Width and Serialization of Classical Planning Problems

Authors: Nir Lipovetzky and Hector Geffner
Presenter: Gage Birchmeier

Background

- Many State Space planners we looked at use a heuristic to guide search
- Improvements to planning usually just meant a better heuristic
- This paper introduces a new way to navigate a search space without a heuristic
- Very similar to iterative deepening methods

Introductory Definitions

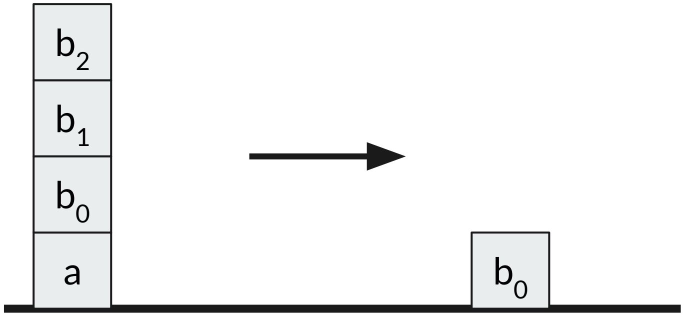
- Consider conjunctions of size *i* in the space of atoms in a planning problem
- A conjunctions *t* is succeeded by *t'* if any optimal plan for *t* can be made into an optimal plan for *t'* by adding one more action
- We can construct a directed graph G^i as follows:
  - The vertices are all possible conjunctions of size at most *i*
  - A directed edge from *t* to *t'* exists iff *t'* succeeds *t*

Width of a Problem

- We can now define the *width* of a formula in a planning problem
- A formula has width w is the minimum w such that G^w contains a conjunction that implies the formula
- Then the width of a planning problem can be defined as the width of the goal

Width of getting block onto table

- In the blocks domain, getting a block onto a table has width 1
- Imagine blocks $b_1, \dots, b_n$ are stacked on top of b_0, which we want to get on the table
- We can connect *clear*(b_n) -> *clear*(b_{n-1}) -> … -> *clear*(b_0) -> *ontable*(b_0) each by the action *putontable*(b_i) which is what gives the problem a width of 1



Advantages of width

- Many common domains have a bounded width given that the goal is a single atom
- Blocks, Logistics, and n-puzzle single goal problems have a width of at most 2
- Planning problems can be solved in time exponential relative to width $O(n^i)$
- The small width of single goal problems shows that complexity comes from having multiple goals, not the domain

Iterative Width Planner

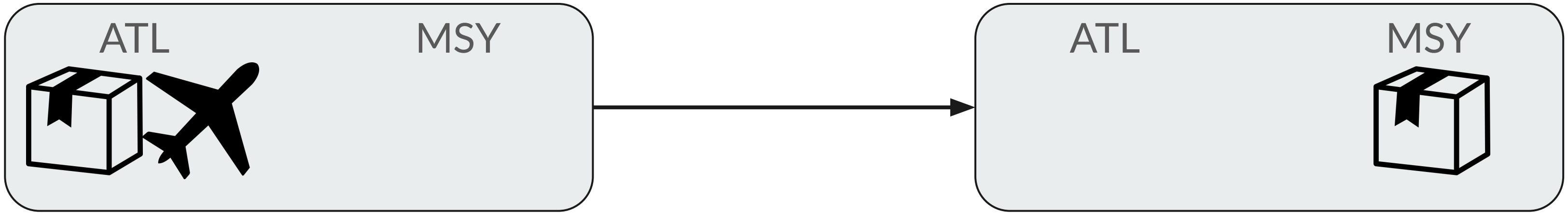
- Iterative Width is a forward-state breadth-first search with pruning
- If the smallest new conjunction produced by a state *s* has size *i* (this particular collection of facts has not been seen as true in another state), then we say it has a *novelty* of *i*. If there is no such conjunction, a state has a novelty of *n+1*.
- The higher the novelty, the less new a state is
- At iteration *i* of IW, we prune any states with *novelty* **greater than *i***
- Compare to Iterative deepening where we prune if the *depth* is **greater than *i***

