State-Space Planning

Stephen G. Ware

CSCI 4525 / 5525

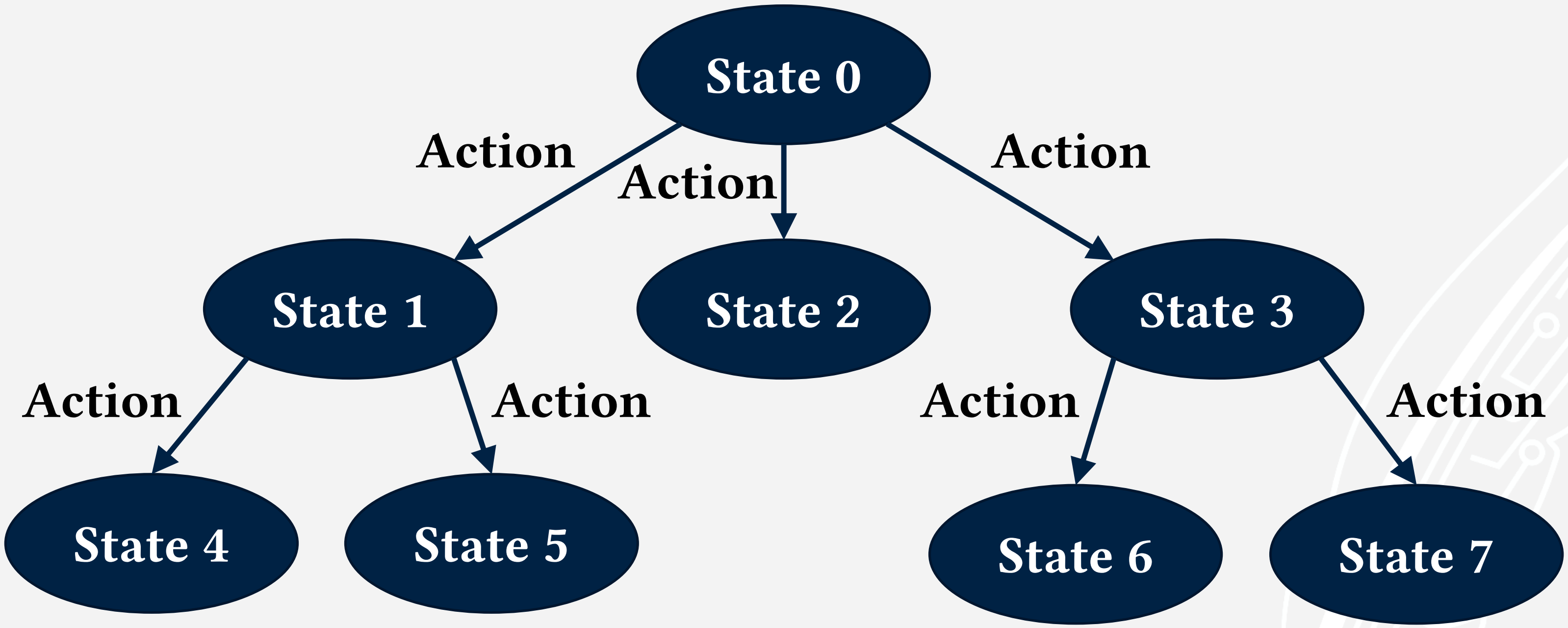
Narrative Intelligence LAB

THE UNIVERSITY of NEW ORLEANS

Short History of Planning

- State-space planning seems the most intuitive way to make planning into a search problem.
- However, a planning problem's state space explodes so quickly that we can't hope to search it unless we have a very accurate heuristic.
- We can reduce the number of decisions we have to make during planning through abstraction.
- Plan graphs provide an abstraction of a planning problem's search space.
- Plan graphs lead to accurate heuristics!

State-Space Search



State-Space Planner

```
Begin with an empty priority queue.
Put the initial state onto the priority queue.
While the queue is not empty:
    Pop a state C off the priority queue.
    If C is a goal state, return the plan to get to C.
    For every step S whose preconditions are satisfied in C:
        Let N be the state after taking step S in state C.
        Push N onto the queue.
Return failure.
```

State-Space Search

Priority Queue:

State-Space Search

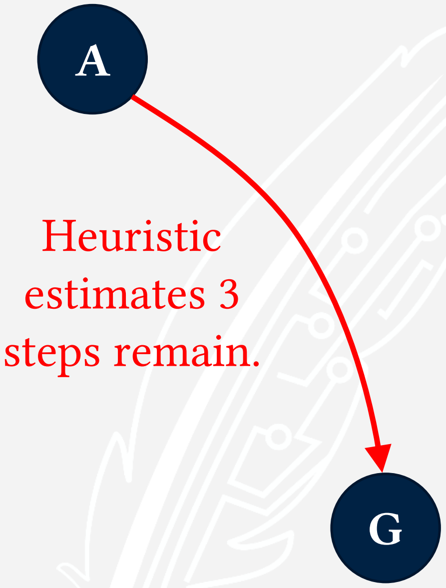
Start by pushing the initial state onto the priority queue.

Priority Queue:

State-Space Search

Start by pushing the initial state onto the priority queue.

Priority Queue:

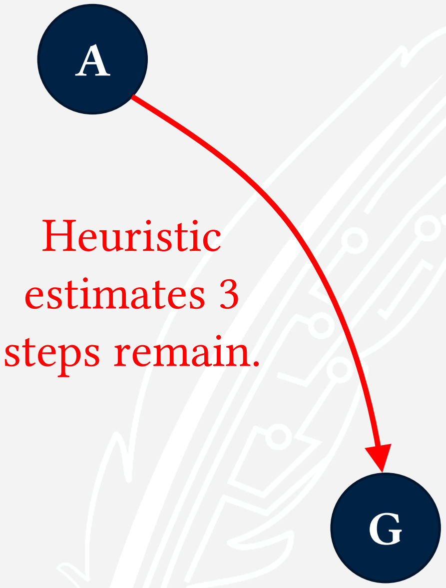


State-Space Search

Start by pushing the initial state onto the priority queue.

Priority Queue:

A: 0 + 3 = 3



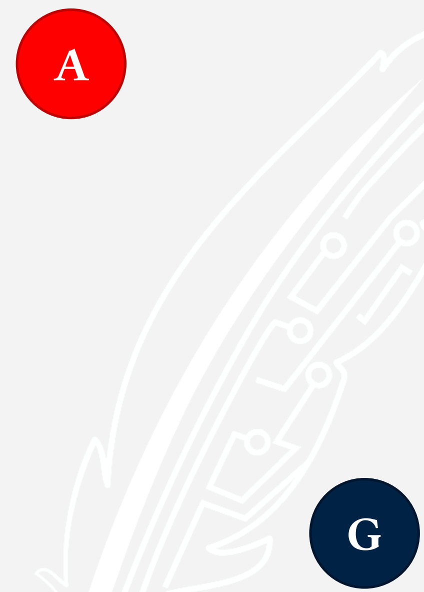
State-Space Search

Pop a state off the queue.

Current State: A

Current Plan: ∅

Priority Queue:



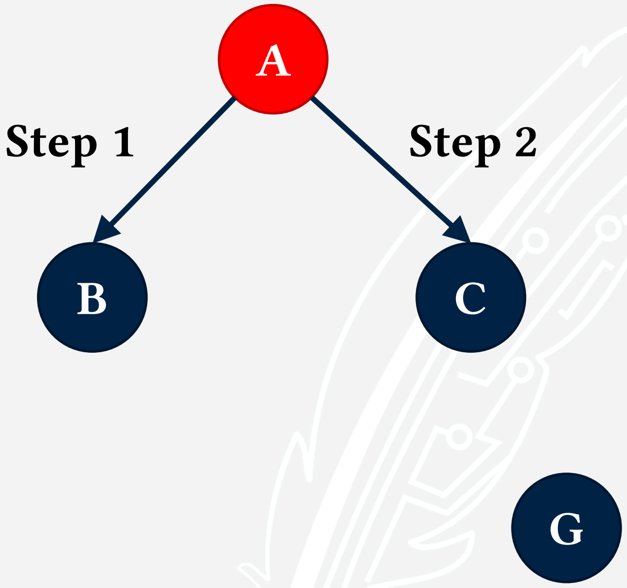
State-Space Search

Expand the current state.

Current State: A

Current Plan: ∅

Priority Queue:



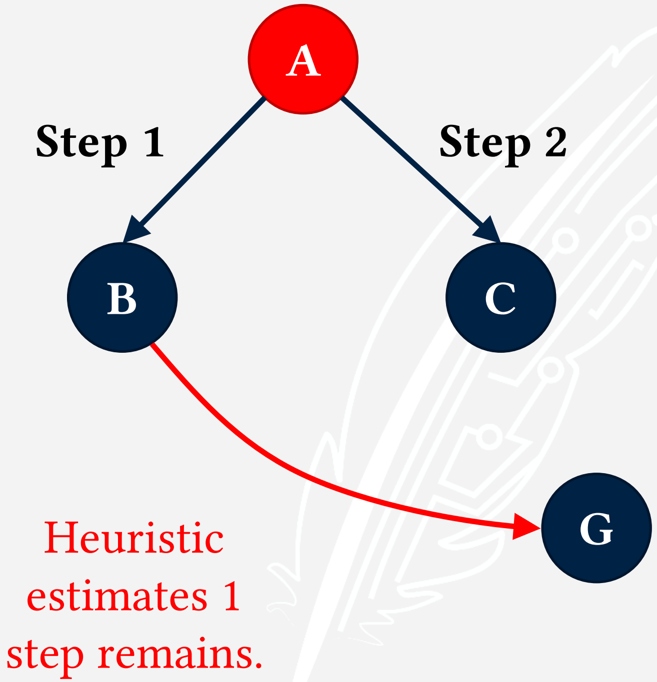
State-Space Search

Put children on the queue.

Current State: A

Current Plan: ∅

Priority Queue:



State-Space Search

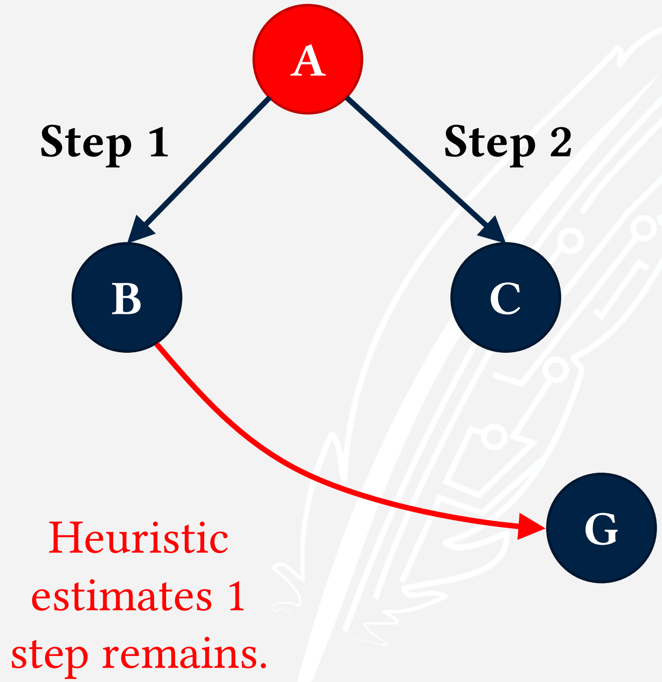
Put children on the queue.

Current State: A

Current Plan: ∅

Priority Queue:

B: 1 + 1 = 2



State-Space Search

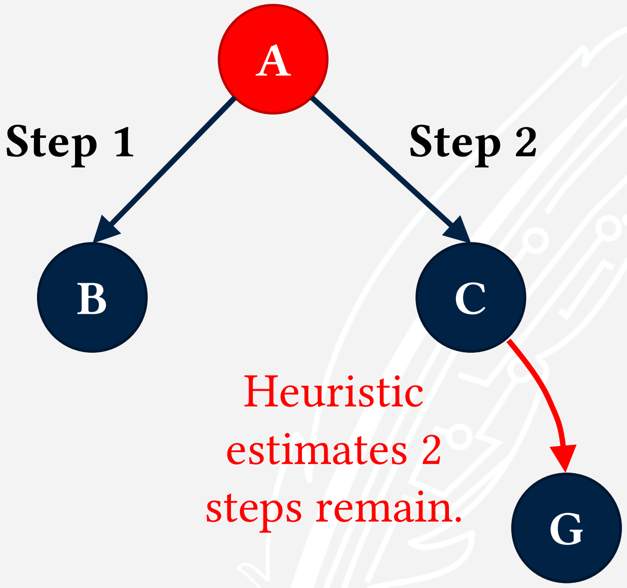
Put children on the queue.

Current State: A

Current Plan: Ø

Priority Queue:

B: 1 + 1 = 2



State-Space Search

Put children on the queue.

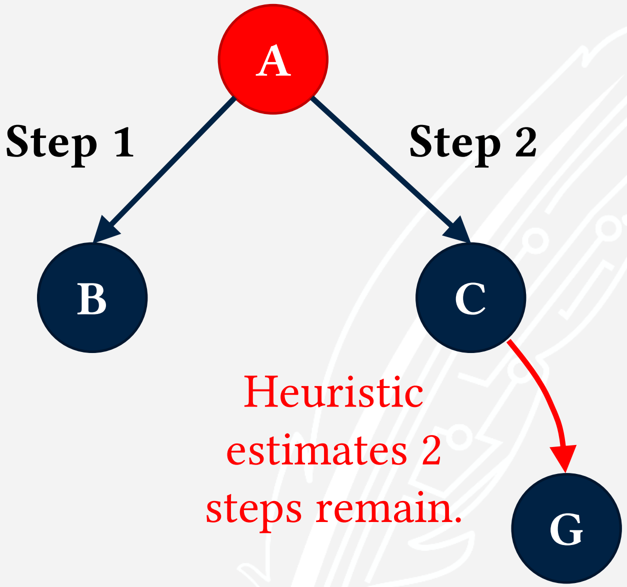
Current State: A

Current Plan: ∅

Priority Queue:

B: 1 + 1 = 2

C: 1 + 2 = 3



State-Space Search

Put children on the queue.

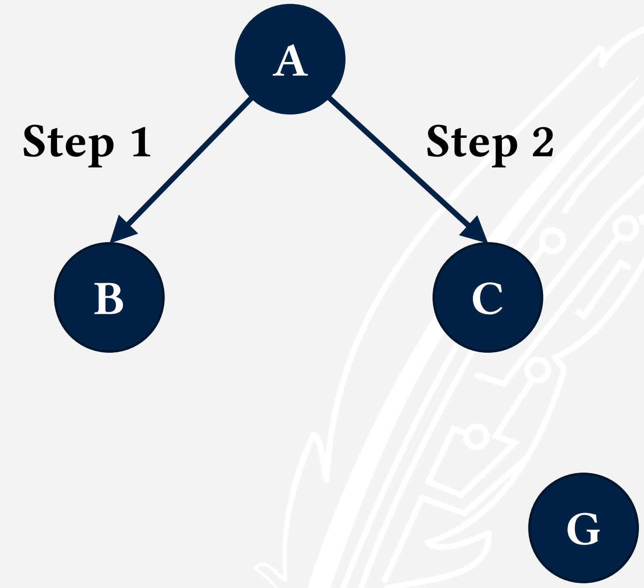
Current State: A

Current Plan: ∅

Priority Queue:

B: 1 + 1 = 2

C: 1 + 2 = 3



State-Space Search

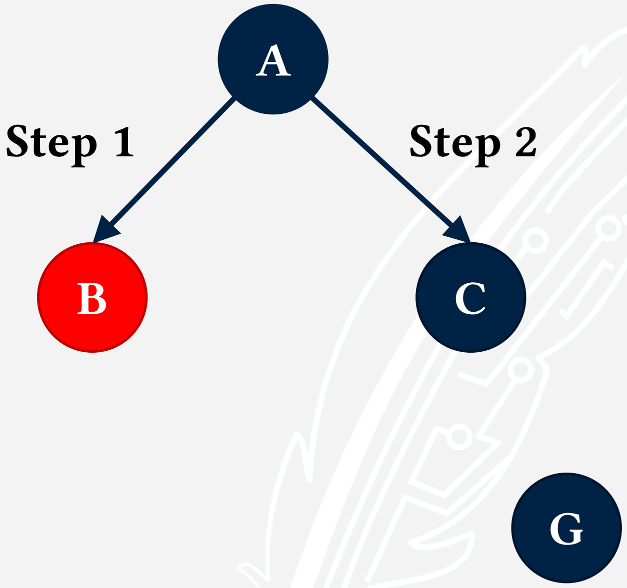
Pop a state off the queue.

