NON LINEAR PLANNING: EXERCISE

Given the following initial state **robot_at(loc_a), handempty, object_at(plate, loc_a), object_at(cube, loc_b),**

We have to reach the goal: **coloured(plate, blue), coloured(cube,red)**

Actions are modelled as follows:

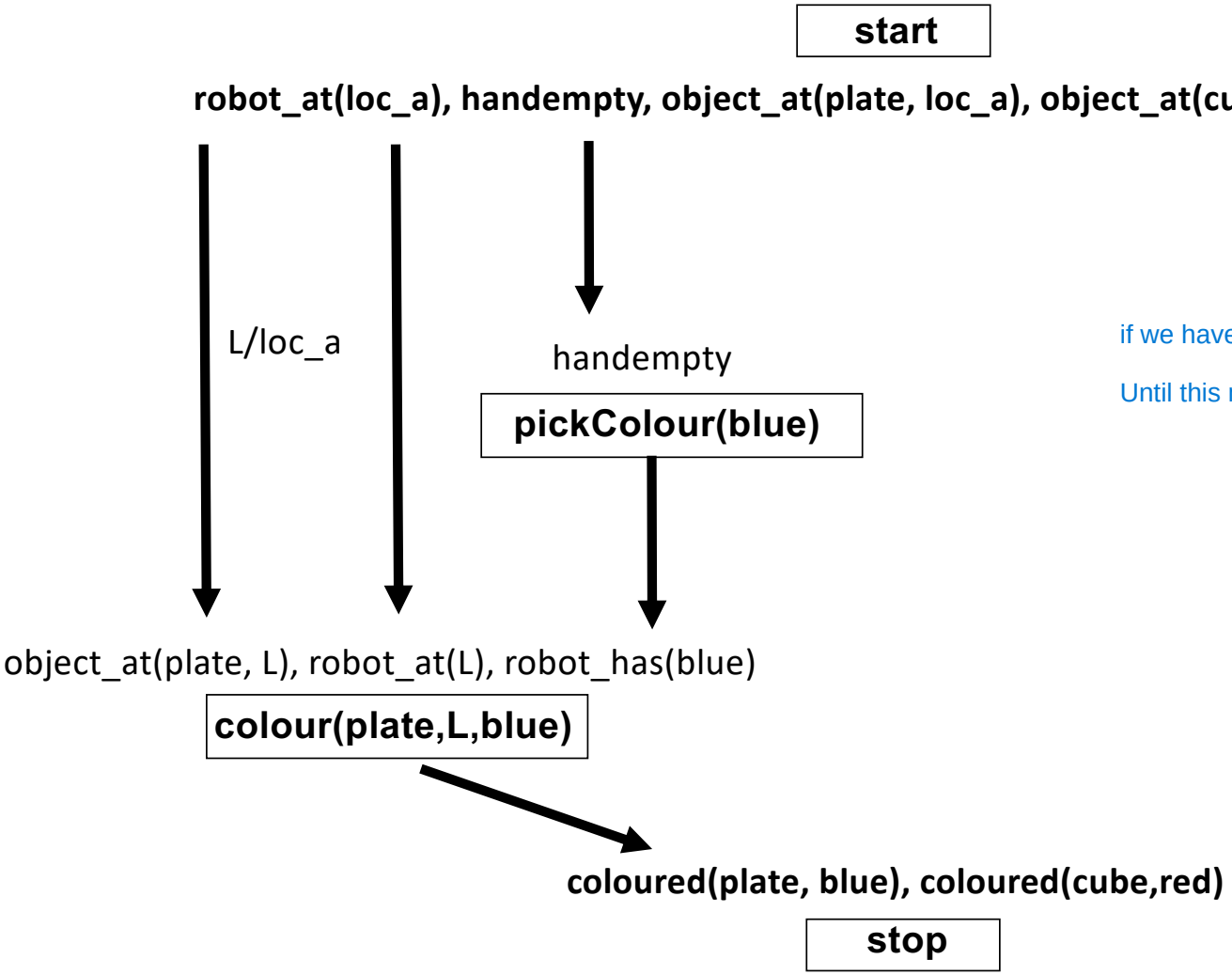
colour(Object, Location, Colour)
PRECOND: object_at(Object, Location), robot_at(Location), robot_has(Colour)
ADD: coloured(Object, Colour)

go(X,Y)
PRECOND: robot_at(X)
DELETE: robot_at(X)
ADD: robot_at(Y)

pickColour(C)
PRECOND: handempty
DELETE: handempty
ADD: robot_has(C)

