Autonomous Greenhouse Management System

Alessia Pontiggia – 1892079
Gerardo Paladino – 1918178
Planning and Reasoning
Sapienza University of Rome

Table of contents

Problem Definition

- The **greenhouse** has:
  - **Plants** and **Vegetables** located in different *positions* that need **resources** like *water*, *food* and/or *pesticides*
  - One (or more) **robotic systems** providing resources
  - They have a **battery** that needs to be recharged once it decreases all its levels or resource
  - They can **share** resources and/or energy
  - A **recharge station** is at a fixed position

- The **goal** is:
  - Maximizing energy saving and feed all the plants

PDDL Domain – Types, Predicates and Functions

```
(:types plant system location resource)

(:predicates
  (needs ?r - resource ?p - plant)
  (healthy ?p - plant ?r - resource)
  (resource-available ?r - resource ?s)
  (at ?s - system ?l - location)
  (at-plant ?p - plant ?l - location)
  (adjacent ?l1 - location ?l2 - location)
  (recharge-station ?l - location)

)
(:functions
  (total-cost)
  (battery-level ?s - system)
)
```

PDDL Domain – Actions

```
(:action move-system
  :parameters (?s - system ?from - location ?to - location)
  :precondition (and
    (at ?s ?from)
    (adjacent ?from ?to)
    (> (battery-level ?s) 0)
  )
  :effect (and
    (not (at ?s ?from))
    (at ?s ?to)
    (increase (total-cost) 10)
    (decrease (battery-level ?s) 1)
  )
)

(:action feed-plant
  :parameters (?p - plant ?s - system ?r - resource ?l - location)
  :precondition (and
    (needs ?r ?p)
    (at ?s ?l)
    (at-plant ?p ?l)
    (resource-available ?r ?s)
    (> (battery-level ?s) 0)
  )
  :effect (and
    (not (needs ?r ?p))
    (healthy ?p ?r)
    (not (resource-available ?r ?s))
    (increase (total-cost) 1)
    (decrease (battery-level ?s) 1)
  )
)
```

PDDL Domain – Actions

```
(:action recharge-system
:parameters (?s - system ?l - location ?r - resource)
:precondition (and
  (at ?s ?l)
  (recharge-station ?l)
  (or
    (<= (battery-level ?s) 1)
    (not (resource-available ?r ?s))
  )
  )
  :effect (and
  (increase (total-cost) 100)
  (when (not (resource-available ?r ?s))
    (resource-available ?r ?s)
  )
  (when (<= (battery-level ?s) 1)
    (increase (battery-level ?s) 10)
  )
  )
  )
```

PDDL Domain – Actions

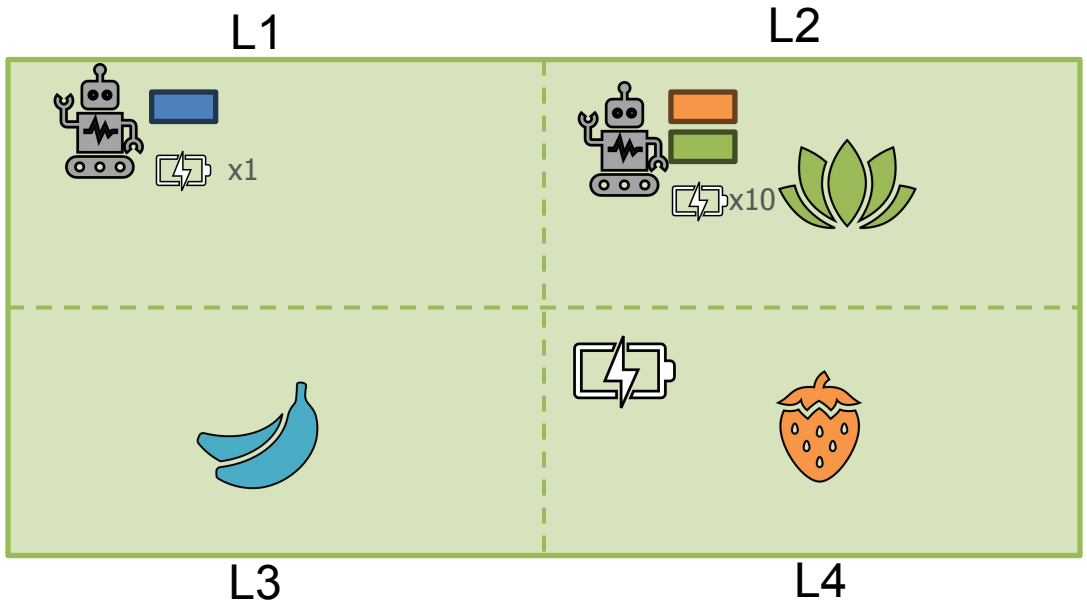
```
(:action share-resource
  :parameters (?s1 - system ?s2 - system ?l1 - location ?r - resource ?p - plant)
  :precondition (and
    (needs ?r ?p)
    (at ?s1 ?l1)
    (at ?s2 ?l1)
    (not (resource-available ?r ?s1))
    (resource-available ?r ?s2)
    (> (battery-level ?s1) 0)
    (> (battery-level ?s2) 0)
  )
  :effect (and
    (resource-available ?r ?s1)
    (not (resource-available ?r ?s2))
  )
)

(:action share-energy
  :parameters (?s1 - system ?s2 - system ?l1 - location)
  :precondition (and
    (at ?s1 ?l1)
    (at ?s2 ?l1)
    (= (battery-level ?s1) 0)
    (> (battery-level ?s2) 0)
  )
  :effect (and
    (increase (battery-level ?s1) (battery-level ?s2))
    (decrease (battery-level ?s2) (battery-level ?s2))
  )
)
)
```

PDDL Problem – (Easy) Solution

= water

= pesticides

= nutrients

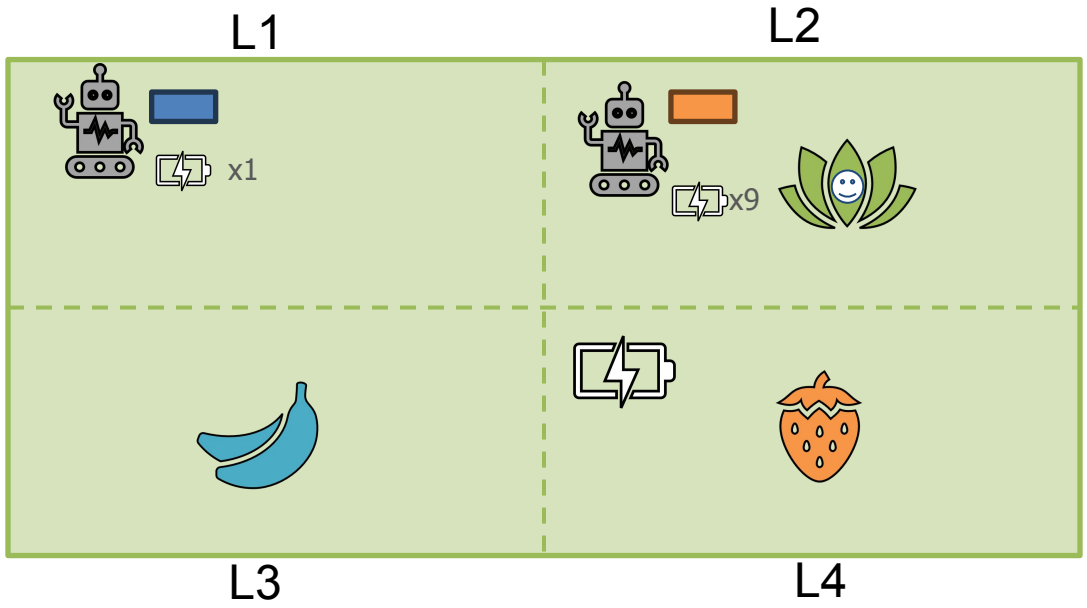


PDDL Problem – (Easy) Solution

= water

= pesticides

= nutrients

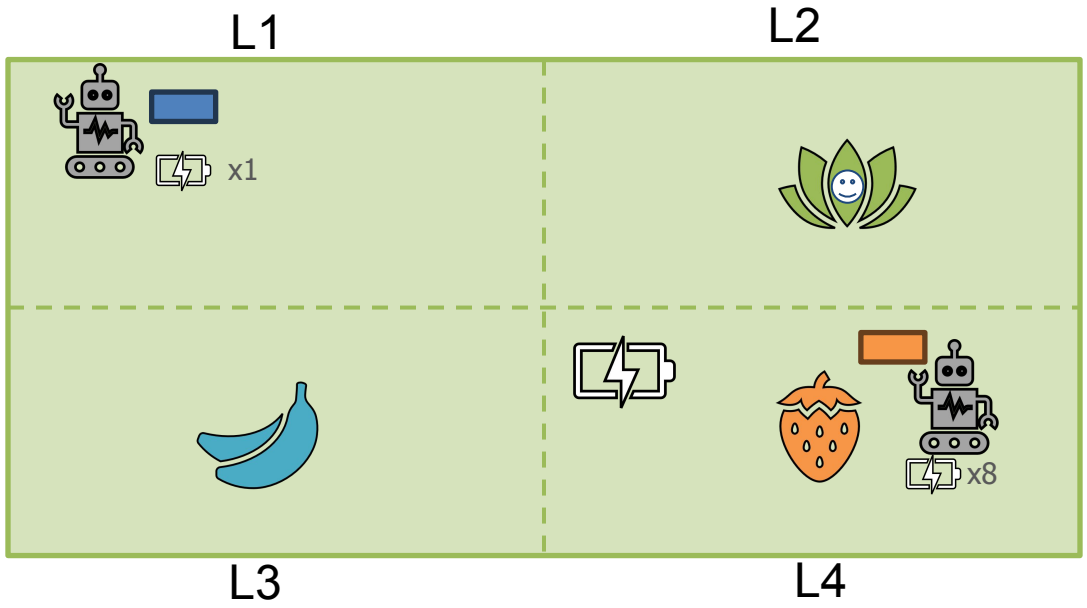


PDDL Problem – (Easy) Solution

= water

= pesticides

= nutrients

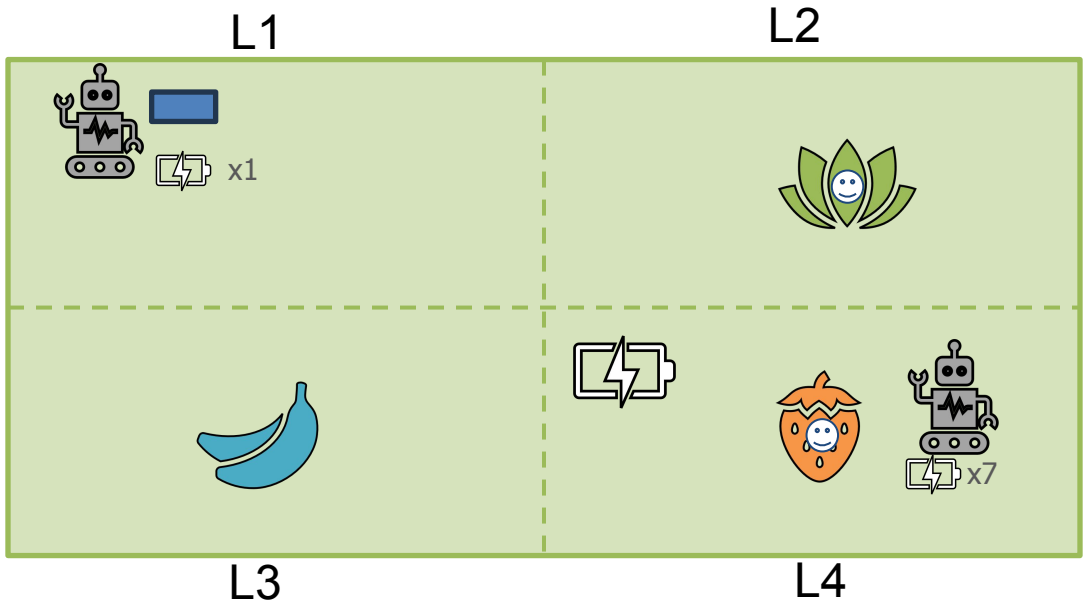


PDDL Problem – (Easy) Solution

= water

= pesticides

= nutrients

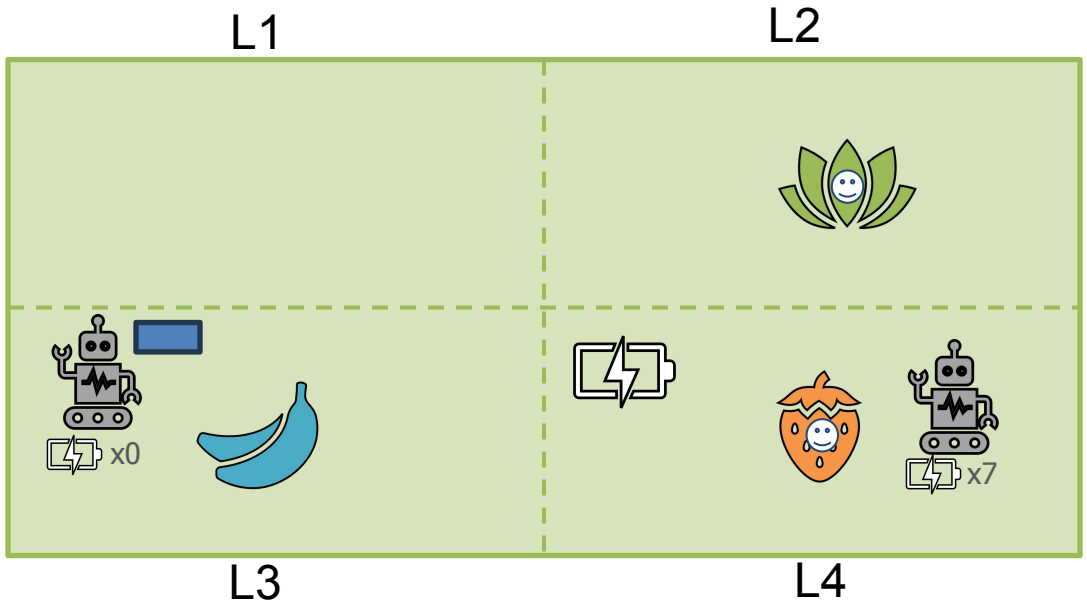


PDDL Problem – (Easy) Solution

= water

= pesticides

= nutrients

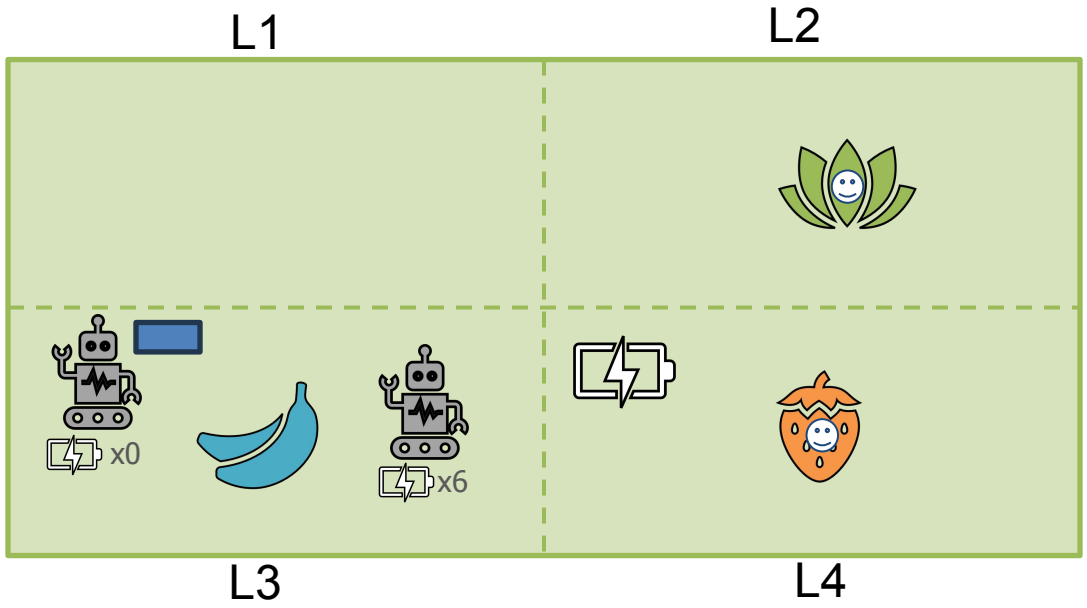


PDDL Problem – (Easy) Solution

= water

= pesticides

= nutrients

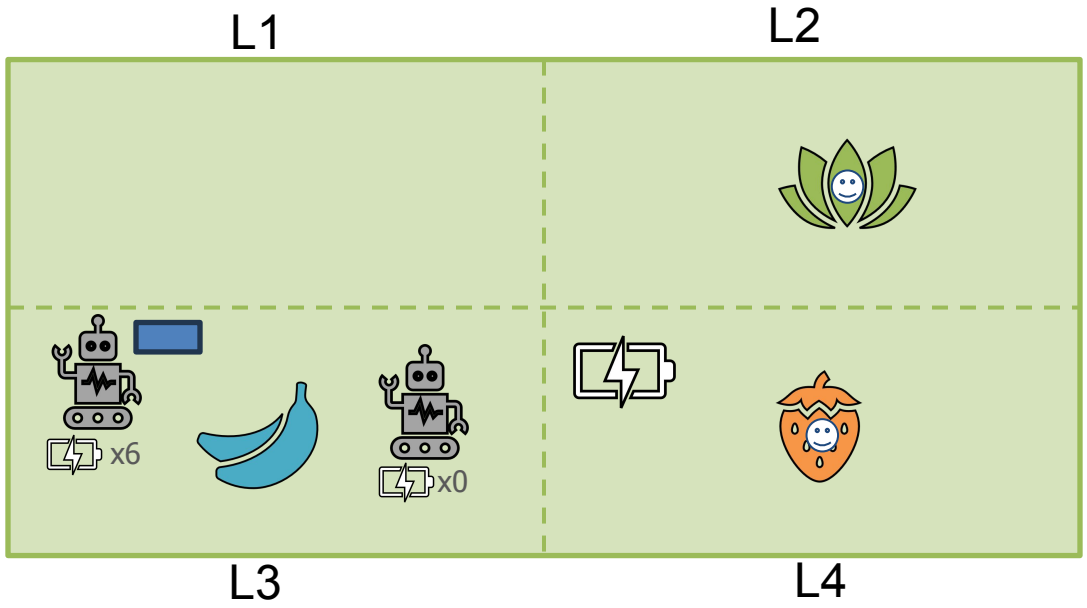


PDDL Problem – (Easy) Solution

= water

= pesticides

= nutrients

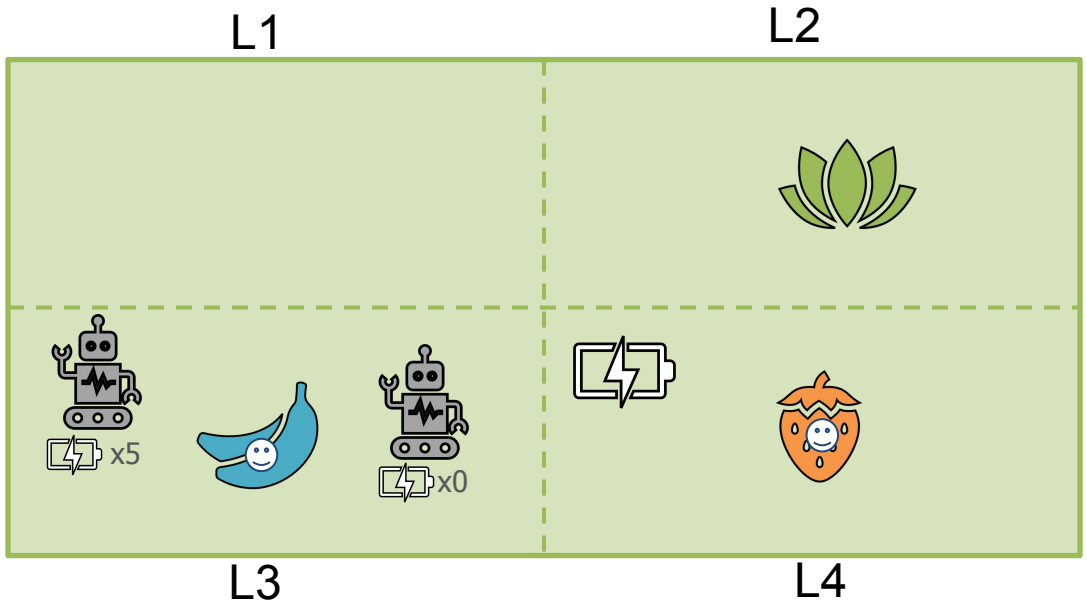


PDDL Problem – (Easy) Solution

= water

= pesticides

= nutrients



PDDL Problem – (Easy) Performances

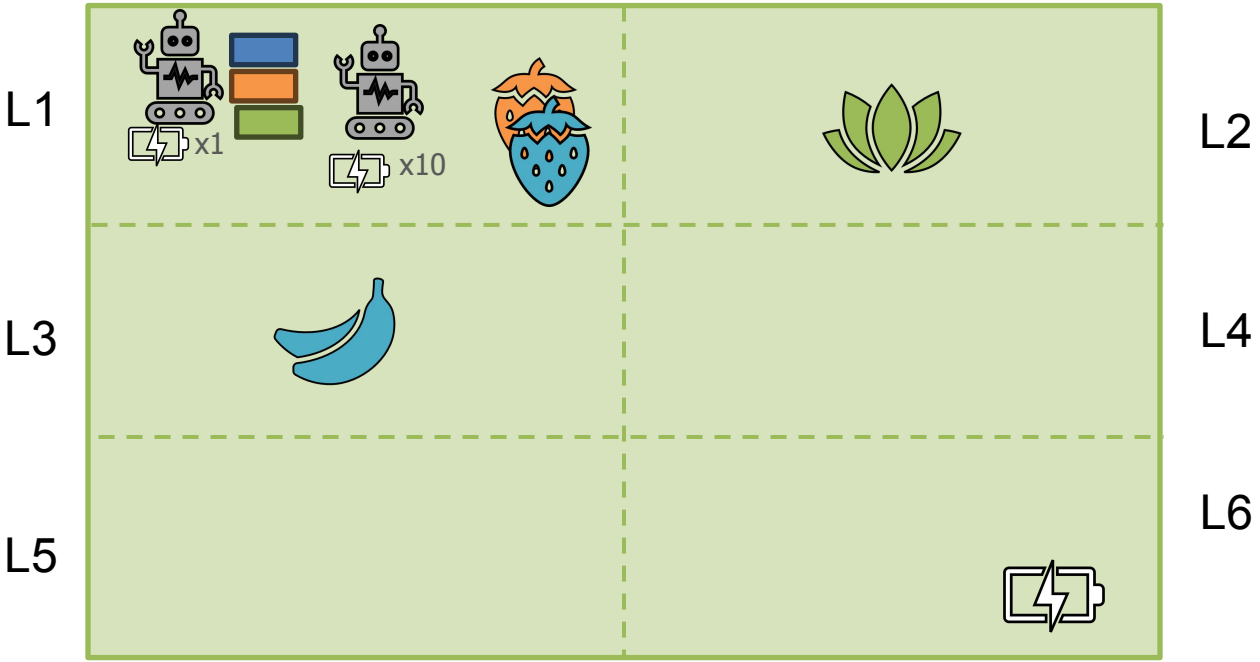
Search	Heuristic	Time (ms)	Expanded Nodes	Length	Cost
Weighted A*	h_max	717	39	10	153
Weighted A*	h_add	662	9	7	33
Weighted A*	h_ff	649	199	95	1213