Current State: B

Current Plan: 1

Priority Queue:

C: 1 + 2 = 3



State-Space Search

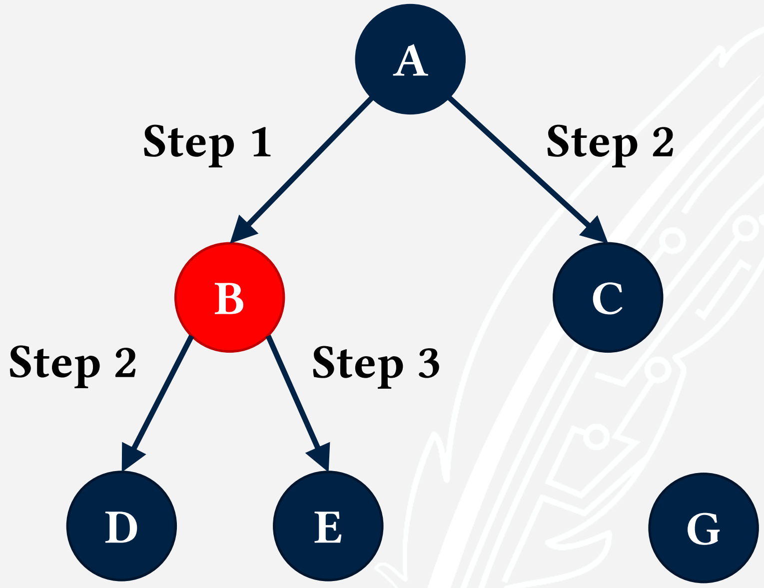
Expand the current state.

Current State: B

Current Plan: 1

Priority Queue:

C: 1 + 2 = 3



State-Space Search

Put children on the queue.

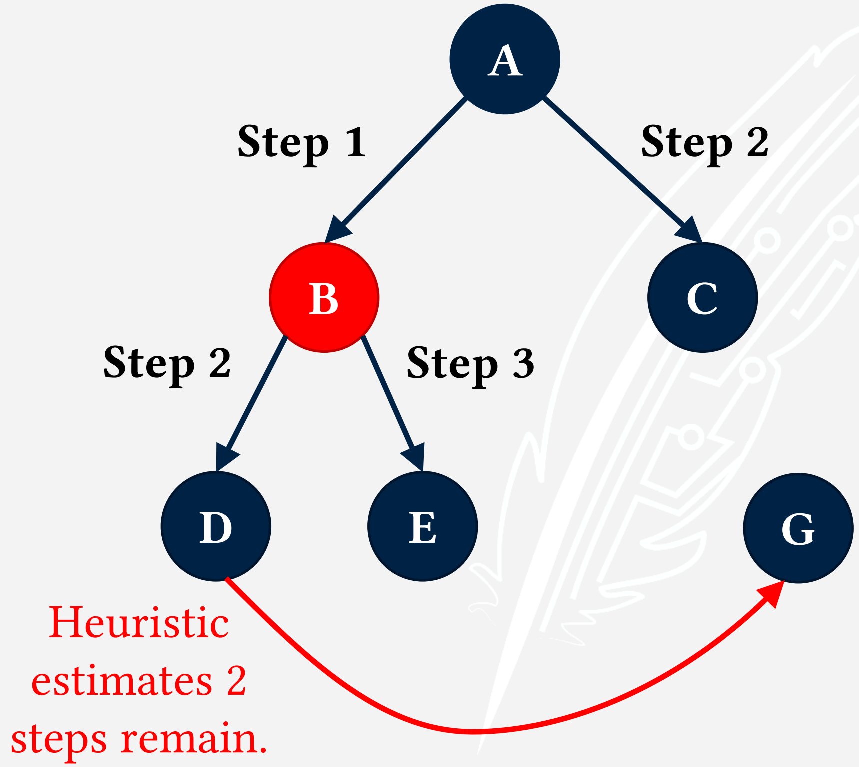
Current State: B

Current Plan: 1

Priority Queue:

C: 1 + 2 = 3

D: 2 + 2 = 4



State-Space Search

Put children on the queue.

Current State: B

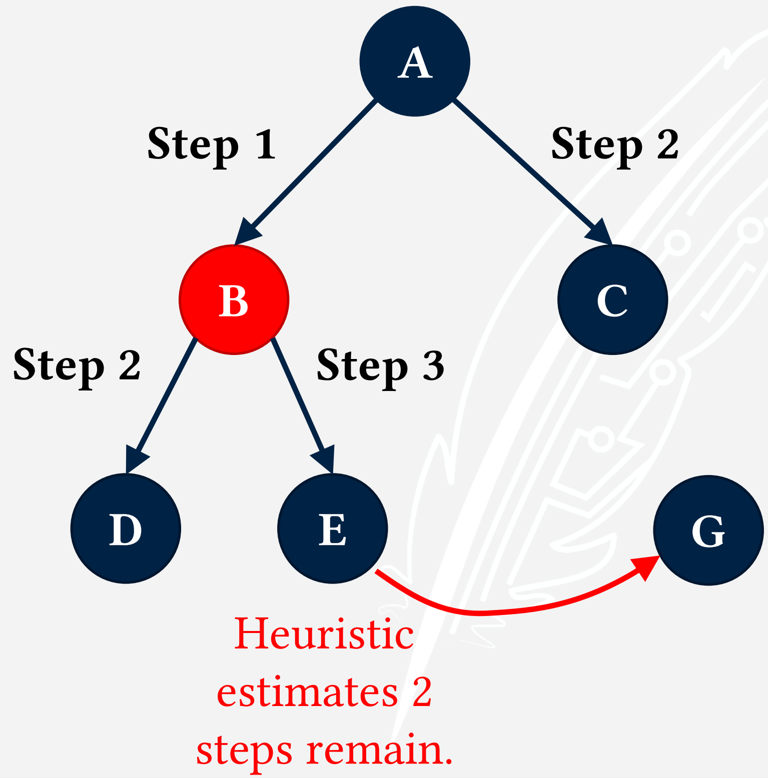
Current Plan: 1

Priority Queue:

C: 1 + 2 = 3

D: 2 + 2 = 4

E: 2 + 2 = 4



State-Space Search

Put children on the queue.

Current State: B

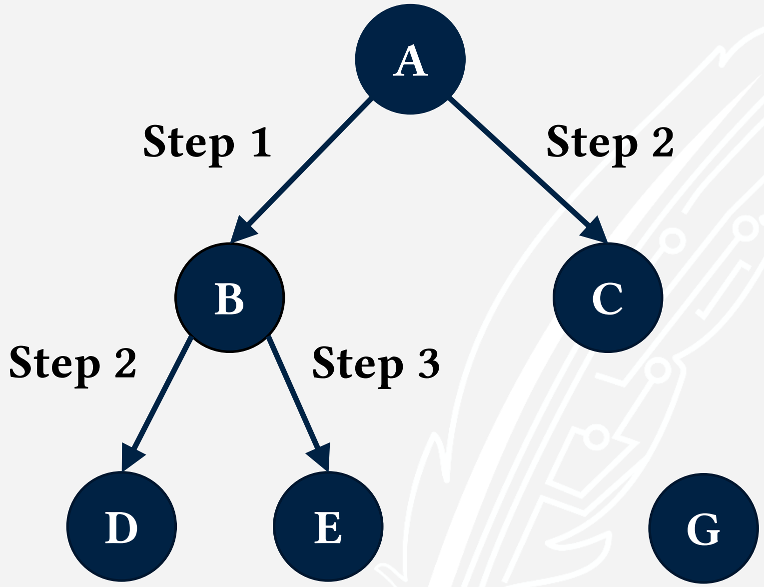
Current Plan: 1

Priority Queue:

C: 1 + 2 = 3

D: 2 + 2 = 4

E: 2 + 2 = 4



State-Space Search

Pop a state off the queue.

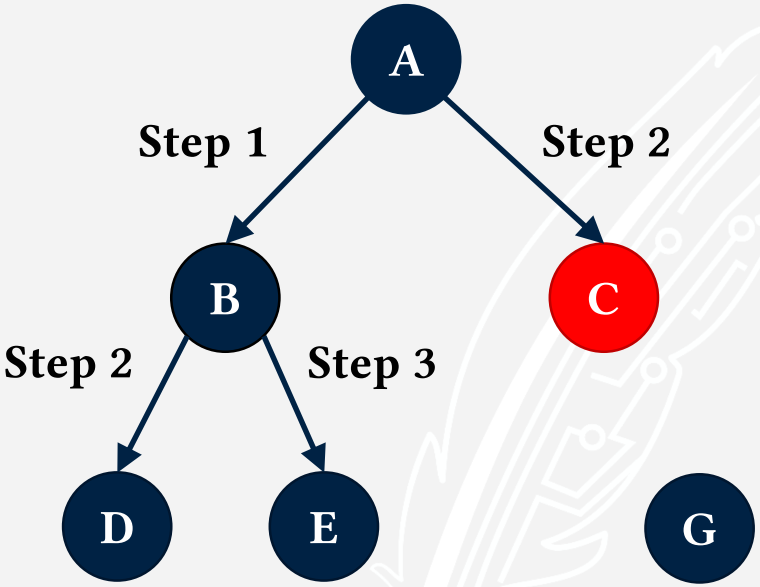
Current State: C

Current Plan: 2

Priority Queue:

D: 2 + 2 = 4

E: 2 + 2 = 4



State-Space Search

Expand the current state.

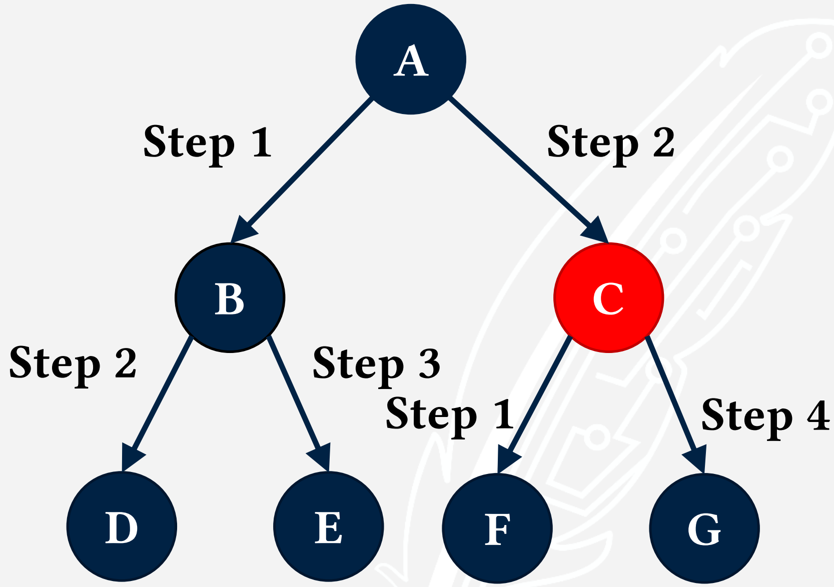
Current State: C

Current Plan: 2

Priority Queue:

D: 2 + 2 = 4

E: 2 + 2 = 4



State-Space Search

Put children on the queue.

Current State: C

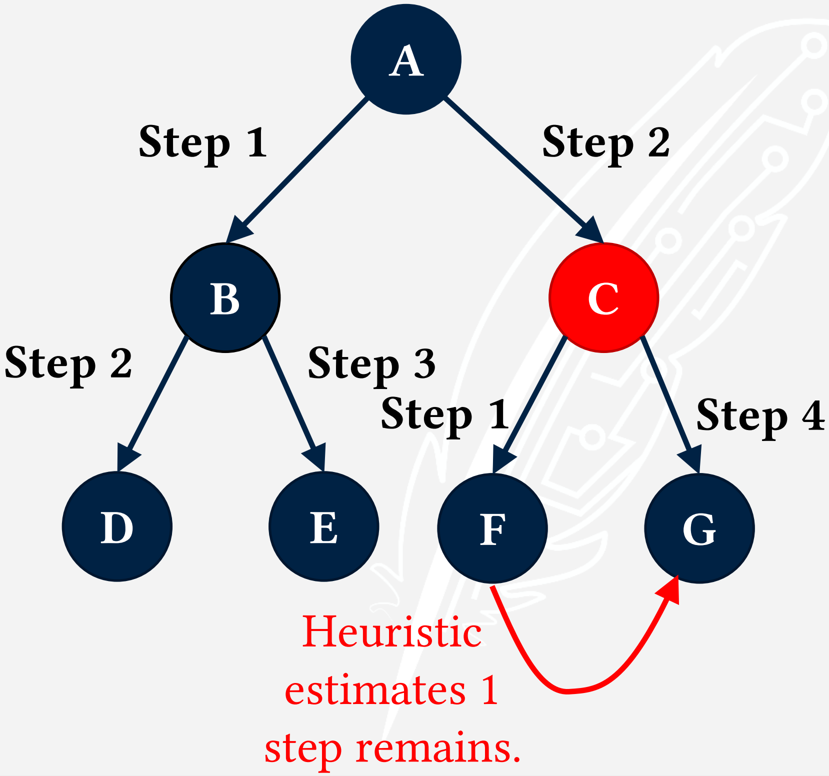
Current Plan: 2

Priority Queue:

F: 2 + 1 = 3

D: 2 + 2 = 4

E: 2 + 2 = 4



State-Space Search

Put children on the queue.