Serialized Iterative Width

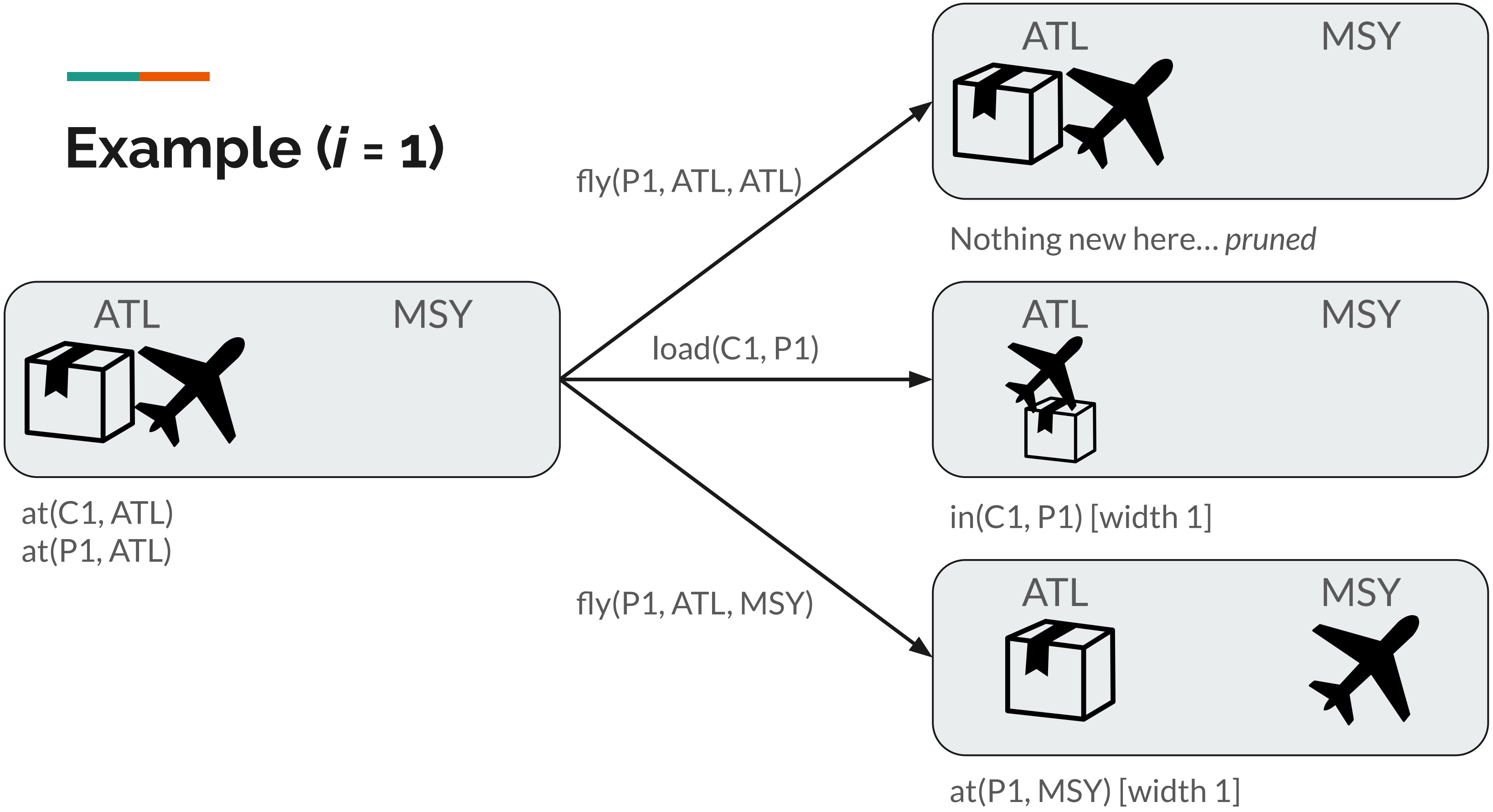
- Another variant on Iterative Width search
- Start with a width of 1, then find the first state *s* that has 1 of the goals
- Then starting from *s*, perform search with a width of 2 to find a state that has the earlier goal plus one new one satisfied
- Continue until all goal clauses are satisfied

Example

- Delivery domain
- Have one plane and one cargo at ATL
- Want the cargo at MSY
- Use Iterative Width Planning to solve