PDDL Problem – (Mid) Solution

= water

= pesticides

= nutrients

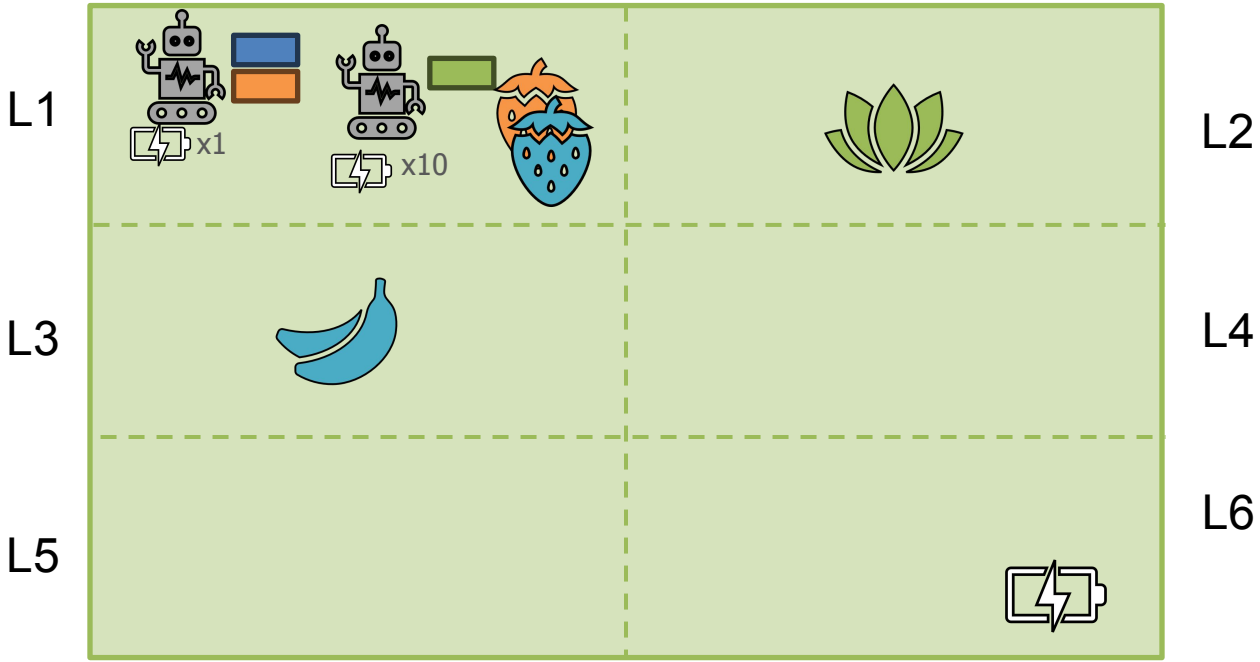


PDDL Problem – (Mid) Solution

= water

= pesticides

= nutrients

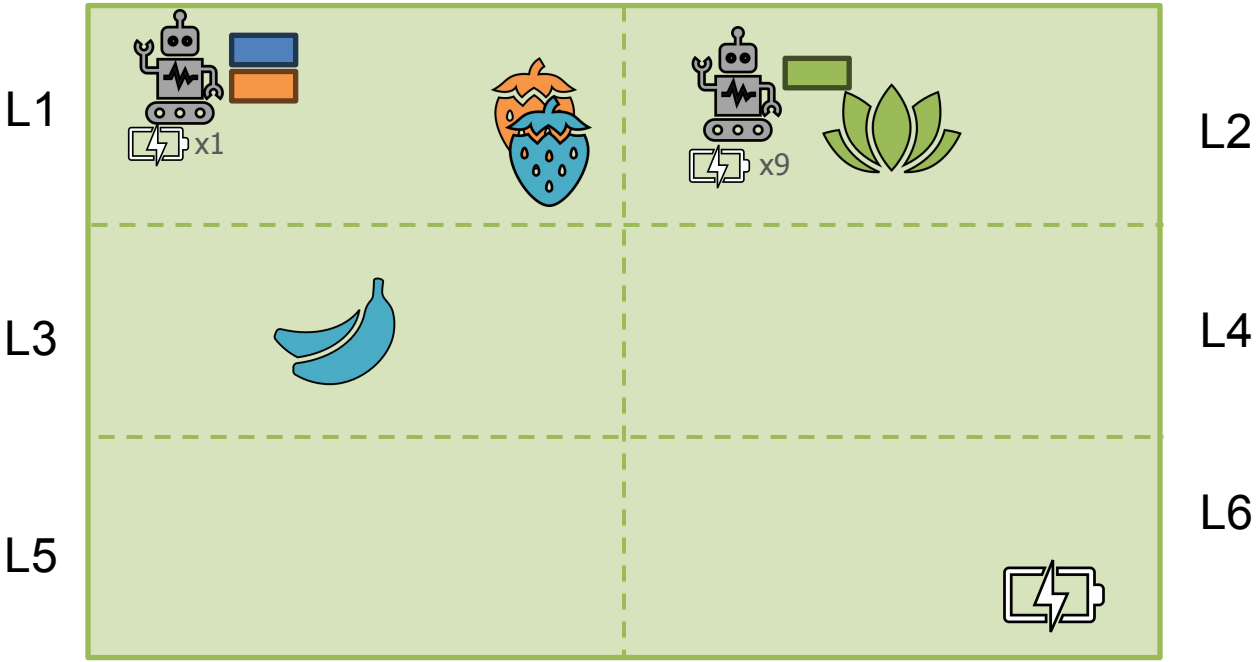


PDDL Problem – (Mid) Solution

= water

= pesticides

= nutrients

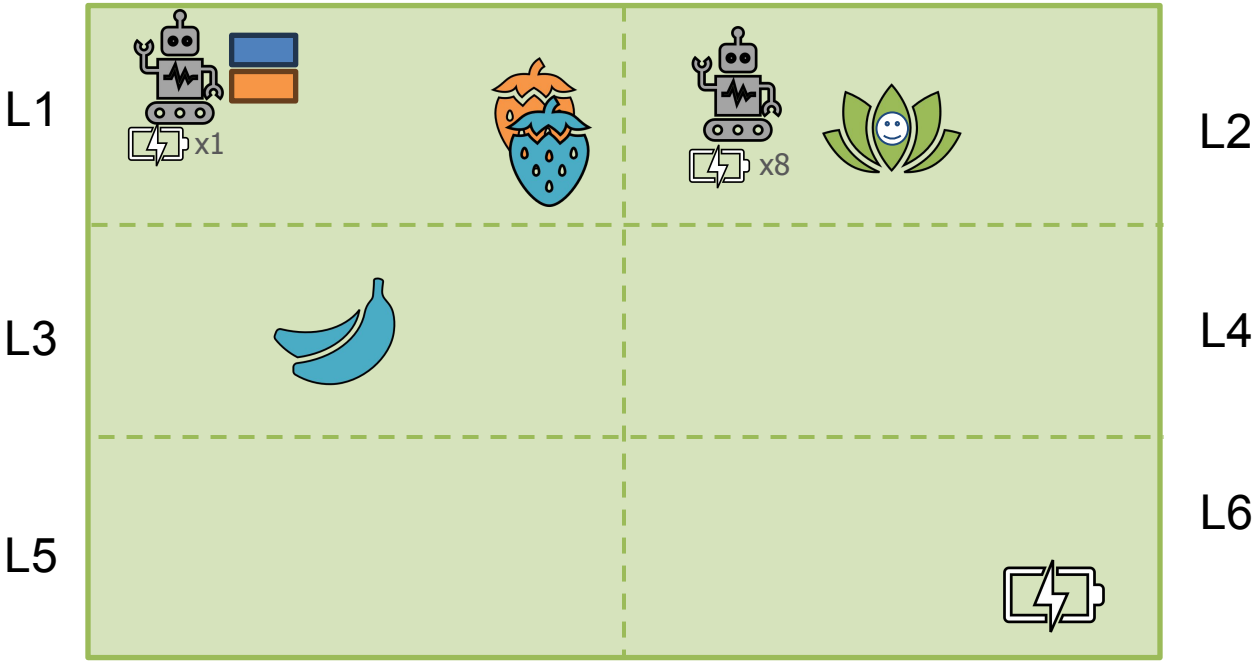


PDDL Problem – (Mid) Solution

= water

= pesticides

= nutrients

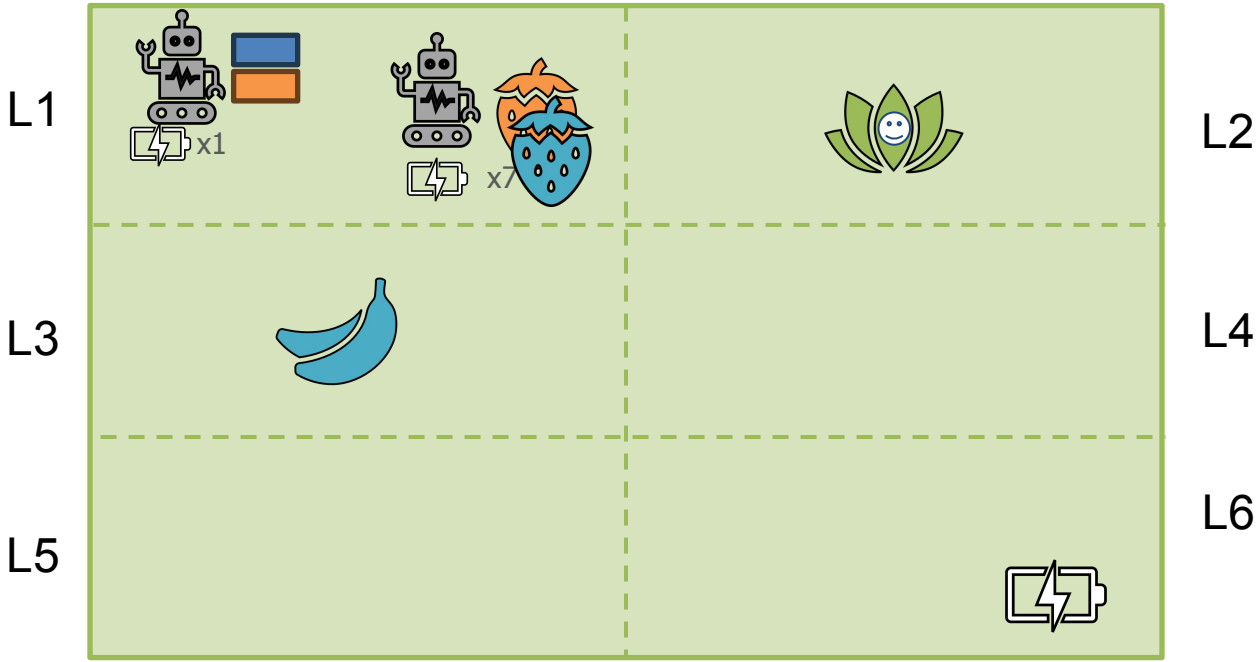


PDDL Problem – (Mid) Solution

= water

= pesticides

= nutrients

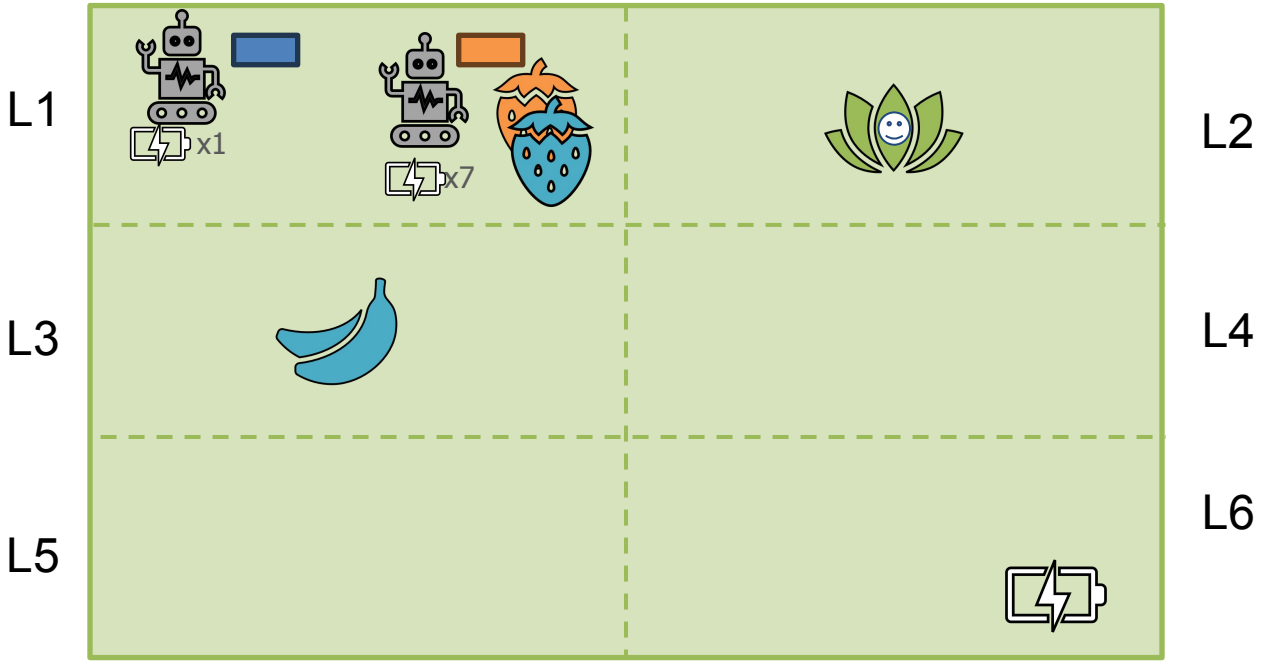


PDDL Problem – (Mid) Solution

= water

= pesticides

= nutrients

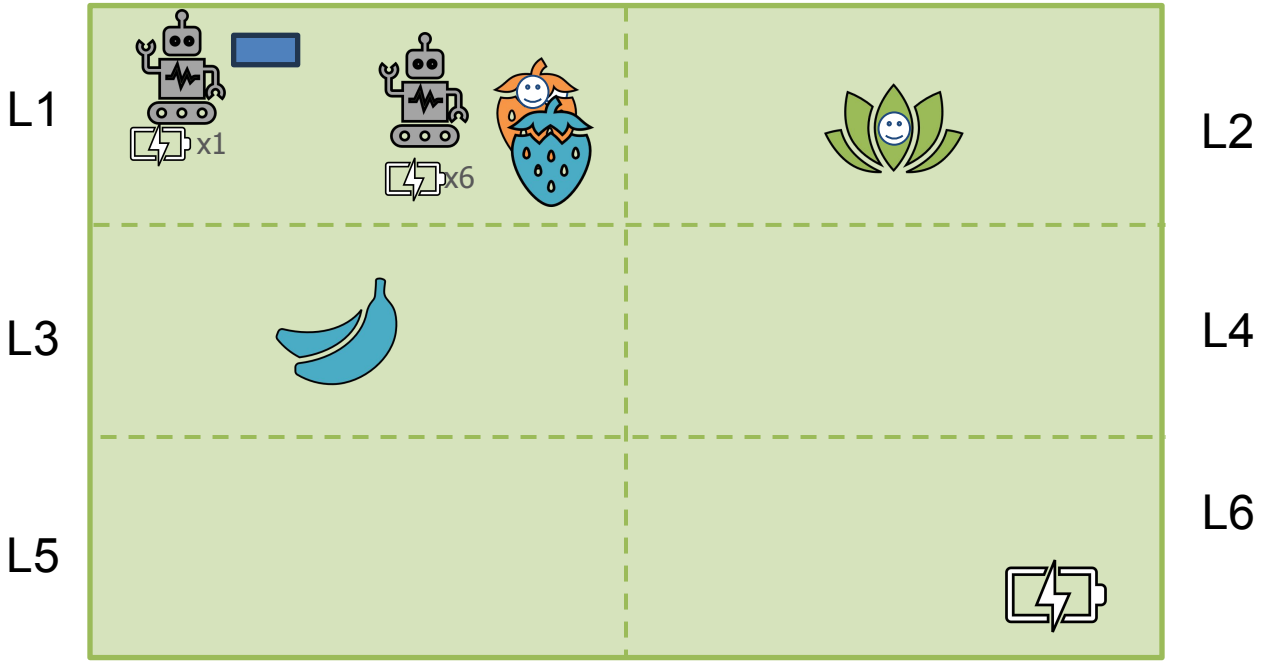


PDDL Problem – (Mid) Solution

= water

= pesticides

= nutrients

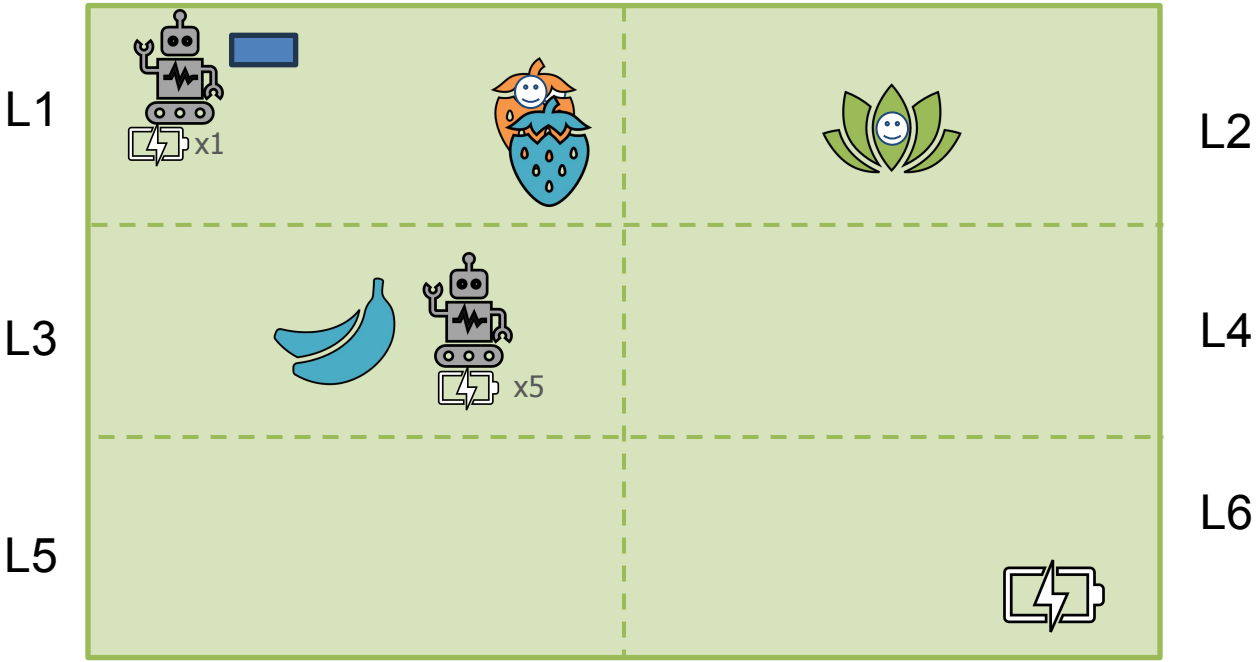


PDDL Problem – (Mid) Solution

= water

= pesticides

= nutrients

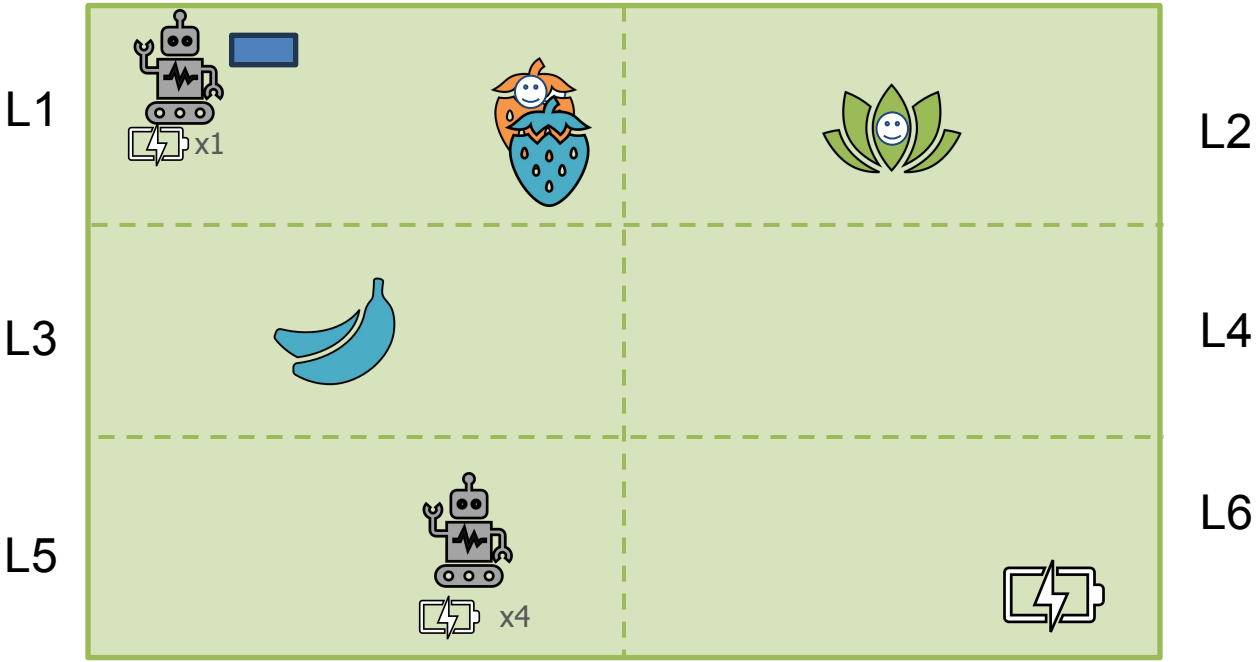


PDDL Problem – (Mid) Solution

= water

= pesticides

= nutrients

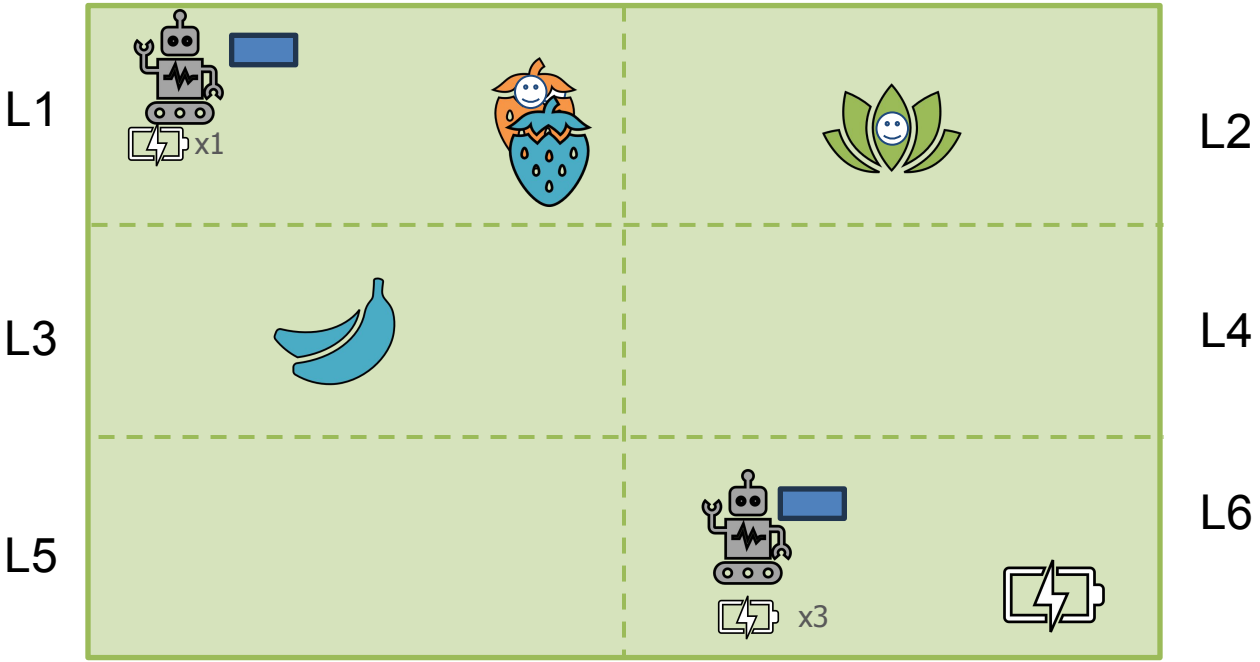


PDDL Problem – (Mid) Solution

= water

= pesticides

= nutrients

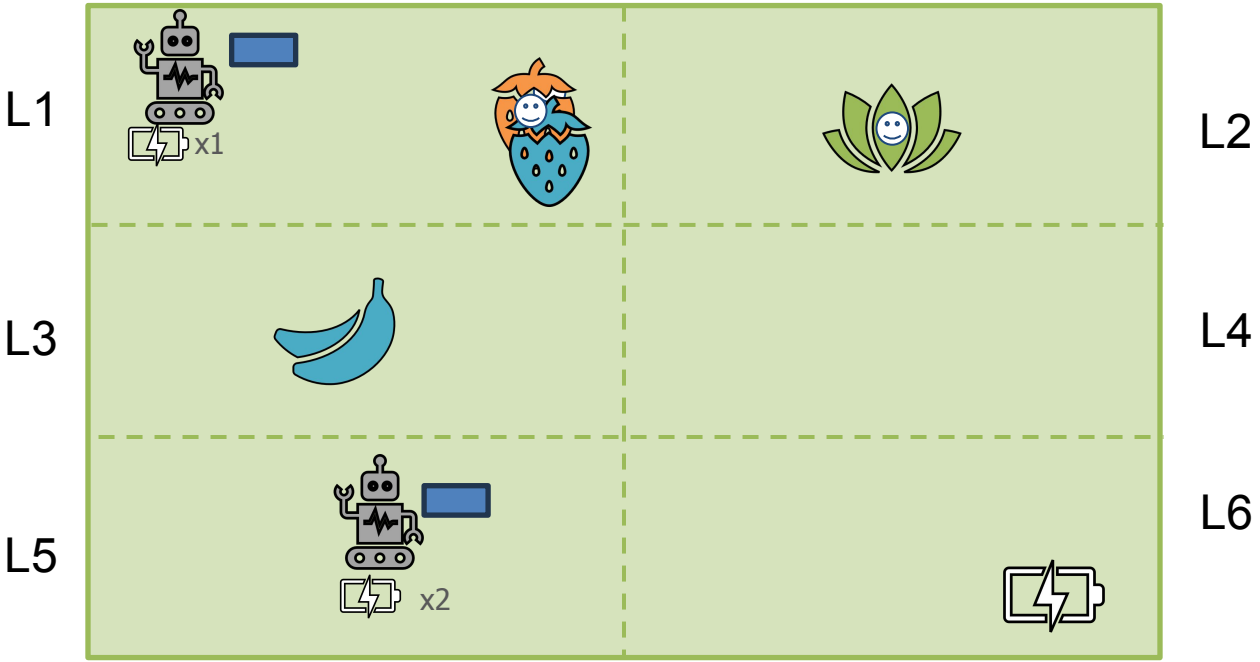


PDDL Problem – (Mid) Solution

= water

= pesticides

= nutrients

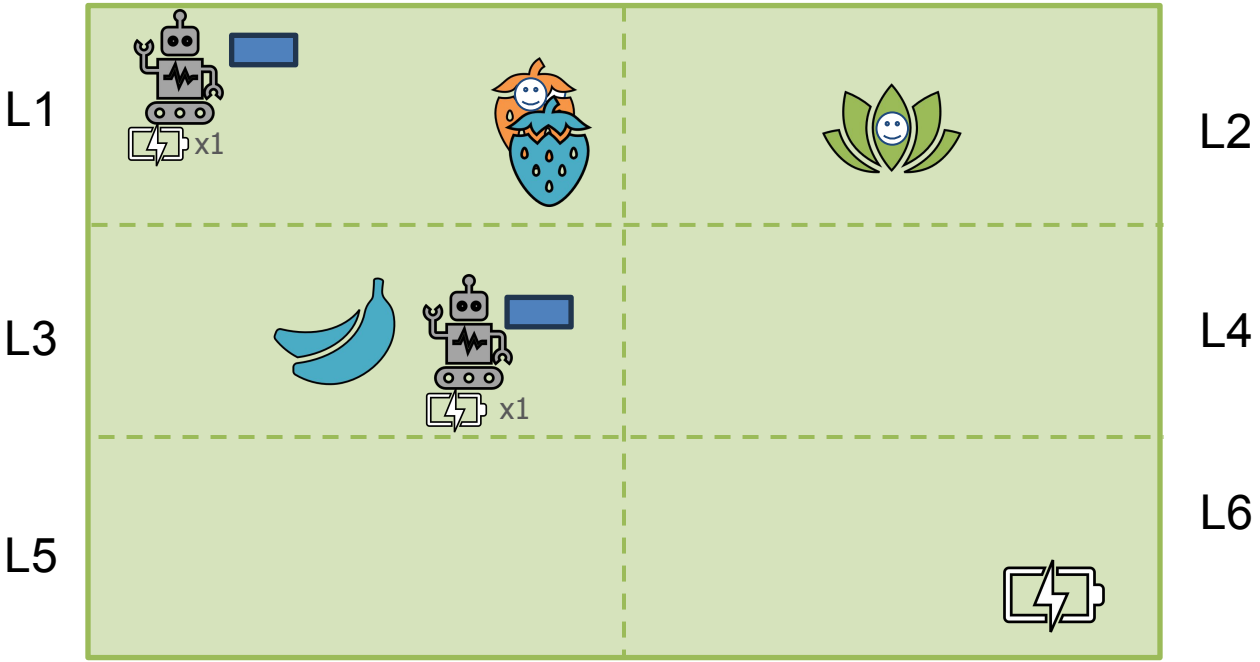


PDDL Problem – (Mid) Solution

= water

= pesticides

= nutrients



PDDL Problem – (Mid) Solution

= water

= pesticides

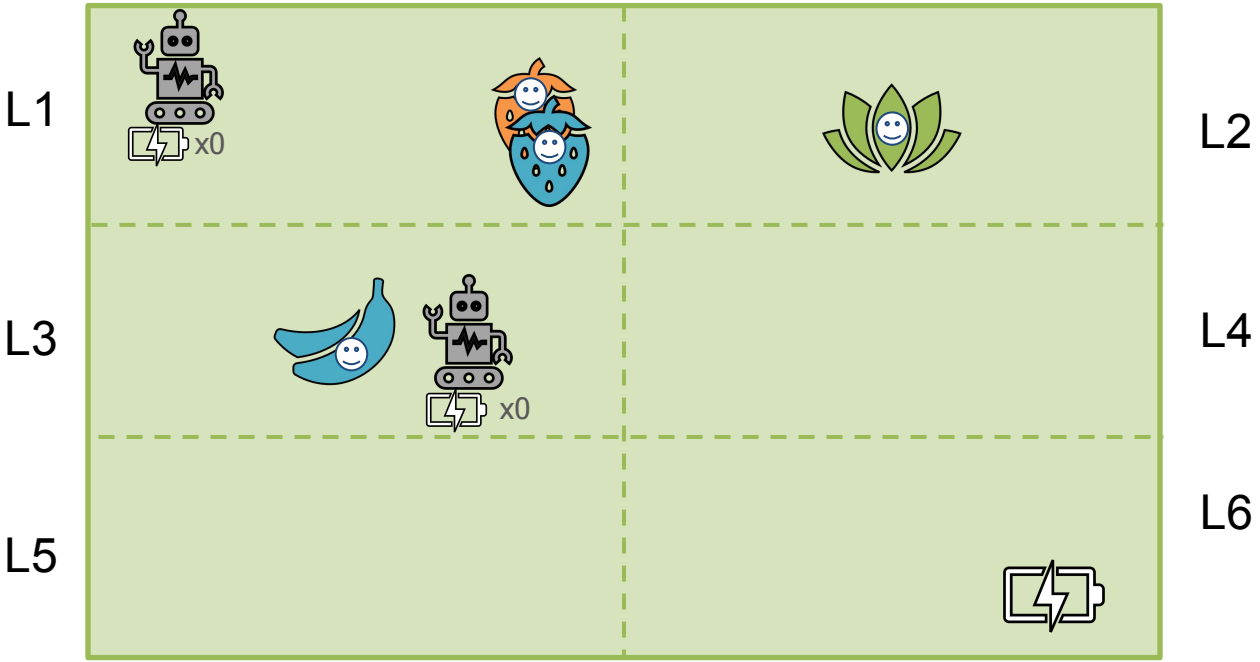
= nutrients

PDDL Problem – (Mid) Solution

= water

= pesticides

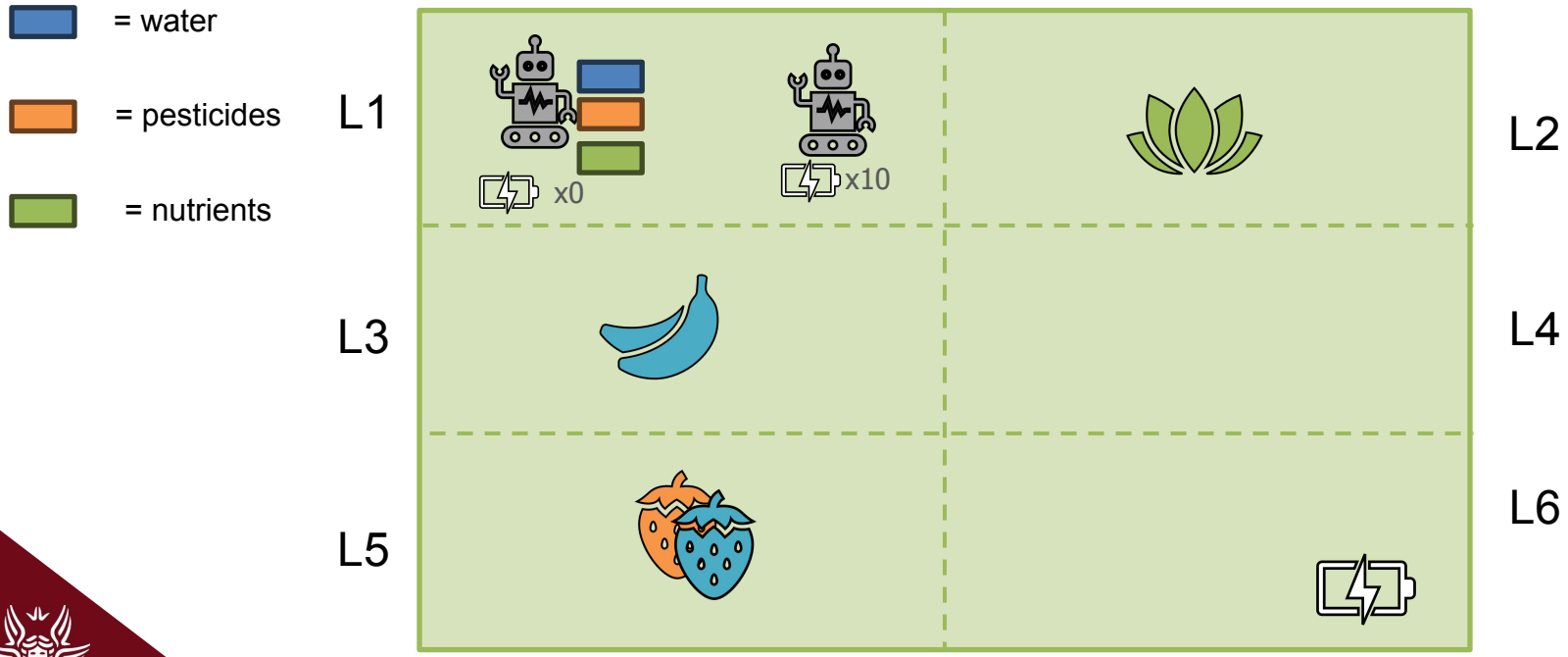
= nutrients



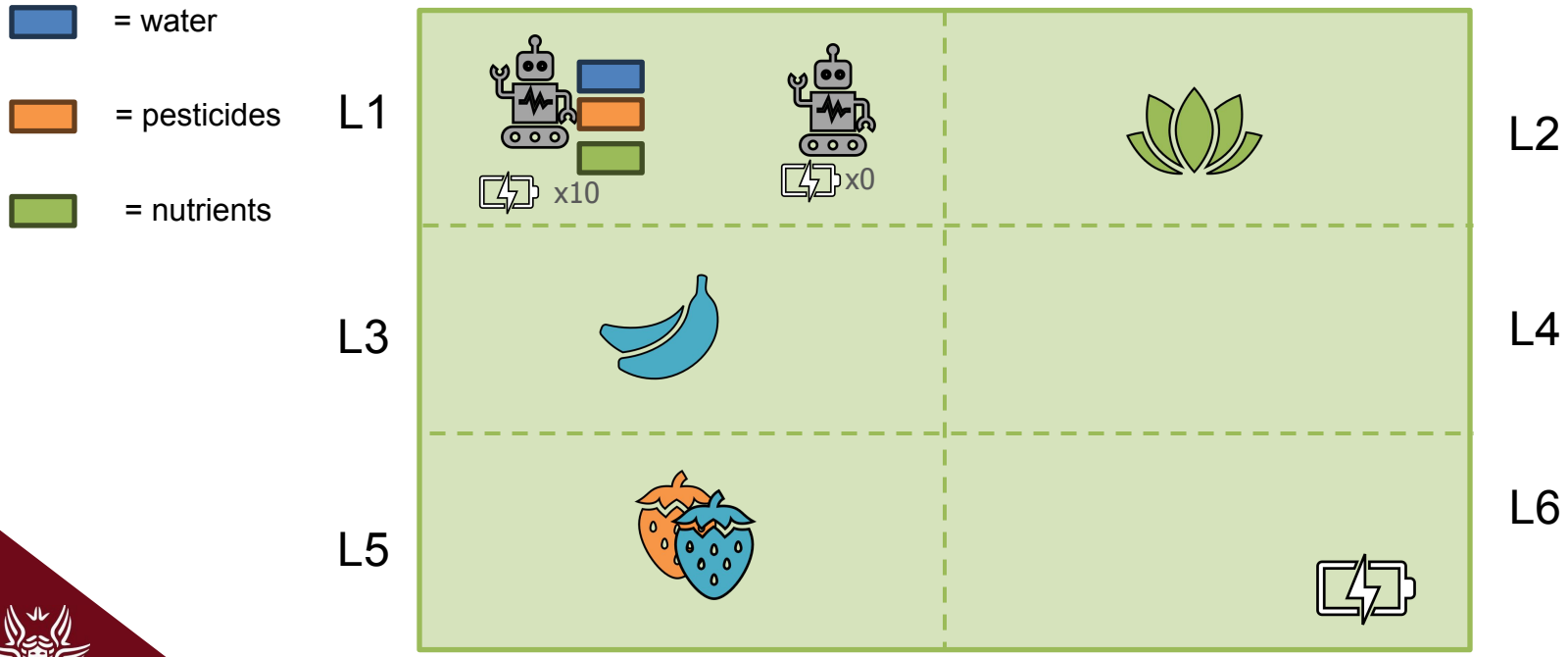
PDDL Problem – (Mid) Performances

Search	Heuristic	Time (ms)	Expanded Nodes	Length	Cost
Weighted A*	h_max	1099	312	23	434
Weighted A*	h_add	874	70	14	174
Weighted A*	h_ff	737	127	48	804

