Prec: onTable(x), clear(x), handFree
- Add: holding(x)
- Del: onTable(x), clear(x), handFree

putdown(x)
- Prec: holding(x)
- Add: clear(x), onTable(x), handFree
- Del: holding(x)

unstack(x, y)
- Prec: on(x, y), clear(x), handFree
- Add: holding(x), clear(y)
- Del: on(x, y), clear(x), handFree

stack(x, y)
- Prec: holding(x), clear(y)
- Add: clear(x), on(x,y), handFree
- Del: clear(y), holding(x)

	clear(a)	clear(b)	clear(c)	handempty()	holding(a)	holding(b)	holding(c)	on(a,a)	on(a,b)	on(a,c)	on(b,a)	on(b,b)	on(b,c)	on(c,a)	on(c,b)	on(c,c)	ontable(a)	ontable(b)	ontable(c)
0	NA	NA	(unstack a c)	NA	(unstack a c)	(pick-up b)	(pick-up c)	(stack a a)	(stack a b)	NA	(stack b a)	(stack b b)	(stack b c)	(stack c a)	(stack c b)	(stack c c)	(put-down a)	NA	NA

Relaxed Plan Extraction for state s and best-supporter function bs

$Open := G \setminus s$; $Closed := \emptyset$; $RPlan := \emptyset$
while $Open \neq \emptyset$ **do**:
  select $g \in Open$
  $Open := Open \setminus \{g\}$; $Closed := Closed \cup \{g\}$;
  $RPlan := RPlan \cup \{bs(g)\}$; $Open := Open \cup (pre_{bs(g)} \setminus (s \cup Closed))$
endwhile
return $RPlan$

Open = {~~clear(C)~~}
Closed = {on(A, B), on(B, C), holding(A), holding(B)} ∪ {clear(C)}
RPlan = {stack(A, B), stack(B, C), unstack(A, C), pickup(B)}

→ Relaxed Plan

Return.

Iteration 5:
- Select *g* from Open   g = clear(C)
- Put *g* into Closed
- Get *bs(g)* and add *bs(g)* into RPlan   bs(clear(C)) = unstack(A,C)
- Get *preconditions* of bs(g) and update Open list if necessary

pre unstack(A, C) = {on(A,C), clear(A), handFree}
pre unstack(A,C) \ (I ∪ Closed) = { }

Problem 1: Get h^{ff}

action cost = 1

RPlan = {stack(A, B), stack(B, C), unstack(A, C), pickup(B)}

h^{ff} is the sum of the cost of actions in the relaxed plan

$h^{ff} = 1 + 1 + 1 + 1 = 4$

h^{ff} = 4 for both $h^{\boldsymbol{max}}$ and $h^{\boldsymbol{add}}$ (because they have the same best supporter functions for all facts)

$h^{ff} \neq |RPlan|$

Problem 2: Iterated Width (IW)

Iterated Width (IW) vs Iterative Deepening (ID)

- Both are blind search algorithms
- ID: DFS with depth limit
- IW: BFS with width limit

Problem 2: Iterated Width (IW)

Find the novelty w(s) of a state s?

Algorithm

- $IW(k)$ = breadth-first search that prunes newly generated states whose $novelty(s) > k$.
- IW is a sequence of calls $IW(k)$ for $i = 0, 1, 2, \ldots$ over problem P until problem solved or i exceeds number of variables in problem

Key definition: the **novelty $w(s)$ of a state s** is the size of the smallest subset of atoms in s that is true for the first time in the search.

- e.g. $w(s) = 1$ if there is **one** atom $p \in s$ such that s is the first state that makes p true.
- Otherwise, $w(s) = 2$ if there are **two** different atoms $p, q \in s$ such that s is the first state that makes $p \wedge q$ true.
- Otherwise, $w(s) = 3$ if there are **three** different atoms...