Current State: C

Current Plan: 2

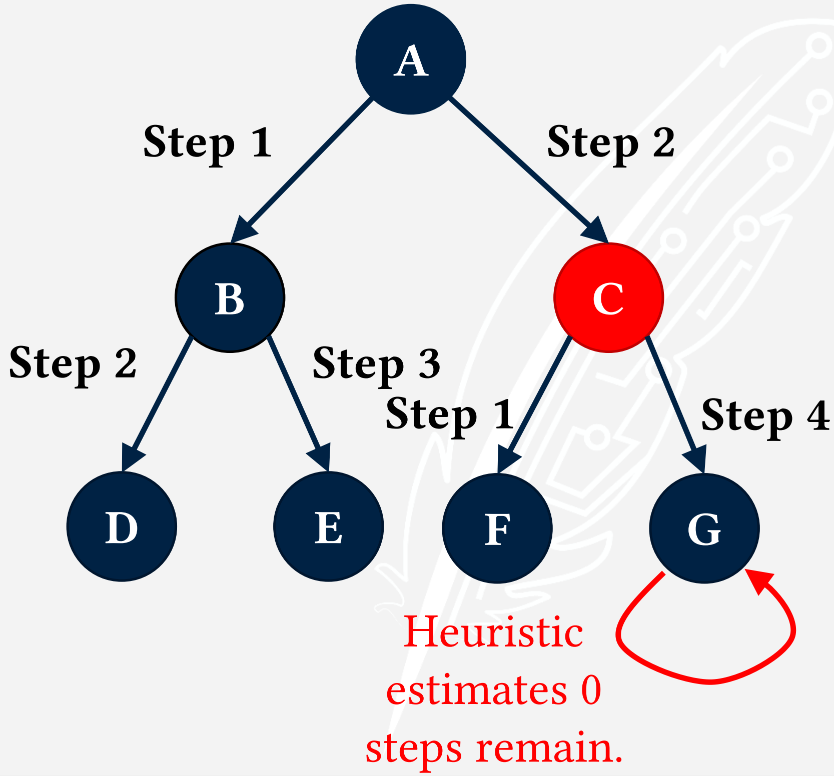
Priority Queue:

G: 2 + 0 = 2

F: 2 + 1 = 3

D: 2 + 2 = 4

E: 2 + 2 = 4



State-Space Search

Put children on the queue.

Current State: C

Current Plan: 2

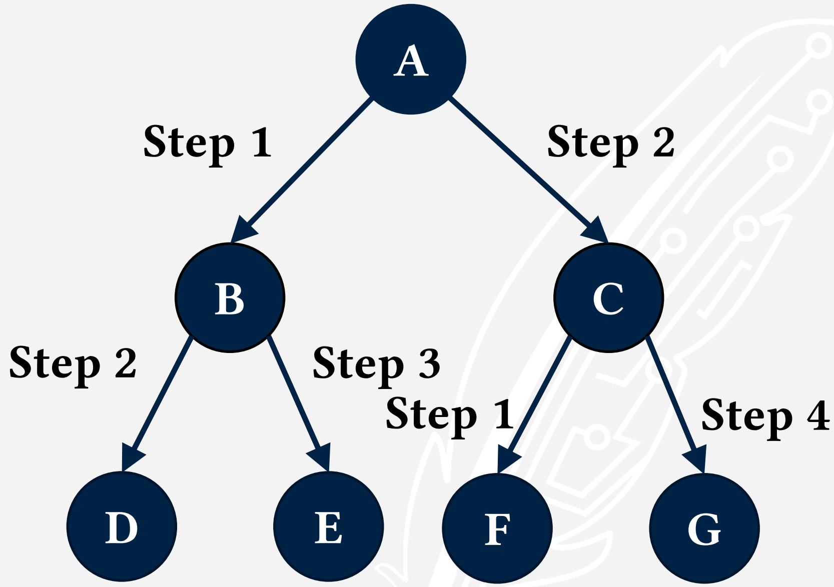
Priority Queue:

G: 2 + 0 = 2

F: 2 + 1 = 3

D: 2 + 2 = 4

E: 2 + 2 = 4



State-Space Search

Pop a state off the queue.

Current State: G

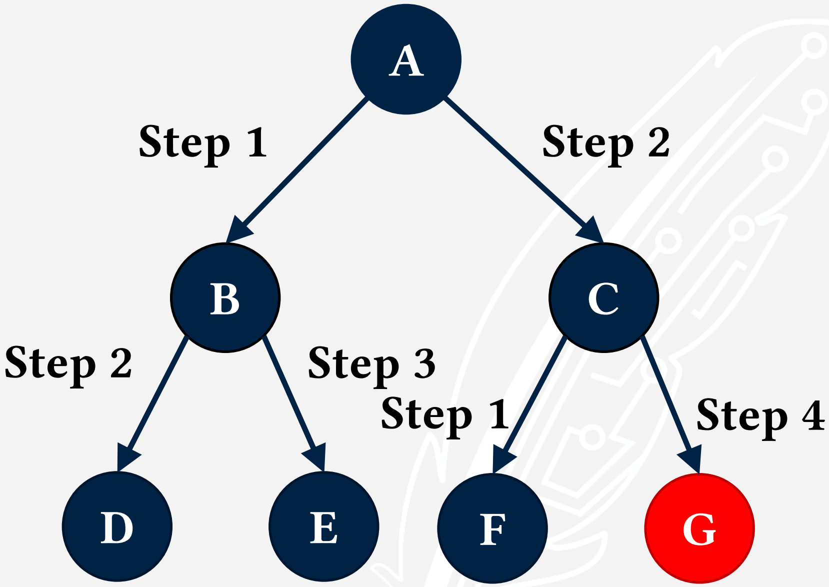
Current Plan: 2, 4

Priority Queue:

F: 2 + 1 = 3

D: 2 + 2 = 4

E: 2 + 2 = 4



State-Space Search

Current state is a goal state!

Current State: G

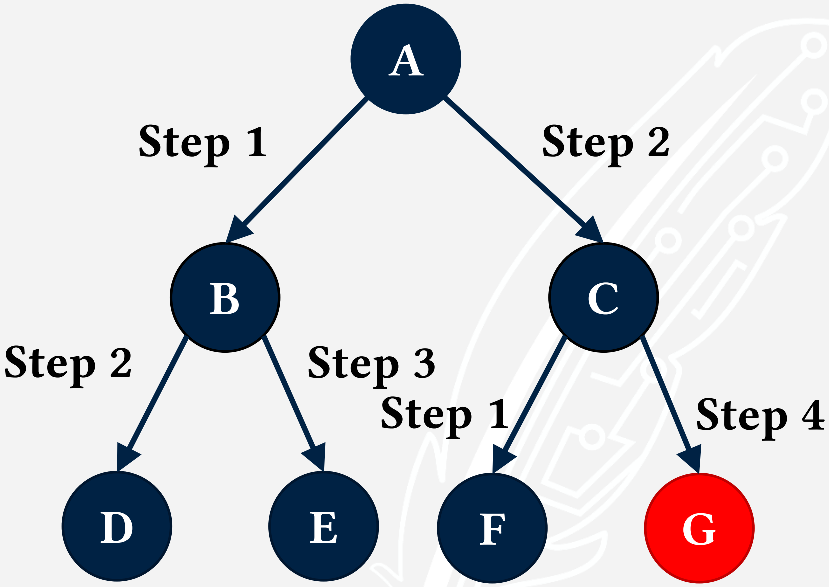
Current Plan: 2, 4

Priority Queue:

F: 2 + 1 = 3

D: 2 + 2 = 4

E: 2 + 2 = 4



State-Space Search

Return plan to reach G.

Current State: G

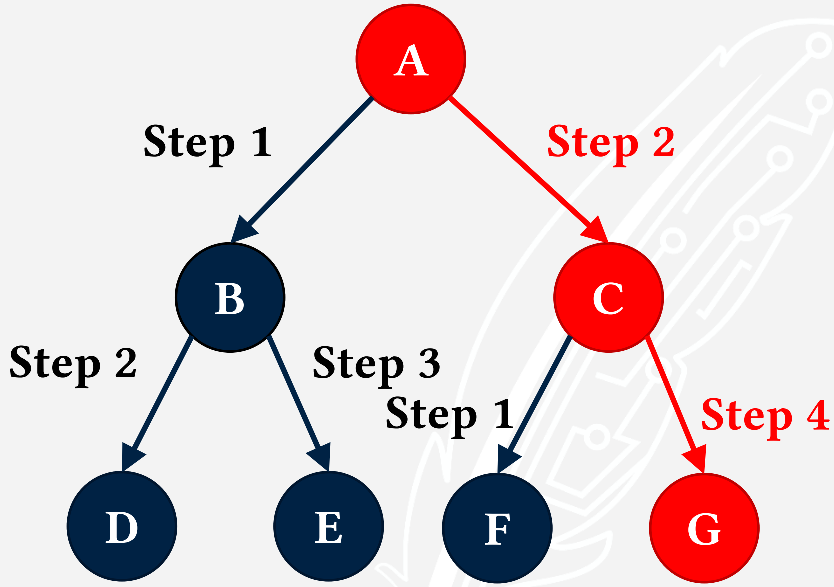
Current Plan: 2, 4

Priority Queue:

F: 2 + 1 = 3

D: 2 + 2 = 4

E: 2 + 2 = 4



Planning Heuristics

The speed of a state-space planner is entirely dependent on its heuristic.

State-space planning algorithms are simple; the complexity and ingenuity comes in how they calculate their heuristics.

Heuristics

A state-space planning heuristic estimates the answer to the following question: "Given some current state, how many more steps need to be taken before a goal state is reached?"

Ideally, a heuristics is:

- Highly accurate
- Admissible
- Fast to calculate

Heuristic Search Planner (HSP)

- Created by Blai Bonet, Gábor Loerincs, and Héctor Geffner
- Perhaps the first viable state-space planner.
- Winner of the first International Planning Competition in 1998

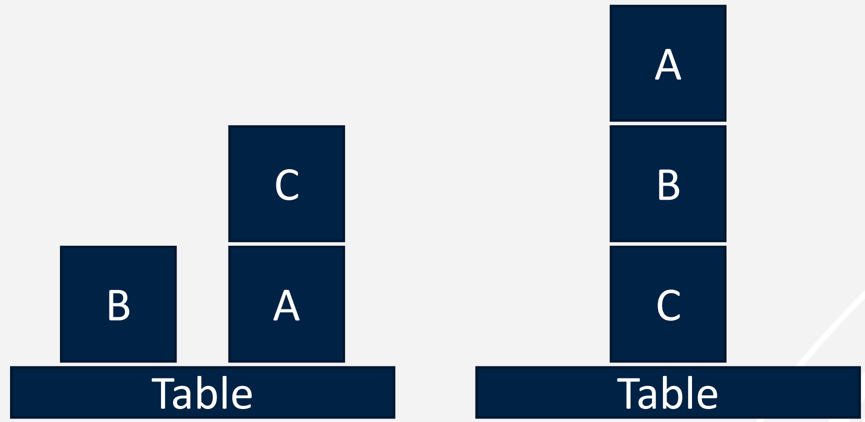
HSP's Heuristic

```
Input: The current state.
Every literal has a cost, initially ∞.
Every literal that is true in the current state has a cost of 0.
The cost of a conjunction is the sum of the costs of its conjuncts.
Do this until the costs of the literals stop changing:
    For every step S:
        For every literal E in the effect of S:
            Let the cost of E be the minimum of:
                1. The current cost of E.
                2. The cost of S's precondition + 1.
Return the cost of the problem's goal.
```

HSP in Blocks World

Weights:

$h(on(A, Table)) = \infty$
$h(\neg on(A, Table)) = \infty$
$h(on(A, B)) = \infty$
$h(on(B, Table)) = \infty$
$h(\neg on(B, Table)) = \infty$
$h(on(B, C)) = \infty$
$h(on(C, Table)) = \infty$
$h(on(C, A)) = \infty$
$h(\neg on(C, A)) = \infty$
$h(clear(A)) = \infty$
$h(clear(B)) = \infty$
$h(\neg clear(B)) = \infty$
$h(clear(C)) = \infty$
$h(\neg clear(C)) = \infty$
$h(clear(Table)) = \infty$



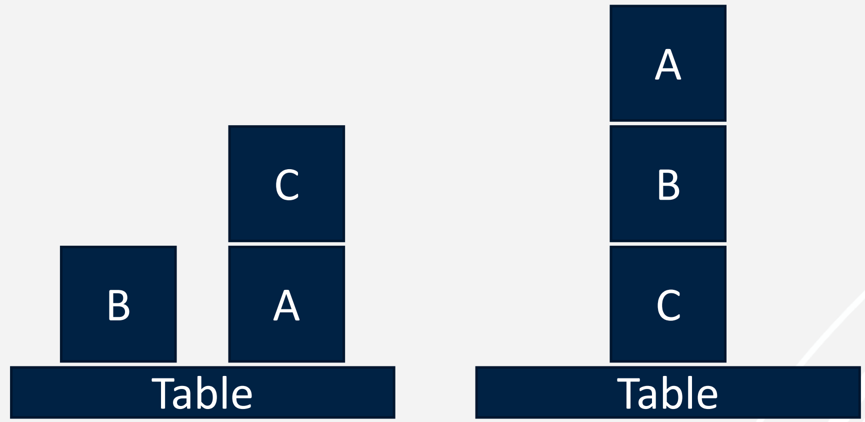
Initial State   Goal

Note: For the sake of a small example, we will only consider *some* of the literals and actions. When HSP computes it heuristic, it considers *all* literals and *all* actions.

HSP in Blocks World

Weights:

$h(on(A, Table)) = \infty$
$h(\neg on(A, Table)) = \infty$
$h(on(A, B)) = \infty$
$h(on(B, Table)) = \infty$
$h(\neg on(B, Table)) = \infty$
$h(on(B, C)) = \infty$
$h(on(C, Table)) = \infty$
$h(on(C, A)) = \infty$
$h(\neg on(C, A)) = \infty$
$h(clear(A)) = \infty$
$h(clear(B)) = \infty$
$h(\neg clear(B)) = \infty$
$h(clear(C)) = \infty$
$h(\neg clear(C)) = \infty$
$h(clear(Table)) = \infty$



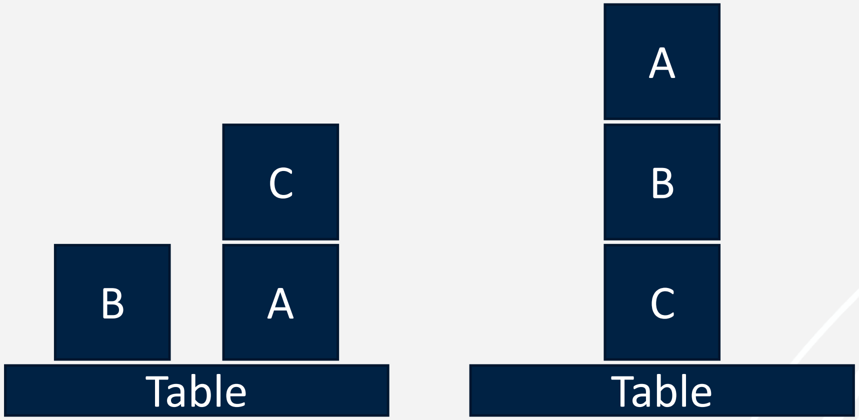
Initial State Goal

Start with the cost of every literal set to ∞.

HSP in Blocks World

Weights:

$h(on(A, Table)) = 0$
$h(\neg on(A, Table)) = \infty$
$h(on(A, B)) = \infty$
$h(on(B, Table)) = 0$
$h(\neg on(B, Table)) = \infty$
$h(on(B, C)) = \infty$
$h(on(C, Table)) = \infty$
$h(on(C, A)) = 0$
$h(\neg on(C, A)) = \infty$
$h(clear(A)) = \infty$
$h(clear(B)) = 0$
$h(\neg clear(B)) = \infty$
$h(clear(C)) = 0$
$h(\neg clear(C)) = \infty$
$h(clear(Table)) = \infty$



Initial State

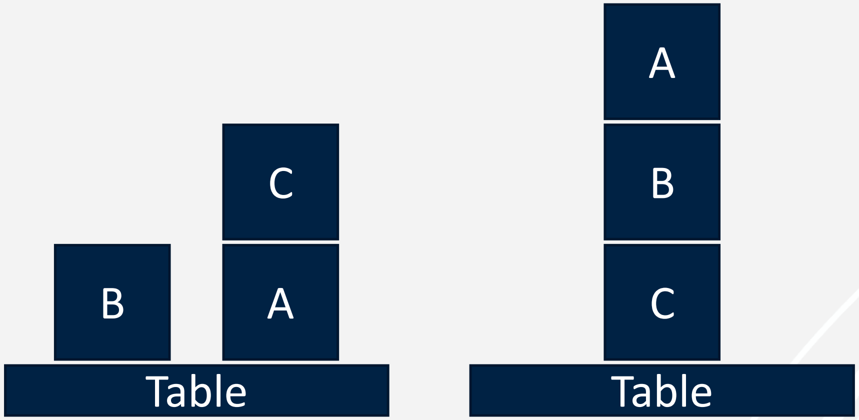
Goal

Set the cost of every literal that is true in the initial state to 0.