PDDL Problem – (Difficult) Solution



PDDL Problem – (Difficult) Solution

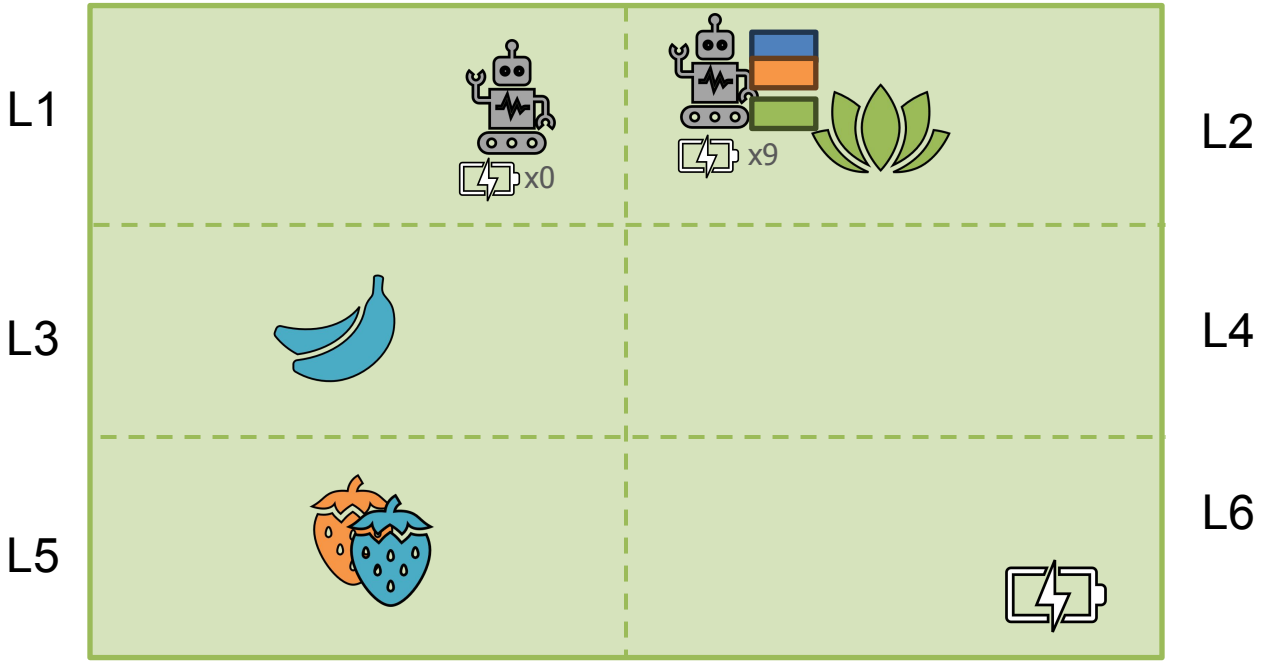


PDDL Problem – (Difficult) Solution

= water

= pesticides

= nutrients

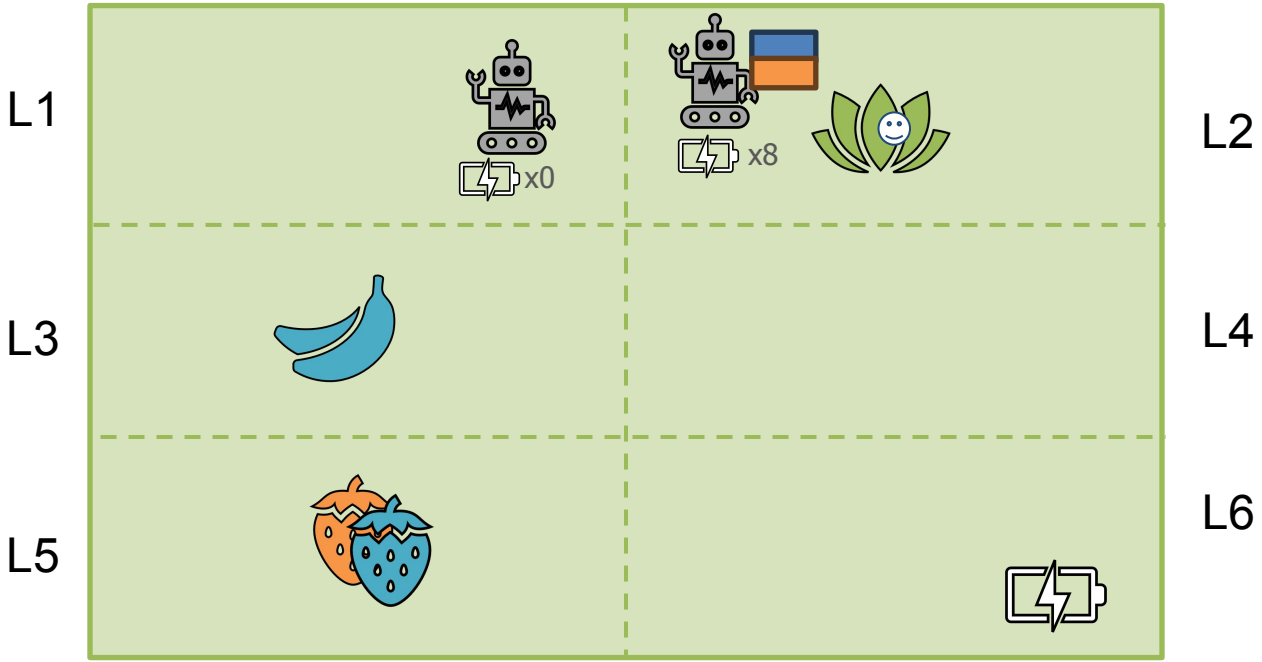


PDDL Problem – (Difficult) Solution

= water

= pesticides

= nutrients

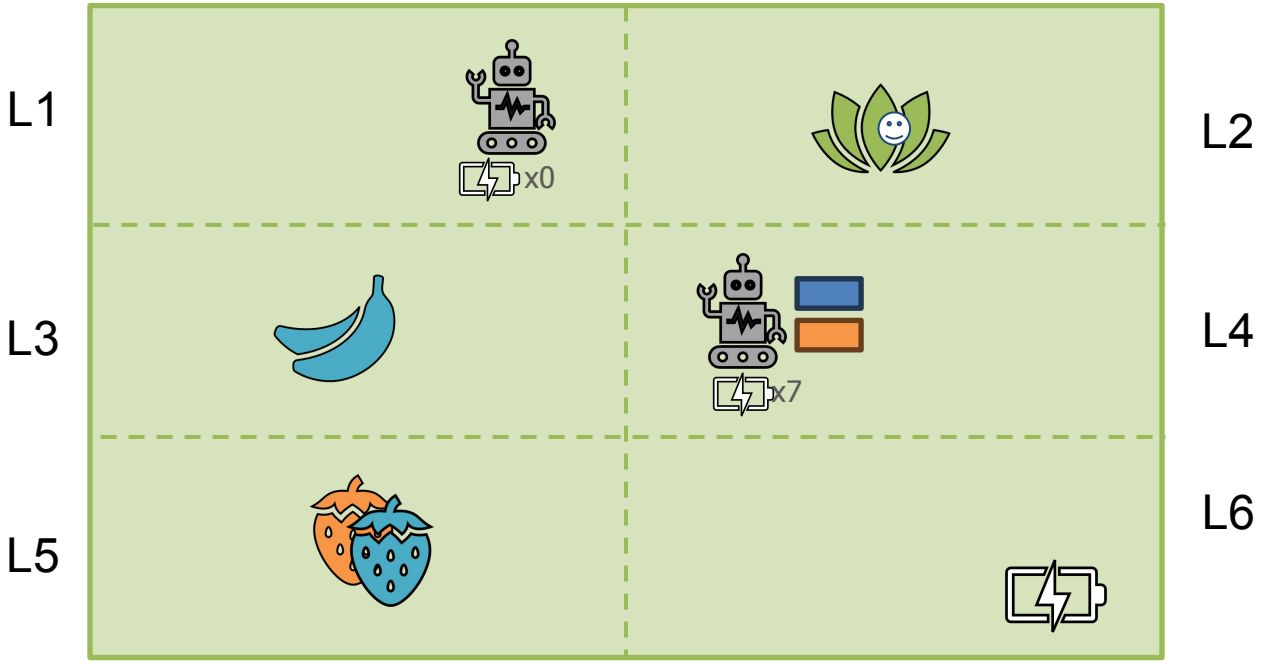


PDDL Problem – (Difficult) Solution

= water

= pesticides

= nutrients

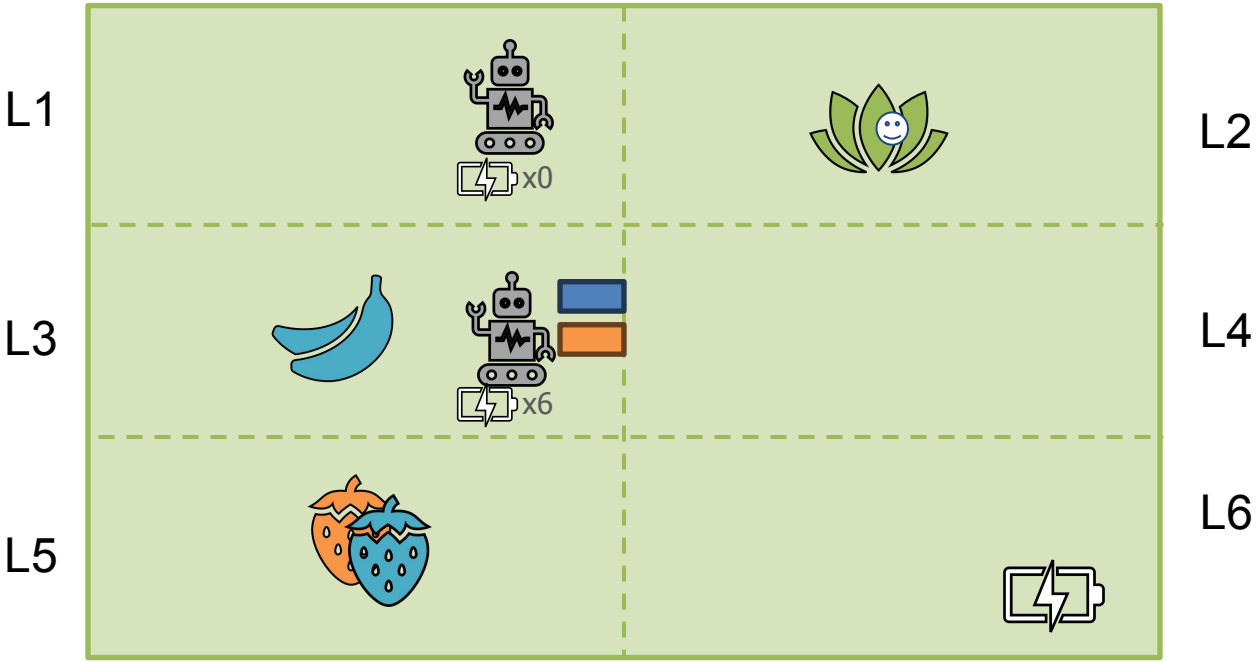


PDDL Problem – (Difficult) Solution

= water

= pesticides

= nutrients

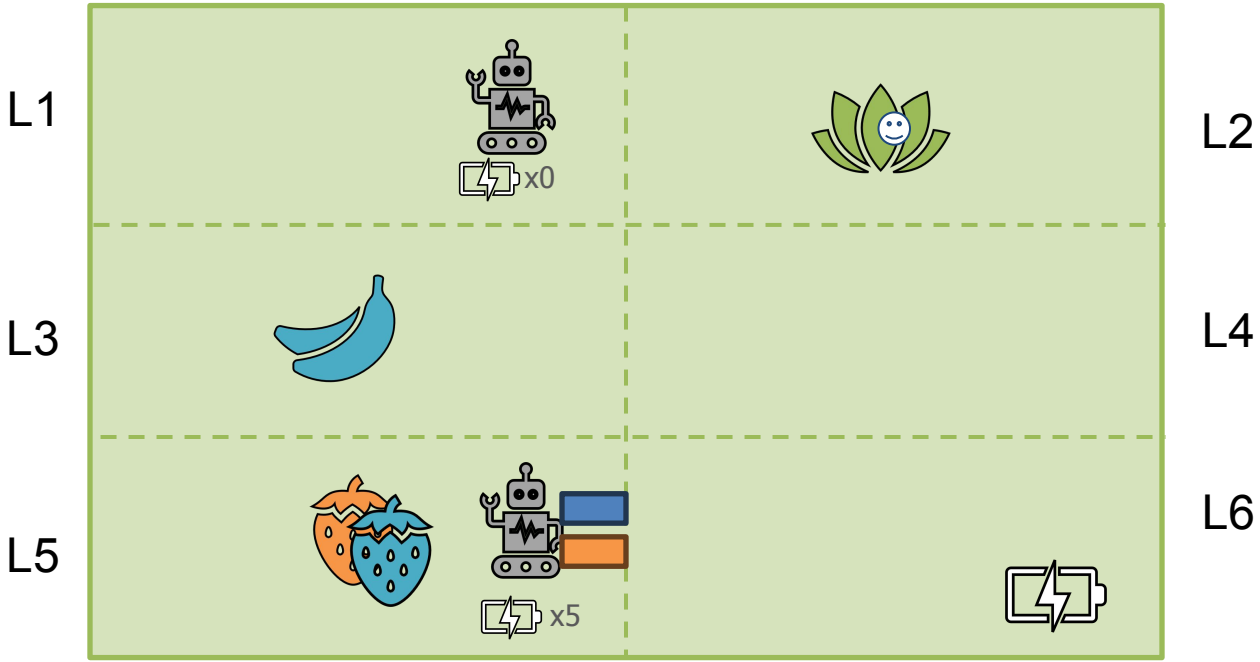


PDDL Problem – (Difficult) Solution

= water

= pesticides

= nutrients

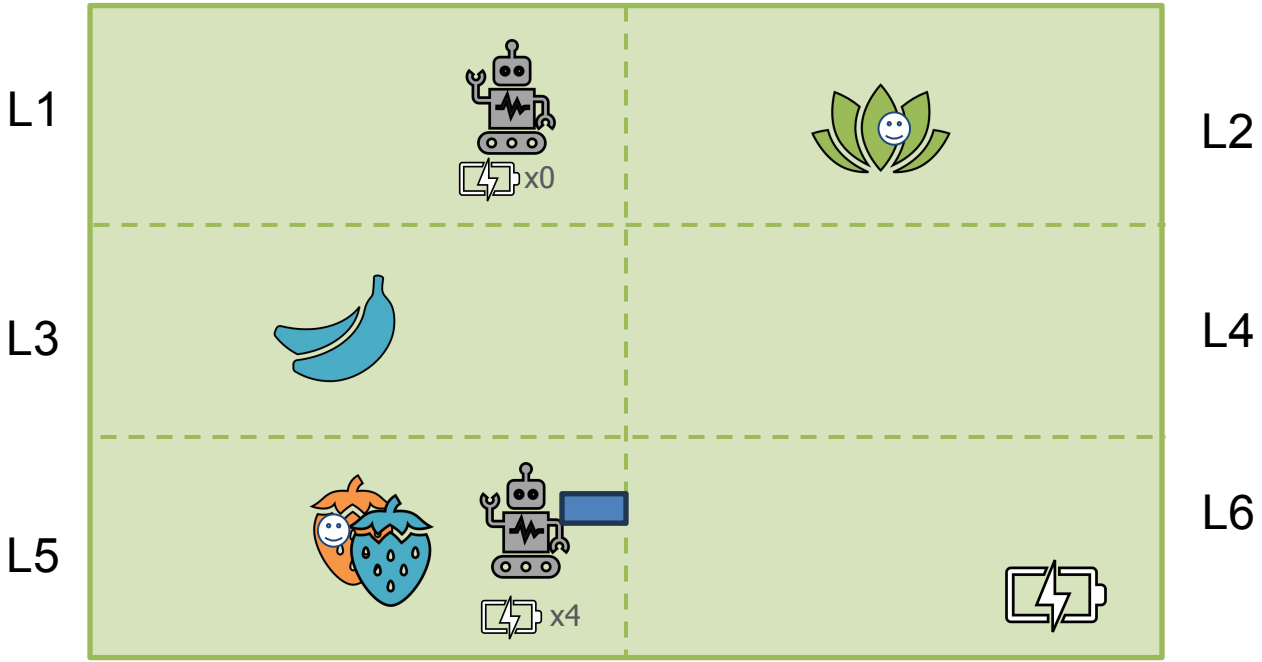


PDDL Problem – (Difficult) Solution

= water

= pesticides

= nutrients

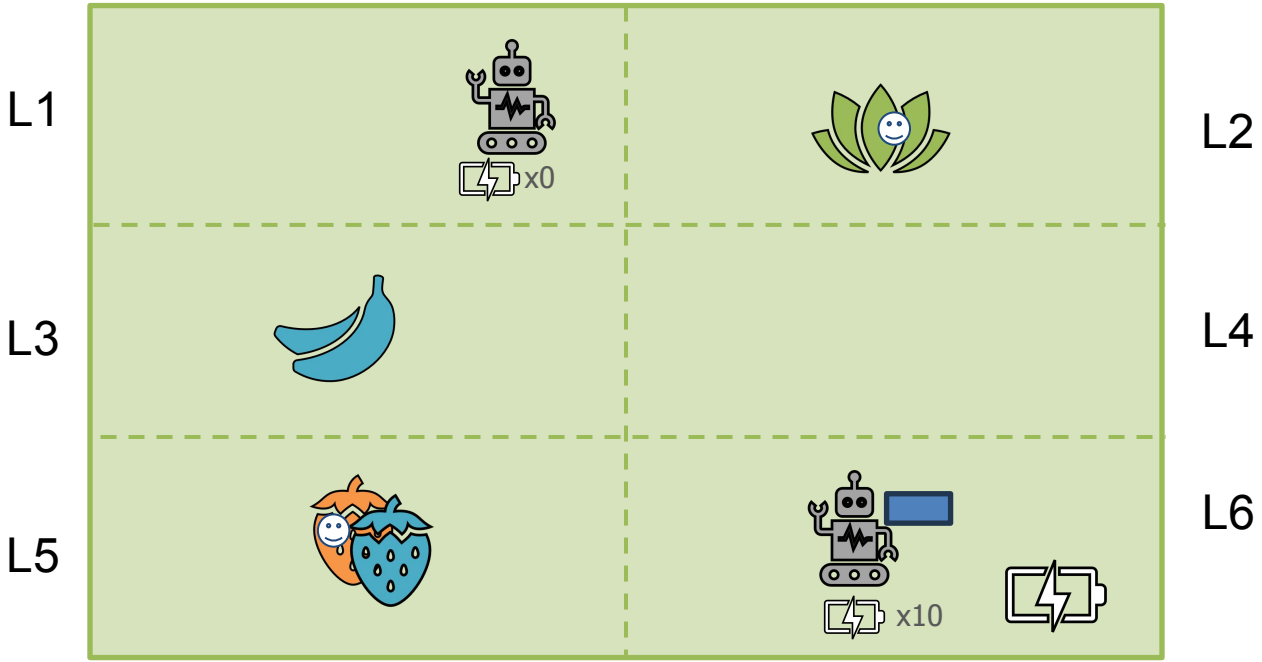


PDDL Problem – (Difficult) Solution

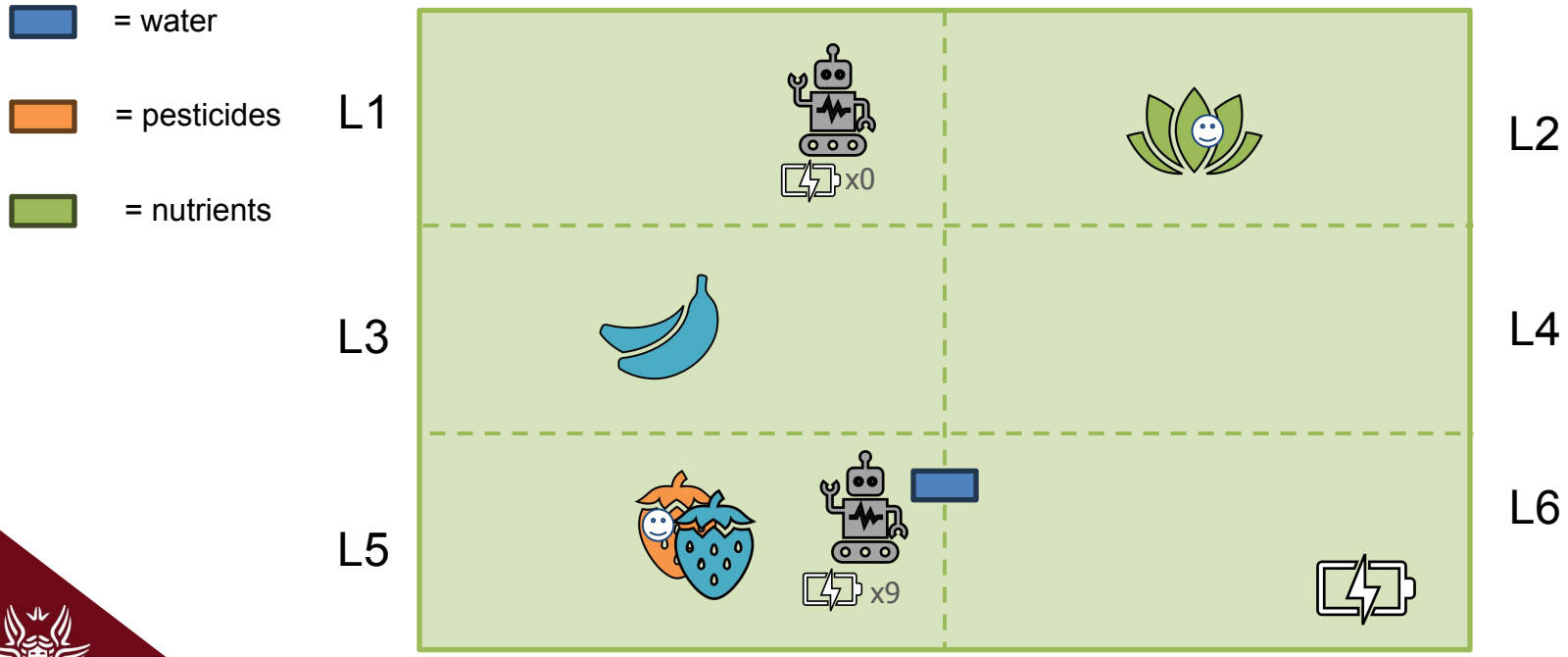
= water

= pesticides

= nutrients



PDDL Problem – (Difficult) Solution

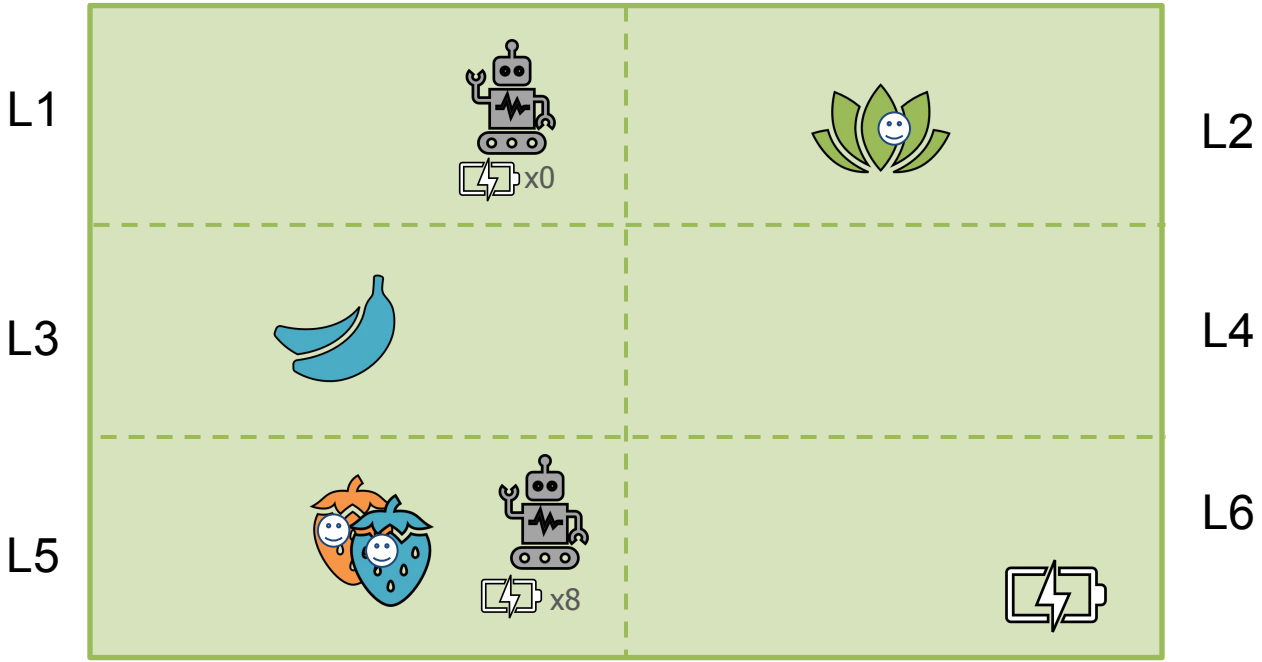


PDDL Problem – (Difficult) Solution

= water

= pesticides

= nutrients

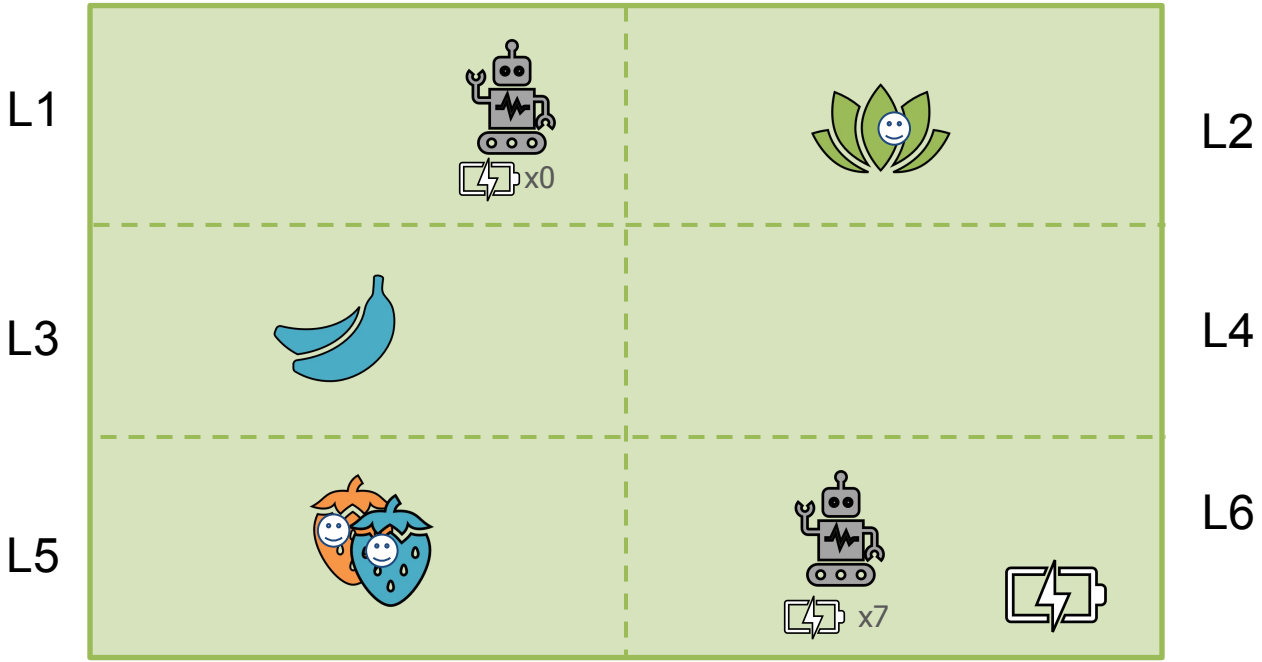


PDDL Problem – (Difficult) Solution

= water

= pesticides

= nutrients

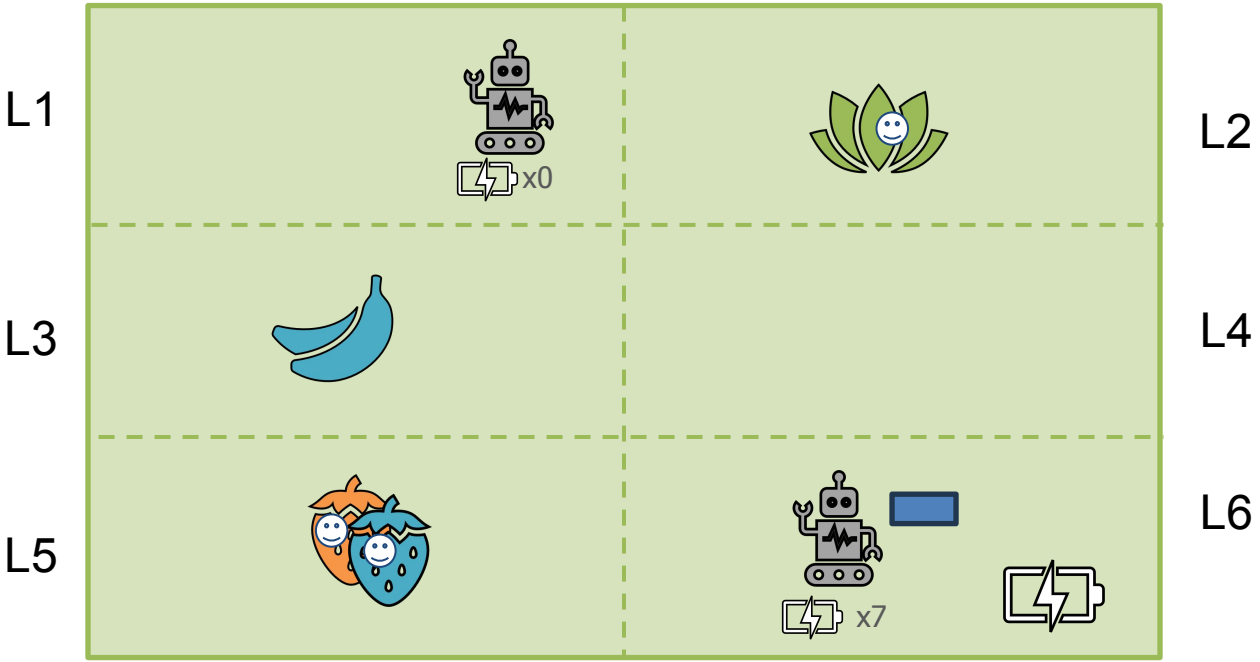


PDDL Problem – (Difficult) Solution

= water

= pesticides

= nutrients

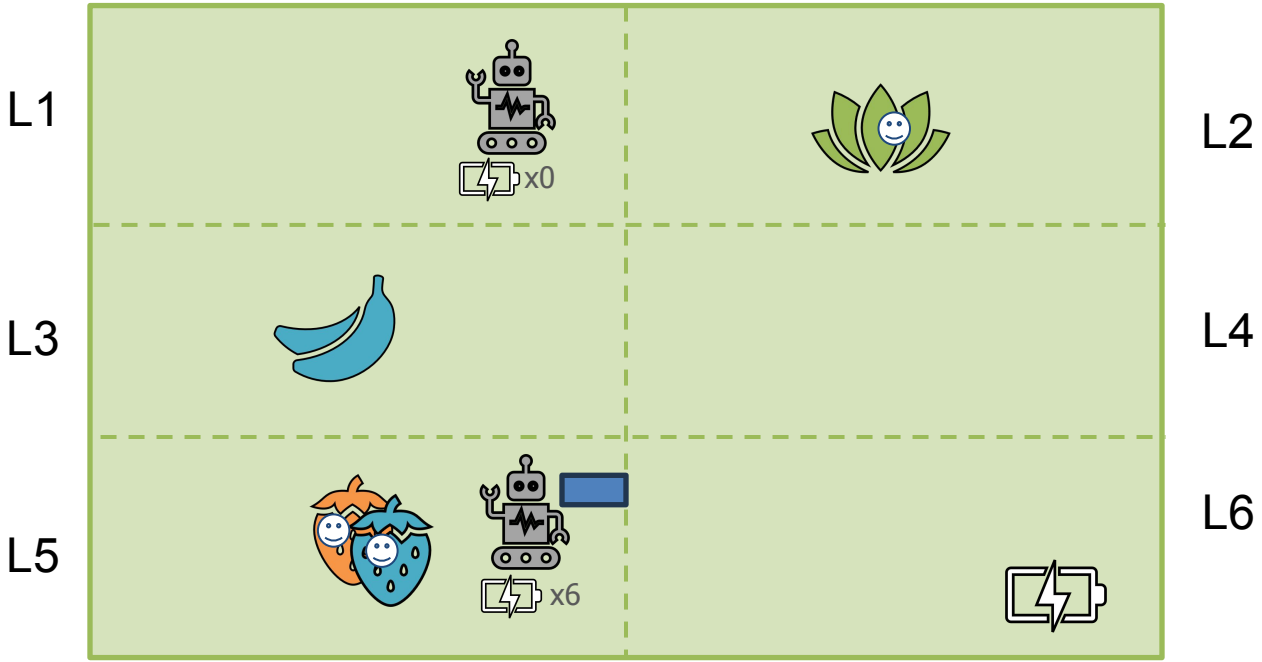


PDDL Problem – (Difficult) Solution

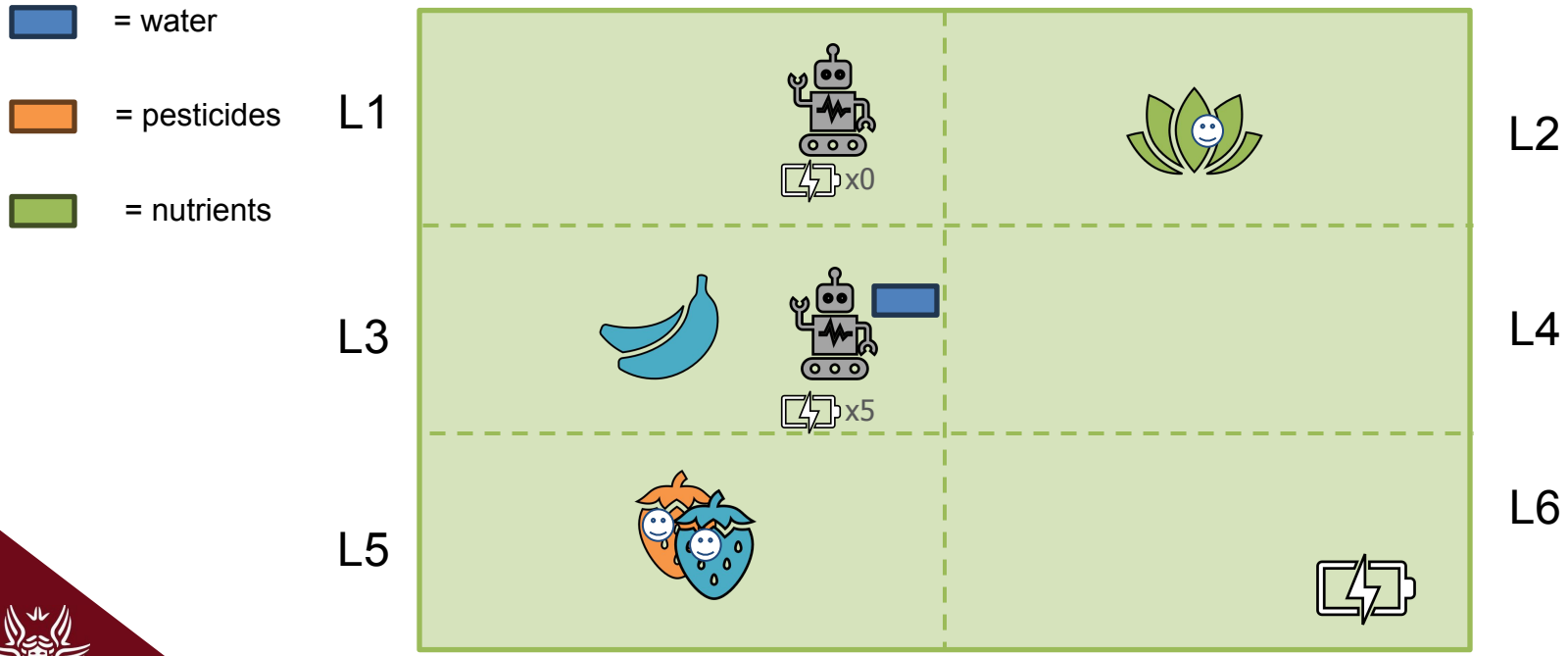
= water

= pesticides

= nutrients



PDDL Problem – (Difficult) Solution

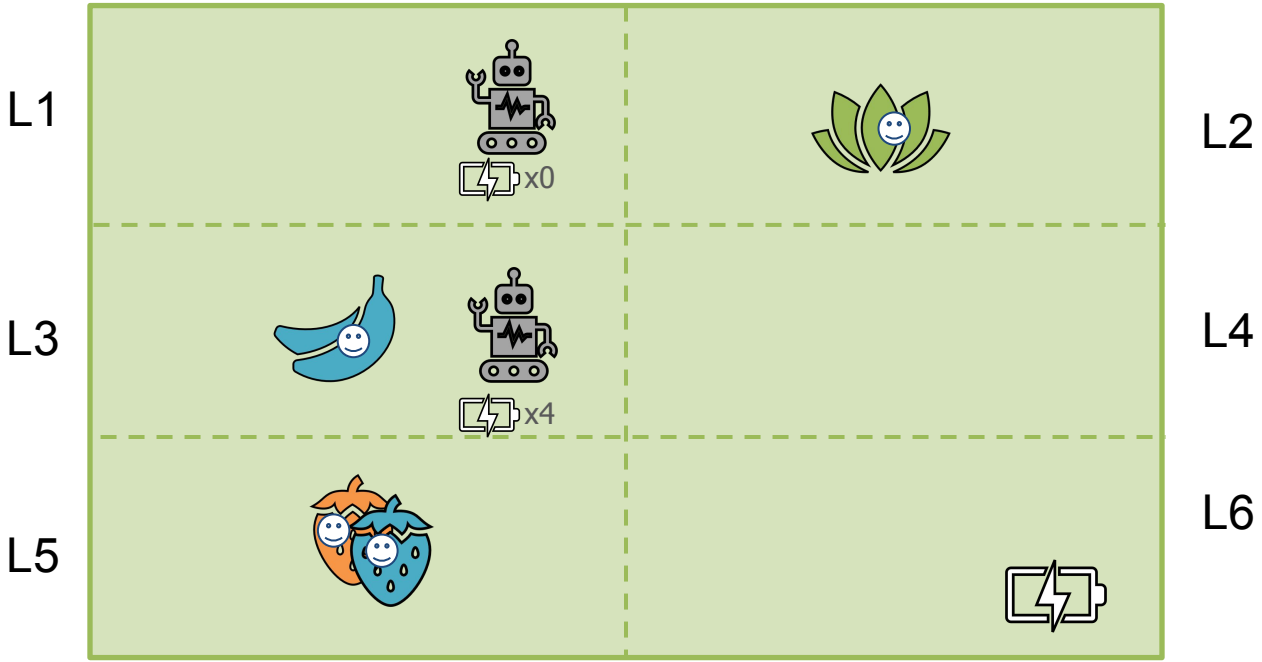


PDDL Problem – (Difficult) Solution

= water

= pesticides

= nutrients



PDDL Problem – (Difficult) Performances

Search	Heuristic	Time (ms)	Expanded Nodes	Length	Cost
Weighted A*	h_max	967	335	58	1024
Weighted A*	h_add	998	428	16	294
Weighted A*	h_ff	689	147	64	854

Situation Calculus Problem

In this case, we have **3 several types** of task:

- **Legality Task** – *Is a single action executable now?*
- **Projection Task** – *What world-state follows a given action sequence?*
- **Controller Task (A*)** – *Which sequence achieves one or more feeding goals with minimum cost?*

Each of them has 3 tasks of increasing difficulty, **easy**, **medium** and **hard**.

Indigolog Domain – Static objects, primitive actions, fluents, constraints

```
robot(r1).  robot(r2).
plant(p1).  plant(p2).  plant(p3).
resource(water).
resource(nutrients).
resource(fertilizer).

cell(c(1,1)). cell(c(1,2)).
cell(c(2,1)). cell(c(2,2)).
cell(c(3,1)). cell(c(3,2)).
```

```
prim_action(move(R,Cf,Ct)).
prim_action(feed_plant(R,P,Res)).
prim_action(charge(R)).
prim_action(transfer_resource(R1,R2,Res,Q)).
prim_action(transfer_energy(R1,R2,Q)).
```

```
adjacent(c(R,C), c(R1,C)) :- R1 is R+1, cell(c(R1,C)).
adjacent(c(R,C), c(R1,C)) :- R1 is R-1, cell(c(R1,C)).
adjacent(c(R,C), c(R,C1)) :- C1 is C+1, cell(c(R,C1)).
adjacent(c(R,C), c(R,C1)) :- C1 is C-1, cell(c(R,C1)).