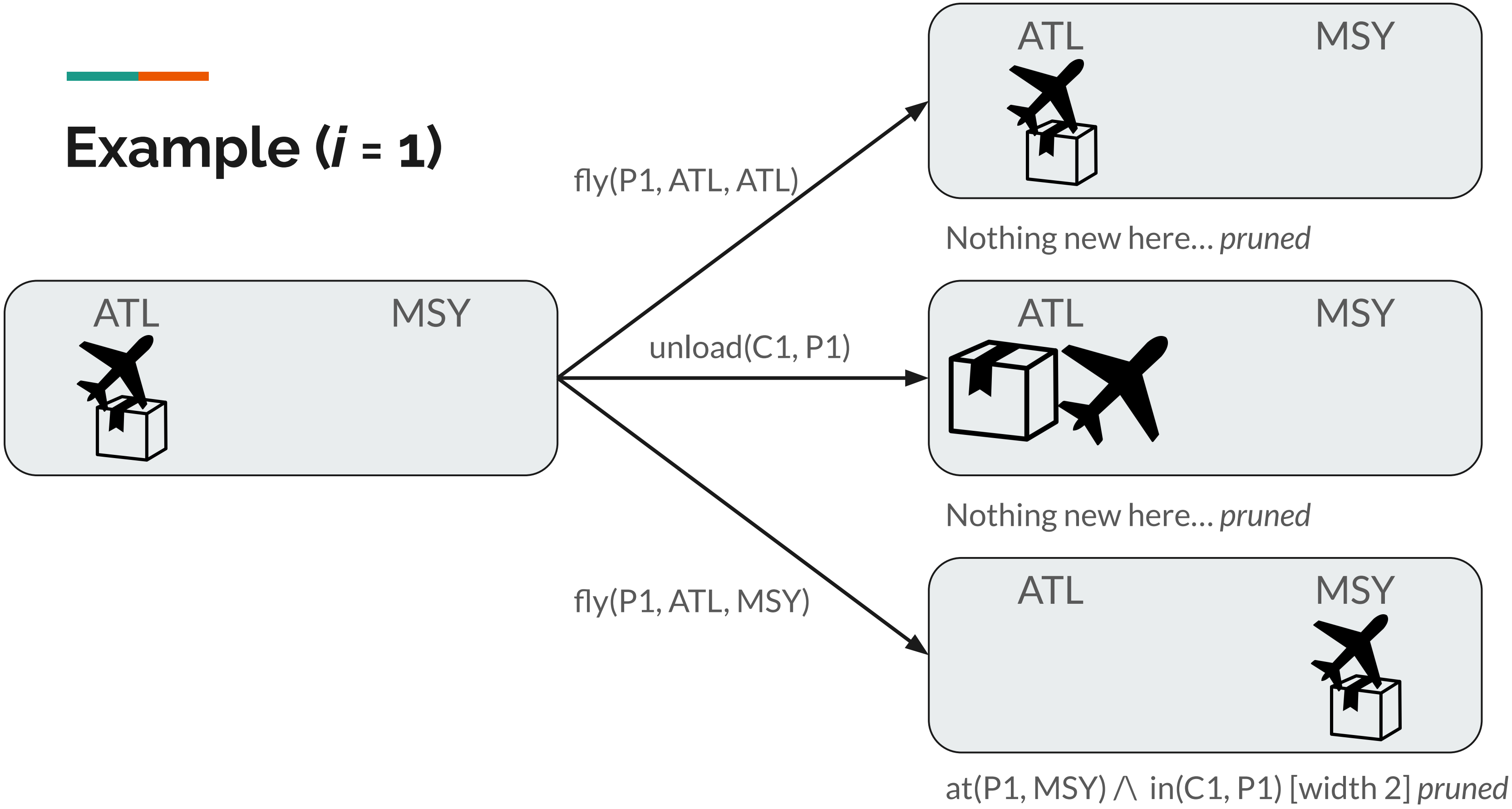


Example (*i* = 1)