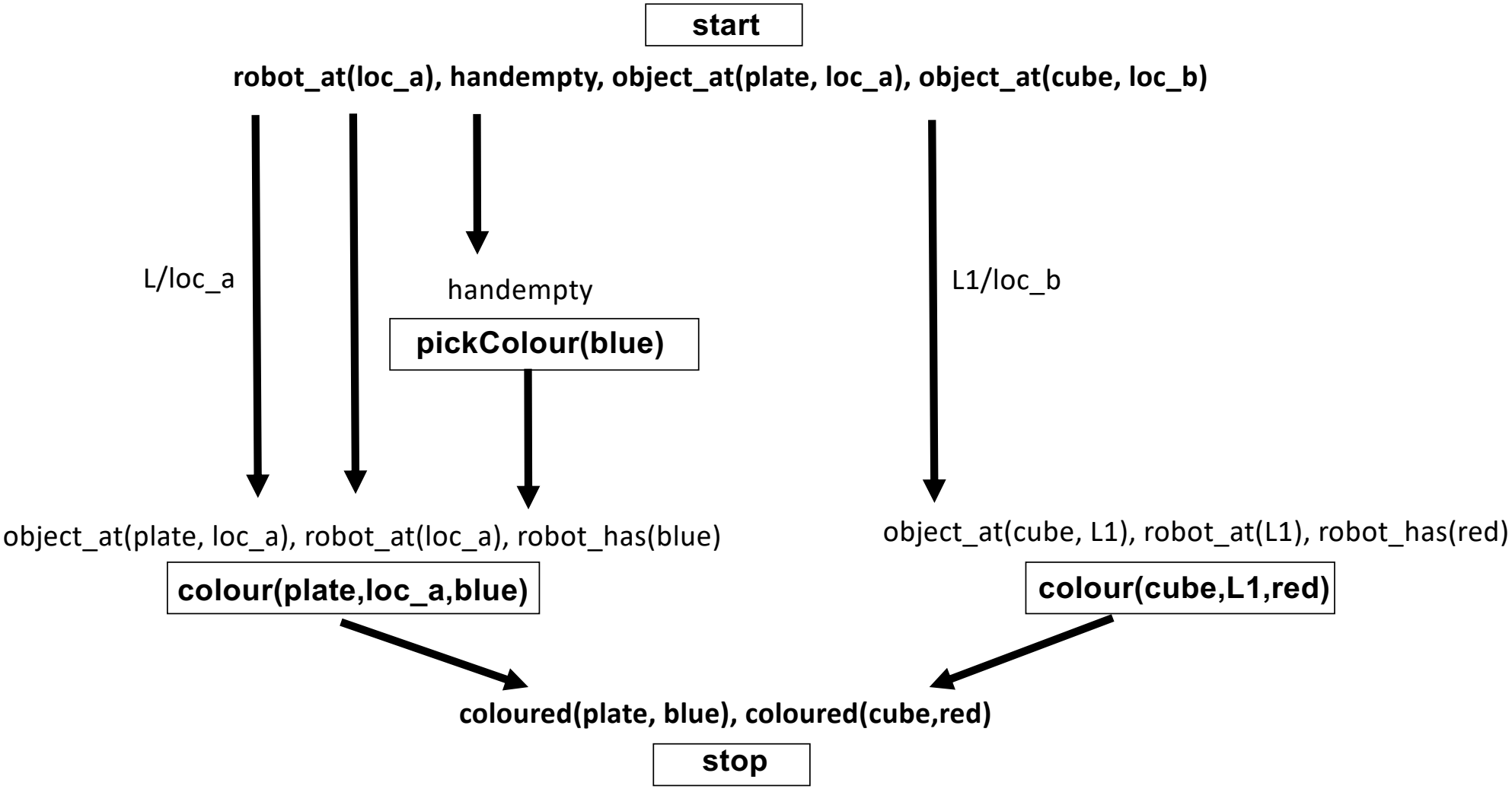
releaseColour(C)
PRECOND: robot_has(C)
DELETE: robot_has(C)
ADD: handempty