max_battery(100).

consumption(move,1).
consumption(feed_plant,10).
consumption(transfer_resource,0).
consumption(transfer_energy,1).
consumption(charge,0).
```

```
:- dynamic   at/3, battery/3, stock/4, fed/3.
```

Indigolog Domain – Preconditions

```
poss(move(R,Cf,Ct),S) :-
    robot(R), cell(Cf), cell(Ct), Cf \= Ct,
    at(R,Cf,S), adjacent(Cf,Ct),
    battery(R,L,S), consumption(move,C), L >= C.

poss(feed_plant(R,P,Res),S) :-
    robot(R), plant(P), resource(Res),
    at(R,C,S), plant_at(P,C),
    stock(R,Res,Q,S), Q > 0,
    battery(R,L,S), consumption(feed_plant,T), L >= T,
    \+ fed(P,true,S).

poss(charge(R),S) :-
    robot(R),
    recharge_station(C), at(R,C,S),
    battery(R,L,S), L < 100.
```

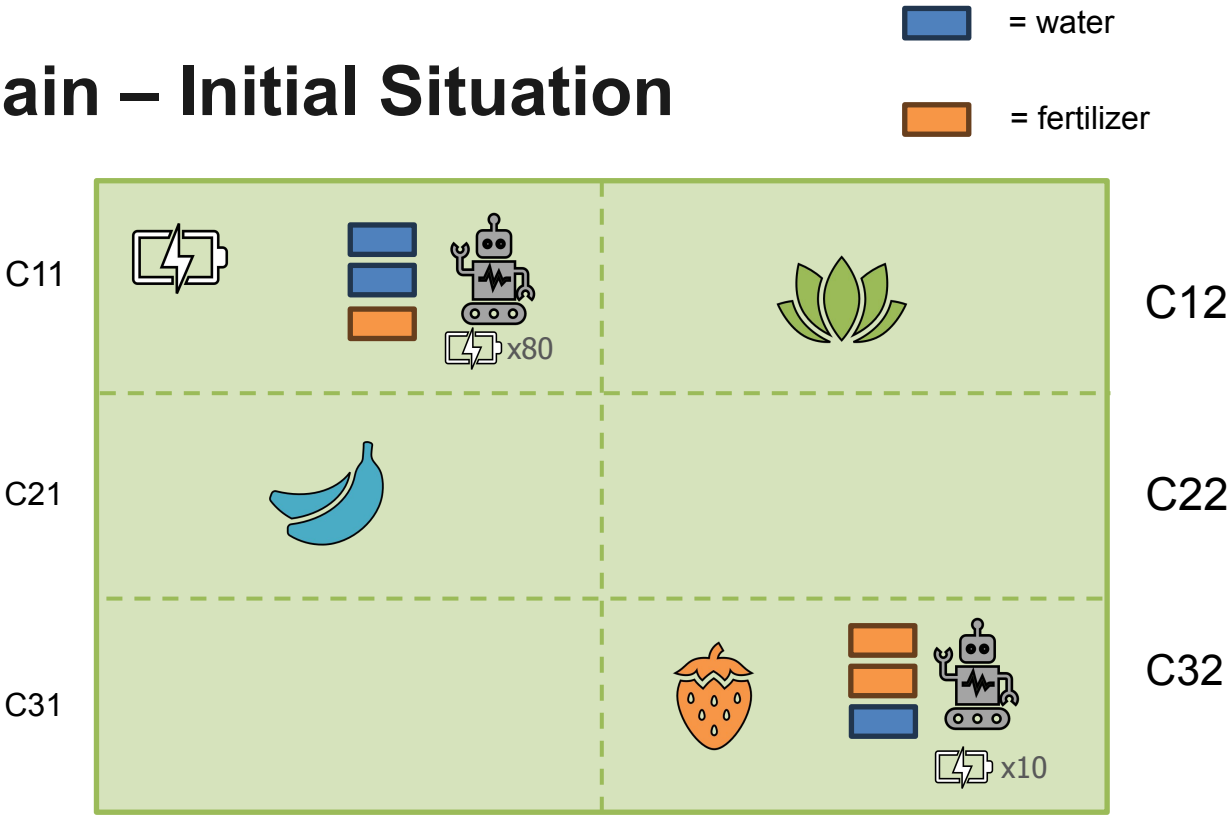
```
poss(transfer_resource(R1,R2,Res,Q),S) :-
    robot(R1), robot(R2), R1 \= R2,
    resource(Res),
    stock(R1,Res,Qmax,S), Qmax > 0,
    Q = 1,                          % istanzia Q (1..Qmax)
    at(R1,C,S), at(R2,C,S).

poss(transfer_energy(R1,R2,Q),S) :-
    robot(R1), robot(R2), R1 \= R2,
    battery(R1,L1,S), L1 > 0,
    Q = 1,                           % istanzia Q (1..L1)
    at(R1,C,S), at(R2,C,S),
    battery(R2,L2,S), max_battery(Max), L2+Q =< Max.
```

Indigolog Domain – Successor State Axioms

```
at(R,C2,do(A,S)) :-
    ( A = move(R,_,C2) )
 ;  ( at(R,C2,S), A \= move(R,_,_) ).

battery(R,L2,do(A,S)) :-
    battery(R,L1,S),
    (   A = move(R,_,_),        consumption(move,C),            L2 is L1-C
    ;   A = feed_plant(R,_,_), consumption(feed_plant,C),       L2 is L1-C
    ;   A = charge(R),          max_battery(L2)
    ;   A = transfer_energy(R,_,Q),                             L2 is L1-Q
    ;   A = transfer_energy(_,R,Q),                             L2 is L1+Q
    ;                          L2 = L1 ).

stock(R,Res,Q2,do(A,S)) :-
    (   A = feed_plant(R,_,Res)            -> stock(R,Res,Q1,S), Q2 is Q1-1
    ;   A = transfer_resource(R,_,Res,Q)   -> stock(R,Res,Q1,S), Q2 is Q1-Q
    ;   A = transfer_resource(_,R,Res,Q)   -> stock(R,Res,Q1,S), Q2 is Q1+Q
    ;                                         stock(R,Res,Q2,S) ).

fed(P,true,do(A,_)) :- A = feed_plant(_,P,_), !.
fed(P,true,do(_,S)) :- fed(P,true,S).
```

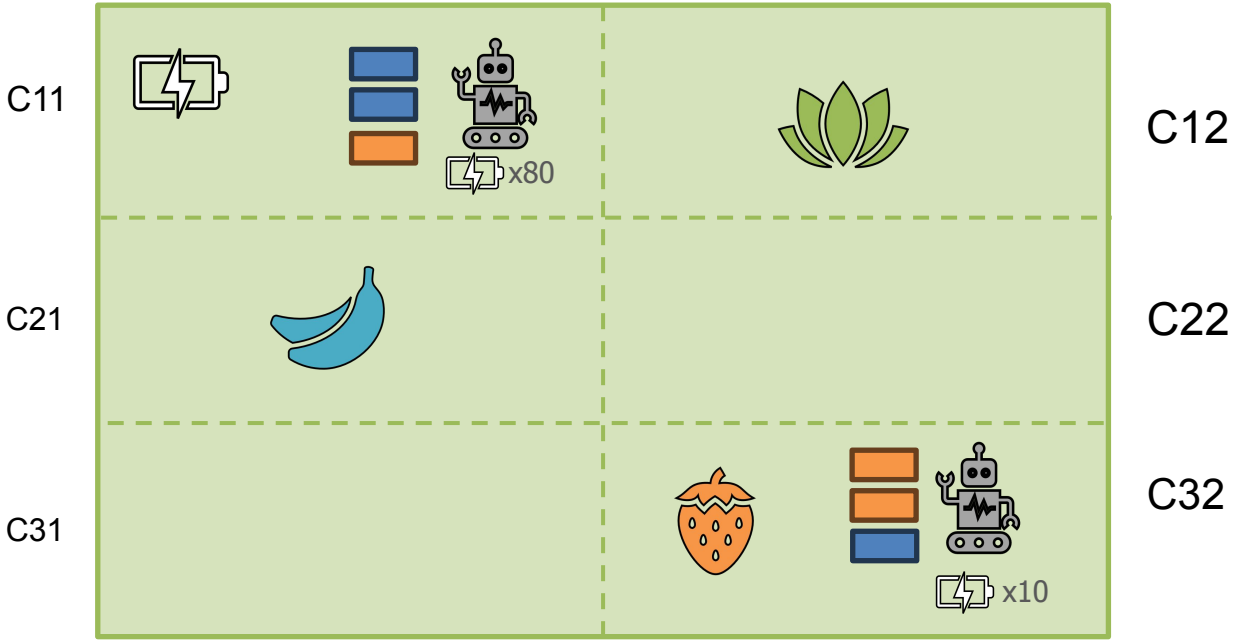
Indigolog Domain – Initial Situation

```
recharge_station(c(1,1)).

plant_at(p1,c(1,2)).
plant_at(p2,c(2,1)).
plant_at(p3,c(3,2)).

at(r1,c(1,1),s0).
at(r2,c(3,2),s0).

battery(r1,80,s0).
battery(r2,10,s0).

stock(r1,water,2,s0).
stock(r1,fertilizer,1,s0).
stock(r2,water,1,s0).
stock(r2,fertilizer,2,s0).
```



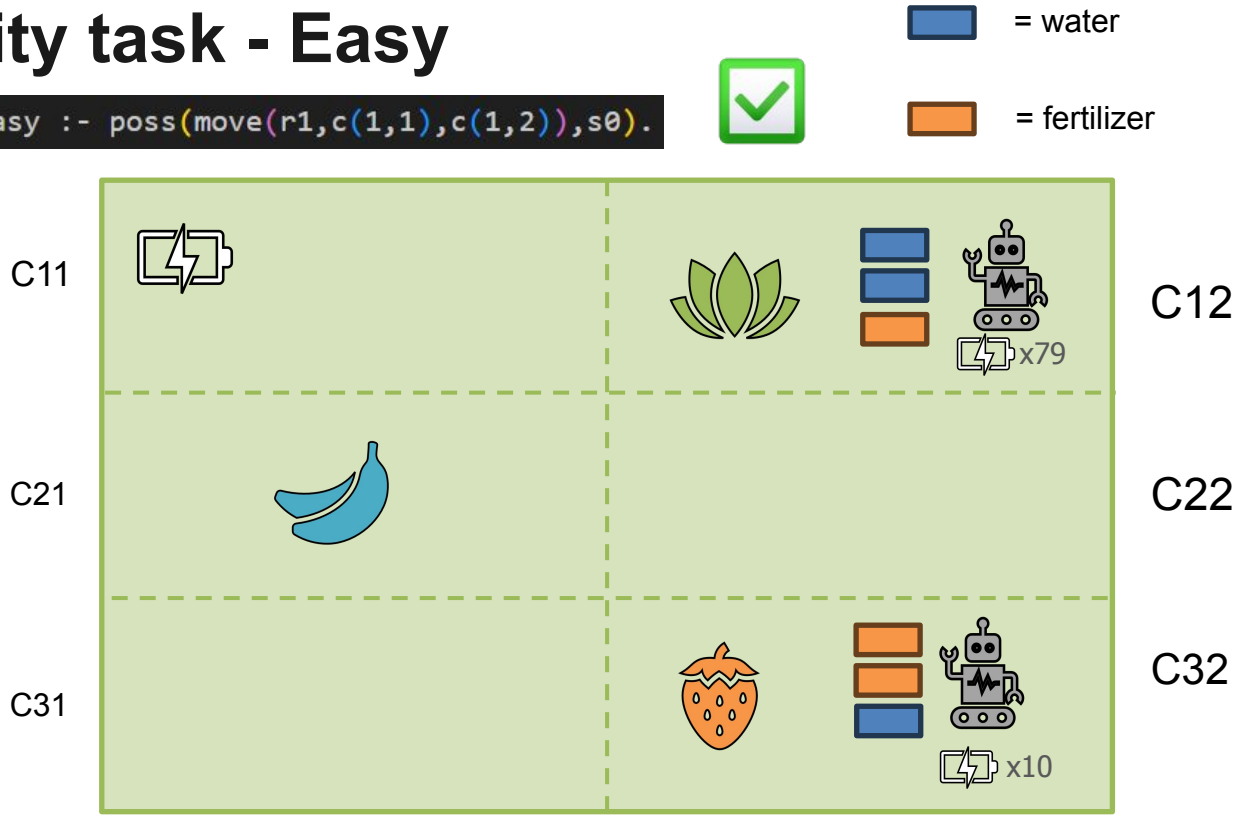
Legality task - Easy

```
legality_easy :- poss(move(r1,c(1,1),c(1,2)),s0).
```



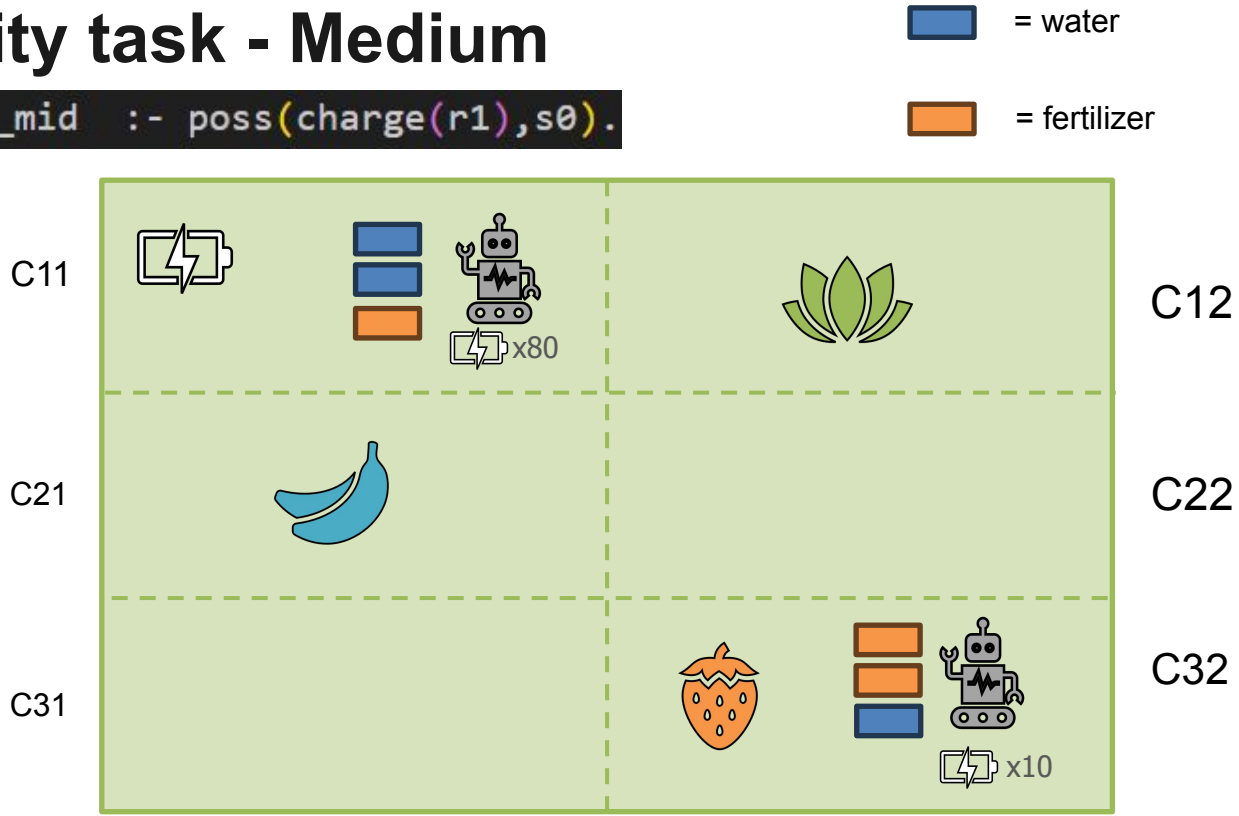
Legality task - Easy

```
legality_easy :- poss(move(r1,c(1,1),c(1,2)),s0).
```



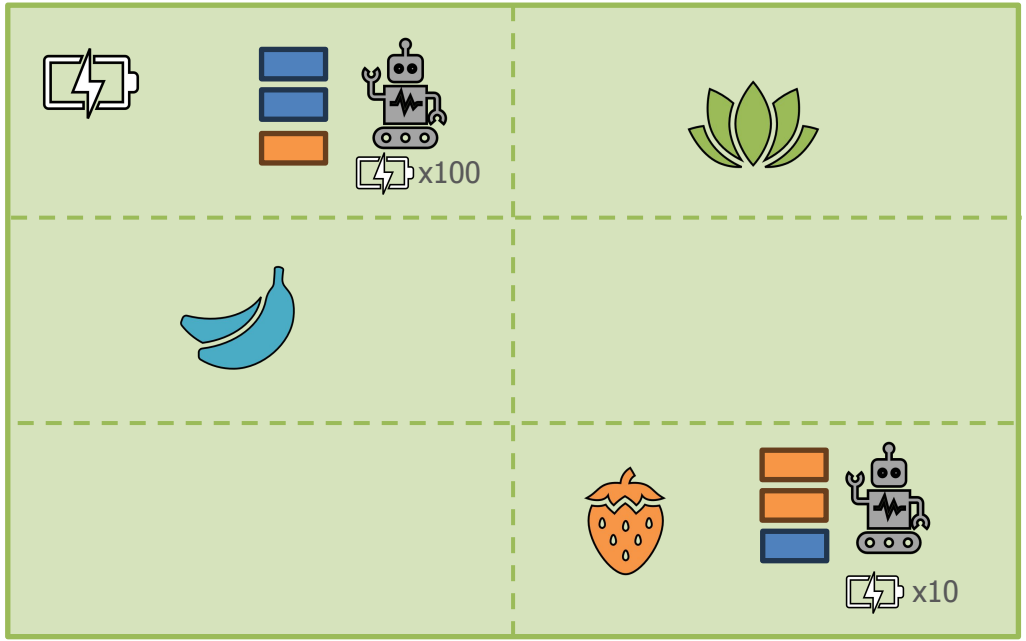
Legality task - Medium

```
legality_mid  :- poss(charge(r1),s0).
```