F = {a, b, c} (3)

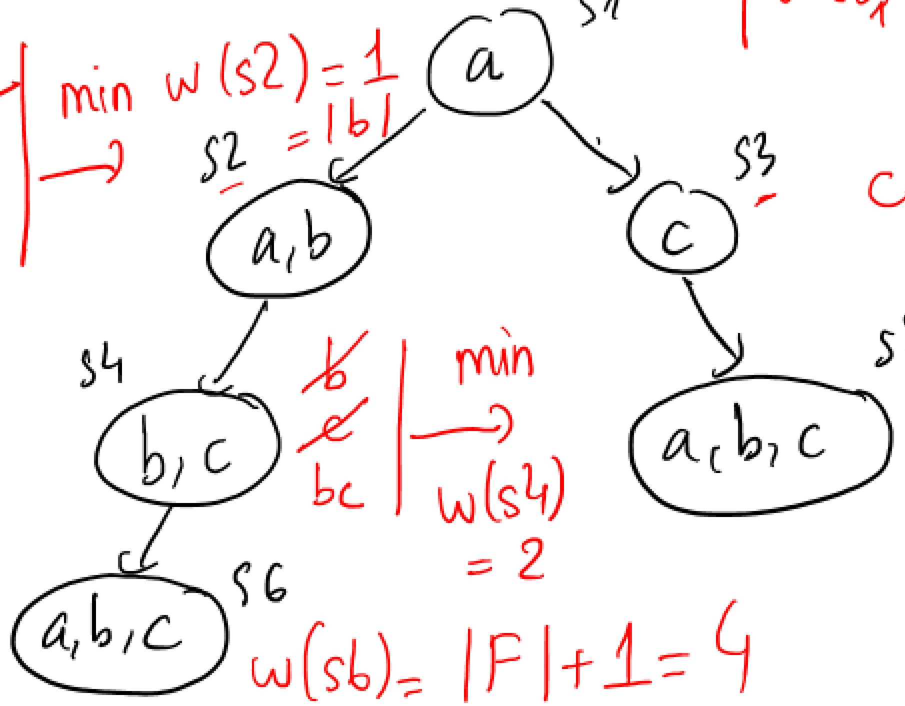
Queue

Novelty table

	Have seen ?
a	True
b	True
c	True
a, b	True
a, c	True
b, c	True
a, b, c	True

all combinations

BFS: left → right



a | $w(s) = 1 = |a|$

~~a~~
b
ab

min $w(s2) = 1 = |b|$

c | $w(s_3) = 1$

~~b~~
~~c~~
bc

min $w(s4) = 2$

~~a~~
~~b~~
~~c~~
~~ab~~
~~bc~~
ac
abc

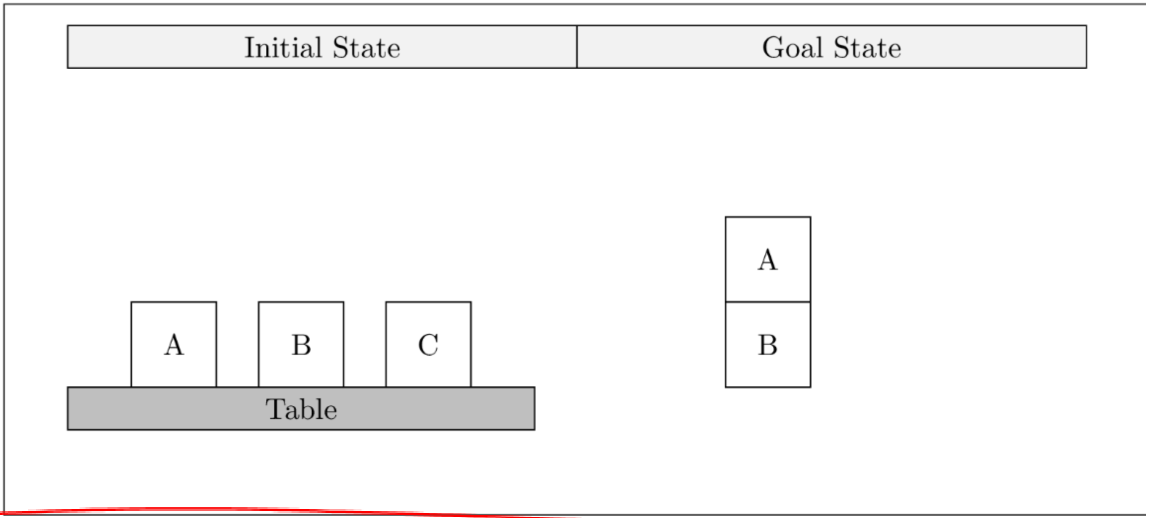
min $w(s5) = 2$

$w(s6) = |F| + 1 = 4$

Problem 2: Iterated Width (IW)