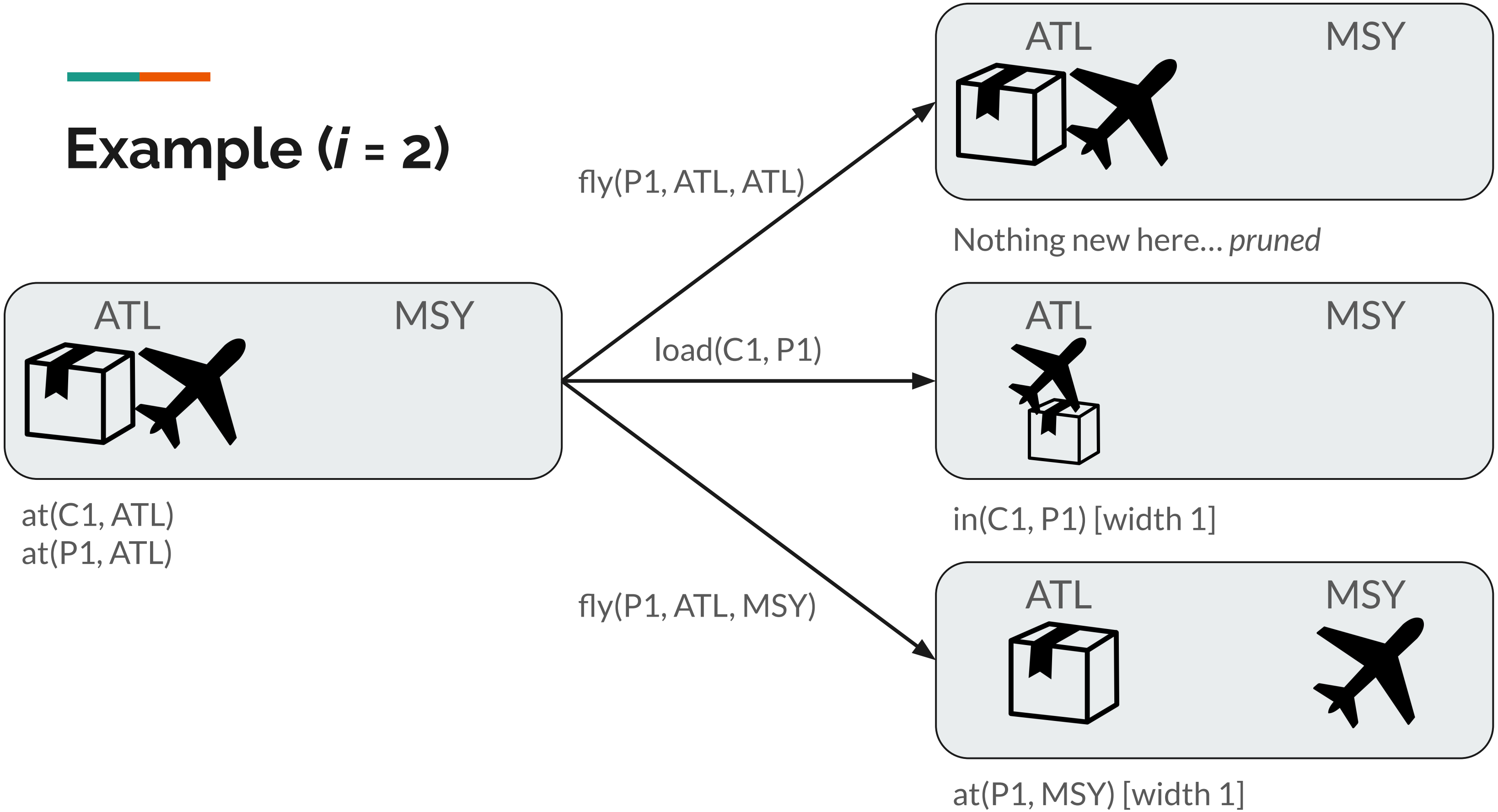


ATL
MSY
at(C1, ATL)
at(P1, ATL)
fly(P1, ATL, ATL)
load(C1, P1)
fly(P1, ATL, MSY)
ATL
MSY
Nothing new here… *pruned*
ATL
MSY
in(C1, P1) [width 1]
ATL
MSY
at(P1, MSY) [width 1]