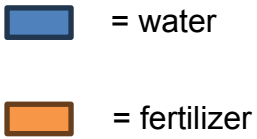
Legality task - Medium

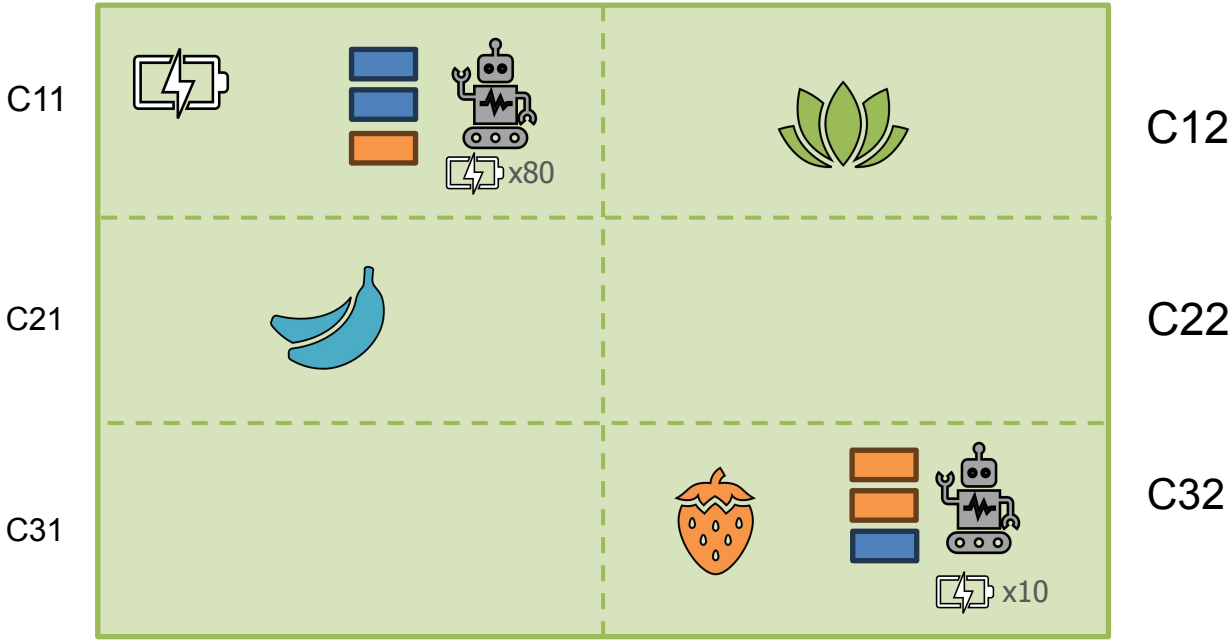
```
legality_mid  :- poss(charge(r1),s0).
```



Legality task - Hard

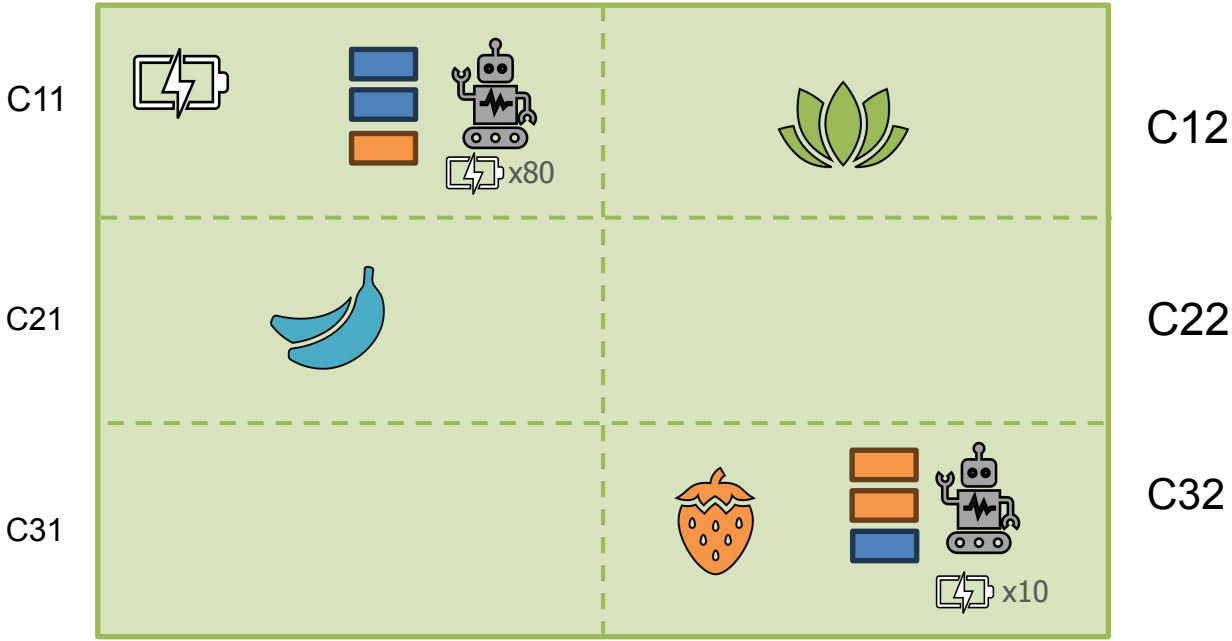
```
legality_hard :- poss(transfer_resource(r2,r1,fertilizer,1),s0).
```





Legality task - Hard

```
legality_hard :- poss(transfer_resource(r2,r1,fertilizer,1),s0).
```

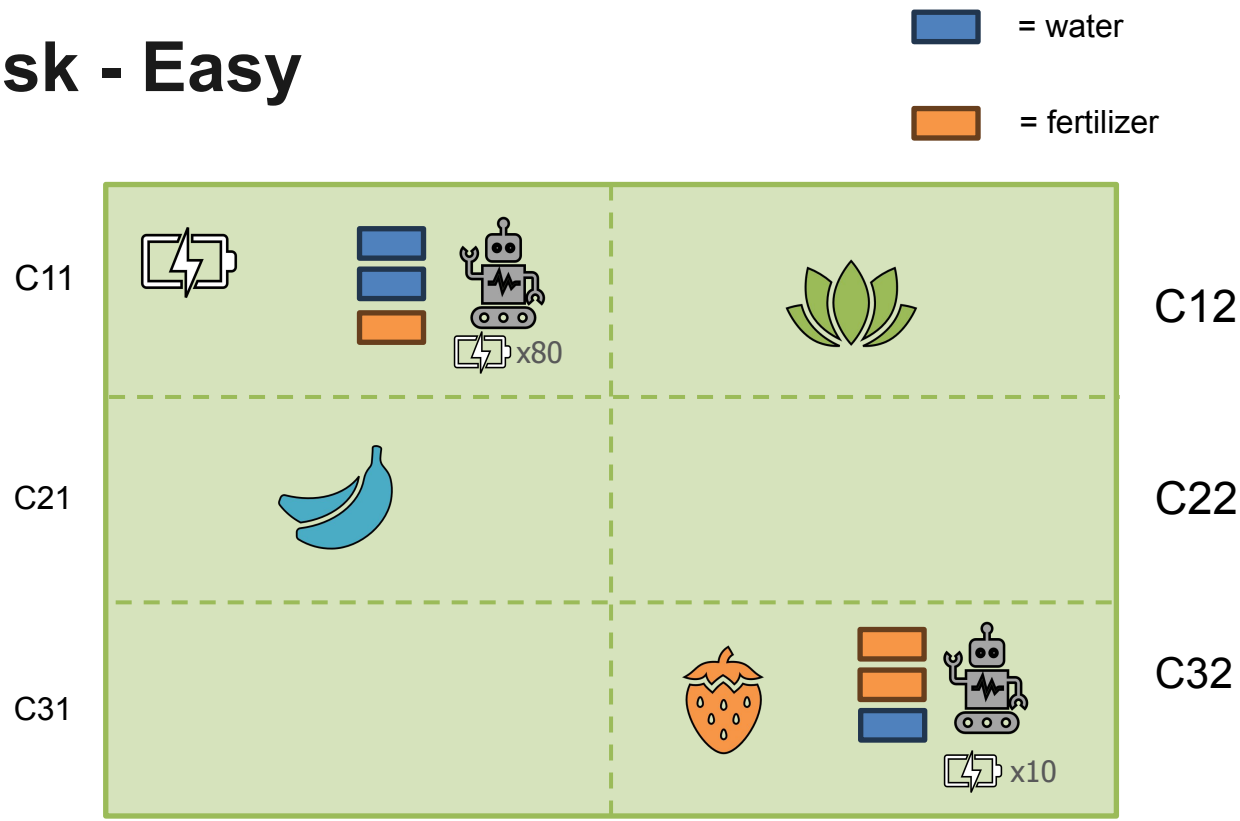


Legality task - Results

```
legality_easy:  true
legality_mid:   true
legality_hard:  false
```

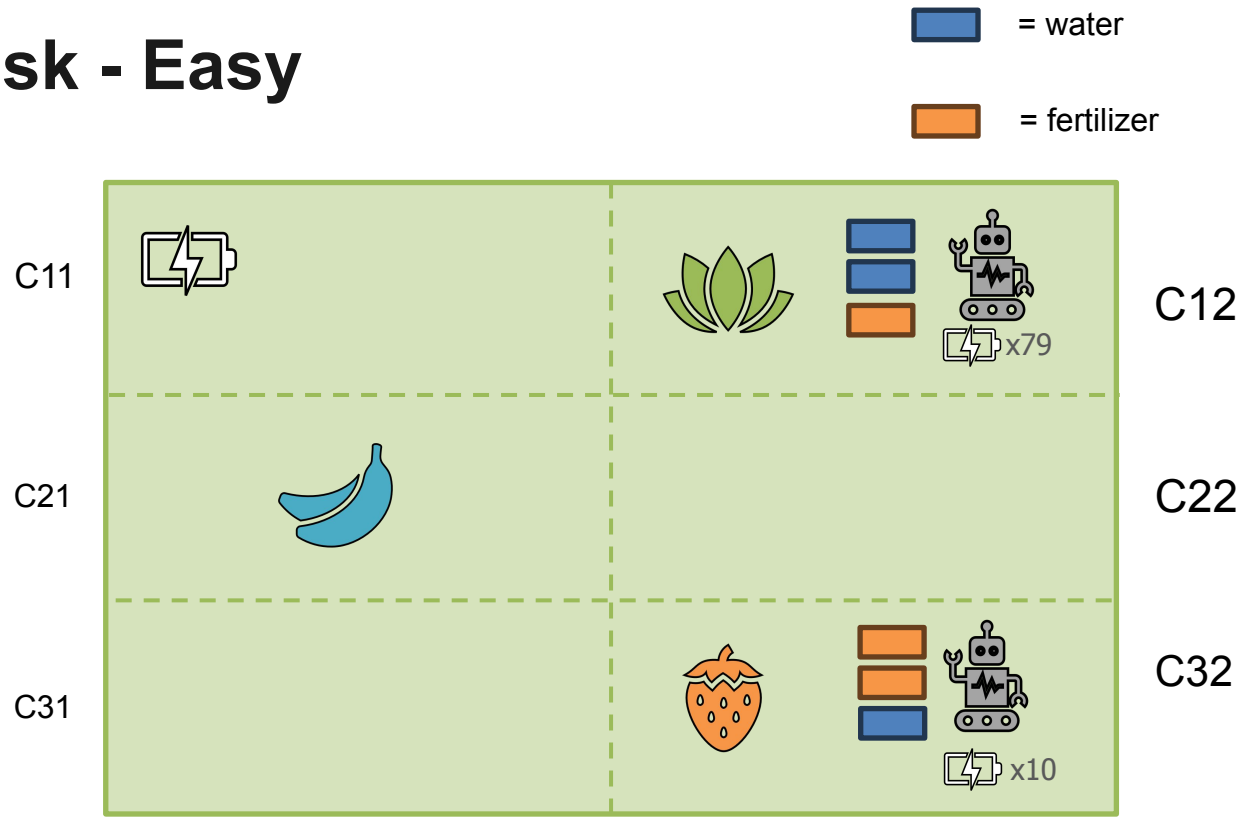
Projection task - Easy

```
projection_easy(Sf) :-
    Actions = [
        move(r1,c(1,1),c(1,2)),
        feed_plant(r1,p1,water)
    ],
    proj(Actions,s0,Sf).
```



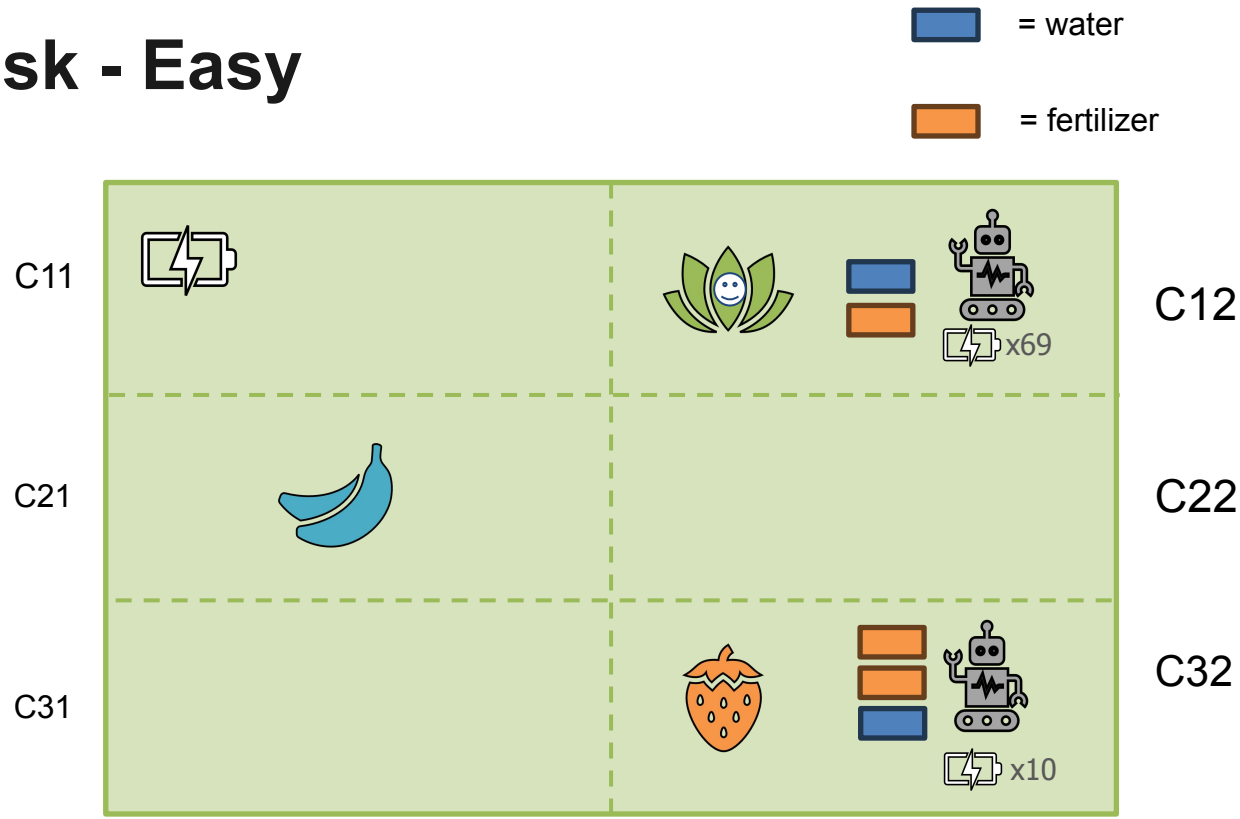
Projection task - Easy

```
projection_easy(Sf) :-
    Actions = [
        move(r1,c(1,1),c(1,2)),
        feed_plant(r1,p1,water)
    ],
    proj(Actions,s0,Sf).
```



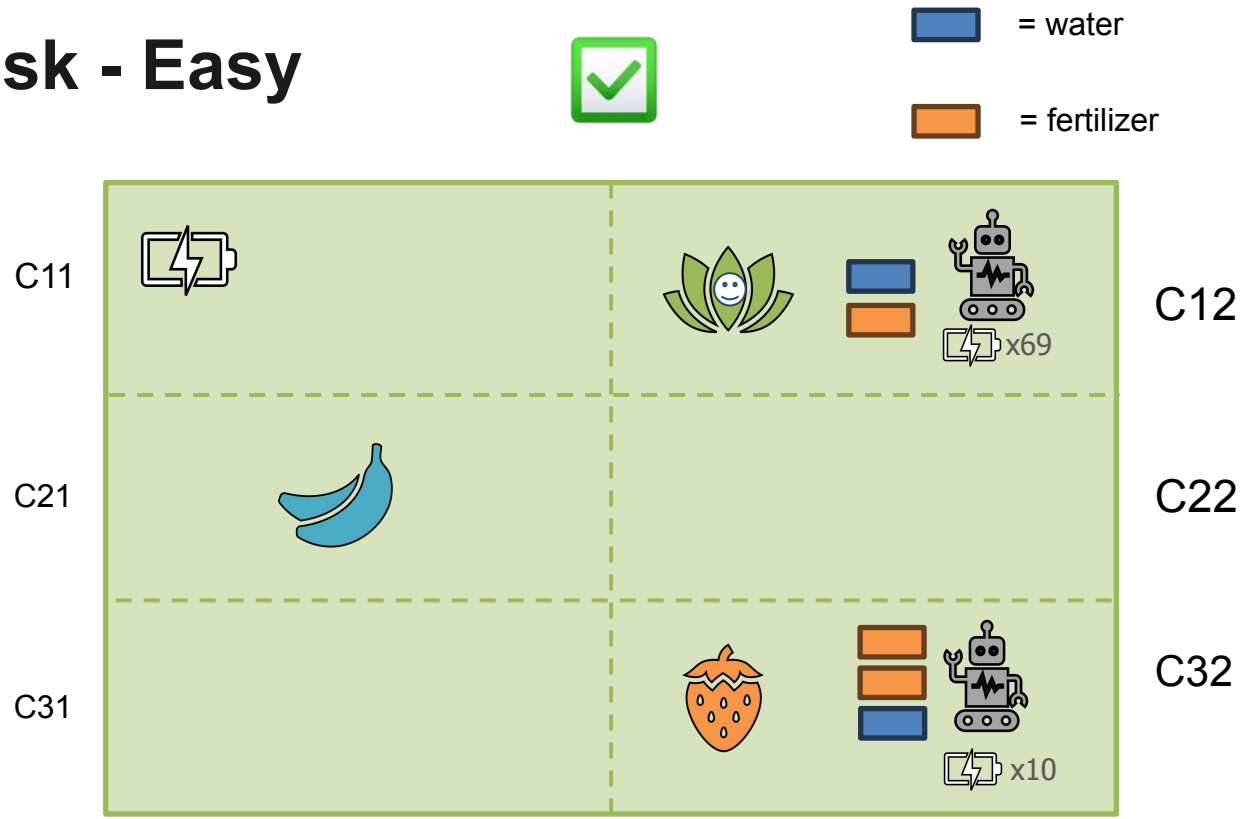
Projection task - Easy

```
projection_easy(Sf) :-
    Actions = [
        move(r1,c(1,1),c(1,2)),
        feed_plant(r1,p1,water)
    ],
    proj(Actions,s0,Sf).
```



Projection task - Easy

```
projection_easy(Sf) :-
    Actions = [
        move(r1,c(1,1),c(1,2)),
        feed_plant(r1,p1,water)
    ],
    proj(Actions,s0,Sf).
```



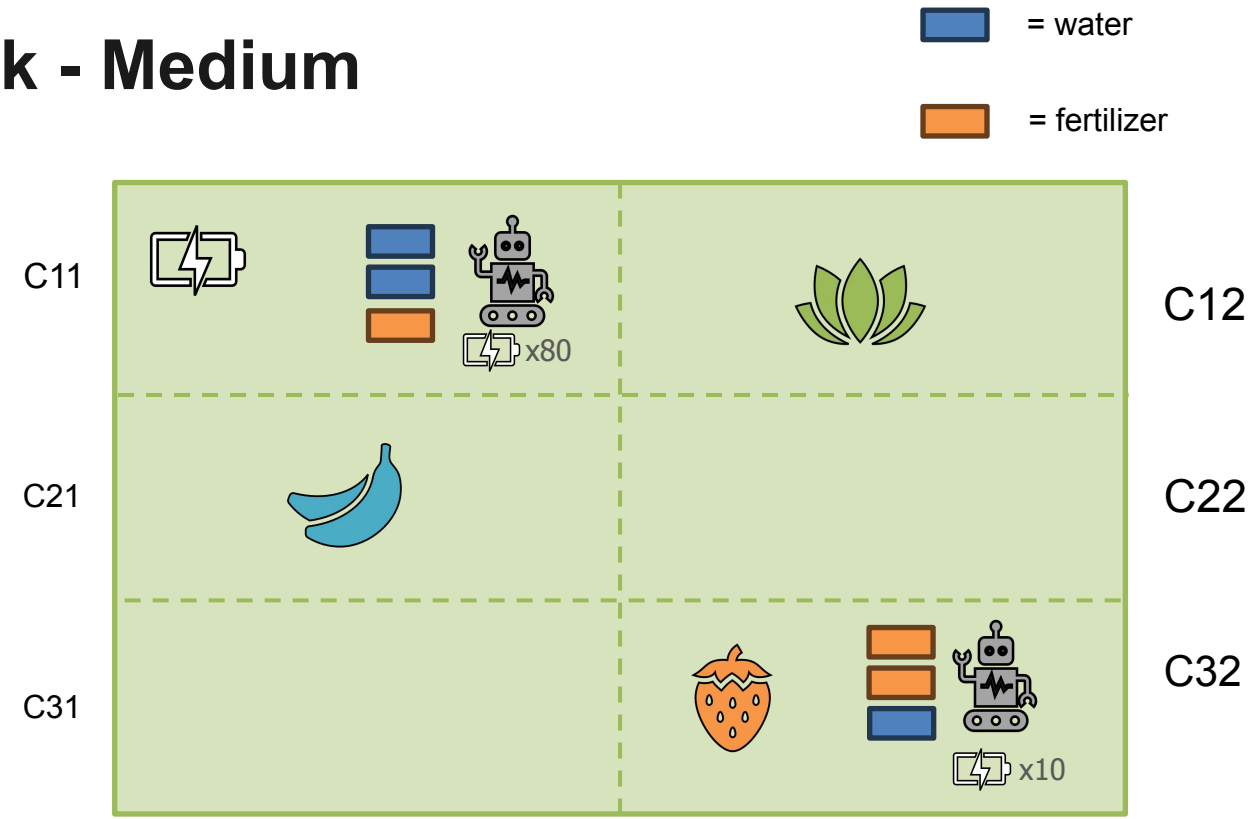
Projection task - Easy - Results

```
projection_easy(Sf) :-
    Actions = [
        move(r1,c(1,1),c(1,2)),
        feed_plant(r1,p1,water)
    ],
    proj(Actions,s0,Sf).
```

```
projection_easy: OK
  r1 @ c(1,2), battery=69
  r2 @ c(3,2), battery=10
  p1: fed
  p2: not_fed
  p3: not_fed
  Sf = do(feed_plant(r1,p1,water),do(move(r1,c(1,1),c(1,2)),s0))
```

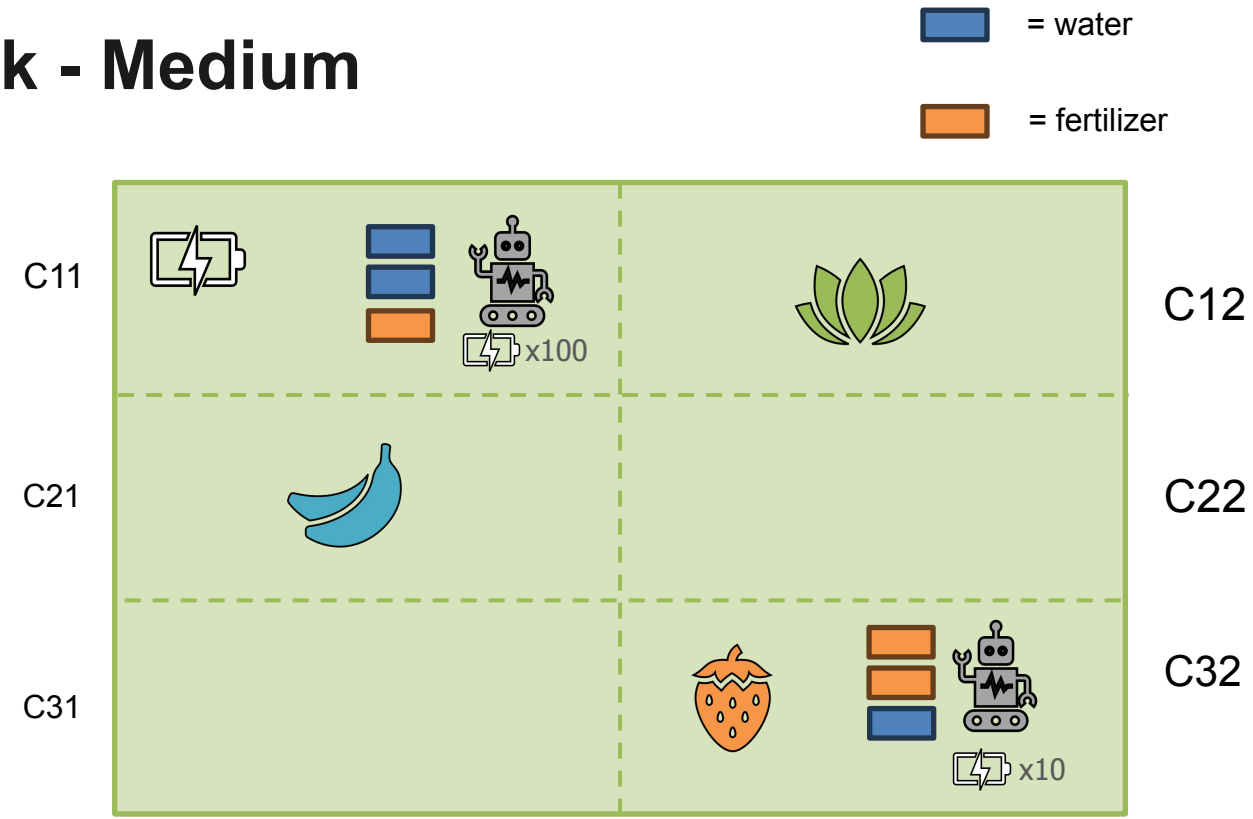
Projection task - Medium

```
projection_mid(Sf) :-
    Actions = [
        charge(r1),
        move(r1,c(1,1),c(2,1)),
        feed_plant(r1,p2,water)
    ],
    proj(Actions,s0,Sf).
```



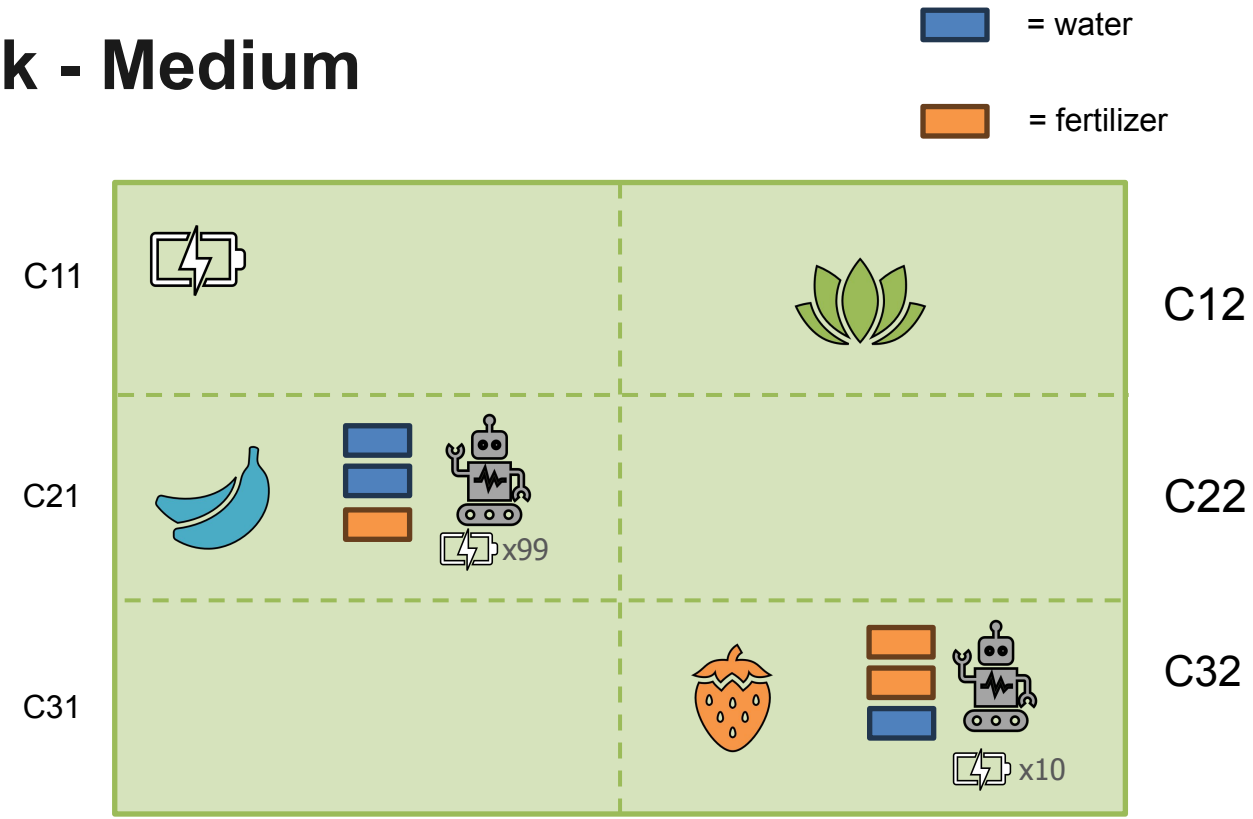
Projection task - Medium

```
projection_mid(Sf) :-
    Actions = [
        charge(r1),
        move(r1,c(1,1),c(2,1)),
        feed_plant(r1,p2,water)
    ],
    proj(Actions,s0,Sf).
```