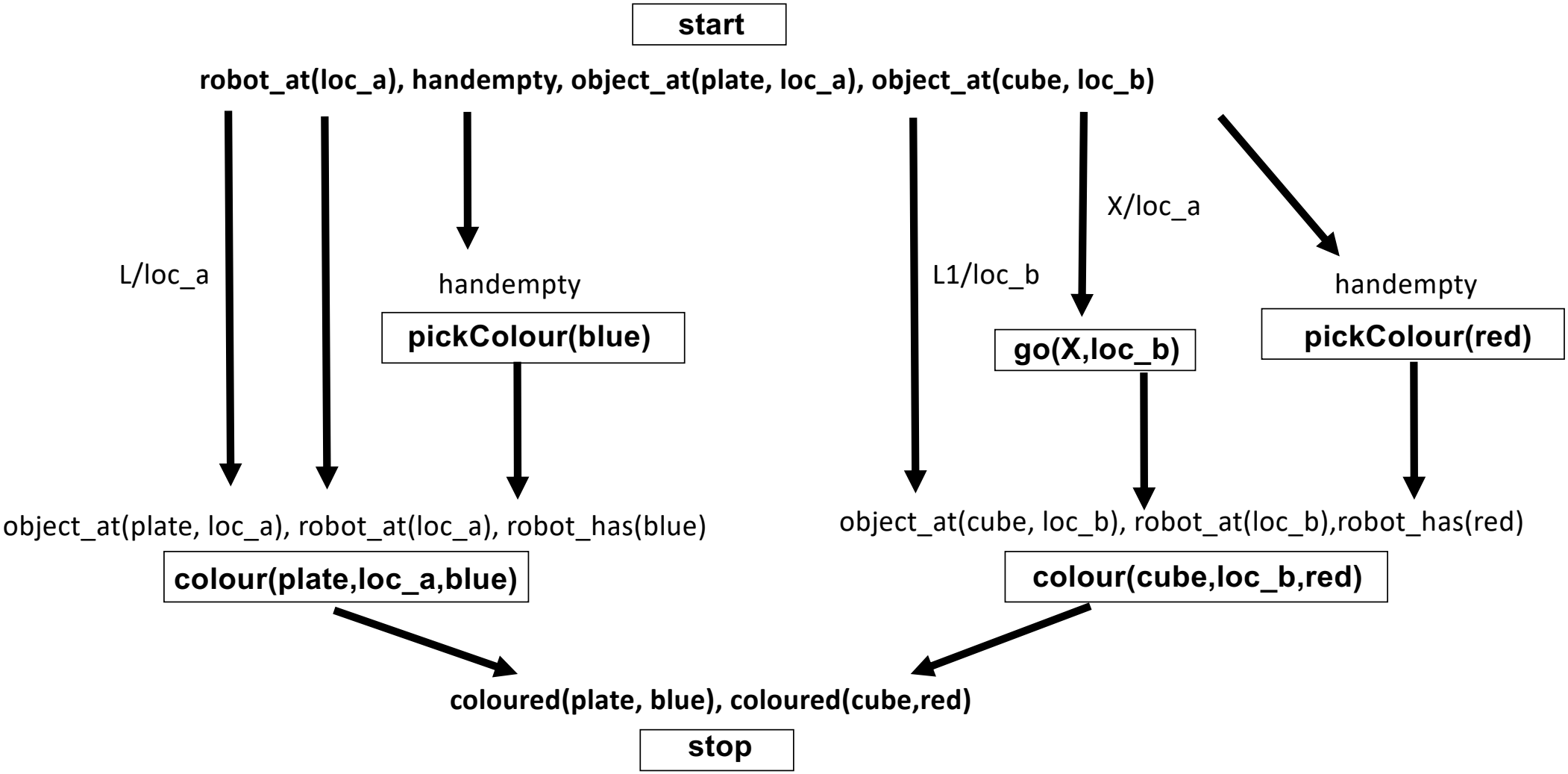
Solve the problem with the POP algorithm, identifying threats and their solution during the process