Show the IW(1): Prune when novelty(s) > 1

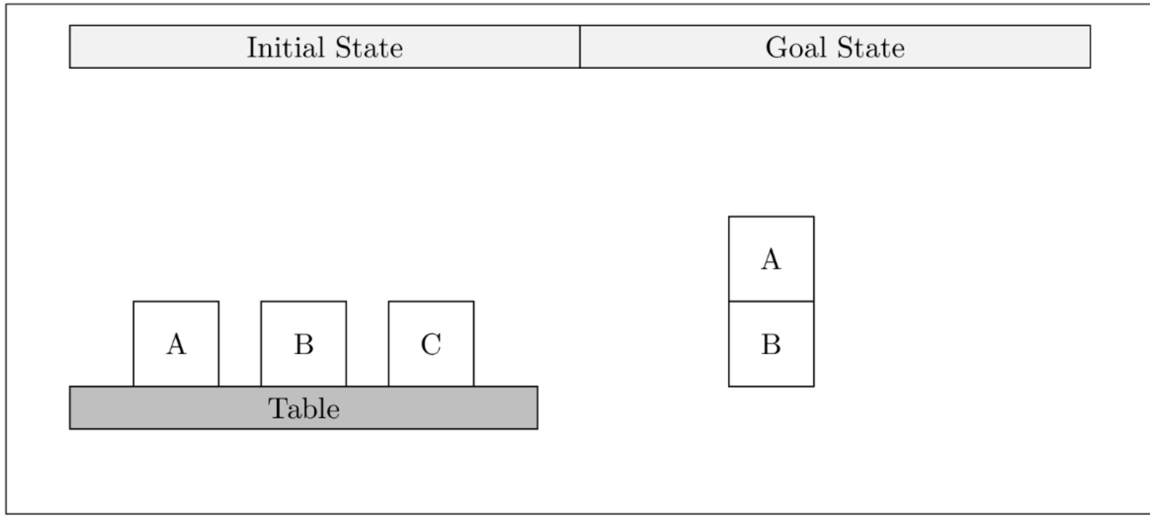
I = {onTable(A), onTable(B), onTable(C), clear(A), clear(B), clear(C), handFree}
G = {on(A, B)}



IW(n): prune nodes when novelty(s) > n

F = {}
Novelty table

Problem 2: Iterated Width (IW)



Show the IW(1): Prune when novelty(s) > 1

I = {onTable(A), onTable(B), onTable(C), clear(A), clear(B), clear(C), handFree}
G = {on(A, B)}

I

w = 1

actions ?

pickup(A)

pickup(B)

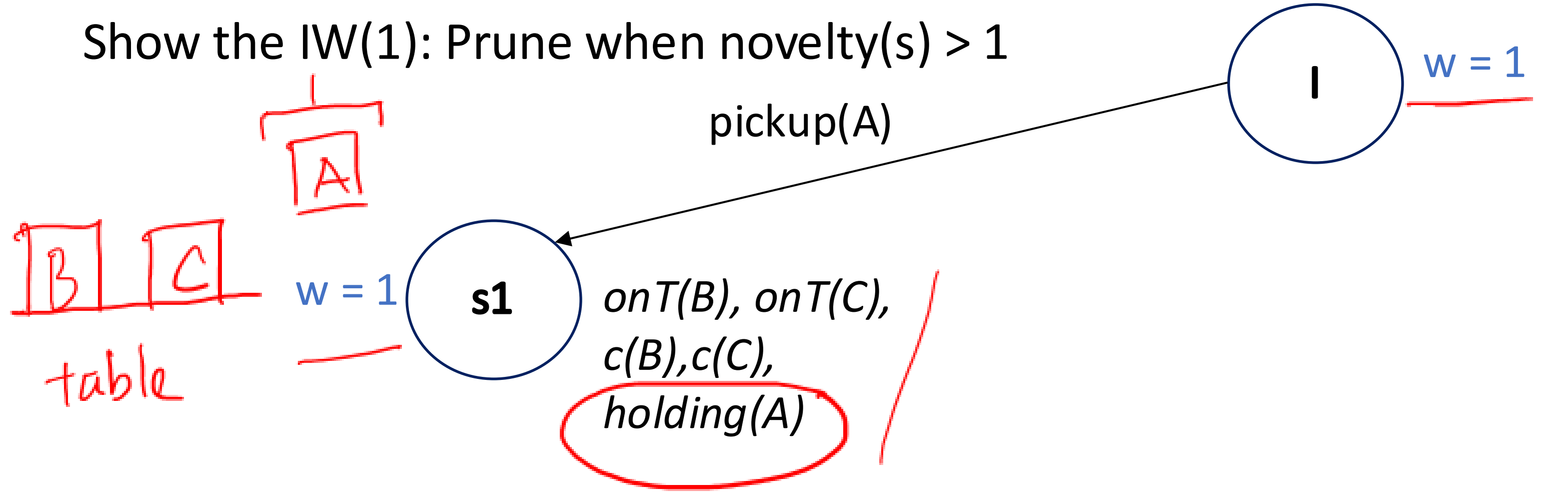
pickup(C)