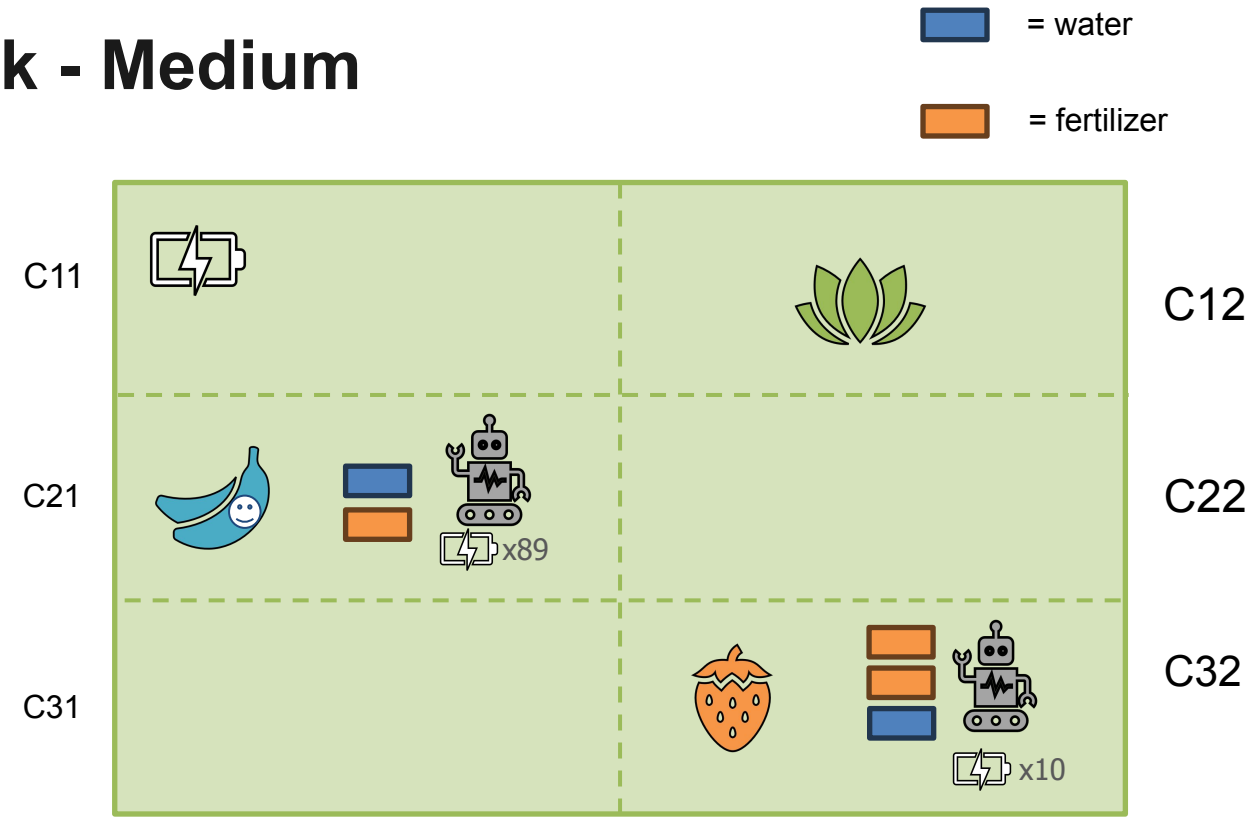
Projection task - Medium

```
projection_mid(Sf) :-
    Actions = [
        charge(r1),
        move(r1,c(1,1),c(2,1)),
        feed_plant(r1,p2,water)
    ],
    proj(Actions,s0,Sf).
```



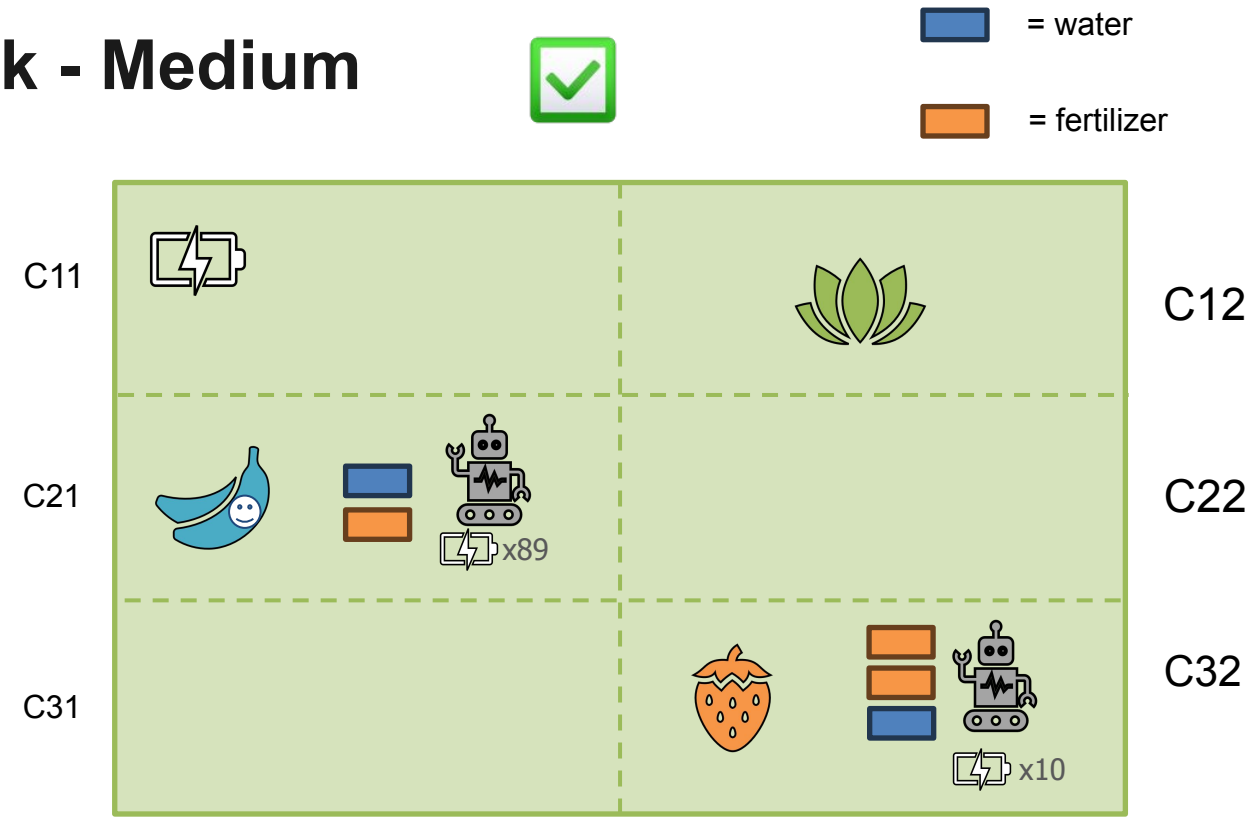
Projection task - Medium

```
projection_mid(Sf) :-
    Actions = [
        charge(r1),
        move(r1,c(1,1),c(2,1)),
        feed_plant(r1,p2,water)
    ],
    proj(Actions,s0,Sf).
```



Projection task - Medium

```
projection_mid(Sf) :-
    Actions = [
        charge(r1),
        move(r1,c(1,1),c(2,1)),
        feed_plant(r1,p2,water)
    ],
    proj(Actions,s0,Sf).
```



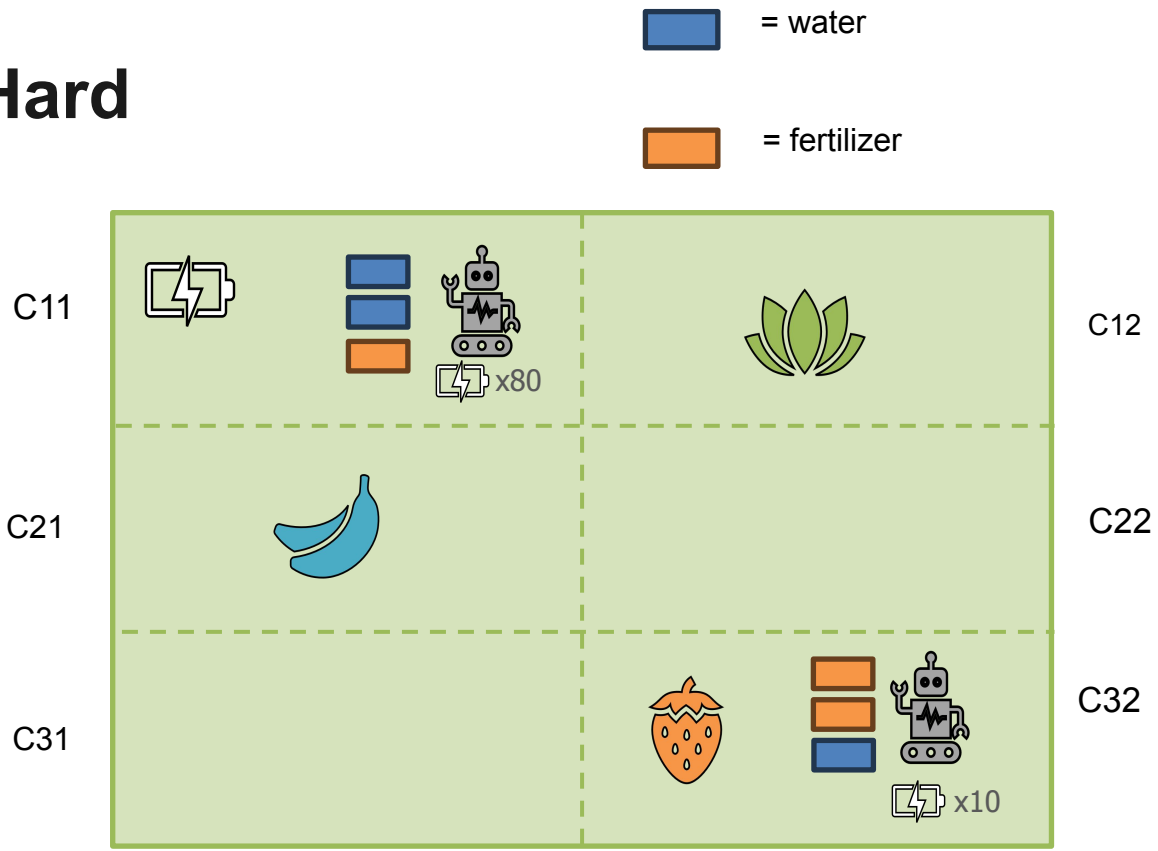
Projection task - Medium - Results

```
projection_mid(Sf) :-
    Actions = [
        charge(r1),
        move(r1,c(1,1),c(2,1)),
        feed_plant(r1,p2,water)
    ],
    proj(Actions,s0,Sf).
```

```
projection_mid:  OK
  r1 @ c(2,1), battery=89
  r2 @ c(3,2), battery=10
  p1: not_fed
  p2: fed
  p3: not_fed
  Sf = do(feed_plant(r1,p2,water),do(move(r1,c(1,1),c(2,1)),do(charge(r1),s0)))
```

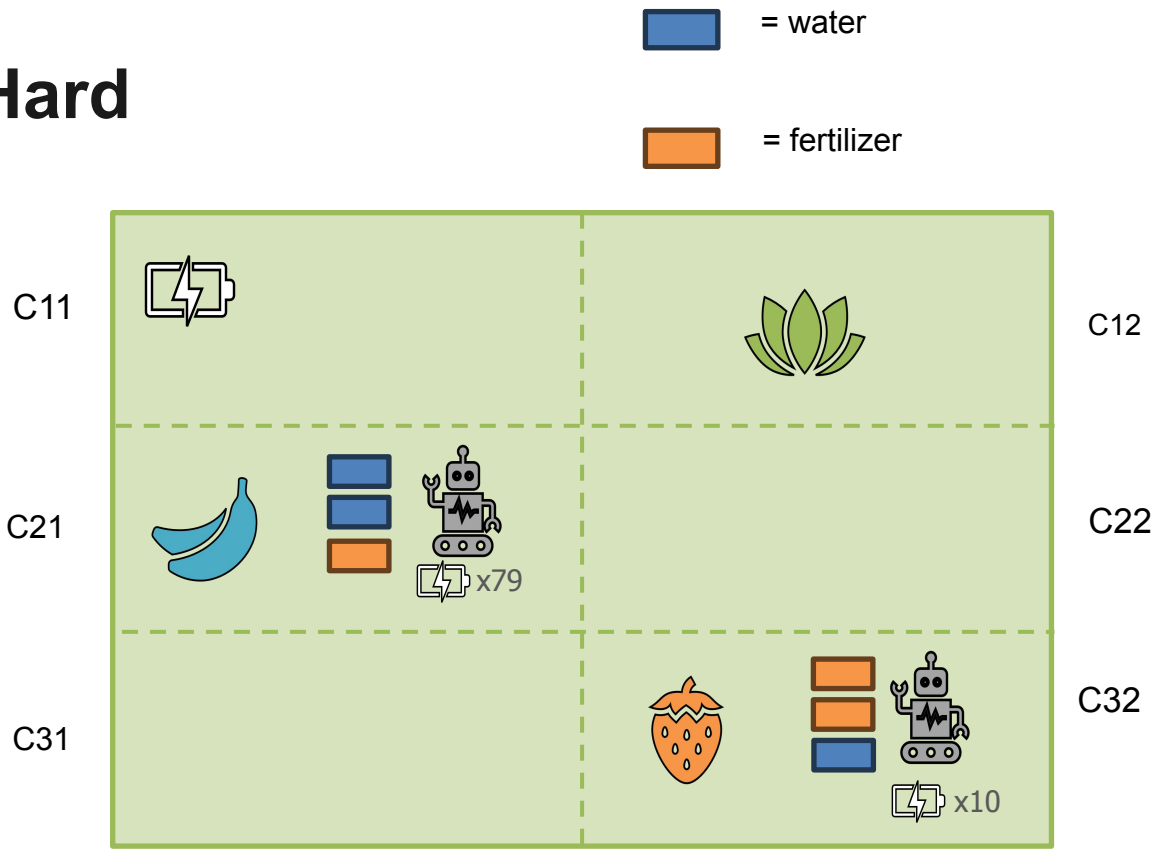
Projection task - Hard

```
projection_hard(Sf) :-
    Actions = [
        move(r1,c(1,1),c(2,1)),
        move(r1,c(2,1),c(2,2)),
        move(r1,c(2,2),c(3,2)),
        transfer_resource(r2,r1,fertilizer,1),
        feed_plant(r2,p3,water)
    ],
    proj(Actions,s0,Sf).
```



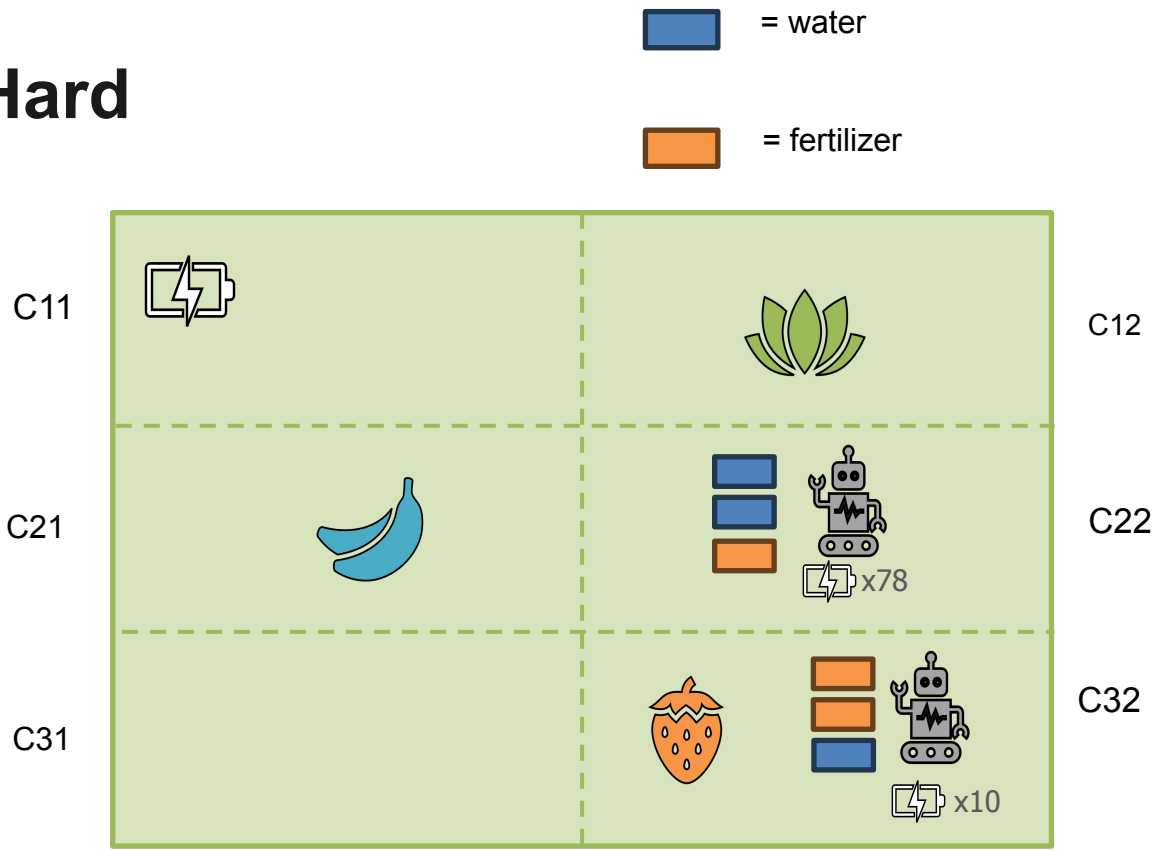
Projection task - Hard

```
projection_hard(Sf) :-
    Actions = [
        move(r1,c(1,1),c(2,1)),
        move(r1,c(2,1),c(2,2)),
        move(r1,c(2,2),c(3,2)),
        transfer_resource(r2,r1,fertilizer,1),
        feed_plant(r2,p3,water)
    ],
    proj(Actions,s0,Sf).
```



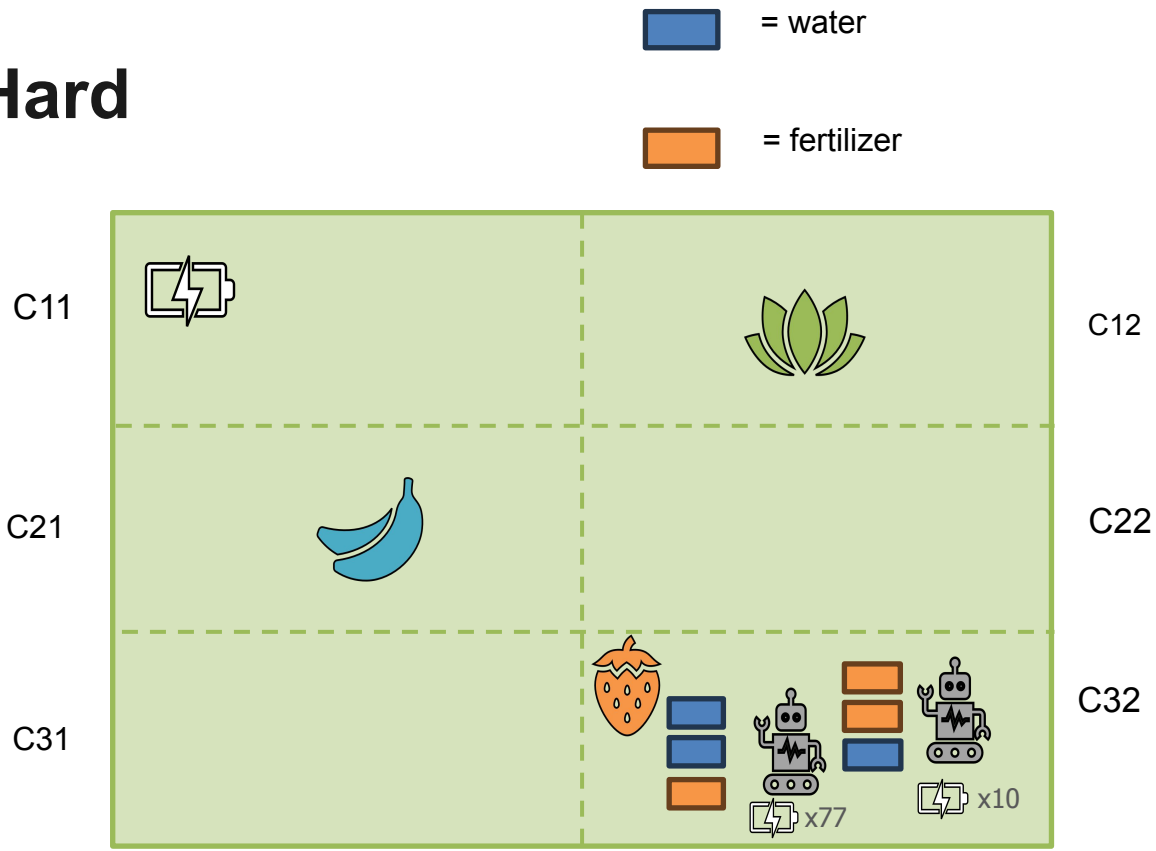
Projection task - Hard

```
projection_hard(Sf) :-
    Actions = [
        move(r1,c(1,1),c(2,1)),
        move(r1,c(2,1),c(2,2)),
        move(r1,c(2,2),c(3,2)),
        transfer_resource(r2,r1,fertilizer,1),
        feed_plant(r2,p3,water)
    ],
    proj(Actions,s0,Sf).
```



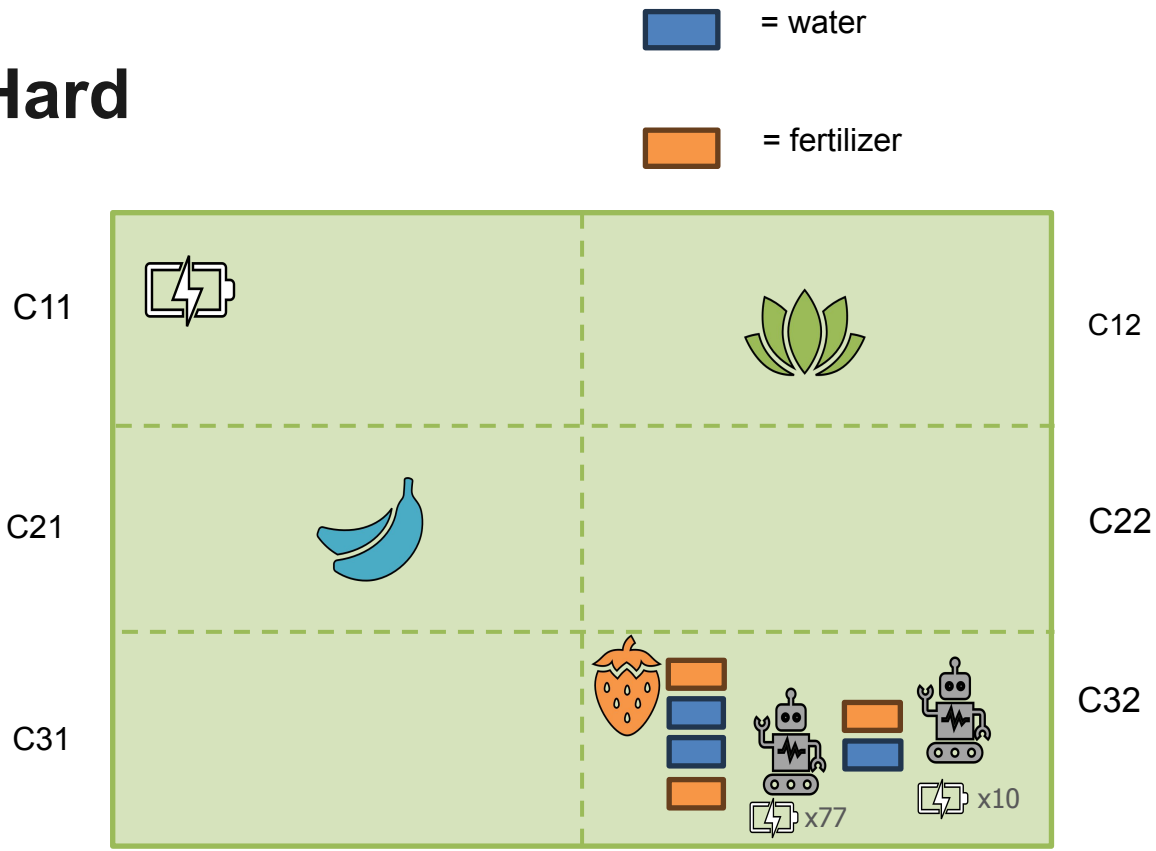
Projection task - Hard

```
projection_hard(Sf) :-
    Actions = [
        move(r1,c(1,1),c(2,1)),
        move(r1,c(2,1),c(2,2)),
        move(r1,c(2,2),c(3,2)),
        transfer_resource(r2,r1,fertilizer,1),
        feed_plant(r2,p3,water)
    ],
    proj(Actions,s0,Sf).
```



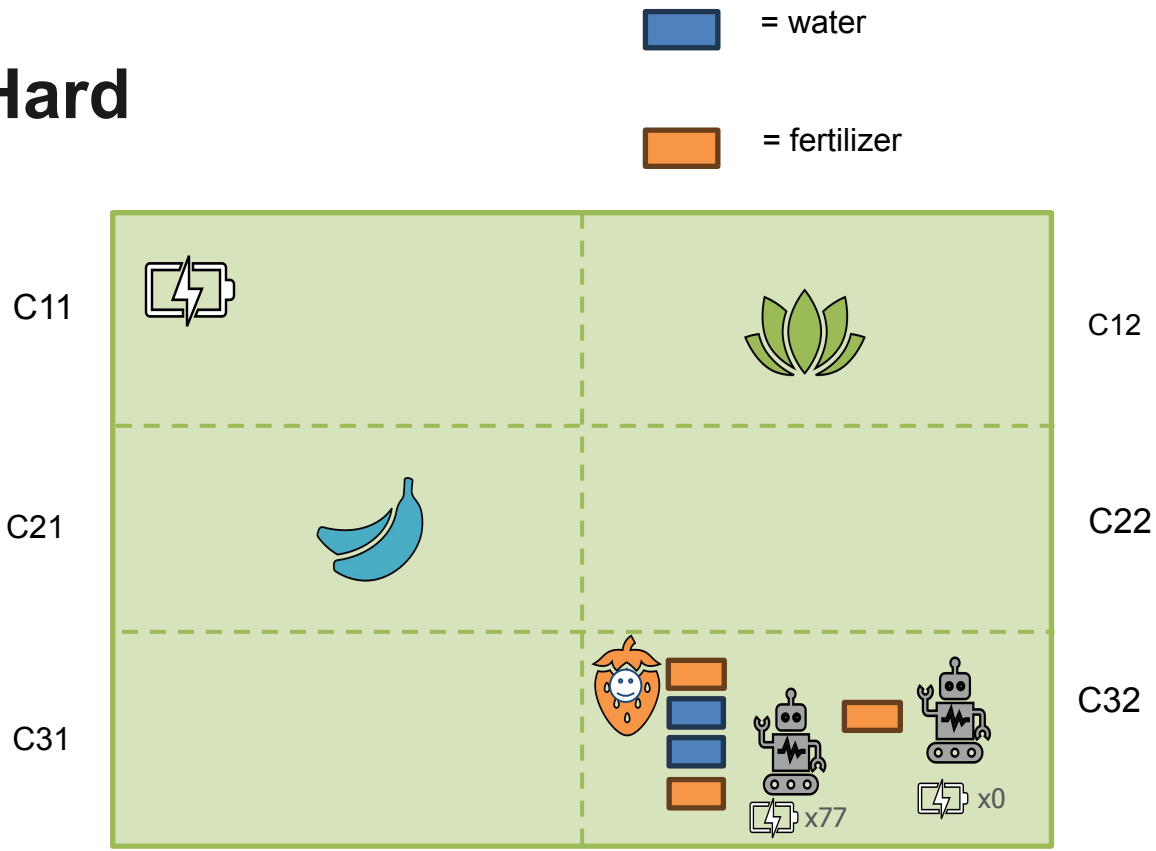
Projection task - Hard

```
projection_hard(Sf) :-
    Actions = [
        move(r1,c(1,1),c(2,1)),
        move(r1,c(2,1),c(2,2)),
        move(r1,c(2,2),c(3,2)),
        transfer_resource(r2,r1,fertilizer,1),
        feed_plant(r2,p3,water)
    ],
    proj(Actions,s0,Sf).
```



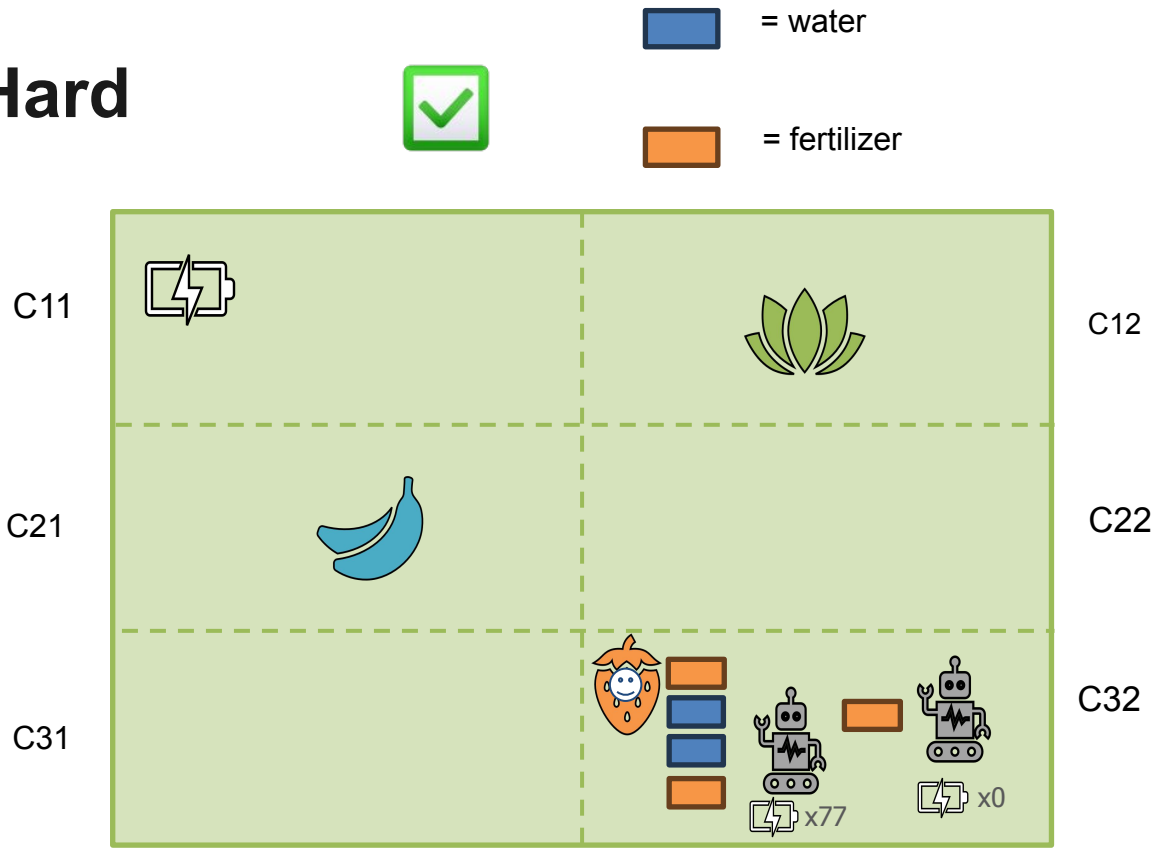
Projection task - Hard

```
projection_hard(Sf) :-
    Actions = [
        move(r1,c(1,1),c(2,1)),
        move(r1,c(2,1),c(2,2)),
        move(r1,c(2,2),c(3,2)),
        transfer_resource(r2,r1,fertilizer,1),
        feed_plant(r2,p3,water)
    ],
    proj(Actions,s0,Sf).
```