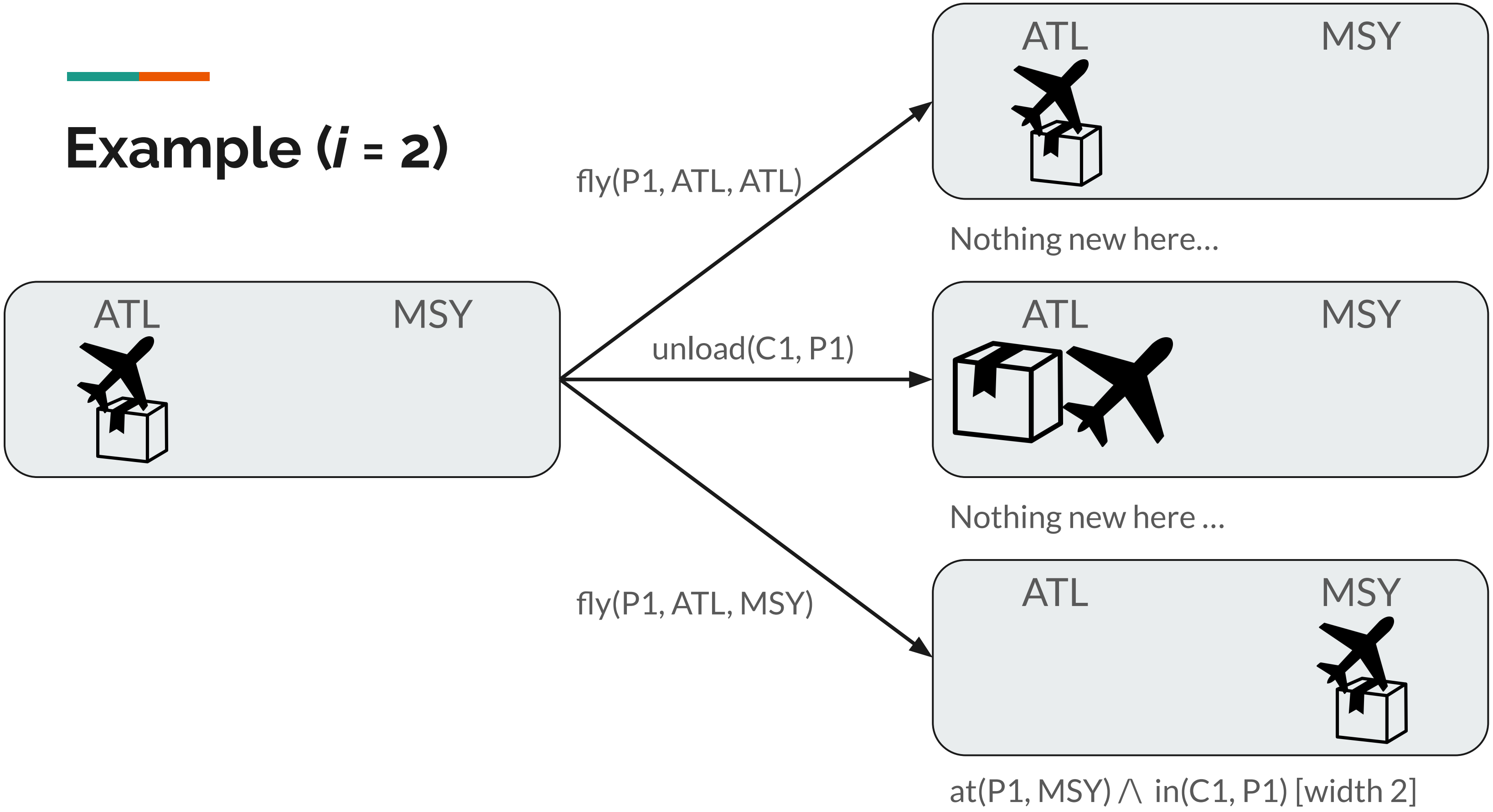
Example (*i* = 1)


ATL
MSY
fly(P1, ATL, ATL)
unload(C1, P1)
fly(P1, ATL, MSY)
ATL
MSY
Nothing new here… *pruned*
ATL
MSY
Nothing new here… *pruned*
ATL
MSY
at(P1, MSY) ∧ in(C1, P1) [width 2] *pruned*