Novelty table:
onTable(A), onTable(B), onTable(C), clear(A), clear(B), clear(C), handFree + all other combinations

Problem 2: Iterated Width (IW)

I = {onTable(A), onTable(B), onTable(C), clear(A), clear(B), clear(C), handFree}
G = {on(A, B)}

Show the IW(1): Prune when novelty(s) > 1

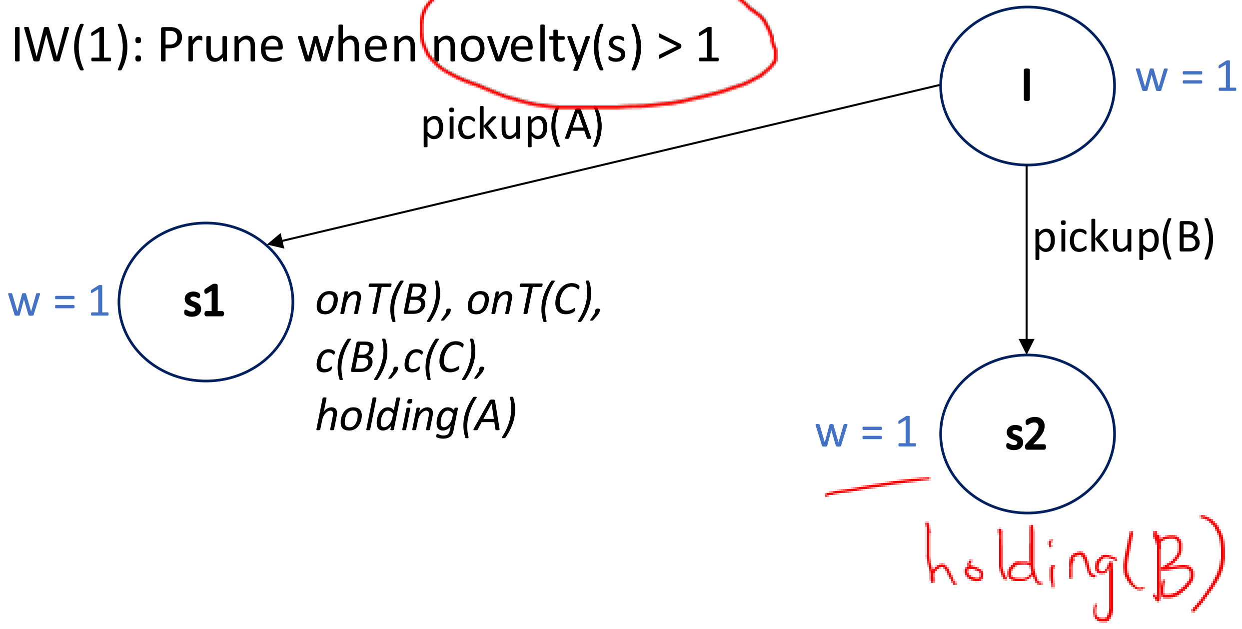


Novelty table
onTable(A), onTable(B), onTable(C), clear(A), clear(B), clear(C), handFree, holding(A)

Problem 2: Iterated Width (IW)

I = {onTable(A), onTable(B), onTable(C), clear(A), clear(B), clear(C), handFree}
G = {on(A, B)}