Projection task - Hard

```
projection_hard(Sf) :-
    Actions = [
        move(r1,c(1,1),c(2,1)),
        move(r1,c(2,1),c(2,2)),
        move(r1,c(2,2),c(3,2)),
        transfer_resource(r2,r1,fertilizer,1),
        feed_plant(r2,p3,water)
    ],
    proj(Actions,s0,Sf).
```



Projection task - Hard - Results

```
projection_hard(Sf) :-
    Actions = [
        move(r1,c(1,1),c(2,1)),
        move(r1,c(2,1),c(2,2)),
        move(r1,c(2,2),c(3,2)),
        transfer_resource(r2,r1,fertilizer,1),
        feed_plant(r2,p3,water)
    ],
    proj(Actions,s0,Sf).
```

```
projection_hard: OK
  r1 @ c(3,2), battery=77
  r2 @ c(3,2), battery=0
  p1: not_fed
  p2: not_fed
  p3: fed
  Sf = do(feed_plant(r2,p3,water),do(transfer_resource(r2,r1,fertilizer,1),
       do(move(r1,c(2,2),c(3,2)),do(move(r1,c(2,1),c(2,2)),do(move(r1,c(1,1),c(2,1)),s0)))))
```

Controller A* task - Goals and Heuristic

```
goal_p1(S) :-
    fed(p1, true, S).

goal_p1p2(S) :-
    fed(p1, true, S),
    fed(p2, true, S).

goal_p1p2p3(S) :-
    fed(p1, true, S),
    fed(p2, true, S),
    fed(p3, true, S).
```

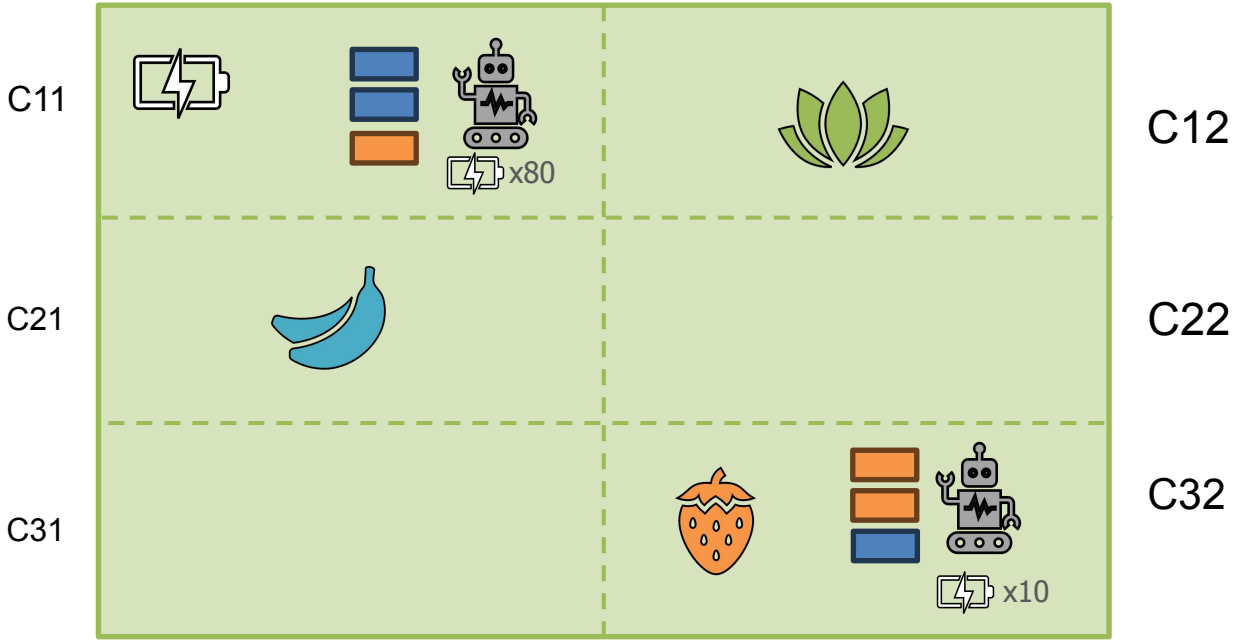
```
h(S, H) :-
    ( goal(S) ->
        H = 0
    ; at(r1, C, S),
      remaining_plants(S, Ps),
      findall(D,
          ( member(P, Ps),
            plant_at(P, Cp),
            manhattan(C, Cp, D)
          ),
          Ds),
      min_list(Ds, H)
    ).
```

The **Heuristic function h(x)** is the minimum (Manhattan) distance between the robot and the nearest plant.

In the A* algorithm, **g(x)** is the plan length.

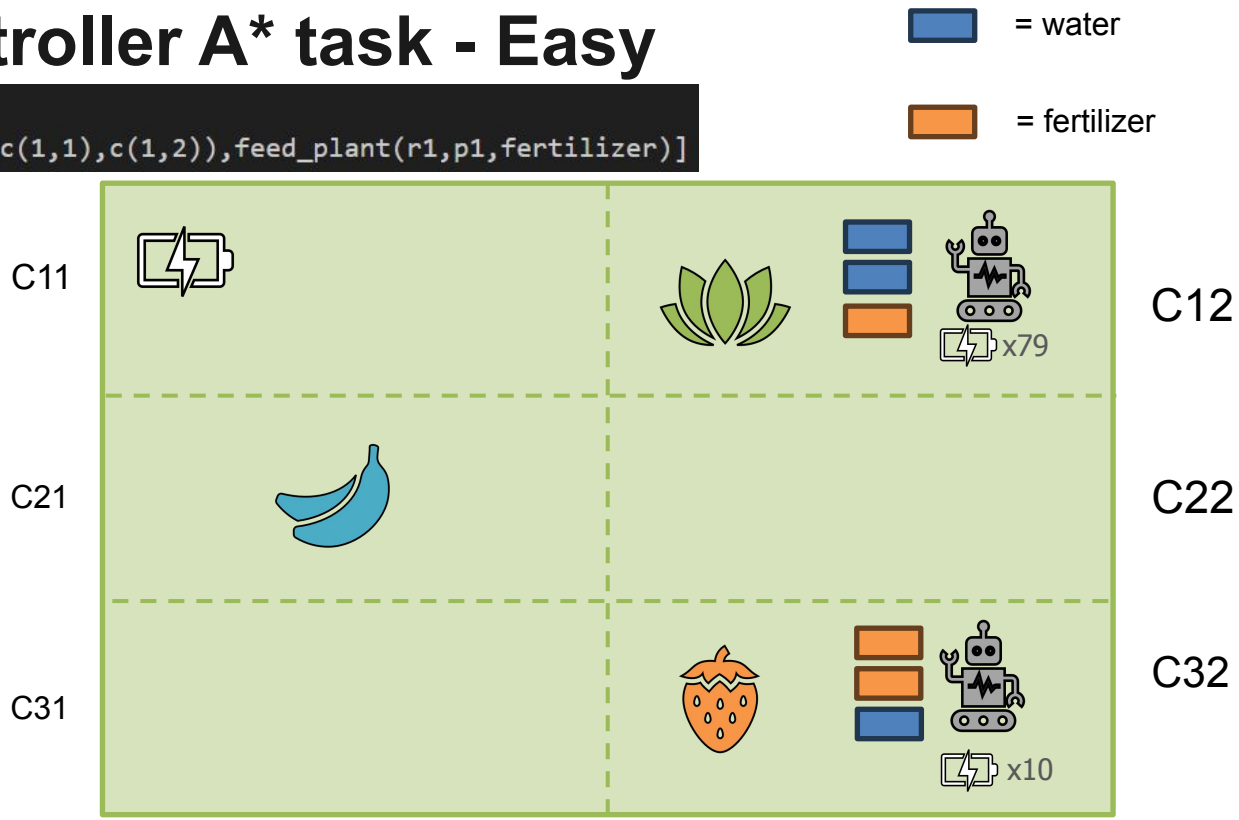
Controller A* task - Easy

```
A*: goal p1 fed
Plan 1: [move(r1,c(1,1),c(1,2)),feed_plant(r1,p1,fertilizer)]
```



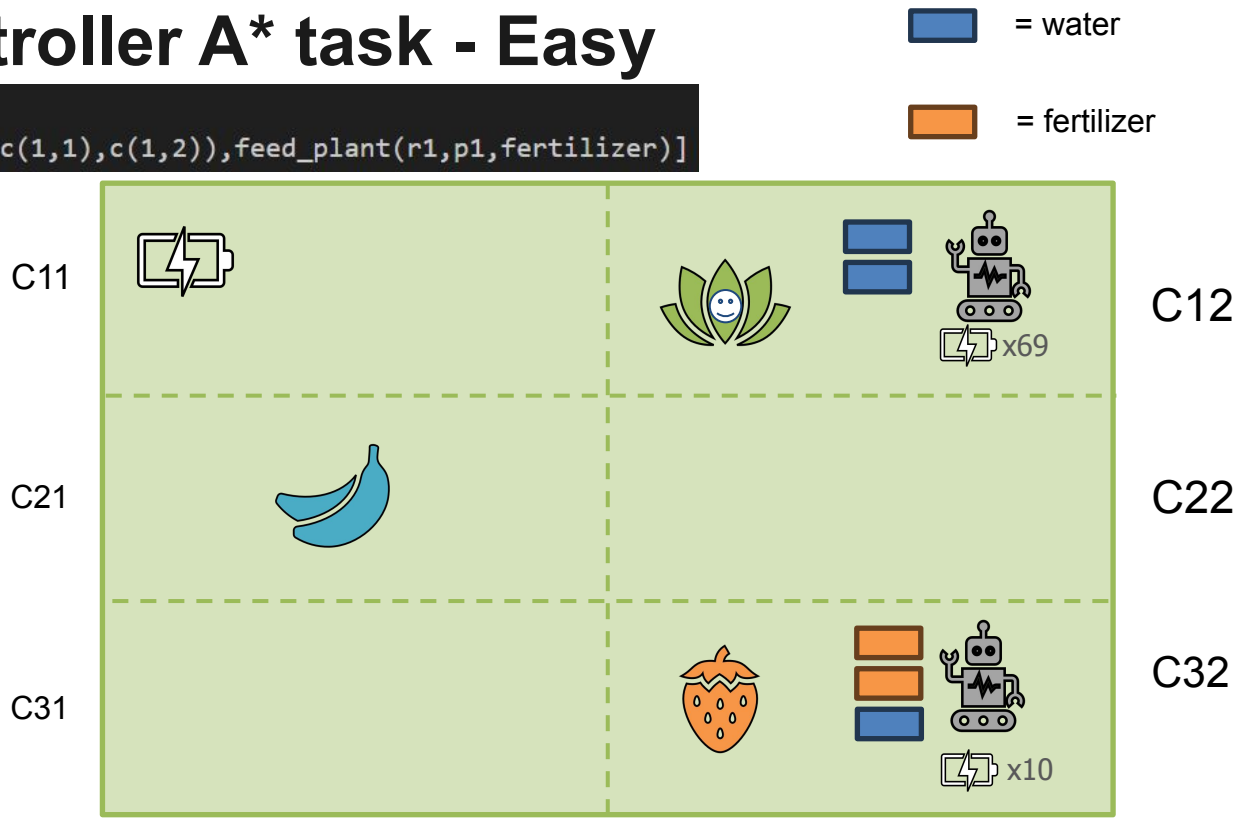
Controller A* task - Easy

```
A*: goal p1 fed
Plan 1: [move(r1,c(1,1),c(1,2)),feed_plant(r1,p1,fertilizer)]
```



Controller A* task - Easy

```
A*: goal p1 fed
Plan 1: [move(r1,c(1,1),c(1,2)),feed_plant(r1,p1,fertilizer)]
```

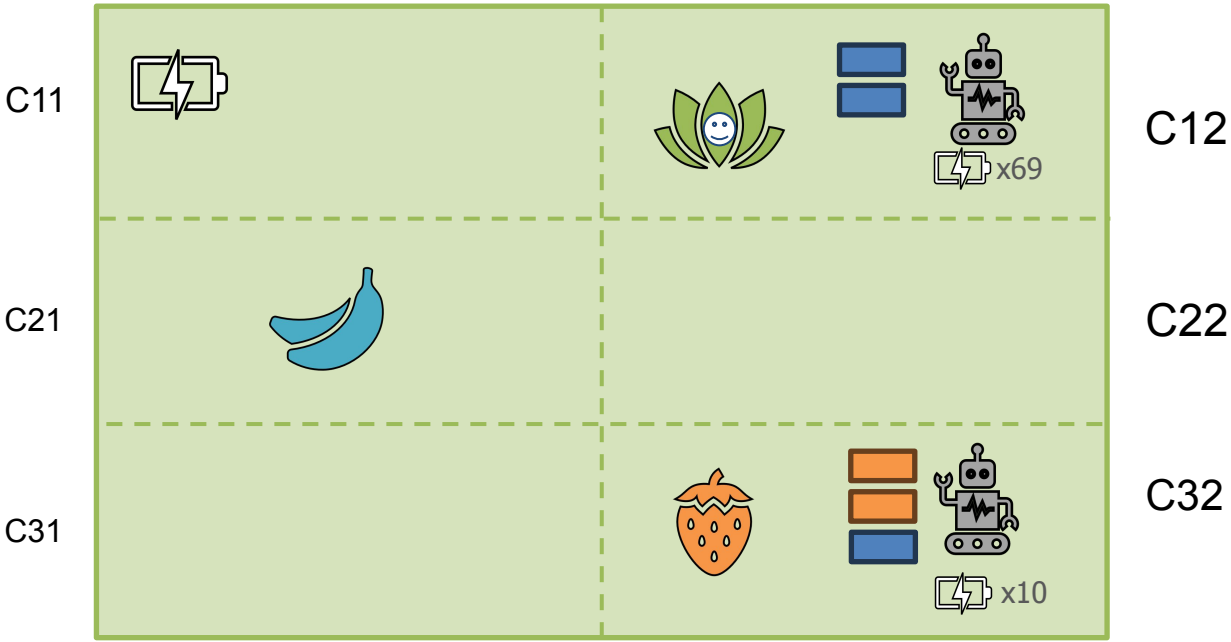


Controller A* task - Easy

```
A*: goal p1 fed
Plan 1: [move(r1,c(1,1),c(1,2)),feed_plant(r1,p1,fertilizer)]
```

= water

= fertilizer



Controller A* task - Easy - Results

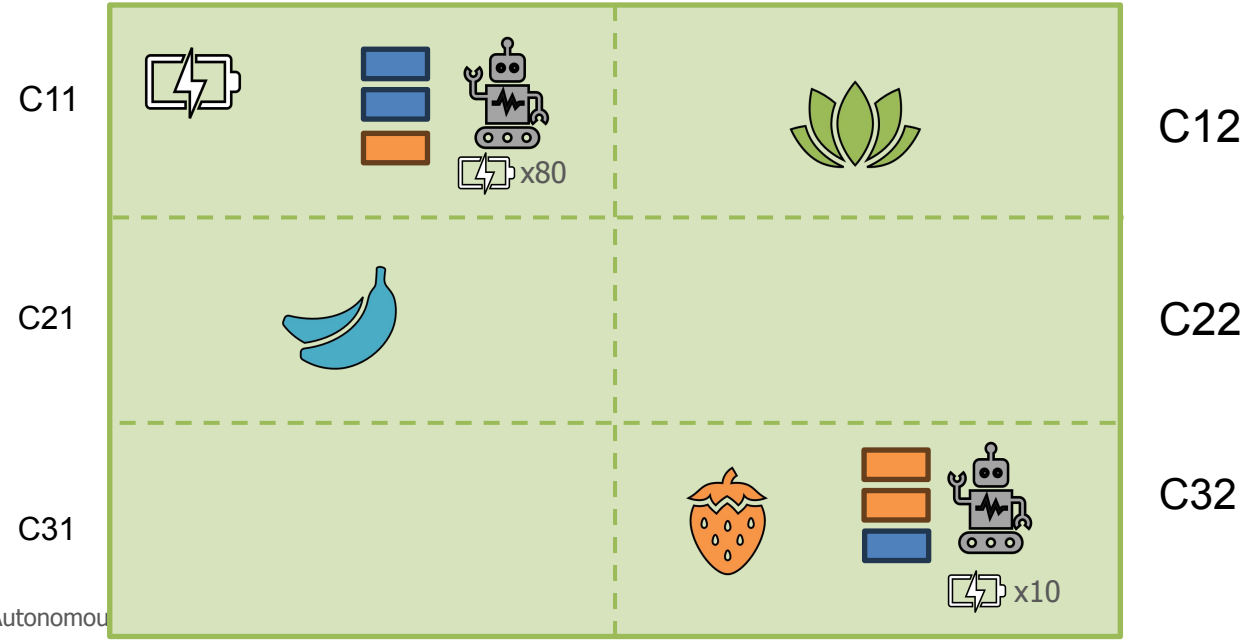
```
A*: goal p1 fed
Plan 1: [move(r1,c(1,1),c(1,2)),feed_plant(r1,p1,fertilizer)]
  r1 @ c(1,2), battery=69
  r2 @ c(3,2), battery=10
  p1: fed
  p2: not_fed
  p3: not_fed
  Sf = do(feed_plant(r1,p1,fertilizer),do(move(r1,c(1,1),c(1,2)),s0))
```

Controller A* task - Medium

```
A*: goal p1 p2 fed
Plan 2: [move(r1,c(1,1),c(2,1)),feed_plant(r1,p2,water),move(r1,c(2,1),c(2,2)),
         move(r1,c(2,2),c(1,2)),feed_plant(r1,p1,water)]
```

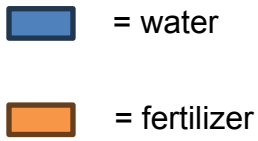
= water

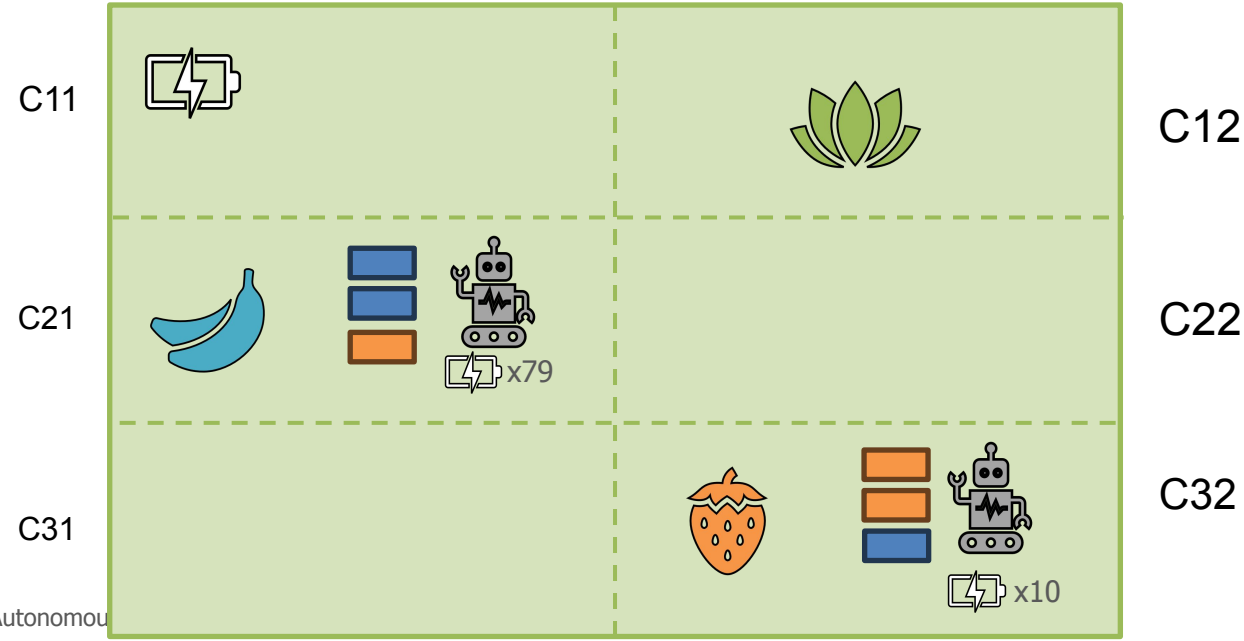
= fertilizer



Controller A* task - Medium

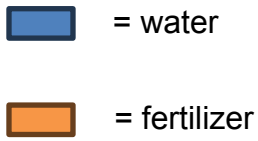
```
A*: goal p1 p2 fed
Plan 2: [move(r1,c(1,1),c(2,1)),feed_plant(r1,p2,water),move(r1,c(2,1),c(2,2)),
         move(r1,c(2,2),c(1,2)),feed_plant(r1,p1,water)]
```

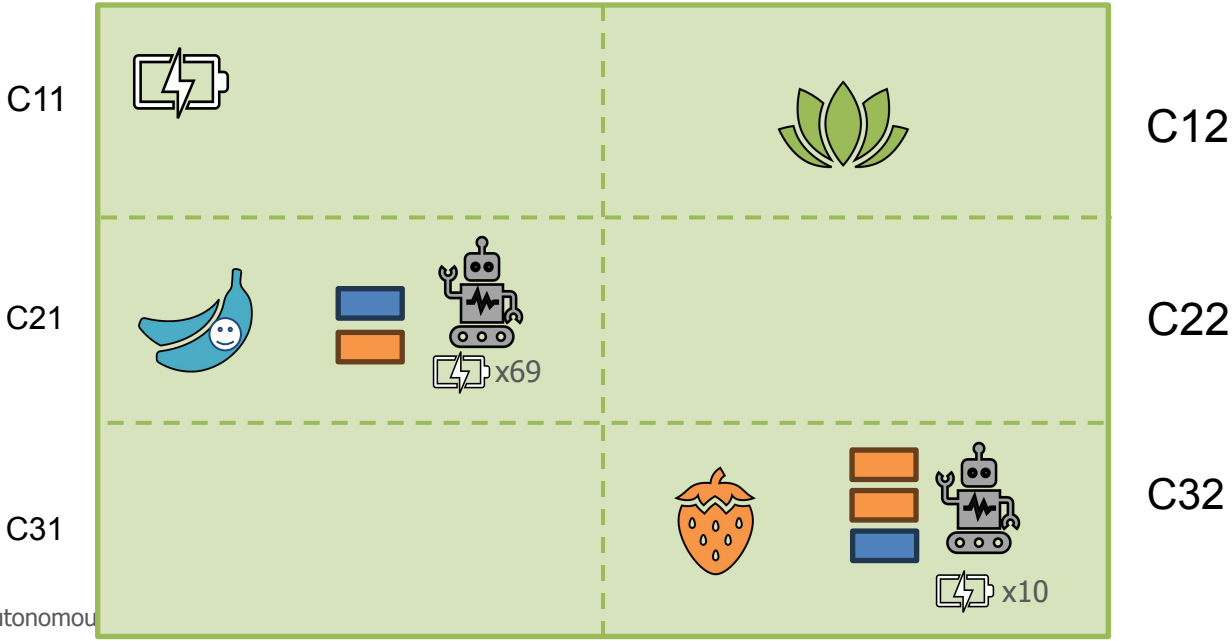




Controller A* task - Medium

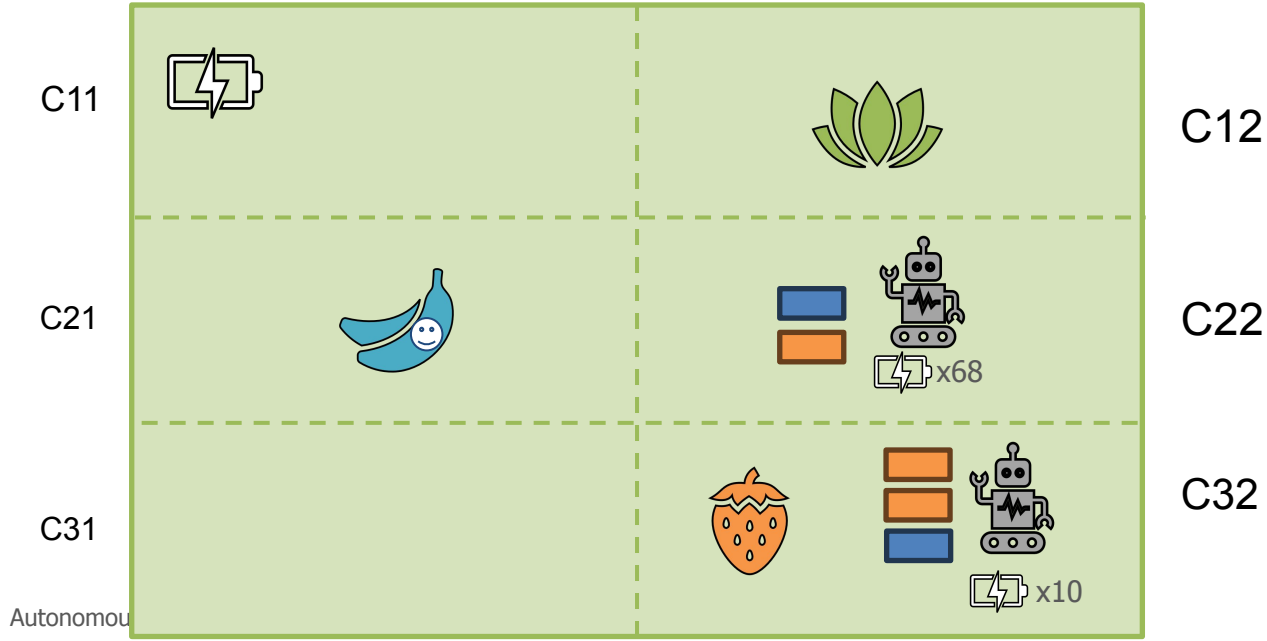
```
A*: goal p1 p2 fed
Plan 2: [move(r1,c(1,1),c(2,1)),feed_plant(r1,p2,water),move(r1,c(2,1),c(2,2)),
         move(r1,c(2,2),c(1,2)),feed_plant(r1,p1,water)]
```





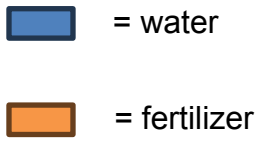
Controller A* task - Medium

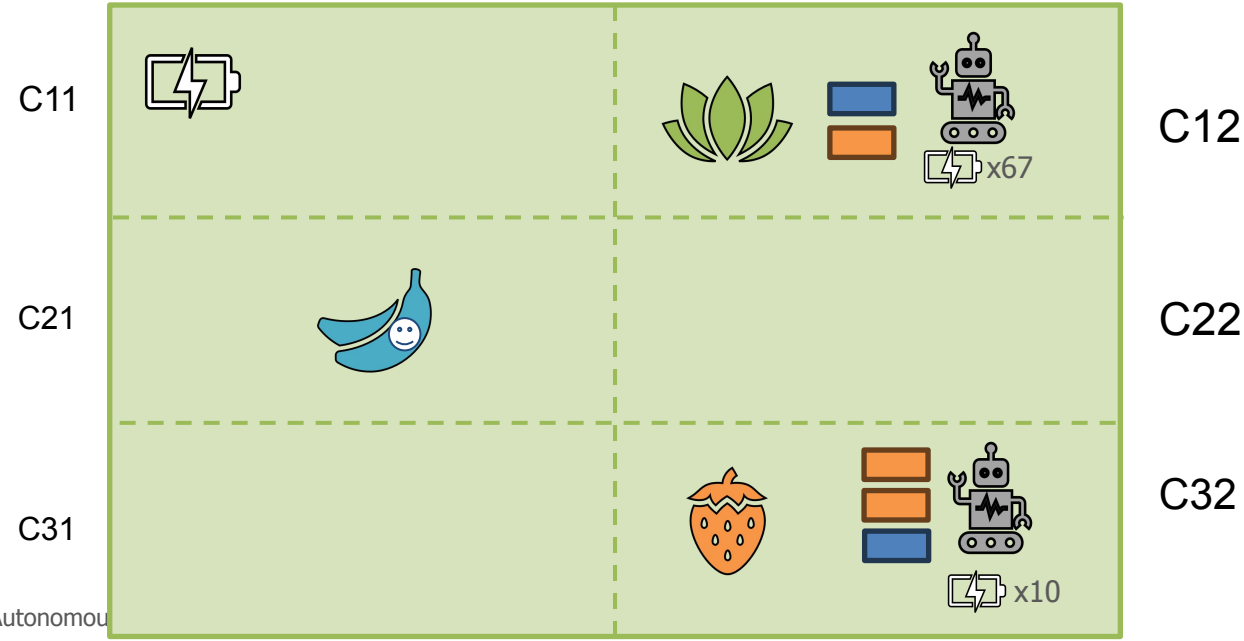
```
A*: goal p1 p2 fed
Plan 2: [move(r1,c(1,1),c(2,1)),feed_plant(r1,p2,water),move(r1,c(2,1),c(2,2)),
         move(r1,c(2,2),c(1,2)),feed_plant(r1,p1,water)]
```



Controller A* task - Medium

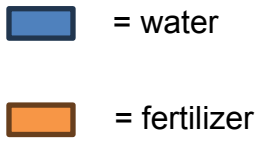
```
A*: goal p1 p2 fed
Plan 2: [move(r1,c(1,1),c(2,1)),feed_plant(r1,p2,water),move(r1,c(2,1),c(2,2)),
         move(r1,c(2,2),c(1,2)),feed_plant(r1,p1,water)]
```