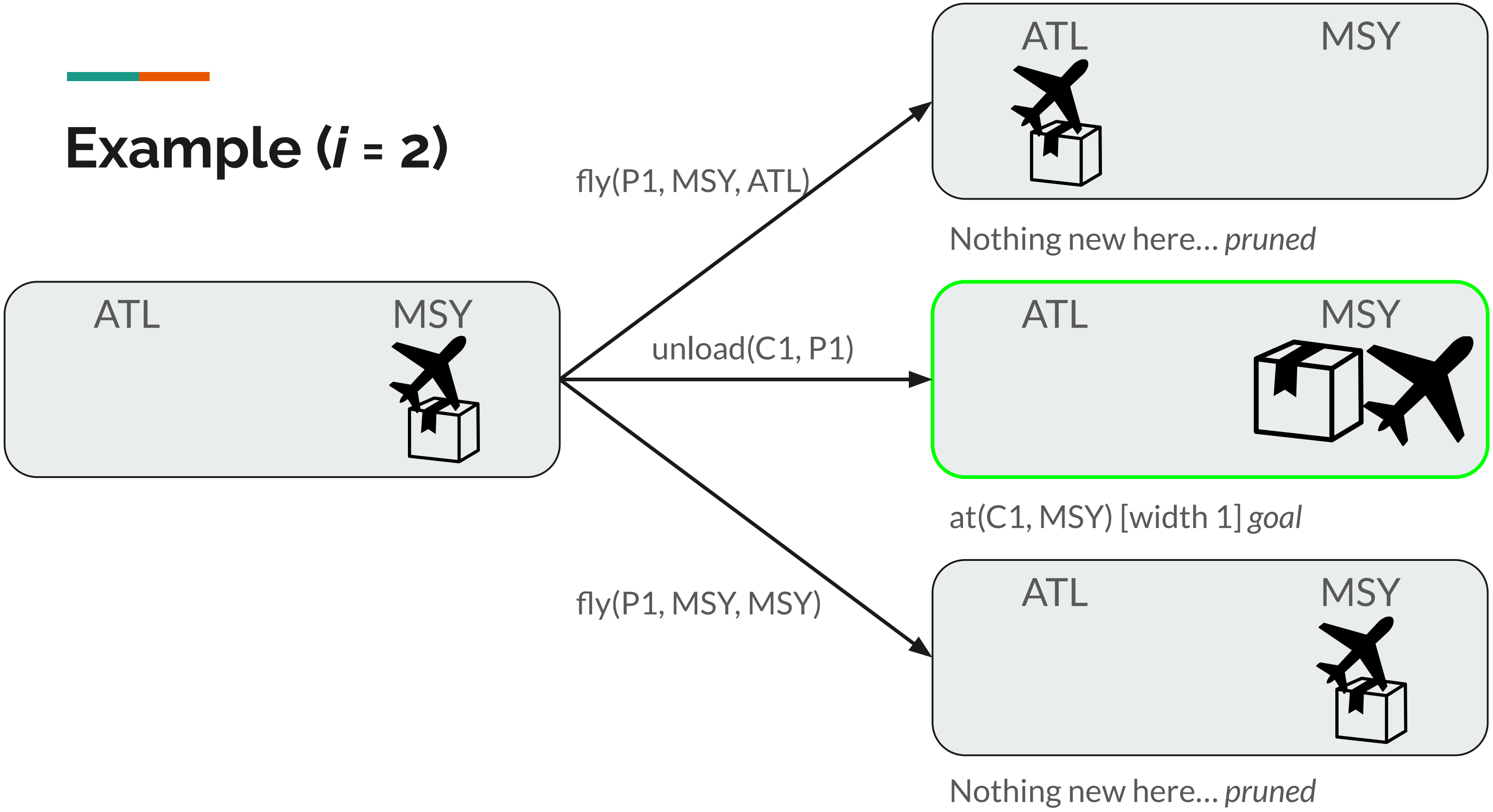
Example (*i* = 2)


ATL
MSY
fly(P1, ATL, ATL)
Nothing new here… *pruned*
load(C1, P1)
ATL
MSY
in(C1, P1) [width 1]
ATL
MSY
at(C1, ATL)
at(P1, ATL)
fly(P1, ATL, MSY)
ATL
MSY
at(P1, MSY) [width 1]

Example (*i* = 2)


ATL
MSY
fly(P1, ATL, ATL)
ATL
MSY
Nothing new here...
unload(C1, P1)
ATL
MSY
Nothing new here ...
fly(P1, ATL, MSY)
ATL
MSY
at(P1, MSY) ∧ in(C1, P1) [width 2]

Example (*i* = 2)


ATL
MSY
fly(P1, MSY, ATL)
unload(C1, P1)
fly(P1, MSY, MSY)
ATL
MSY
Nothing new here... *pruned*
ATL
MSY
at(C1, MSY) [width 1] *goal*
ATL
MSY
Nothing new here... *pruned*