Show the IW(1): Prune when novelty(s) > 1

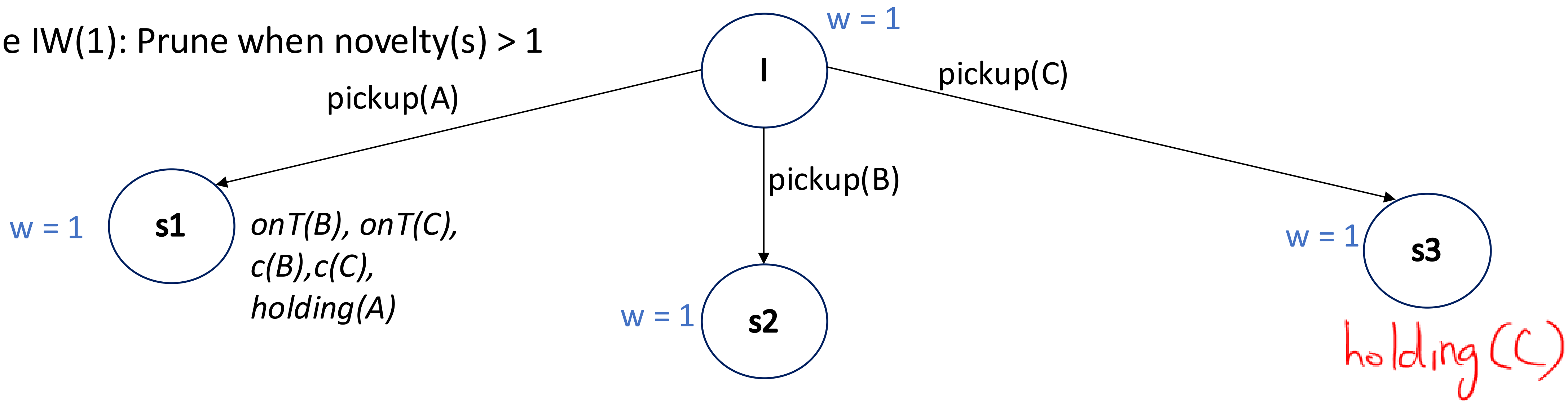


Novelty table
onTable(A), onTable(B), onTable(C), clear(A), clear(B), clear(C), handFree,
holding(A), holding(B)

Problem 2: Iterated Width (IW)

I = {onTable(A), onTable(B), onTable(C), clear(A), clear(B), clear(C), handFree}
G = {on(A, B)}

Show the IW(1): Prune when novelty(s) > 1

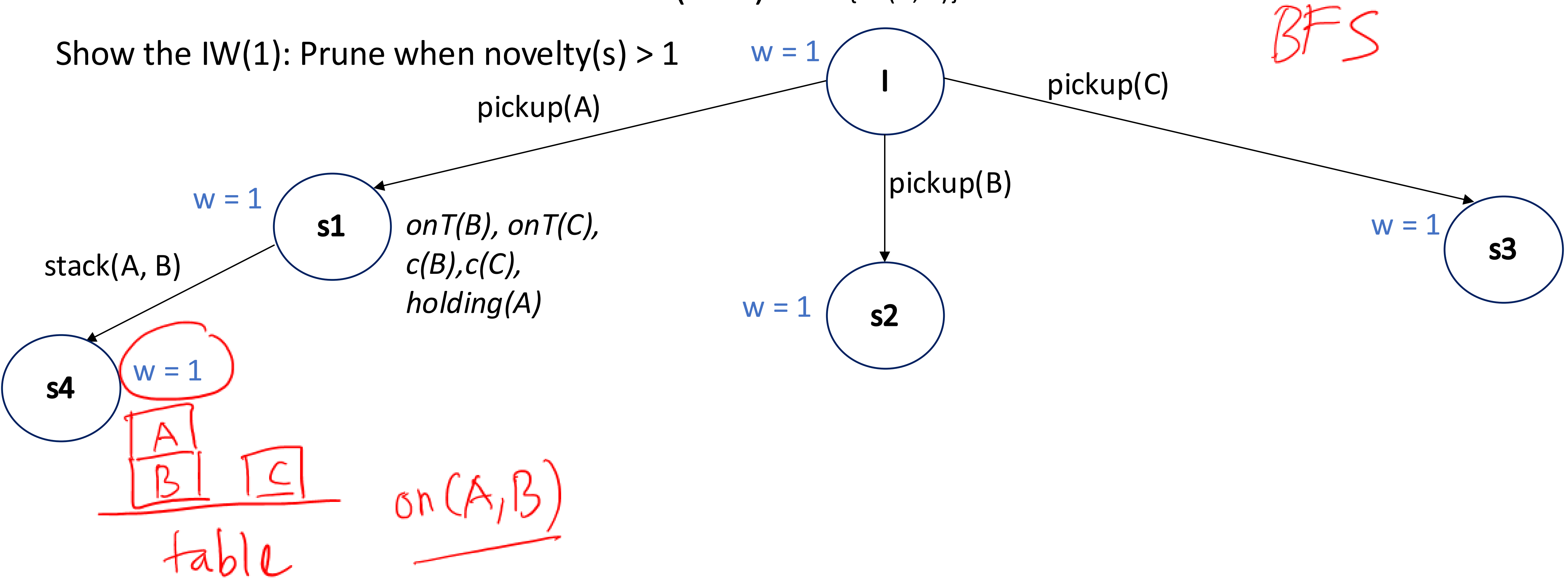


Novelty table
onTable(A), onTable(B), onTable(C), clear(A), clear(B), clear(C), handFree,
holding(A), holding(B), holding(C)