HSP in Blocks World

Weights:

$h(on(A, Table)) = 0$
$h(\neg on(A, Table)) = \infty$
$h(on(A, B)) = \infty$
$h(on(B, Table)) = 0$
$h(\neg on(B, Table)) = \infty$
$h(on(B, C)) = \infty$
$h(on(C, Table)) = \infty$
$h(on(C, A)) = 0$
$h(\neg on(C, A)) = \infty$
$h(clear(A)) = \infty$
$h(clear(B)) = 0$
$h(\neg clear(B)) = \infty$
$h(clear(C)) = 0$
$h(\neg clear(C)) = \infty$
$h(clear(Table)) = \infty$



Initial State

Goal

Update the cost of every effect of every step.

HSP in Blocks World

Weights:

$h(on(A, Table)) = 0$
$h(\neg on(A, Table)) = \infty$
$h(on(A, B)) = \infty$
$h(on(B, Table)) = 0$
$h(\neg on(B, Table)) = \infty$
$h(on(B, C)) = \infty$
$h(on(C, Table)) = \infty$
$h(on(C, A)) = 0$
$h(\neg on(C, A)) = \infty$
$h(clear(A)) = \infty$
$h(clear(B)) = 0$
$h(\neg clear(B)) = \infty$
$h(clear(C)) = 0$
$h(\neg clear(C)) = \infty$
$h(clear(Table)) = \infty$

$$on(A, Table) \wedge clear(A) \wedge clear(B)$$
$$move(A, Table, B)$$
$$on(A, B) \wedge \neg on(A, Table) \wedge clear(Table) \wedge \neg clear(B)$$

Cost of precondition:
$h(on(A, Table) \wedge clear(A) \wedge clear(B)) = ?$

HSP in Blocks World

Weights:

$h(on(A, Table)) = 0$

$h(\neg on(A, Table)) = \infty$

$h(on(A, B)) = \infty$

$h(on(B, Table)) = 0$

$h(\neg on(B, Table)) = \infty$

$h(on(B, C)) = \infty$

$h(on(C, Table)) = \infty$

$h(on(C, A)) = 0$

$h(\neg on(C, A)) = \infty$

$h(clear(A)) = \infty$

$h(clear(B)) = 0$

$h(\neg clear(B)) = \infty$

$h(clear(C)) = 0$

$h(\neg clear(C)) = \infty$

$h(clear(Table)) = \infty$

$$on(A, Table) \wedge clear(A) \wedge clear(B)$$

$$move(A, Table, B)$$

$$on(A, B) \wedge \neg on(A, Table) \wedge clear(Table) \wedge \neg clear(B)$$

Cost of precondition:

$h(on(A, Table) \wedge clear(A) \wedge clear(B)) = \infty$

HSP in Blocks World

Weights:

$h(on(A, Table)) = 0$
$h(\neg on(A, Table)) = \infty$
$h(on(A, B)) = \infty$
$h(on(B, Table)) = 0$
$h(\neg on(B, Table)) = \infty$
$h(on(B, C)) = \infty$
$h(on(C, Table)) = \infty$
$h(on(C, A)) = 0$
$h(\neg on(C, A)) = \infty$
$h(clear(A)) = \infty$
$h(clear(B)) = 0$
$h(\neg clear(B)) = \infty$
$h(clear(C)) = 0$
$h(\neg clear(C)) = \infty$
$h(clear(Table)) = \infty$

$$on(A, Table) \wedge clear(A) \wedge clear(B)$$
$$move(A, Table, B)$$
$$on(A, B) \wedge \neg on(A, Table) \wedge clear(Table) \wedge \neg clear(B)$$

Cost of precondition:
$h(on(A, Table) \wedge clear(A) \wedge clear(B)) = \infty$

Set $h(on(A, B)) = \min(\infty, \infty + 1)$

Set $h(\neg on(A, Table)) = \min(\infty, \infty + 1)$

Set $h(clear(Table)) = \min(\infty, \infty + 1)$

Set $h(\neg clear(B)) = \min(\infty, \infty + 1)$

HSP in Blocks World

Weights:

$h(on(A, Table)) = 0$
$h(\neg on(A, Table)) = \infty$
$h(on(A, B)) = \infty$
$h(on(B, Table)) = 0$
$h(\neg on(B, Table)) = \infty$
$h(on(B, C)) = \infty$
$h(on(C, Table)) = \infty$
$h(on(C, A)) = 0$
$h(\neg on(C, A)) = \infty$
$h(clear(A)) = \infty$
$h(clear(B)) = 0$
$h(\neg clear(B)) = \infty$
$h(clear(C)) = 0$
$h(\neg clear(C)) = \infty$
$h(clear(Table)) = \infty$

$$on(C, A) \land clear(C)$$
$$moveToTable(C, A)$$
$$on(C, Table) \land \neg on(C, A) \land clear(A)$$

Cost of precondition:
$h(on(C, A) \land clear(C)) = ?$

HSP in Blocks World

Weights:

$h(on(A, Table)) = 0$
$h(\neg on(A, Table)) = \infty$
$h(on(A, B)) = \infty$
$h(on(B, Table)) = 0$
$h(\neg on(B, Table)) = \infty$
$h(on(B, C)) = \infty$
$h(on(C, Table)) = \infty$
$h(on(C, A)) = 0$
$h(\neg on(C, A)) = \infty$
$h(clear(A)) = \infty$
$h(clear(B)) = 0$
$h(\neg clear(B)) = \infty$
$h(clear(C)) = 0$
$h(\neg clear(C)) = \infty$
$h(clear(Table)) = \infty$

$on(C, A) \wedge clear(C)$
$moveToTable(C, A)$
$on(C, Table) \wedge \neg on(C, A) \wedge clear(A)$

Cost of precondition:
$h(on(C, A) \wedge clear(C)) = 0$

Set $h(on(C, Table)) = \min(\infty, 0 + 1)$

Set $h(\neg on(C, A)) = \min(\infty, 0 + 1)$

Set $h(clear(A)) = \min(\infty, 0 + 1)$

HSP in Blocks World

Weights:

$h(on(A, Table)) = 0$
$h(\neg on(A, Table)) = \infty$
$h(on(A, B)) = \infty$
$h(on(B, Table)) = 0$
$h(\neg on(B, Table)) = \infty$
$h(on(B, C)) = \infty$
$h(on(C, Table)) = 1$
$h(on(C, A)) = 0$
$h(\neg on(C, A)) = 1$
$h(clear(A)) = 1$
$h(clear(B)) = 0$
$h(\neg clear(B)) = \infty$
$h(clear(C)) = 0$
$h(\neg clear(C)) = \infty$
$h(clear(Table)) = \infty$

$$on(C, A) \wedge clear(C)$$
$$moveToTable(C, A)$$
$$on(C, Table) \wedge \neg on(C, A) \wedge clear(A)$$

Cost of precondition:
$h(on(C, A) \wedge clear(C)) = 0$

Set $h(on(C, Table)) = \min(\infty, 0 + 1)$

Set $h(\neg on(C, A)) = \min(\infty, 0 + 1)$

Set $h(clear(A)) = \min(\infty, 0 + 1)$

HSP in Blocks World

Weights:

$h\big(on(A,Table)\big) = 0$
$h\big(\neg on(A,Table)\big) = \infty$
$h\big(on(A,B)\big) = \infty$
$h\big(on(B,Table)\big) = 0$
$h\big(\neg on(B,Table)\big) = \infty$
$h\big(on(B,C)\big) = \infty$
$h\big(on(C,Table)\big) = 1$
$h\big(on(C,A)\big) = 0$
$h\big(\neg on(C,A)\big) = 1$
$h\big(clear(A)\big) = 1$
$h\big(clear(B)\big) = 0$
$h\big(\neg clear(B)\big) = \infty$
$h\big(clear(C)\big) = 0$
$h\big(\neg clear(C)\big) = \infty$
$h\big(clear(Table)\big) = \infty$

$on(B,Table) \land clear(B) \land clear(C)$
$move(B,Table,C)$
$on(B,C) \land \neg on(B,Table) \land clear(Table) \land \neg clear(C)$

Cost of precondition:
$h(on(B,Table) \land clear(B) \land clear(C)) = ?$

HSP in Blocks World

Weights:

$h(on(A, Table)) = 0$
$h(\neg on(A, Table)) = \infty$
$h(on(A, B)) = \infty$
$h(on(B, Table)) = 0$
$h(\neg on(B, Table)) = \infty$
$h(on(B, C)) = \infty$
$h(on(C, Table)) = 1$
$h(on(C, A)) = 0$
$h(\neg on(C, A)) = 1$
$h(clear(A)) = 1$
$h(clear(B)) = 0$
$h(\neg clear(B)) = \infty$
$h(clear(C)) = 0$
$h(\neg clear(C)) = \infty$
$h(clear(Table)) = \infty$

$$\boxed{\begin{array}{c} on(B, Table) \wedge clear(B) \wedge clear(C) \\ move(B, Table, C) \\ on(B, C) \wedge \neg on(B, Table) \wedge clear(Table) \wedge \neg clear(C) \end{array}}$$

Cost of precondition:
$h(on(B, Table) \wedge clear(B) \wedge clear(C)) = 0$

Set $h(on(B, C)) = \min(\infty, 0 + 1)$

Set $h(\neg on(B, Table)) = \min(\infty, 0 + 1)$

Set $h(clear(Table)) = \min(\infty, 0 + 1)$

Set $h(\neg clear(C)) = \min(\infty, 0 + 1)$

HSP in Blocks World

Weights:

$h(on(A, Table)) = 0$
$h(\neg on(A, Table)) = \infty$
$h(on(A, B)) = \infty$
$h(on(B, Table)) = 0$
$h(\neg on(B, Table)) = 1$
$h(on(B, C)) = 1$
$h(on(C, Table)) = 1$
$h(on(C, A)) = 0$
$h(\neg on(C, A)) = 1$
$h(clear(A)) = 1$
$h(clear(B)) = 0$
$h(\neg clear(B)) = \infty$
$h(clear(C)) = 0$
$h(\neg clear(C)) = 1$
$h(clear(Table)) = 1$

$$on(B, Table) \wedge clear(B) \wedge clear(C)$$
$$move(B, Table, C)$$
$$on(B, C) \wedge \neg on(B, Table) \wedge clear(Table) \wedge \neg clear(C)$$

Cost of precondition:
$h(on(B, Table) \wedge clear(B) \wedge clear(C)) = 0$

Set $h(on(B, C)) = \min(\infty, 0 + 1)$

Set $h(\neg on(B, Table)) = \min(\infty, 0 + 1)$

Set $h(clear(Table)) = \min(\infty, 0 + 1)$

Set $h(\neg clear(C)) = \min(\infty, 0 + 1)$

HSP in Blocks World

Weights:

$h(on(A, Table)) = 0$
$h(\neg on(A, Table)) = \infty$
$h(on(A, B)) = \infty$
$h(on(B, Table)) = 0$
$h(\neg on(B, Table)) = 1$
$h(on(B, C)) = 1$
$h(on(C, Table)) = 1$
$h(on(C, A)) = 0$
$h(\neg on(C, A)) = 1$
$h(clear(A)) = 1$
$h(clear(B)) = 0$
$h(\neg clear(B)) = \infty$
$h(clear(C)) = 0$
$h(\neg clear(C)) = 1$
$h(clear(Table)) = 1$

Some costs changed, so we need to do another round of updates.

HSP in Blocks World

Weights:

$h\big(on(A,Table)\big) = 0$
$h\big(\neg on(A,Table)\big) = \infty$
$h\big(on(A,B)\big) = \infty$
$h\big(on(B,Table)\big) = 0$
$h\big(\neg on(B,Table)\big) = 1$
$h\big(on(B,C)\big) = 1$
$h\big(on(C,Table)\big) = 1$
$h\big(on(C,A)\big) = 0$
$h\big(\neg on(C,A)\big) = 1$
$h\big(clear(A)\big) = 1$
$h\big(clear(B)\big) = 0$
$h\big(\neg clear(B)\big) = \infty$
$h\big(clear(C)\big) = 0$
$h\big(\neg clear(C)\big) = 1$
$h\big(clear(Table)\big) = 1$

$$on(A,Table) \wedge clear(A) \wedge clear(B)$$
$$move(A,Table,B)$$
$$on(A,B) \wedge \neg on(A,Table) \wedge clear(Table) \wedge \neg clear(B)$$

Cost of precondition:
$h\big(on(A,Table) \wedge clear(A) \wedge clear(B)\big) = ?$

HSP in Blocks World

Weights:

$h(on(A,Table)) = 0$
$h(\neg on(A,Table)) = \infty$
$h(on(A,B)) = \infty$
$h(on(B,Table)) = 0$
$h(\neg on(B,Table)) = 1$
$h(on(B,C)) = 1$
$h(on(C,Table)) = 1$
$h(on(C,A)) = 0$
$h(\neg on(C,A)) = 1$
$h(clear(A)) = 1$
$h(clear(B)) = 0$
$h(\neg clear(B)) = \infty$
$h(clear(C)) = 0$
$h(\neg clear(C)) = 1$
$h(clear(Table)) = 1$

$$on(A,Table) \wedge clear(A) \wedge clear(B)$$
$$move(A,Table,B)$$
$$on(A,B) \wedge \neg on(A,Table) \wedge clear(Table) \wedge \neg clear(B)$$

Cost of precondition:
$h(on(A,Table) \wedge clear(A) \wedge clear(B)) = 1$

Set $h(on(A,B)) = \min(\infty, 1 + 1)$

Set $h(\neg on(A,Table)) = \min(\infty, 1 + 1)$

Set $h(clear(Table)) = \min(1,1 + 1)$

Set $h(\neg clear(B)) = \min(\infty, 1 + 1)$

HSP in Blocks World

Weights:

$h(on(A, Table)) = 0$

$h(\neg on(A, Table)) = 2$

$h(on(A, B)) = 2$

$h(on(B, Table)) = 0$

$h(\neg on(B, Table)) = 1$

$h(on(B, C)) = 1$

$h(on(C, Table)) = 1$

$h(on(C, A)) = 0$

$h(\neg on(C, A)) = 1$

$h(clear(A)) = 1$

$h(clear(B)) = 0$

$h(\neg clear(B)) = 2$

$h(clear(C)) = 0$

$h(\neg clear(C)) = 1$

$h(clear(Table)) = 1$

$$on(A, Table) \wedge clear(A) \wedge clear(B)$$
$$move(A, Table, B)$$
$$on(A, B) \wedge \neg on(A, Table) \wedge clear(Table) \wedge \neg clear(B)$$

Cost of precondition:

$h(on(A, Table) \wedge clear(A) \wedge clear(B)) = 1$

Set $h(on(A, B)) = \min(\infty, 1 + 1)$

Set $h(\neg on(A, Table)) = \min(\infty, 1 + 1)$

Set $h(clear(Table)) = \min(1, 1 + 1)$

Set $h(\neg clear(B)) = \min(\infty, 1 + 1)$

HSP in Blocks World

Weights:

$h(on(A, Table)) = 0$

$h(\neg on(A, Table)) = 2$

$h(on(A, B)) = 2$

$h(on(B, Table)) = 0$

$h(\neg on(B, Table)) = 1$

$h(on(B, C)) = 1$

$h(on(C, Table)) = 1$

$h(on(C, A)) = 0$

$h(\neg on(C, A)) = 1$

$h(clear(A)) = 1$

$h(clear(B)) = 0$

$h(\neg clear(B)) = 2$

$h(clear(C)) = 0$

$h(\neg clear(C)) = 1$

$h(clear(Table)) = 1$

$$on(C, A) \land clear(C)$$
$$moveToTable(C, A)$$
$$on(C, Table) \land \neg on(C, A) \land clear(A)$$

Cost of precondition:

$h(on(C, A) \land clear(C)) = 0$

Set $h(on(C, Table)) = \min(1, 0 + 1)$

Set $h(\neg on(C, A)) = \min(1, 0 + 1)$

Set $h(clear(A)) = \min(1, 0 + 1)$

NIL

HSP in Blocks World

Weights:

$h\big(on(A,Table)\big) = 0$

$h\big(\neg on(A,Table)\big) = 2$

$h\big(on(A,B)\big) = 2$

$h\big(on(B,Table)\big) = 0$

$h\big(\neg on(B,Table)\big) = 1$

$h\big(on(B,C)\big) = 1$

$h\big(on(C,Table)\big) = 1$

$h\big(on(C,A)\big) = 0$

$h\big(\neg on(C,A)\big) = 1$

$h\big(clear(A)\big) = 1$

$h\big(clear(B)\big) = 0$

$h\big(\neg clear(B)\big) = 2$

$h\big(clear(C)\big) = 0$

$h\big(\neg clear(C)\big) = 1$

$h\big(clear(Table)\big) = 1$

$$on(B,Table) \wedge clear(B) \wedge clear(C)$$
$$move(B,Table,C)$$
$$on(B,C) \wedge \neg on(B,Table) \wedge clear(Table) \wedge \neg clear(C)$$

Cost of precondition:

$h(on(B,Table) \wedge clear(B) \wedge clear(C)) = 0$

Set $h\big(on(B,C)\big) = \min(1,0+1)$

Set $h\big(\neg on(B,Table)\big) = \min(1,0+1)$

Set $h\big(clear(Table)\big) = \min(1,0+1)$

Set $h\big(\neg clear(C)\big) = \min(1,0+1)$

HSP in Blocks World

Weights:

$h(on(A, Table)) = 0$
$h(\neg on(A, Table)) = 2$
$h(on(A, B)) = 2$
$h(on(B, Table)) = 0$
$h(\neg on(B, Table)) = 1$
$h(on(B, C)) = 1$
$h(on(C, Table)) = 1$
$h(on(C, A)) = 0$
$h(\neg on(C, A)) = 1$
$h(clear(A)) = 1$
$h(clear(B)) = 0$
$h(\neg clear(B)) = 2$
$h(clear(C)) = 0$
$h(\neg clear(C)) = 1$
$h(clear(Table)) = 1$

Some costs changed, so we need to do another round of updates.

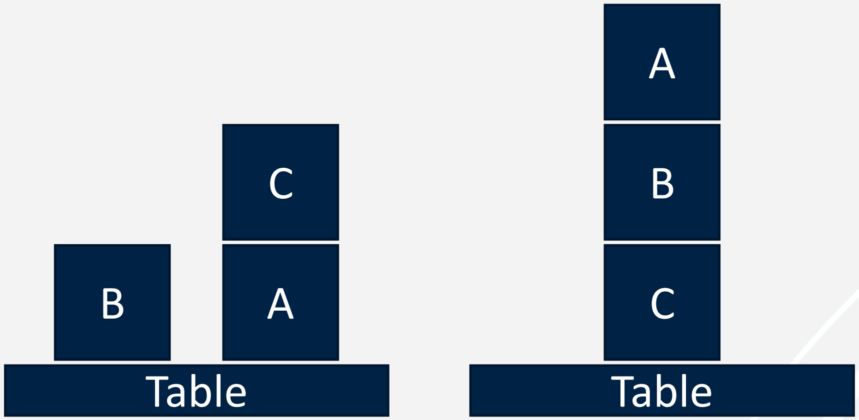
Spoiler: Nothing will change this round.

After a round where no weights change, we are done.

HSP in Blocks World

Weights:

$h(on(A, Table)) = 0$
$h(\neg on(A, Table)) = 2$
$h(on(A, B)) = 2$
$h(on(B, Table)) = 0$
$h(\neg on(B, Table)) = 1$
$h(on(B, C)) = 1$
$h(on(C, Table)) = 1$
$h(on(C, A)) = 0$
$h(\neg on(C, A)) = 1$
$h(clear(A)) = 1$
$h(clear(B)) = 0$
$h(\neg clear(B)) = 2$
$h(clear(C)) = 0$
$h(\neg clear(C)) = 1$
$h(clear(Table)) = 1$



Initial State Goal

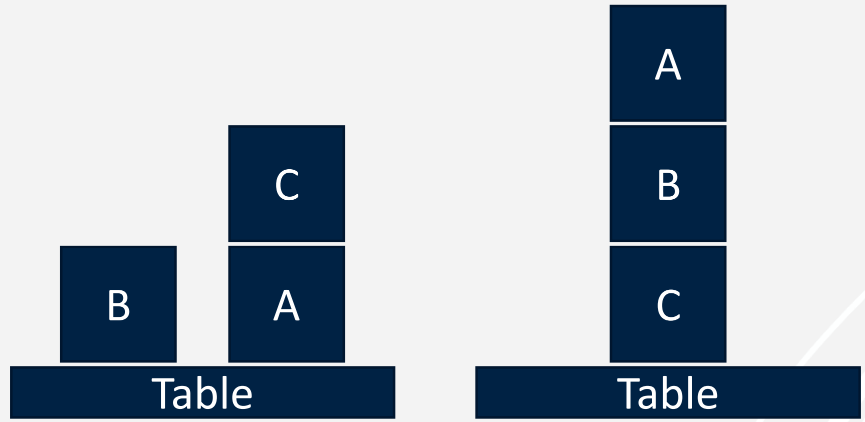
What is the estimated cost of the goal?

$$h(on(A, B) \wedge on(B, C)) = ?$$

HSP in Blocks World

Weights:

$h(on(A, Table)) = 0$
$h(\neg on(A, Table)) = 2$
$h(on(A, B)) = 2$
$h(on(B, Table)) = 0$
$h(\neg on(B, Table)) = 1$
$h(on(B, C)) = 1$
$h(on(C, Table)) = 1$
$h(on(C, A)) = 0$
$h(\neg on(C, A)) = 1$
$h(clear(A)) = 1$
$h(clear(B)) = 0$
$h(\neg clear(B)) = 2$
$h(clear(C)) = 0$
$h(\neg clear(C)) = 1$
$h(clear(Table)) = 1$



Initial State   Goal

What is the estimated cost of the goal?

$h(on(A, B) \wedge on(B, C)) = 2 + 1 = 3$

HSP's Heuristic

Every literal has a cost, initially ∞.

Every literal that is true in the initial state has a cost of 0.

The cost of a conjunction is the sum of the costs of its conjuncts.

Do this until the costs of the literals stop changing:

- For every step S:
  - For every literal E in the effect of S:
    - Let the cost of E be the minimum of:
      1. The current cost of E.
      2. The cost of S's precondition + 1.

Planning Problems

During planning, goals can:

- **Interfere:** Progress toward one goal undoes progress toward another goal.
- **Synergize:** Progress toward one goal also makes progress toward another goal.

HSP’s Heuristic

- HSP does not account for interference.

(Sometimes called the “ignore delete list” assumption)

- HSP does not account for synergy.

(Because the cost of a goal is the sum of its parts)

- HSP may overestimate, and thus is not admissible.
- In practice, HSP was the first heuristic that was accurate enough to allow for state-space planning.
- Heuristic is efficient to compute.

Analyzing HSP's Heuristic

- The heuristic is based on a relaxed version of the planning problem that is much easier to solve but still provides a good approximation of the original.
- Relaxed problem: We can never get farther from a goal, only closer (i.e. ignore interference).
- The cost of achieving some literal is 1 + the cost of achieving the precondition of any step which has that literal as an effect.

Fast-Forward (FF)

- Created by Bernhard Nebel and Jörg Hoffmann
- Top performer in the second (2000) and third (2002) International Planning Competitions
- Observation: HSP's estimates are very similar to those obtained from a plan graph
- Idea: Use a plan graph to estimate the difficulty of a goal and to find a solution to the relaxed problem
- Benefit: Plan graphs account for synergy in goals

FF Heuristic

Given a current state S and a goal G:

- Construct a plan graph such that layer 0 is S.
- Extend the plan graph until all goals in G appear.
- (Do not use persistence steps or mutexes.)
- Extract a solution as graphplan does.
- Return the size of the resulting plan.

Cargo Problem

Initial State: $at(C1, ATL) \wedge at(C2, ATL) \wedge at(P1, ATL)$

Goal: $at(C1, MSY) \wedge at(C2, MSY)$

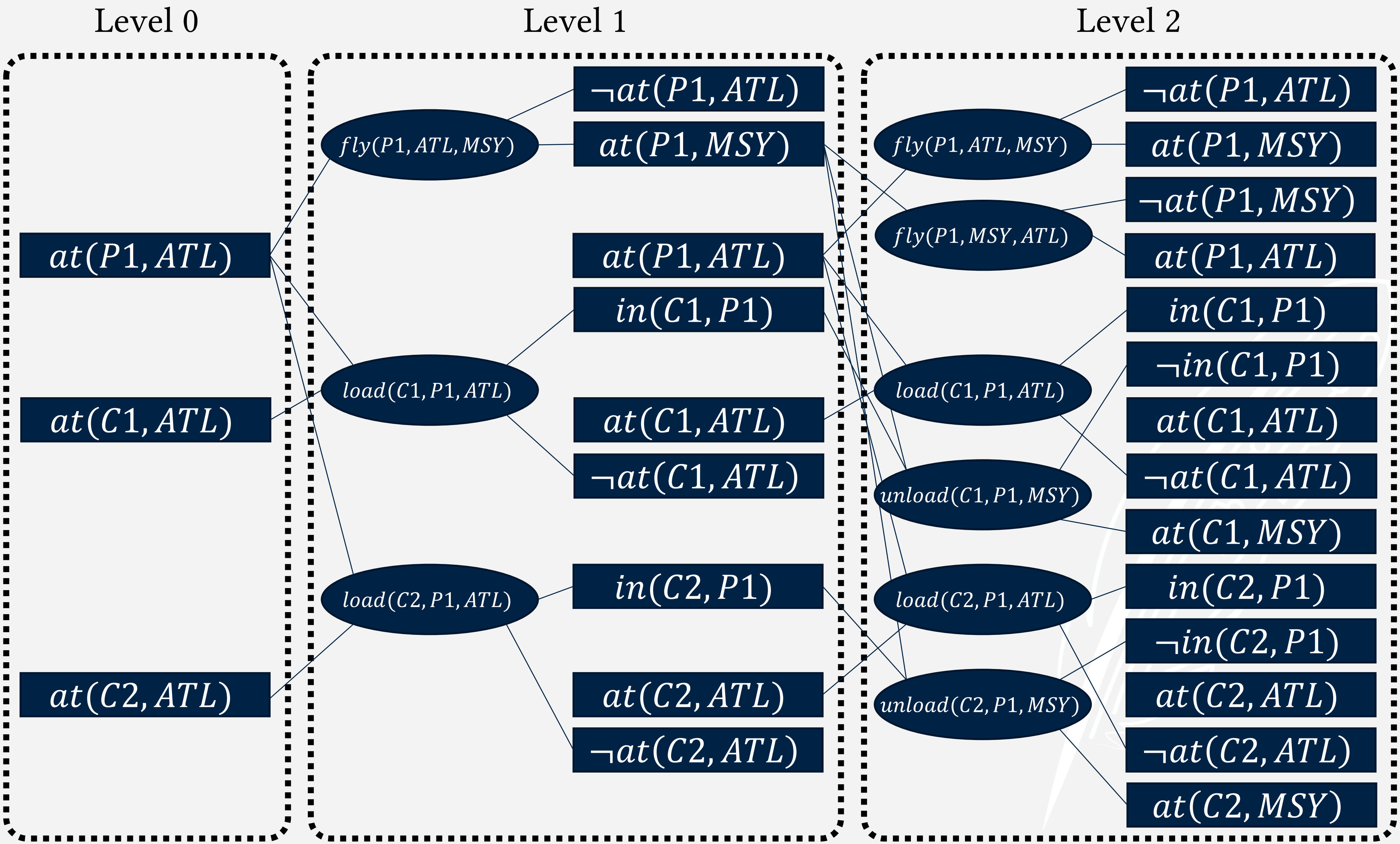
Plan:

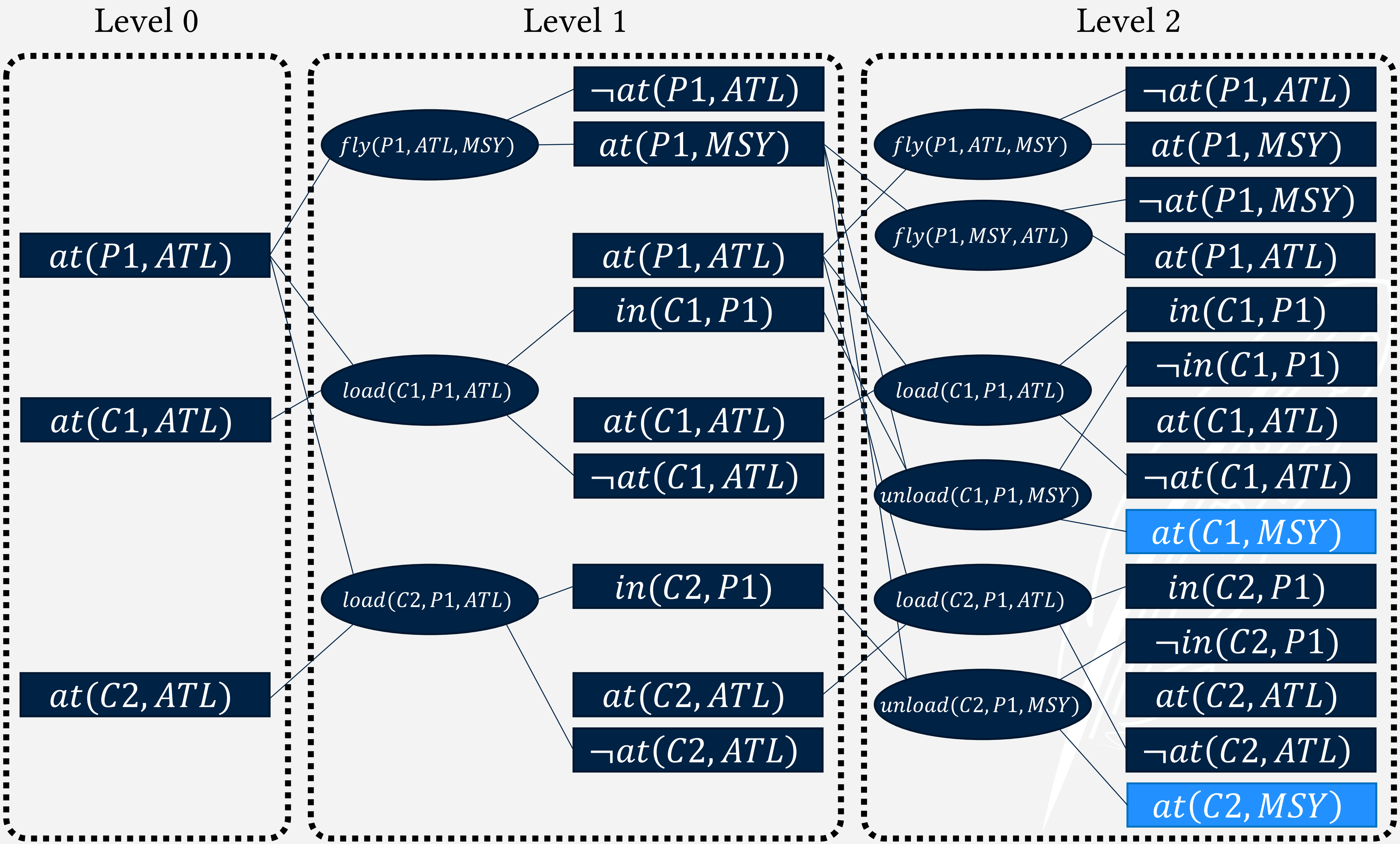
1. $load(C1, P1, ATL)$
2. $load(C2, P1, ATL)$
3. $fly(P1, ATL, MSY)$
4. $unload(C1, P1, MSY)$
5. $unload(C2, P1, MSY)$