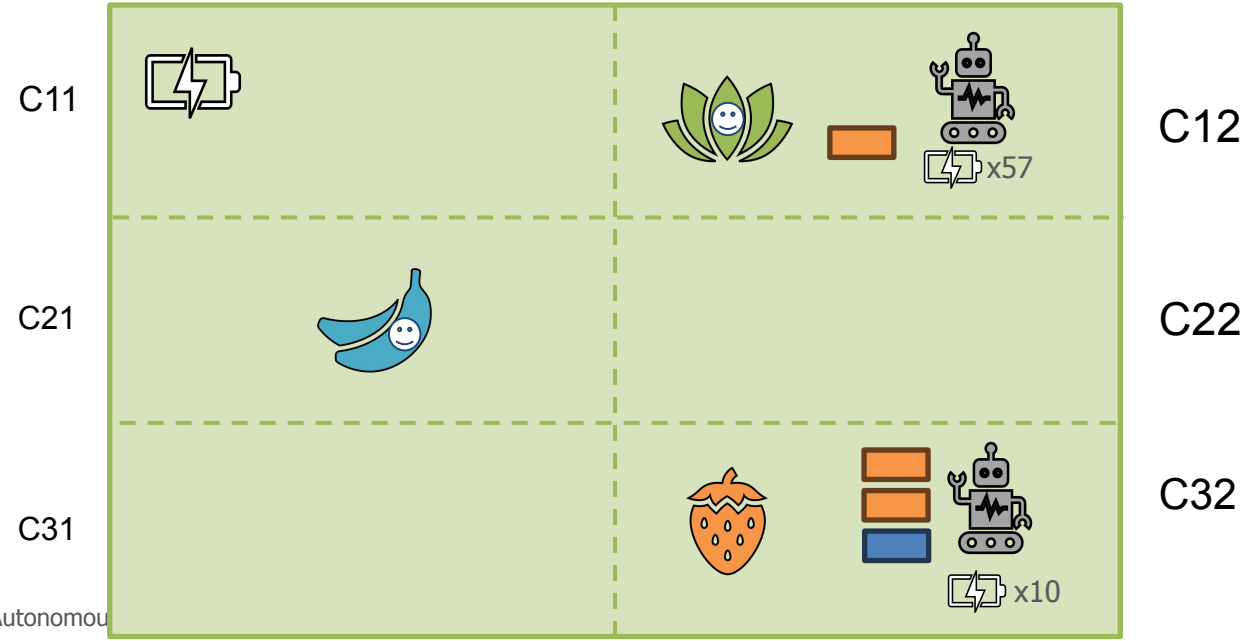




Controller A* task - Medium

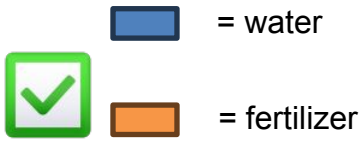
```
A*: goal p1 p2 fed
Plan 2: [move(r1,c(1,1),c(2,1)),feed_plant(r1,p2,water),move(r1,c(2,1),c(2,2)),
         move(r1,c(2,2),c(1,2)),feed_plant(r1,p1,water)]
```

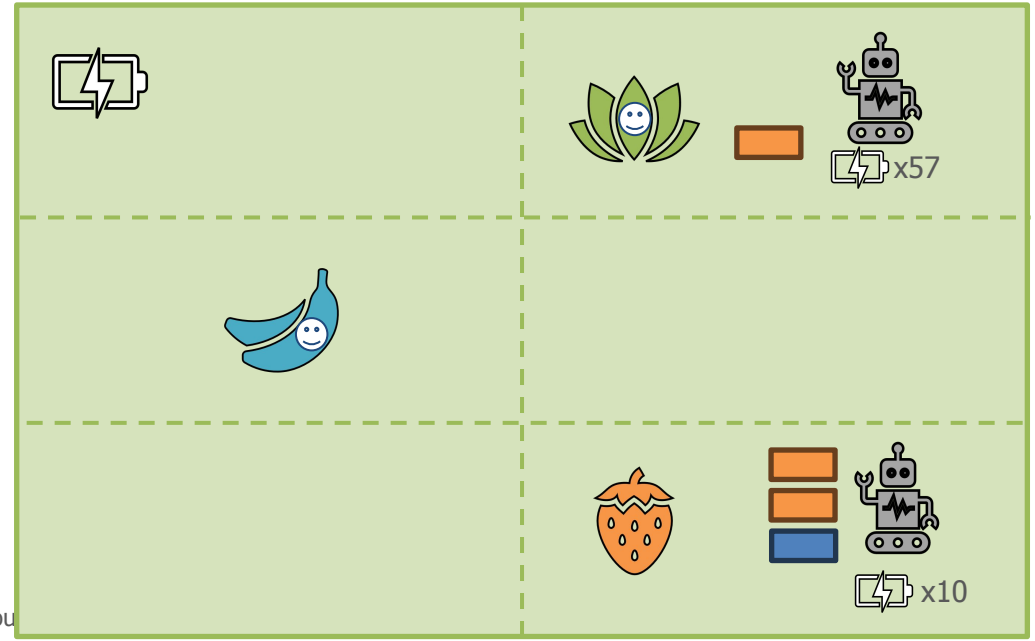




Controller A* task - Medium

```
A*: goal p1 p2 fed
Plan 2: [move(r1,c(1,1),c(2,1)),feed_plant(r1,p2,water),move(r1,c(2,1),c(2,2)),
         move(r1,c(2,2),c(1,2)),feed_plant(r1,p1,water)]
```



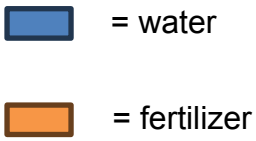


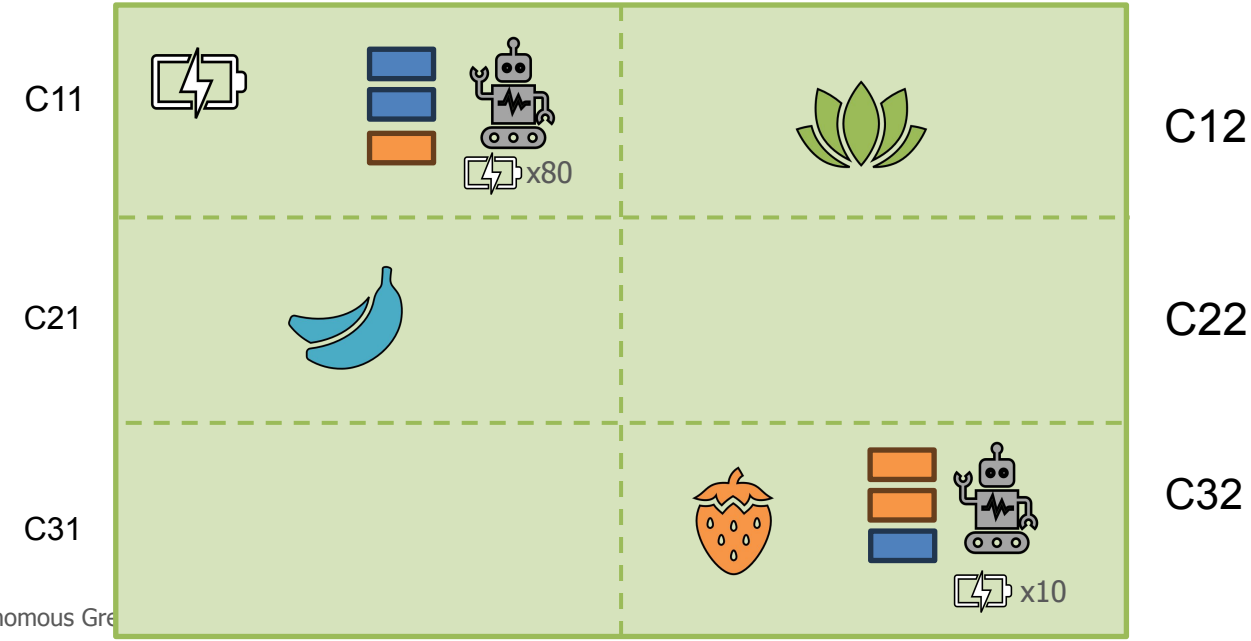
Controller A* task - Medium - Results

```
A*: goal p1 p2 fed
Plan 2: [move(r1,c(1,1),c(2,1)),feed_plant(r1,p2,water),move(r1,c(2,1),c(2,2)),
         move(r1,c(2,2),c(1,2)),feed_plant(r1,p1,water)]
  r1 @ c(1,2), battery=57
  r2 @ c(3,2), battery=10
  p1: fed
  p2: fed
  p3: not_fed
  Sf = do(feed_plant(r1,p1,water),do(move(r1,c(2,2),c(1,2)),do(move(r1,c(2,1),c(2,2)),
       do(feed_plant(r1,p2,water),do(move(r1,c(1,1),c(2,1)),s0)))))
```

Controller A* task - Hard

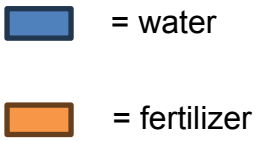
```
A*: goal all p fed
Plan 3: [move(r1,c(1,1),c(2,1)),feed_plant(r2,p3,fertilizer),feed_plant(r1,p2,water),
         move(r1,c(2,1),c(2,2)),move(r1,c(2,2),c(1,2)),feed_plant(r1,p1,water)]
```

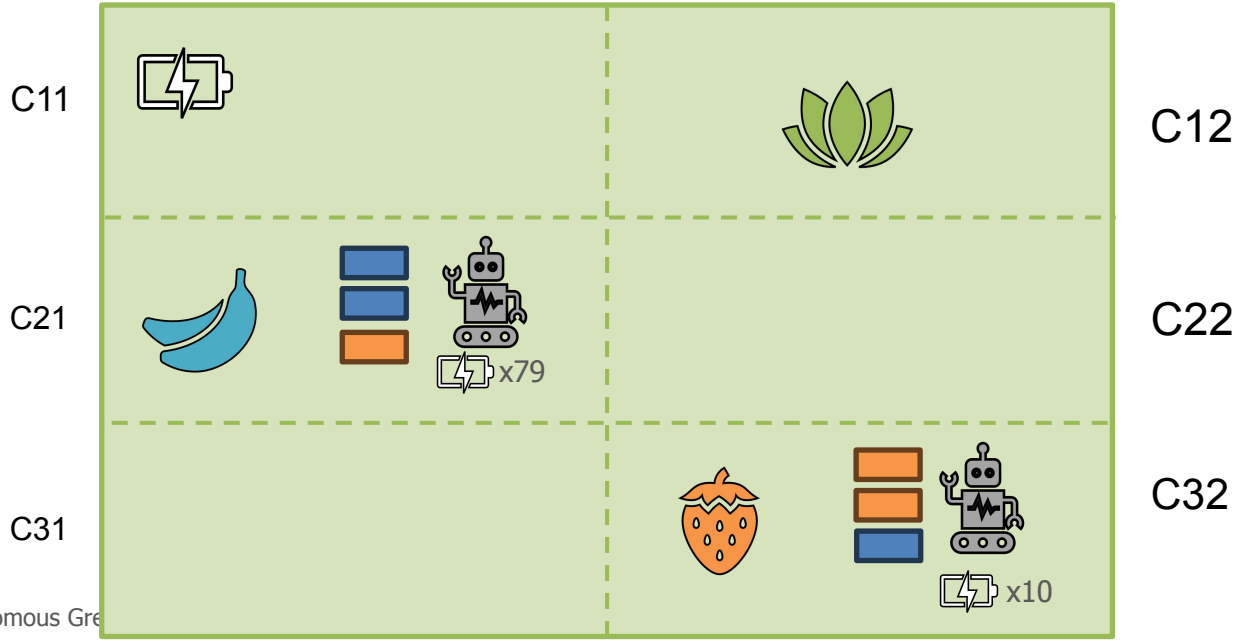




Controller A* task - Hard

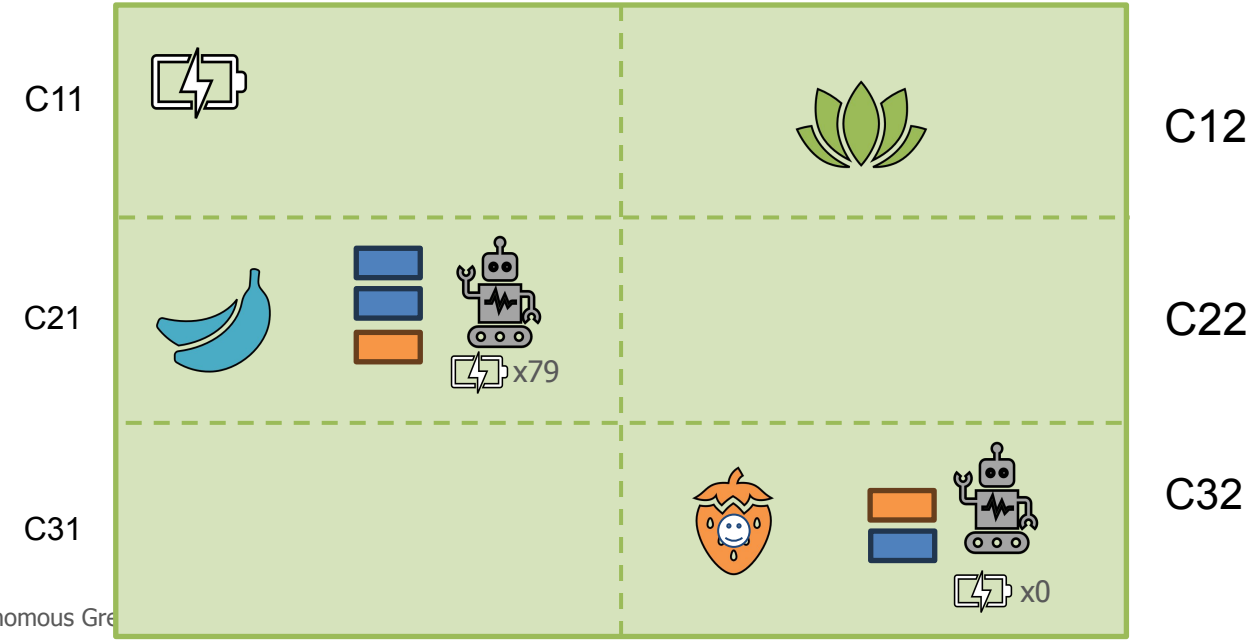
```
A*: goal all p fed
Plan 3: [move(r1,c(1,1),c(2,1)),feed_plant(r2,p3,fertilizer),feed_plant(r1,p2,water),
         move(r1,c(2,1),c(2,2)),move(r1,c(2,2),c(1,2)),feed_plant(r1,p1,water)]
```





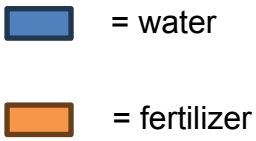
Controller A* task - Hard

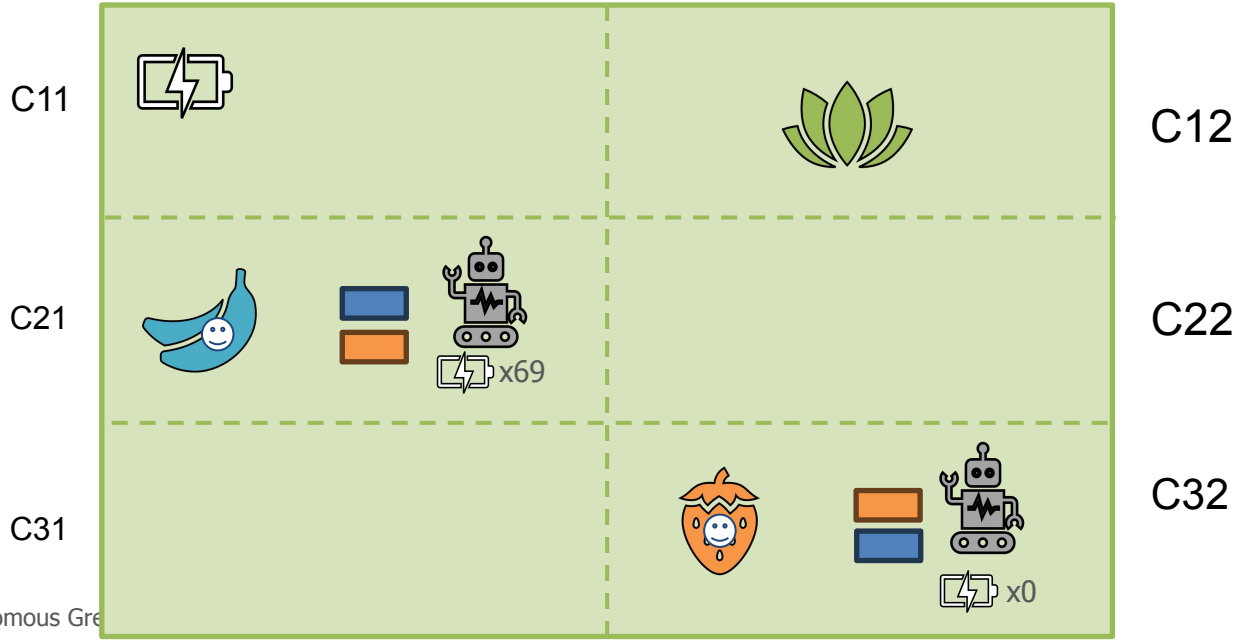
```
A*: goal all p fed
Plan 3: [move(r1,c(1,1),c(2,1)),feed_plant(r2,p3,fertilizer),feed_plant(r1,p2,water),
         move(r1,c(2,1),c(2,2)),move(r1,c(2,2),c(1,2)),feed_plant(r1,p1,water)]
```



Controller A* task - Hard

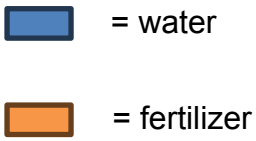
```
A*: goal all p fed
Plan 3: [move(r1,c(1,1),c(2,1)),feed_plant(r2,p3,fertilizer),feed_plant(r1,p2,water),
         move(r1,c(2,1),c(2,2)),move(r1,c(2,2),c(1,2)),feed_plant(r1,p1,water)]
```

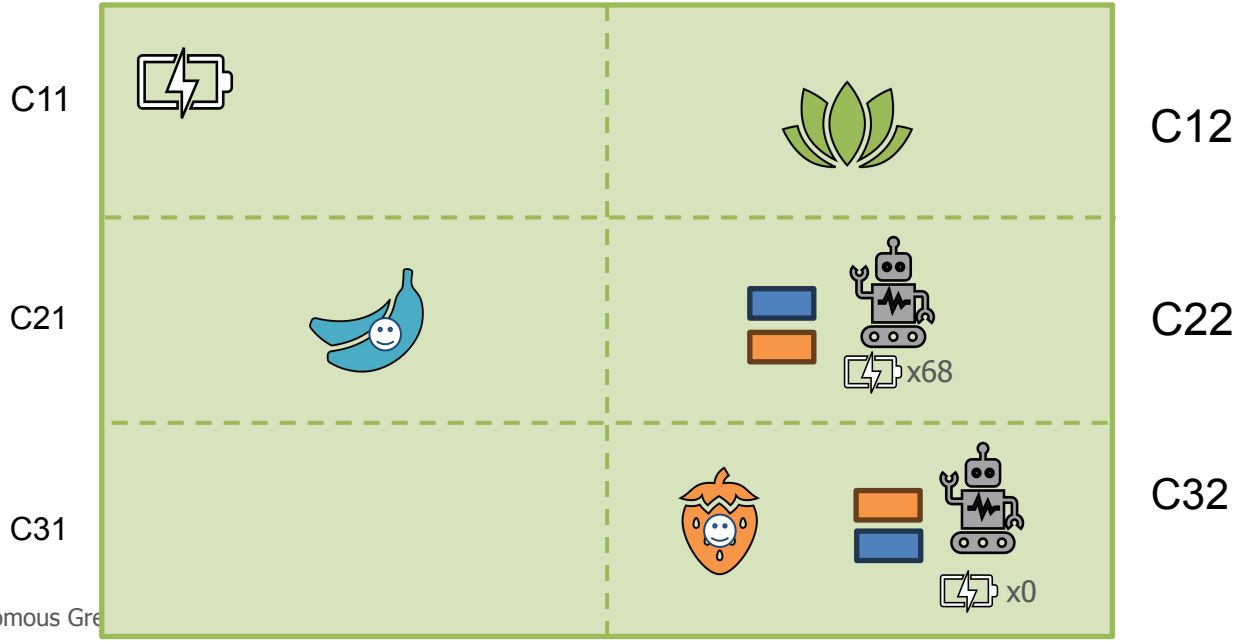




Controller A* task - Hard

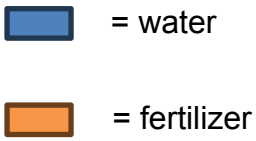
```
A*: goal all p fed
Plan 3: [move(r1,c(1,1),c(2,1)),feed_plant(r2,p3,fertilizer),feed_plant(r1,p2,water),
         move(r1,c(2,1),c(2,2)),move(r1,c(2,2),c(1,2)),feed_plant(r1,p1,water)]
```

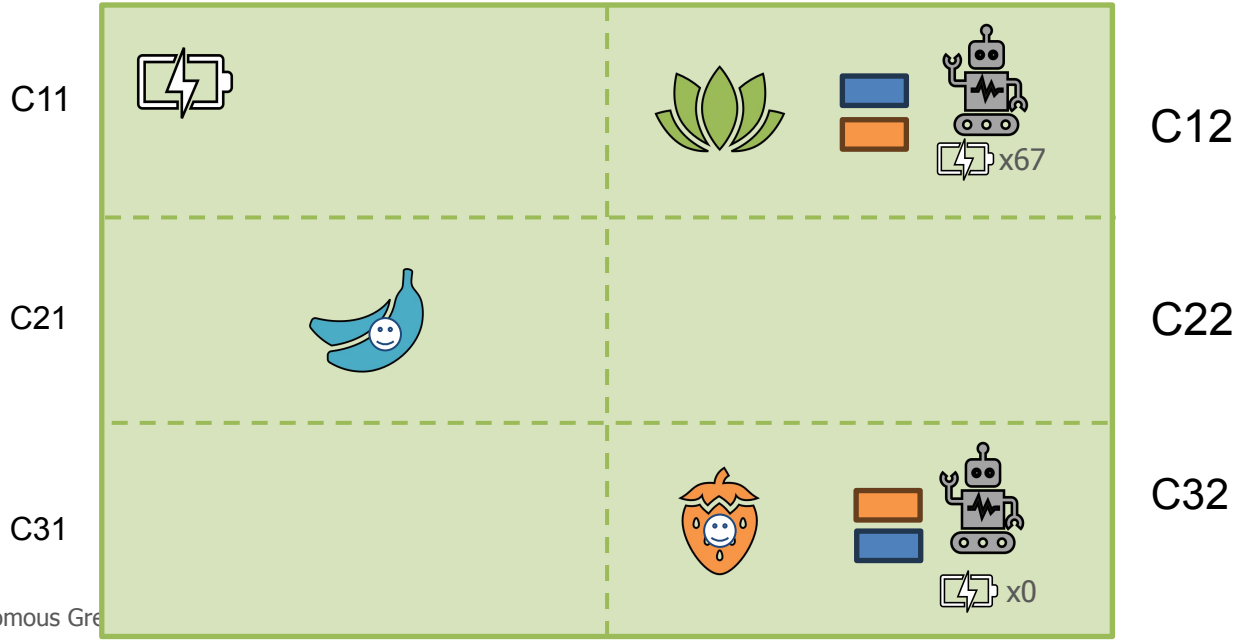




Controller A* task - Hard

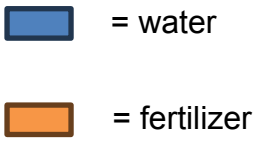
```
A*: goal all p fed
Plan 3: [move(r1,c(1,1),c(2,1)),feed_plant(r2,p3,fertilizer),feed_plant(r1,p2,water),
         move(r1,c(2,1),c(2,2)),move(r1,c(2,2),c(1,2)),feed_plant(r1,p1,water)]
```

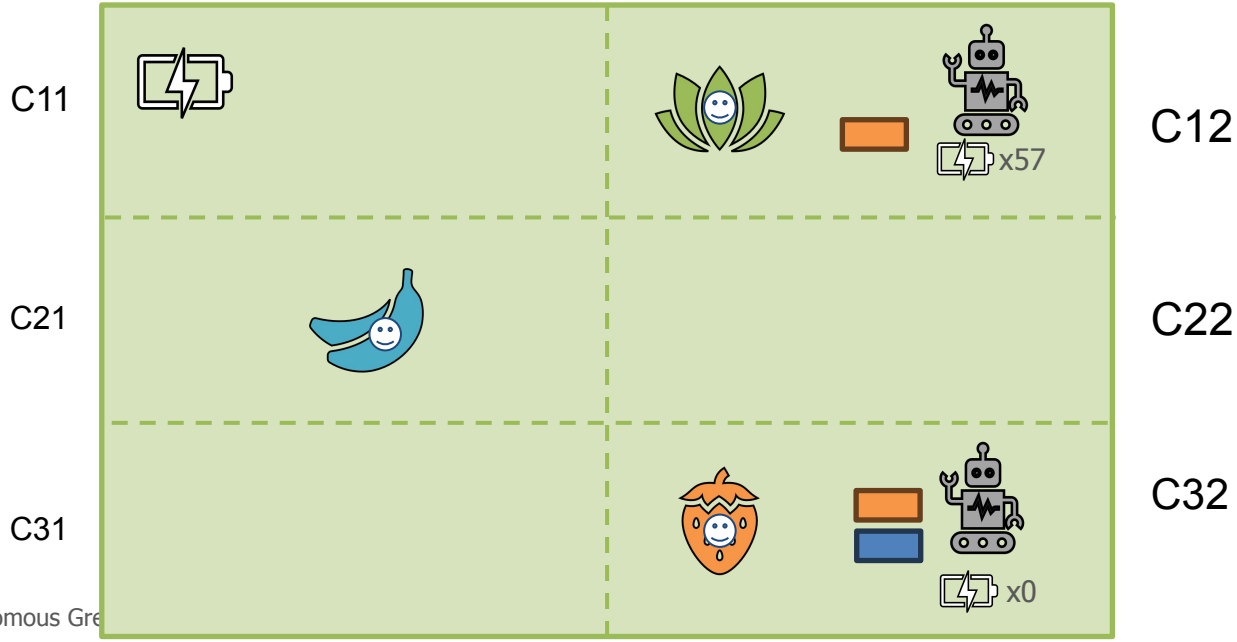




Controller A* task - Hard

```
A*: goal all p fed
Plan 3: [move(r1,c(1,1),c(2,1)),feed_plant(r2,p3,fertilizer),feed_plant(r1,p2,water),
         move(r1,c(2,1),c(2,2)),move(r1,c(2,2),c(1,2)),feed_plant(r1,p1,water)]
```



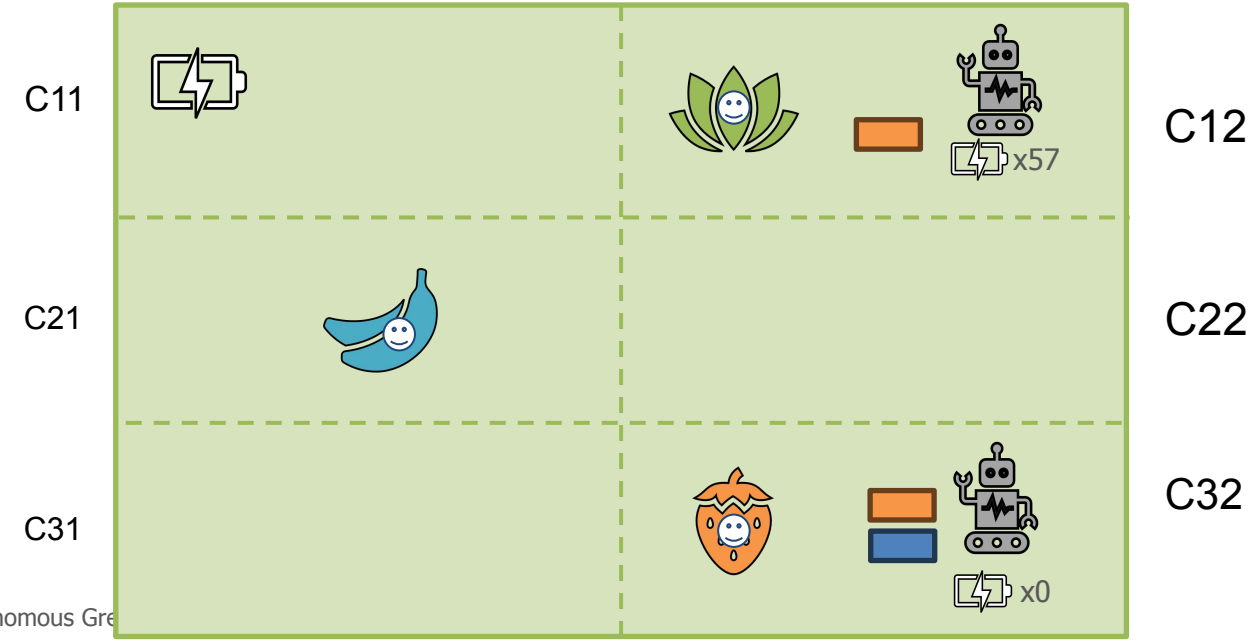


Controller A* task - Hard

```
A*: goal all p fed
Plan 3: [move(r1,c(1,1),c(2,1)),feed_plant(r2,p3,fertilizer),feed_plant(r1,p2,water),
         move(r1,c(2,1),c(2,2)),move(r1,c(2,2),c(1,2)),feed_plant(r1,p1,water)]
```

= water

= fertilizer



Controller A* task - Hard

```
A*: goal all p fed
Plan 3: [move(r1,c(1,1),c(2,1)),feed_plant(r2,p3,fertilizer),feed_plant(r1,p2,water),
         move(r1,c(2,1),c(2,2)),move(r1,c(2,2),c(1,2)),feed_plant(r1,p1,water)]
  r1 @ c(1,2), battery=57
  r2 @ c(3,2), battery=0
  p1: fed
  p2: fed
  p3: fed
  Sf = do(feed_plant(r1,p1,water),do(move(r1,c(2,2),c(1,2)),do(move(r1,c(2,1),c(2,2)),
       do(feed_plant(r1,p2,water),do(feed_plant(r2,p3,fertilizer),
       do(move(r1,c(1,1),c(2,1)),s0))))))
```

Contacts

- **Alessia Pontiggia:**
  pontiggia.1892079@studenti.uniroma1.it

- **Gerardo Paladino:**
  paladino.1918178@studenti.uniroma1.it

- **Code:**
  https://github.com/alessiapontiggia/PR-Project

Thank you for the attention!