Problem 2: Iterated Width (IW)

I = {onTable(A), onTable(B), onTable(C), clear(A), clear(B), clear(C), handFree}
G = {on(A, B)}

Show the IW(1): Prune when novelty(s) > 1

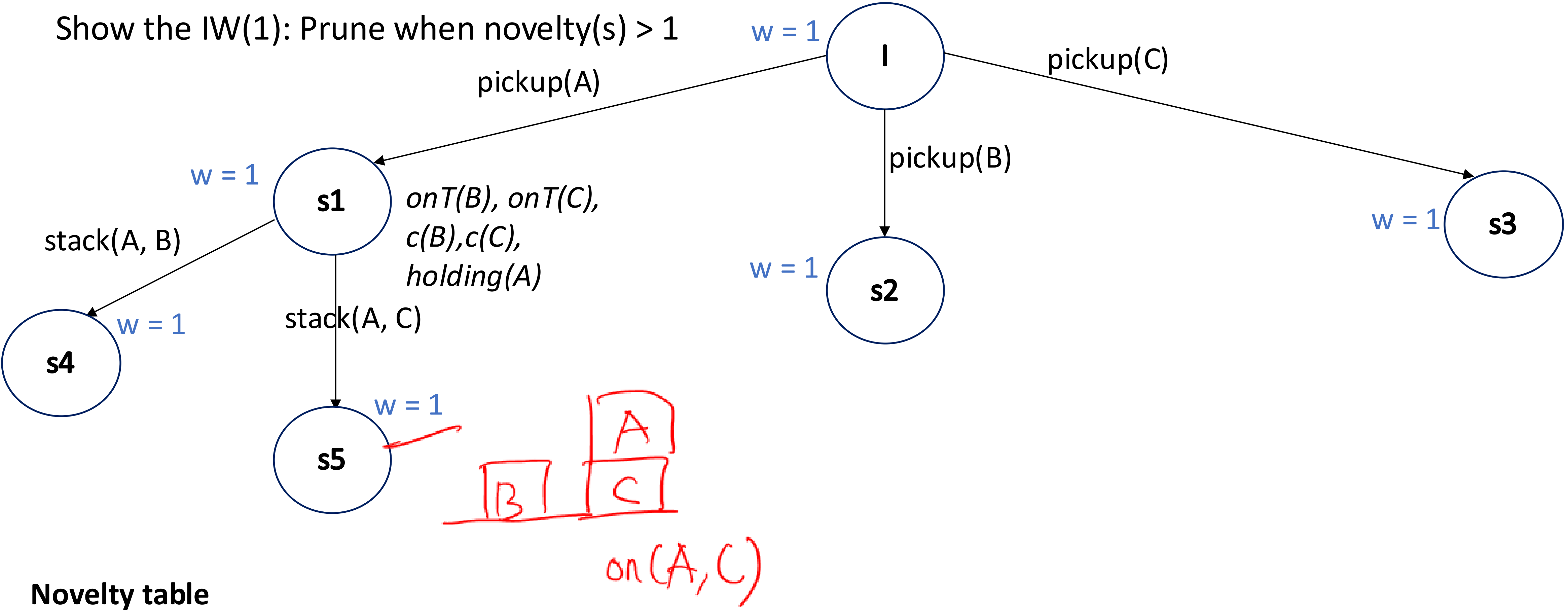


Novelty table
onTable(A), onTable(B), onTable(C), clear(A), clear(B), clear(C), handFree,
holding(A), holding(B), holding(C),
on(A, B)

Problem 2: Iterated Width (IW)

I = {onTable(A), onTable(B), onTable(C), clear(A), clear(B), clear(C), handFree}
G = {on(A, B)}