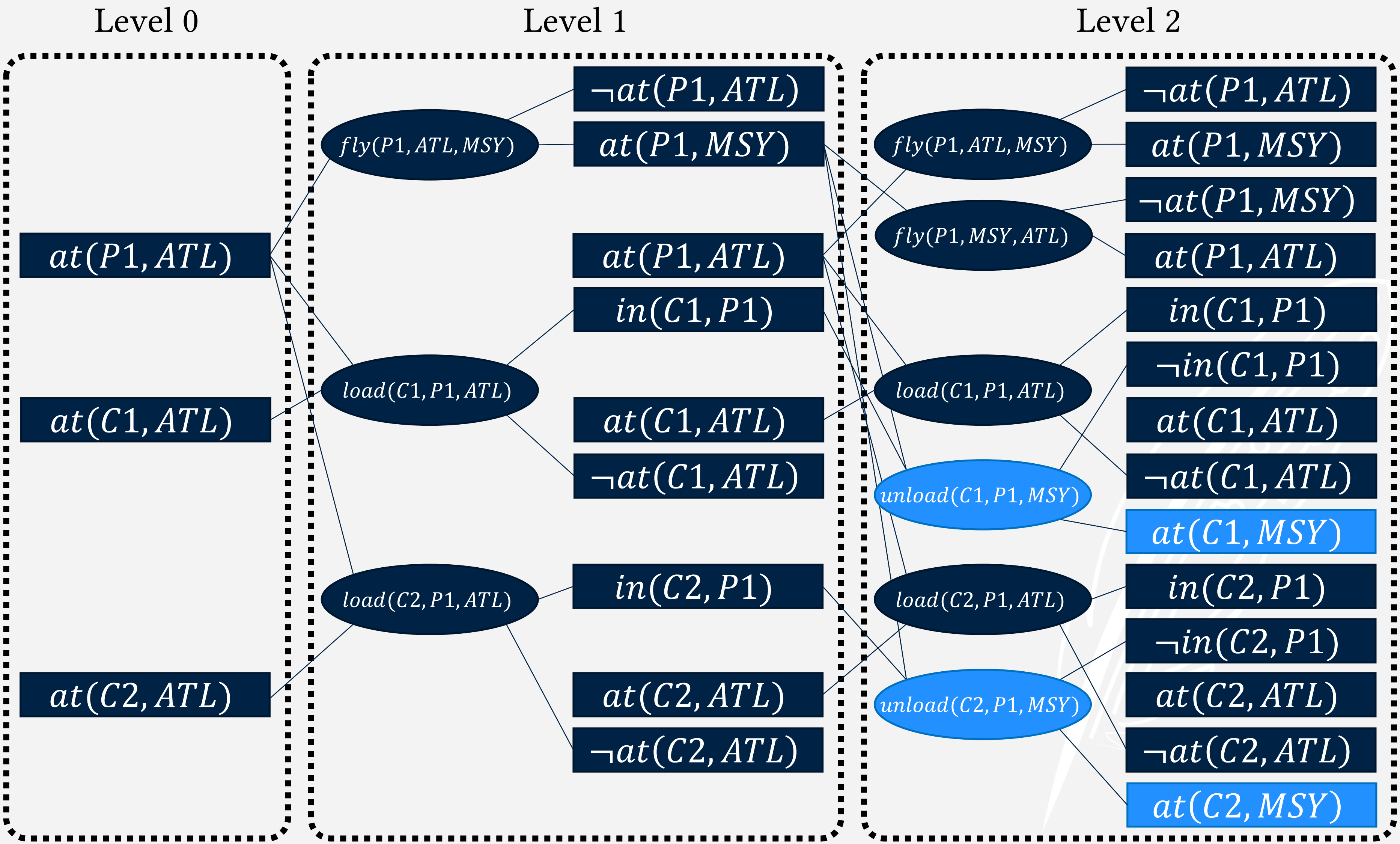
HSP Estimate: ? steps

Plan Graph Estimate: ? steps

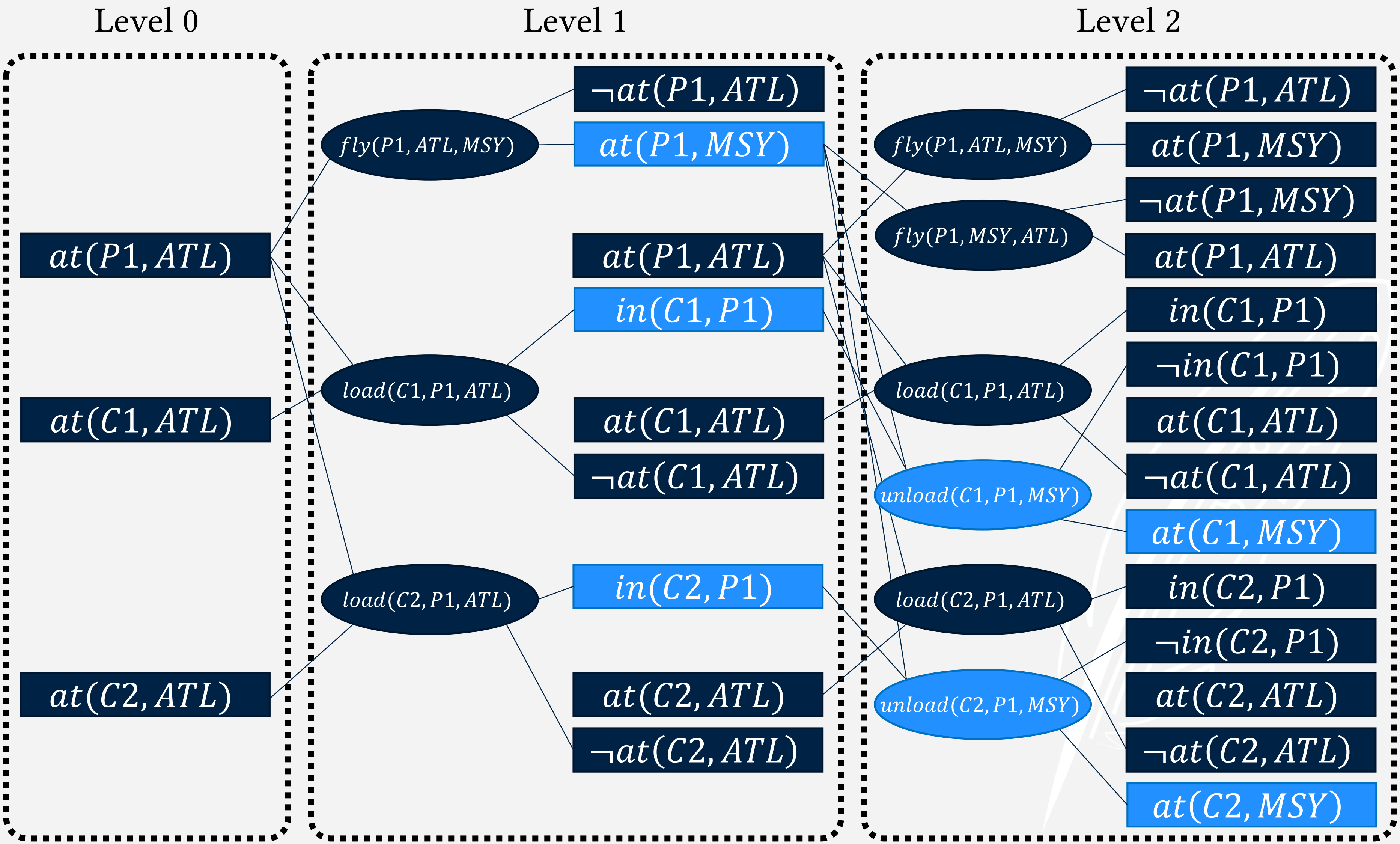
FF Estimate: ? steps

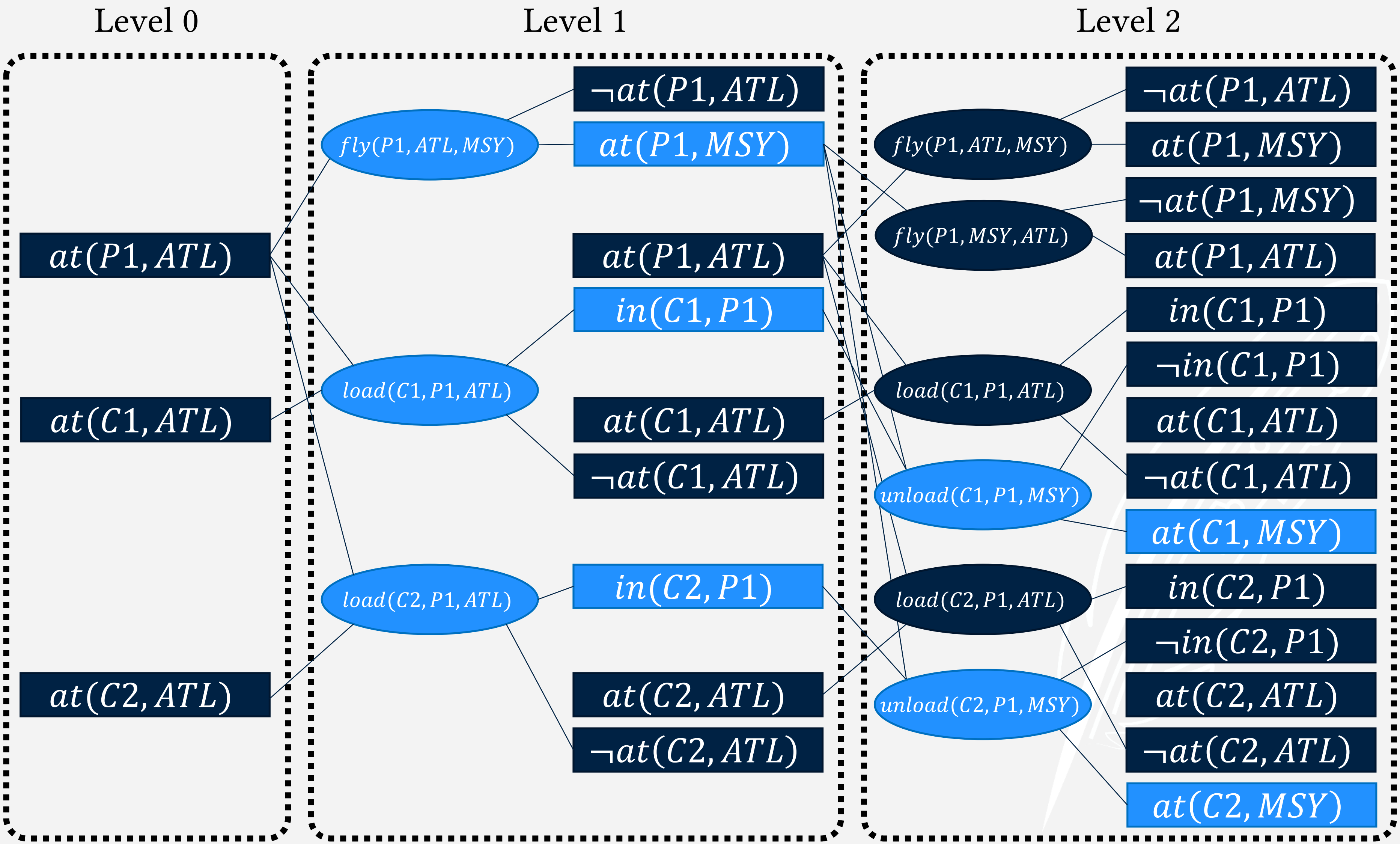

Level 0
Level 1
Level 2
at(P1, ATL)
at(C1, ATL)
at(C2, ATL)
fly(P1, ATL, MSY)
load(C1, P1, ATL)
load(C2, P1, ATL)
¬at(P1, ATL)
at(P1, MSY)
at(P1, ATL)
in(C1, P1)
at(C1, ATL)
¬at(C1, ATL)
in(C2, P1)
at(C2, ATL)
¬at(C2, ATL)
fly(P1, ATL, MSY)
fly(P1, MSY, ATL)
load(C1, P1, ATL)
unload(C1, P1, MSY)
load(C2, P1, ATL)
unload(C2, P1, MSY)
¬at(P1, ATL)
at(P1, MSY)
¬at(P1, MSY)
at(P1, ATL)
in(C1, P1)
¬in(C1, P1)
at(C1, ATL)
¬at(C1, ATL)
at(C1, MSY)
in(C2, P1)
¬in(C2, P1)
at(C2, ATL)
¬at(C2, ATL)
at(C2, MSY)