Show the IW(1): Prune when novelty(s) > 1

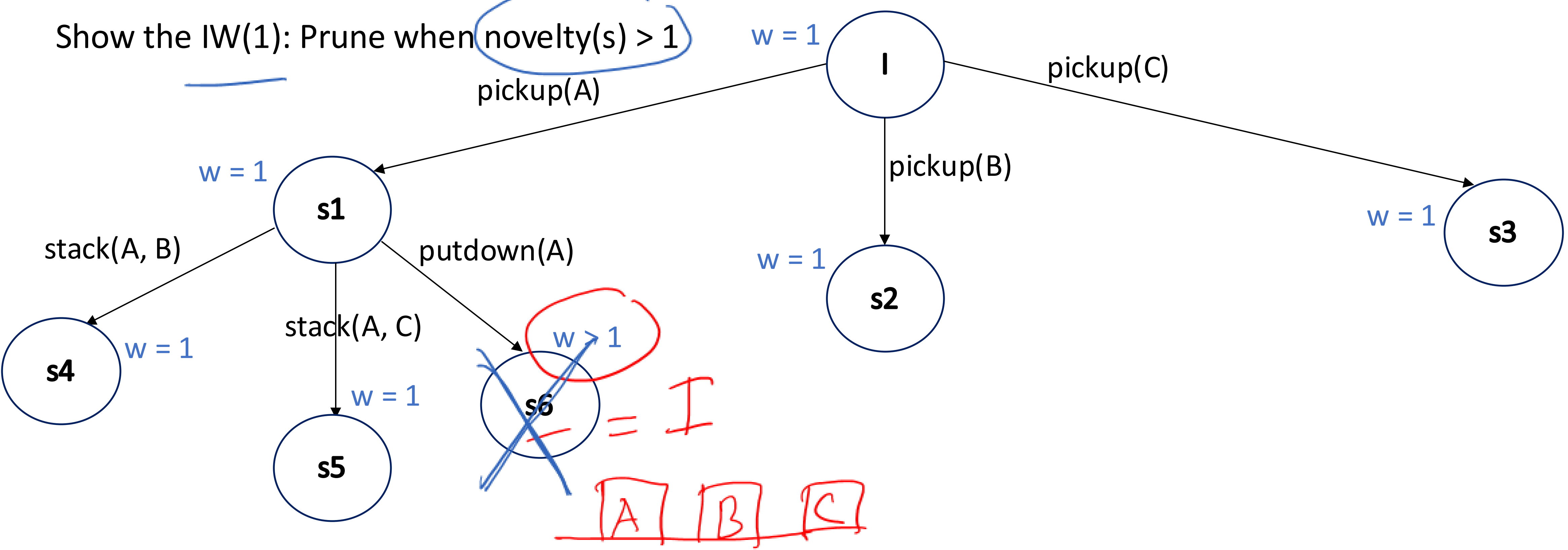


Novelty table
onTable(A), onTable(B), onTable(C), clear(A), clear(B), clear(C), handFree,
holding(A), holding(B), holding(C),
on(A, B), on(A, C)

Problem 2: Iterated Width (IW)

I = {onTable(A), onTable(B), onTable(C), clear(A), clear(B), clear(C), handFree}
G = {on(A, B)}

Show the IW(1): Prune when novelty(s) > 1

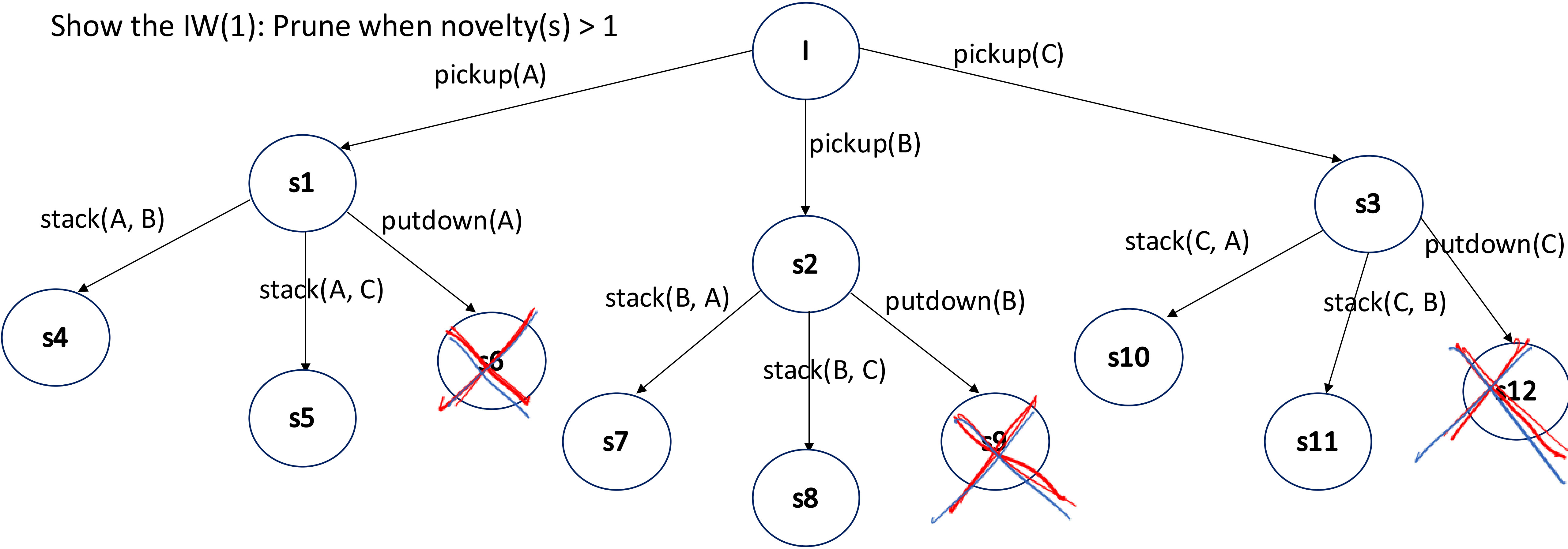


Novelty table
onTable(A), onTable(B), onTable(C), clear(A), clear(B), clear(C), handFree,
holding(A), holding(B), holding(C),
on(A, B), on(A, C)

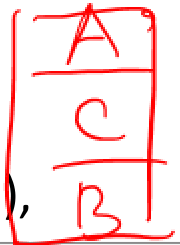
Problem 2: Iterated Width (IW)

I = {onTable(A), onTable(B), onTable(C), clear(A), clear(B), clear(C), handFree}
G = {on(A, B)}

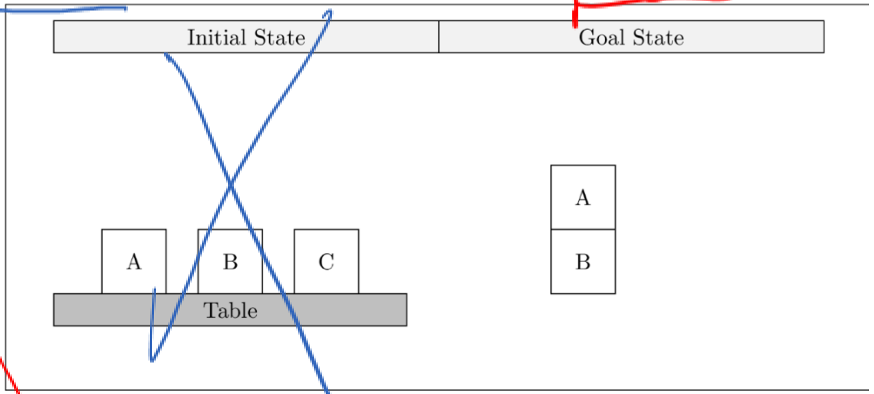
Show the IW(1): Prune when novelty(s) > 1



Problem 2: Iterated Width (IW)

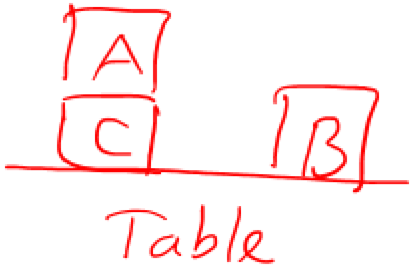


Task 2: Can you think of an initial situation where IW(1) cannot find a solution for the goal on(A,B), but IW(2) does, explain your answer?



Find a new initial state?

- Find a solution with IW(2)
- Can't find a solution with IW(1)



(1) on(A,B) (goal state)

$I \rightarrow S_1 \longrightarrow S_2 \longrightarrow \dots \longrightarrow S_t \longrightarrow G$

S_t must have holding(A) and clear(B)

(2) Saw holding(A) at S_a

Saw clear(B) at S_b

$a, b < t$

$w(S_t) = 2$

Problem 2: Iterated Width (IW)

Task 2: Can you think of an initial situation where IW(1) cannot find a solution for the goal on(A,B), but IW(2) does, explain your answer?