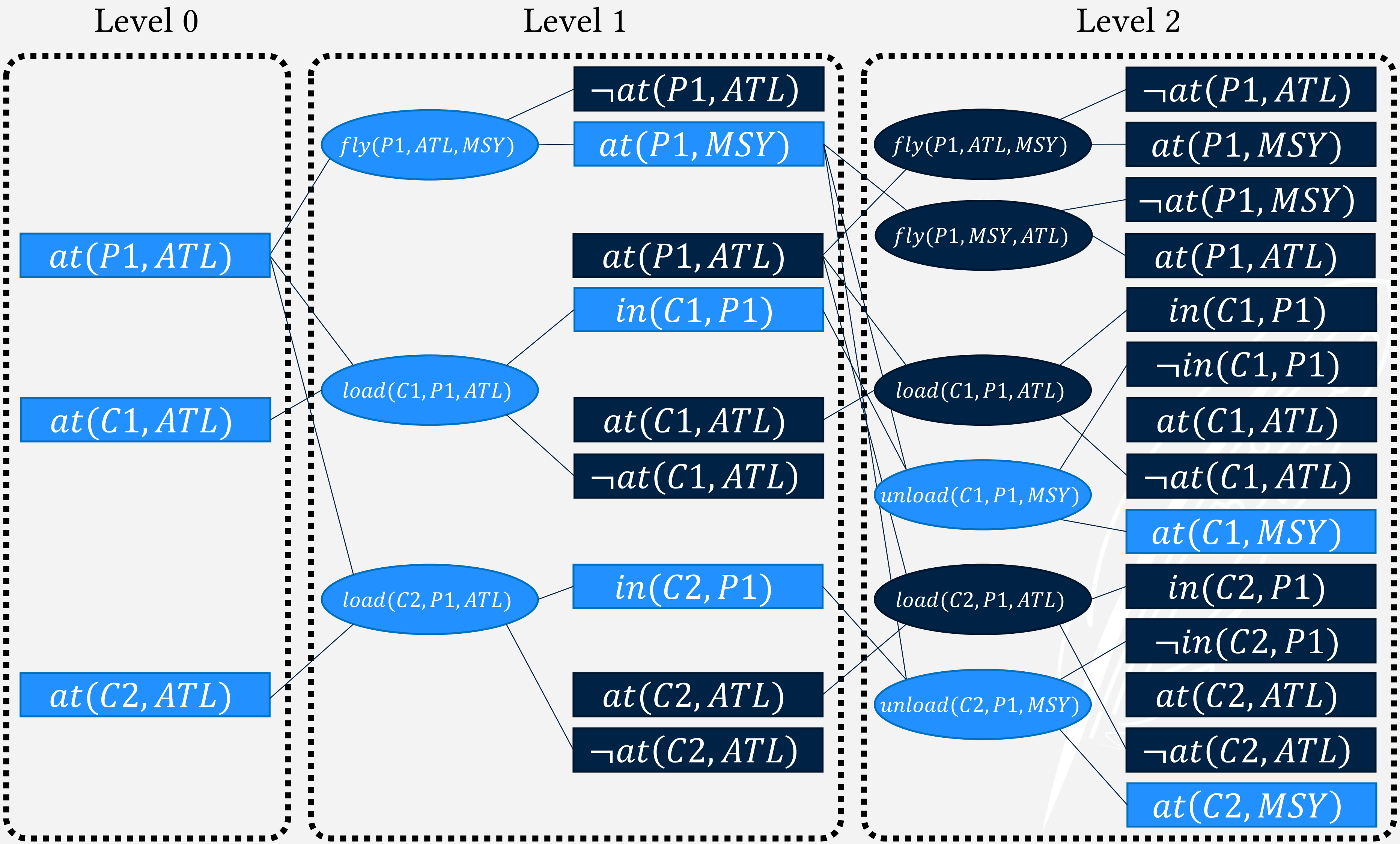

Level 0
Level 1
Level 2
at(P1, ATL)
at(C1, ATL)
at(C2, ATL)
fly(P1, ATL, MSY)
load(C1, P1, ATL)
load(C2, P1, ATL)
¬at(P1, ATL)
at(P1, MSY)
at(P1, ATL)
in(C1, P1)
at(C1, ATL)
¬at(C1, ATL)
in(C2, P1)
at(C2, ATL)
¬at(C2, ATL)
fly(P1, ATL, MSY)
fly(P1, MSY, ATL)
load(C1, P1, ATL)
unload(C1, P1, MSY)
load(C2, P1, ATL)
unload(C2, P1, MSY)
¬at(P1, ATL)
at(P1, MSY)
¬at(P1, MSY)
at(P1, ATL)
in(C1, P1)
¬in(C1, P1)
at(C1, ATL)
¬at(C1, ATL)
at(C1, MSY)
in(C2, P1)
¬in(C2, P1)
at(C2, ATL)
¬at(C2, ATL)
at(C2, MSY)



Level 0
Level 1
Level 2
at(P1, ATL)
at(C1, ATL)
at(C2, ATL)
fly(P1, ATL, MSY)
load(C1, P1, ATL)
load(C2, P1, ATL)
¬at(P1, ATL)
at(P1, MSY)
at(P1, ATL)
in(C1, P1)
at(C1, ATL)
¬at(C1, ATL)
in(C2, P1)
at(C2, ATL)
¬at(C2, ATL)
fly(P1, ATL, MSY)
fly(P1, MSY, ATL)
load(C1, P1, ATL)
unload(C1, P1, MSY)
load(C2, P1, ATL)
unload(C2, P1, MSY)
¬at(P1, ATL)
at(P1, MSY)
¬at(P1, MSY)
at(P1, ATL)
in(C1, P1)
¬in(C1, P1)
at(C1, ATL)
¬at(C1, ATL)
at(C1, MSY)
in(C2, P1)
¬in(C2, P1)
at(C2, ATL)
¬at(C2, ATL)
at(C2, MSY)


Level 0
Level 1
Level 2
at(P1, ATL)
at(C1, ATL)
at(C2, ATL)
fly(P1, ATL, MSY)
load(C1, P1, ATL)
load(C2, P1, ATL)
¬at(P1, ATL)
at(P1, MSY)
at(P1, ATL)
in(C1, P1)
at(C1, ATL)
¬at(C1, ATL)
in(C2, P1)
at(C2, ATL)
¬at(C2, ATL)
fly(P1, ATL, MSY)
fly(P1, MSY, ATL)
load(C1, P1, ATL)
unload(C1, P1, MSY)
load(C2, P1, ATL)
unload(C2, P1, MSY)
¬at(P1, ATL)
at(P1, MSY)
¬at(P1, MSY)
at(P1, ATL)
in(C1, P1)
¬in(C1, P1)
at(C1, ATL)
¬at(C1, ATL)
at(C1, MSY)
in(C2, P1)
¬in(C2, P1)
at(C2, ATL)
¬at(C2, ATL)
at(C2, MSY)


Level 0
Level 1
Level 2
at(P1, ATL)
at(C1, ATL)
at(C2, ATL)
fly(P1, ATL, MSY)
load(C1, P1, ATL)
load(C2, P1, ATL)
¬at(P1, ATL)
at(P1, MSY)
at(P1, ATL)
in(C1, P1)
at(C1, ATL)
¬at(C1, ATL)
in(C2, P1)
at(C2, ATL)
¬at(C2, ATL)
fly(P1, ATL, MSY)
fly(P1, MSY, ATL)
load(C1, P1, ATL)
unload(C1, P1, MSY)
load(C2, P1, ATL)
unload(C2, P1, MSY)
¬at(P1, ATL)
at(P1, MSY)
¬at(P1, MSY)
at(P1, ATL)
in(C1, P1)
¬in(C1, P1)
at(C1, ATL)
¬at(C1, ATL)
at(C1, MSY)
in(C2, P1)
¬in(C2, P1)
at(C2, ATL)
¬at(C2, ATL)
at(C2, MSY)



Level 0
Level 1
Level 2
at(P1, ATL)
at(C1, ATL)
at(C2, ATL)
fly(P1, ATL, MSY)
load(C1, P1, ATL)
load(C2, P1, ATL)
¬at(P1, ATL)
at(P1, MSY)
at(P1, ATL)
in(C1, P1)
at(C1, ATL)
¬at(C1, ATL)
in(C2, P1)
at(C2, ATL)
¬at(C2, ATL)
fly(P1, ATL, MSY)
fly(P1, MSY, ATL)
load(C1, P1, ATL)
unload(C1, P1, MSY)
load(C2, P1, ATL)
unload(C2, P1, MSY)
¬at(P1, ATL)
at(P1, MSY)
¬at(P1, MSY)
at(P1, ATL)
in(C1, P1)
¬in(C1, P1)
at(C1, ATL)
¬at(C1, ATL)
at(C1, MSY)
in(C2, P1)
¬in(C2, P1)
at(C2, ATL)
¬at(C2, ATL)
at(C2, MSY)

Cargo Problem

Initial State: $at(C1, ATL) \wedge at(C2, ATL) \wedge at(P1, ATL)$

Goal: $at(C1, MSY) \wedge at(C2, MSY)$

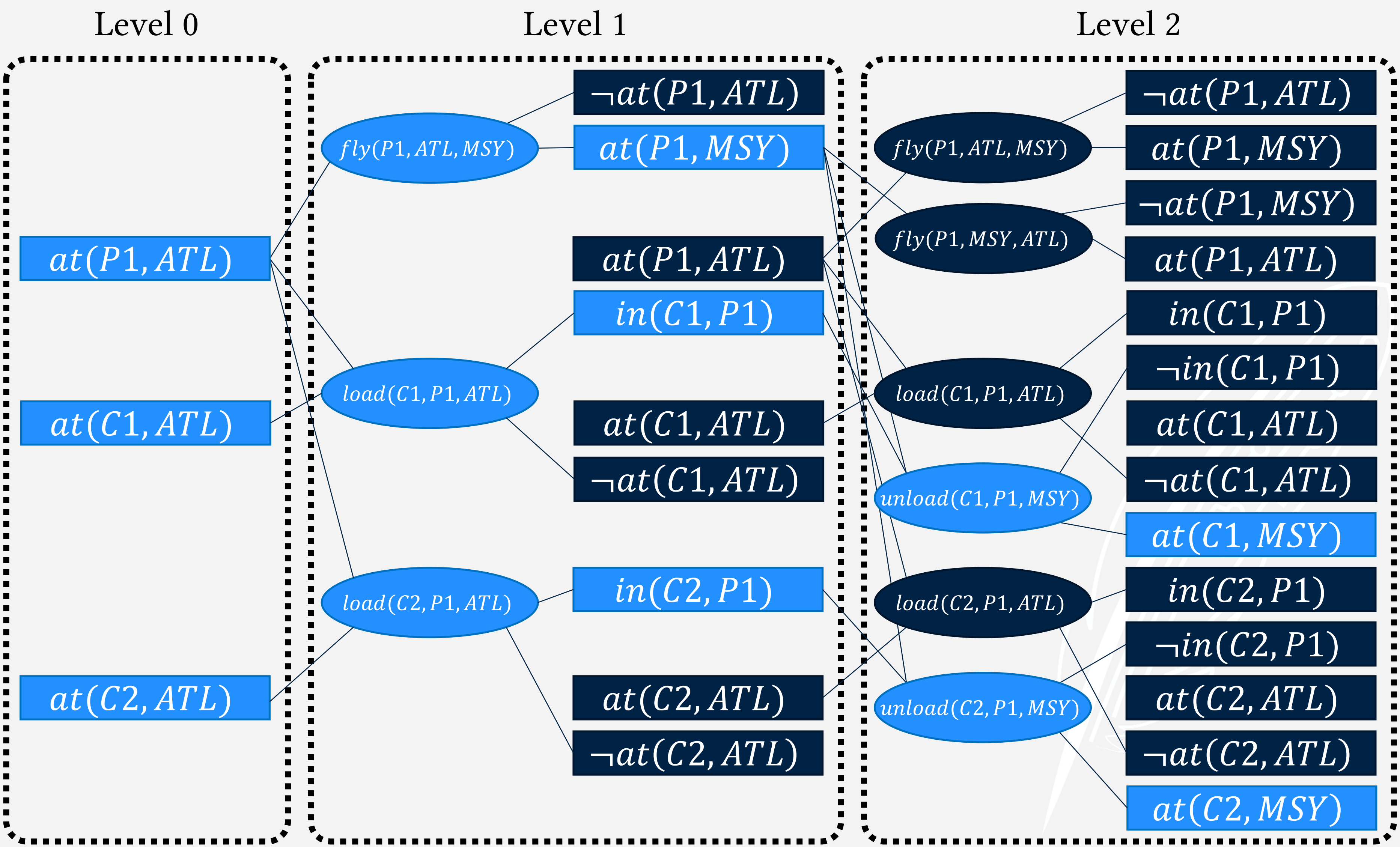
Plan:

1. $load(C1, P1, ATL)$
2. $load(C2, P1, ATL)$
3. $fly(P1, ATL, MSY)$
4. $unload(C1, P1, MSY)$
5. $unload(C2, P1, MSY)$

HSP Estimate: ? steps

Plan Graph Estimate: ? steps

FF Estimate: ? steps



Level 0
Level 1
Level 2
at(P1, ATL)
at(C1, ATL)
at(C2, ATL)
fly(P1, ATL, MSY)
load(C1, P1, ATL)
load(C2, P1, ATL)
¬at(P1, ATL)
at(P1, MSY)
at(P1, ATL)
in(C1, P1)
at(C1, ATL)
¬at(C1, ATL)
in(C2, P1)
at(C2, ATL)
¬at(C2, ATL)
fly(P1, ATL, MSY)
fly(P1, MSY, ATL)
load(C1, P1, ATL)
unload(C1, P1, MSY)
load(C2, P1, ATL)
unload(C2, P1, MSY)
¬at(P1, ATL)
at(P1, MSY)
¬at(P1, MSY)
at(P1, ATL)
in(C1, P1)
¬in(C1, P1)
at(C1, ATL)
¬at(C1, ATL)
at(C1, MSY)
in(C2, P1)
¬in(C2, P1)
at(C2, ATL)
¬at(C2, ATL)
at(C2, MSY)

Cargo Problem

Initial State: $at(C1, ATL) \land at(C2, ATL) \land at(P1, ATL)$

Goal: $at(C1, MSY) \land at(C2, MSY)$

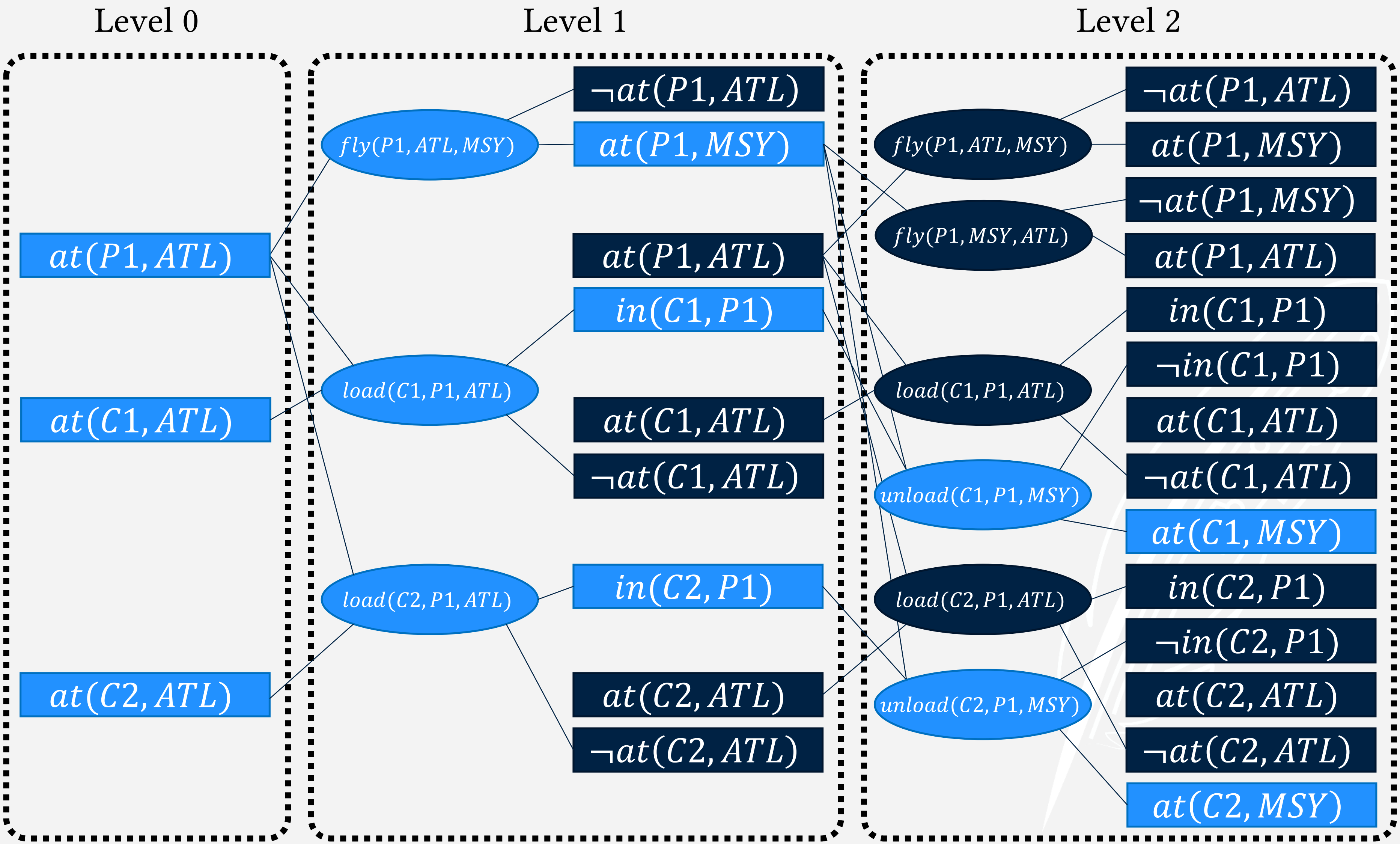
Plan:

1. $load(C1, P1, ATL)$
2. $load(C2, P1, ATL)$
3. $fly(P1, ATL, MSY)$
4. $unload(C1, P1, MSY)$
5. $unload(C2, P1, MSY)$

HSP Estimate: 6 steps

Plan Graph Estimate: ? steps

FF Estimate: ? steps


Level 0
Level 1
Level 2
at(P1, ATL)
at(C1, ATL)
at(C2, ATL)
fly(P1, ATL, MSY)
load(C1, P1, ATL)
load(C2, P1, ATL)
¬at(P1, ATL)
at(P1, MSY)
at(P1, ATL)
in(C1, P1)
at(C1, ATL)
¬at(C1, ATL)
in(C2, P1)
at(C2, ATL)
¬at(C2, ATL)
fly(P1, ATL, MSY)
fly(P1, MSY, ATL)
load(C1, P1, ATL)
unload(C1, P1, MSY)
load(C2, P1, ATL)
unload(C2, P1, MSY)
¬at(P1, ATL)
at(P1, MSY)
¬at(P1, MSY)
at(P1, ATL)
in(C1, P1)
¬in(C1, P1)
at(C1, ATL)
¬at(C1, ATL)
at(C1, MSY)
in(C2, P1)
¬in(C2, P1)
at(C2, ATL)
¬at(C2, ATL)
at(C2, MSY)

Cargo Problem

Initial State: $at(C1, ATL) \wedge at(C2, ATL) \wedge at(P1, ATL)$

Goal: $at(C1, MSY) \wedge at(C2, MSY)$

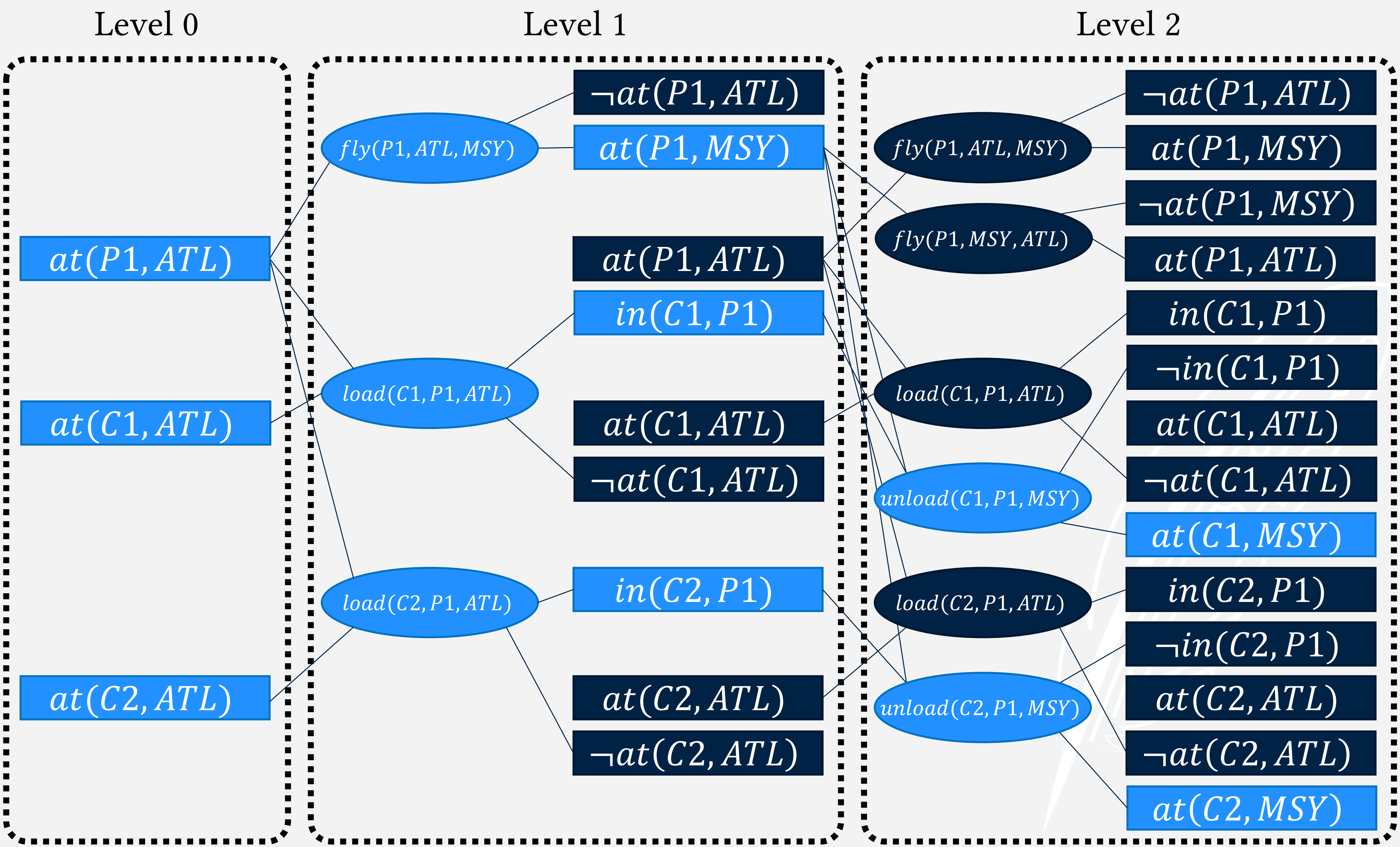
Plan:

1. $load(C1, P1, ATL)$
2. $load(C2, P1, ATL)$
3. $fly(P1, ATL, MSY)$
4. $unload(C1, P1, MSY)$
5. $unload(C2, P1, MSY)$

HSP Estimate: 6 steps

Plan Graph Estimate: 2 steps

FF Estimate: ? steps



Level 0
Level 1
Level 2
$at(P1, ATL)$
$at(C1, ATL)$
$at(C2, ATL)$
$fly(P1, ATL, MSY)$
$load(C1, P1, ATL)$
$load(C2, P1, ATL)$
$\neg at(P1, ATL)$
$at(P1, MSY)$
$at(P1, ATL)$
$in(C1, P1)$
$at(C1, ATL)$
$\neg at(C1, ATL)$
$in(C2, P1)$
$at(C2, ATL)$
$\neg at(C2, ATL)$
$fly(P1, ATL, MSY)$
$fly(P1, MSY, ATL)$
$load(C1, P1, ATL)$
$unload(C1, P1, MSY)$
$load(C2, P1, ATL)$
$unload(C2, P1, MSY)$
$\neg at(P1, ATL)$
$at(P1, MSY)$
$\neg at(P1, MSY)$
$at(P1, ATL)$
$in(C1, P1)$
$\neg in(C1, P1)$
$at(C1, ATL)$
$\neg at(C1, ATL)$
$at(C1, MSY)$
$in(C2, P1)$
$\neg in(C2, P1)$
$at(C2, ATL)$
$\neg at(C2, ATL)$
$at(C2, MSY)$

Cargo Problem

Initial State: $at(C1, ATL) \land at(C2, ATL) \land at(P1, ATL)$

Goal: $at(C1, MSY) \land at(C2, MSY)$

Plan:

1. $load(C1, P1, ATL)$
2. $load(C2, P1, ATL)$
3. $fly(P1, ATL, MSY)$
4. $unload(C1, P1, MSY)$
5. $unload(C2, P1, MSY)$

HSP Estimate: 6 steps

Plan Graph Estimate: 2 steps

FF Estimate: 5 steps

Graphplan vs. HSP vs. FF

- All three are solving a relaxed version of the problem by never deleting facts.
- Simply using the level of the plan graph at which the goal first appears is admissible but often underestimates dramatically.
- HSP does not account for synergy between goals and so is more prone to overestimate.
- FF accounts for synergy between goals and so often gives more accurate estimates.

Mutexes

FF does not compute mutexes when extending the plan graph. This is for two reasons:

- In practice, the extra accuracy gained by using mutexes is not worth the cost of computing them.
- Graphplan solution extraction is P-SPACE-hard because of mutexes. FF must re-compute the plan graph at every iteration of the search, so extracting a solution with mutexes is way too expensive. Without mutexes, solution extraction can be done greedily and is only P-TIME-hard.

Fast-Downward

- Created by Malte Helmert and Silvia Richter
- Winner of the fourth International Planning Competition (2004)
- Every winning planner since then up until the present has been based on FD

Fast-Downward

- Translates propositional problem representation into a variable / value representation, which allows for smaller and faster data structures.
- Computes domain transition graph to describe how a variable's value can change.
- Relaxed plans extracted from domain transition graphs.

Propositional Representation

Given 1 plane and 4 airports, there are 8 literals the need to be expessed:

1. $at(P1, ATL)$
2. $\neg at(P1, ATL)$
3. $at(P1, MSY)$
4. $\neg at(P1, MSY)$
5. $at(P1, SFO)$
6. $\neg at(P1, SFO)$
7. $at(P1, DFW)$
8. $\neg at(P1, DFW)$

Traditionally, a state has been represented as an array of Boolean variables. Given a problem with p planes and a airports, how many indices are needed in this array?

$$p \cdot a$$

Variable / Value Representation

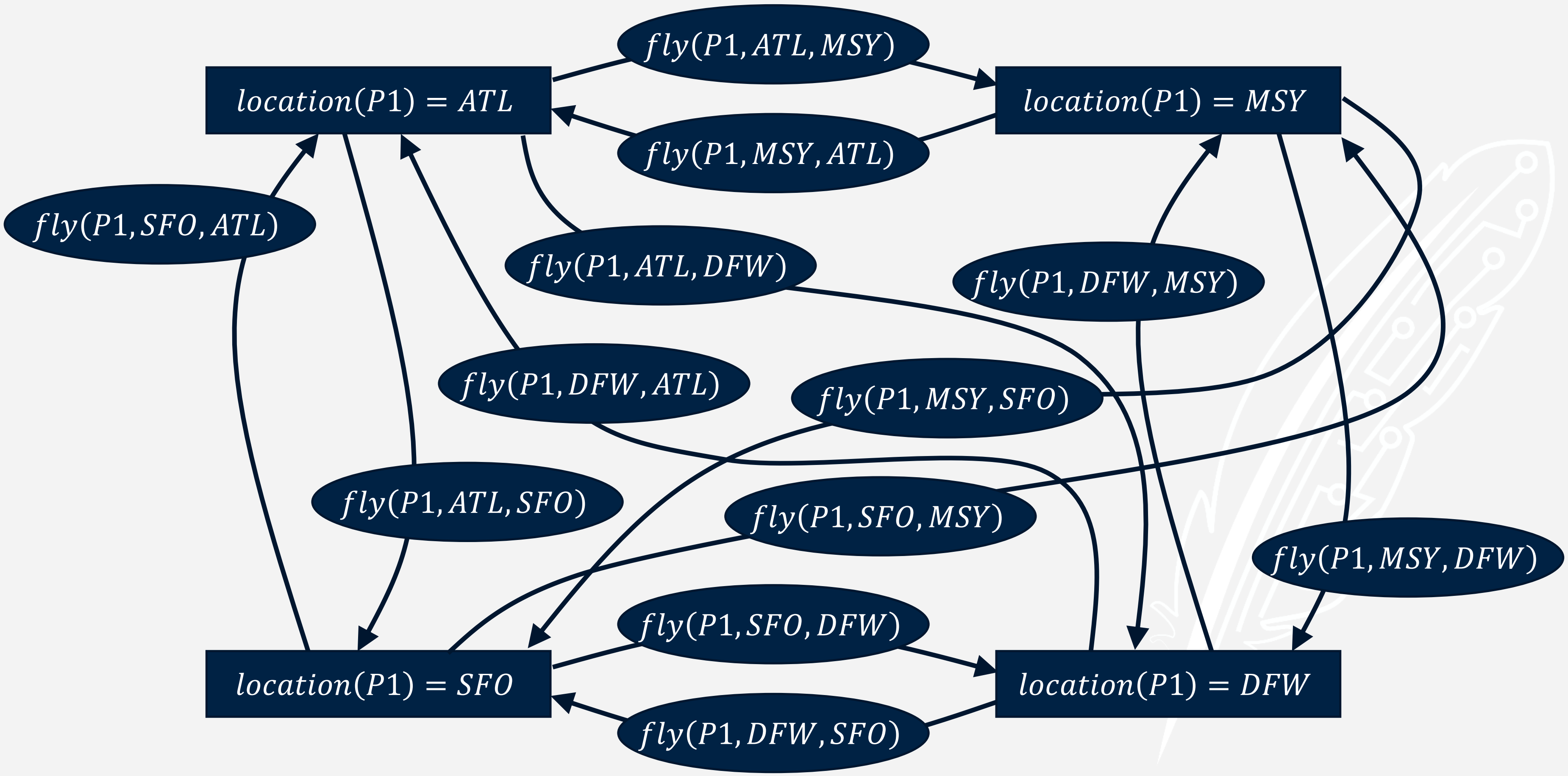
Rather than representing literals as Boolean values, FD infers a set of variables which can have one of many possible values:

1. $location(P1) = ATL$
2. $location(P1) = MSY$
3. $location(P1) = SFO$
4. $location(P1) = DFW$

Given a problem with p planes and a airports, how many indices are needed in this new array?

p

Domain Transition Graphs



NIL

FD Heuristic (Overview)

- FD calculates it heuristic by considering domain transition graphs (DTGs).
- Given the current value of a variable (current node in the DTG) , follow edges (steps) until we reach the value of that variable in the goal.
- The order in which variables are considered is decided based on how goals interact.
- FD is essentially breaking down each goal into a causal chain (remember causal links?).

HSP vs. FF vs. FD

All three planners calculate their heuristics by solving a relaxed version of the problem that is only P-TIME-hard instead of P-SPACE-hard:

- HSP does not account for synergy or interference.
- FF accounts for synergy but not interference.
- FD accounts for synergy and some interference.

History of Planning

- State space planning would be the most straight-forward way to approach the problem, but the search space explodes so quickly that this is only viable with highly accurate heuristics.
- Other approaches to planning (e.g. POCL and Graphplan) are developed. They use abstraction to reduce the number of decisions the planner must make.

History of Planning

- Abstraction leads to the development of accurate heuristics.
- Heuristics become progressively more accurate by solving relaxed problems which are more and more similar to the original problem.
- Other ideas developed early in the history of planning research (e.g. causal links) are sometimes helpful in developing new approaches.