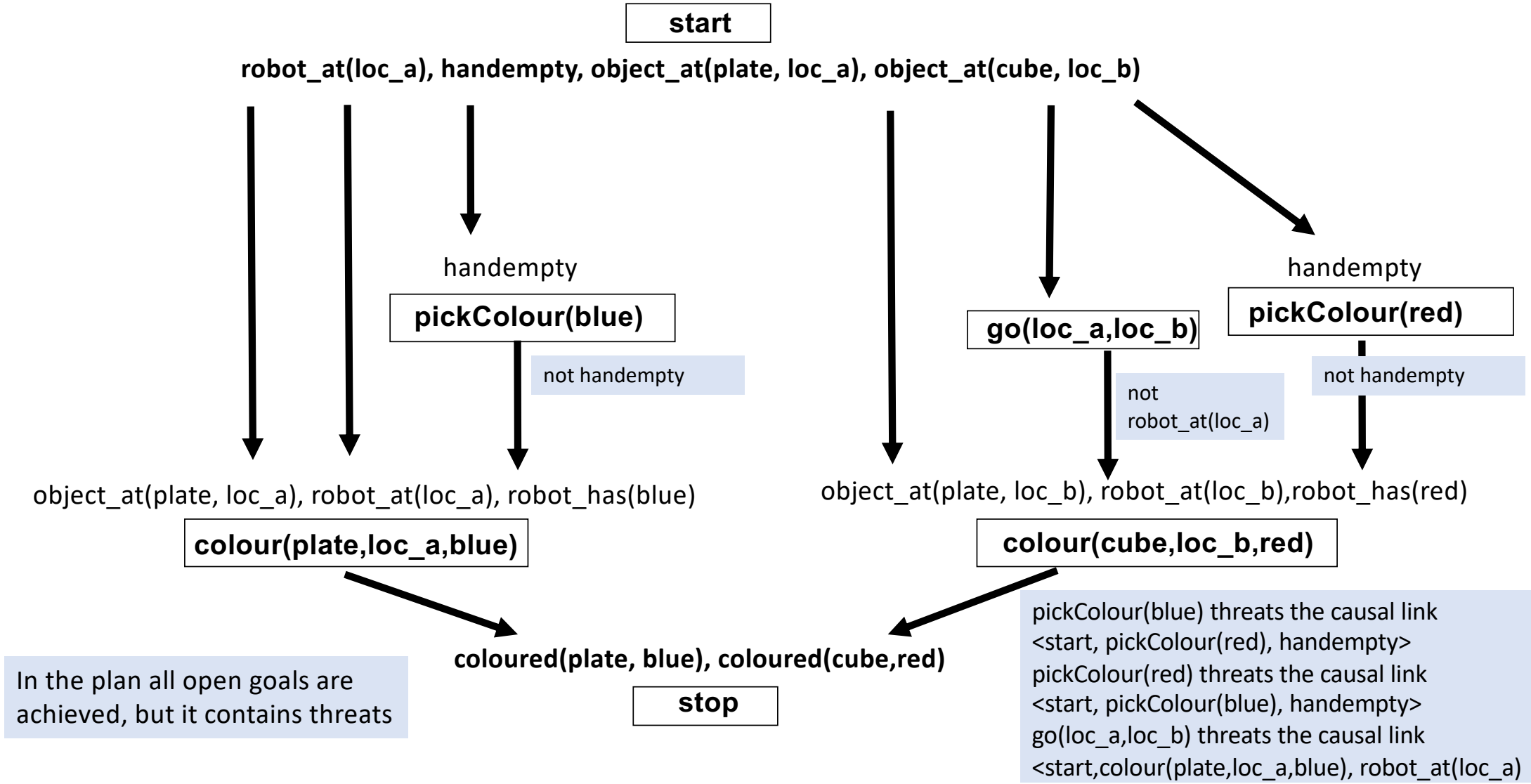


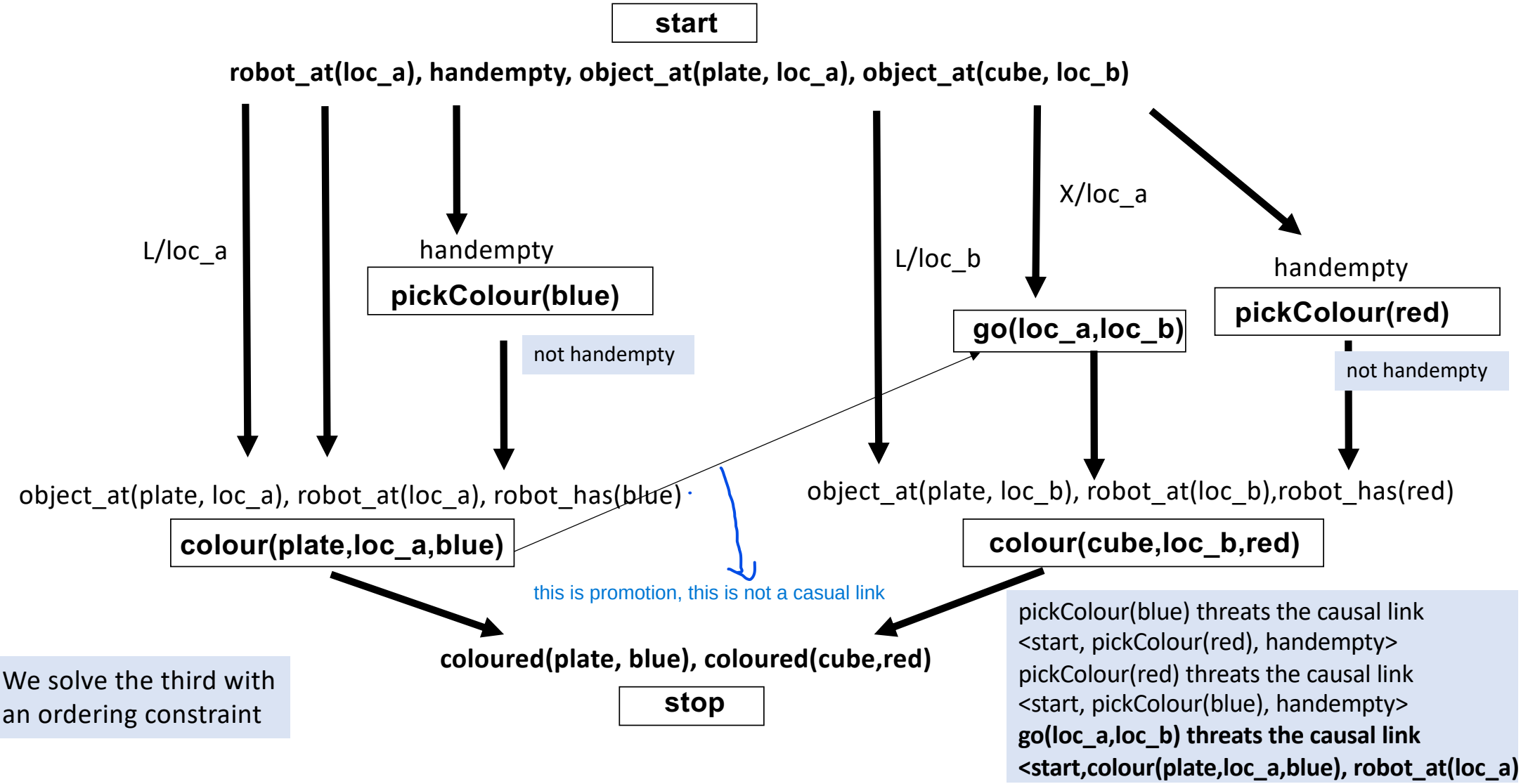
if we have only a single chain of actions we have no threats

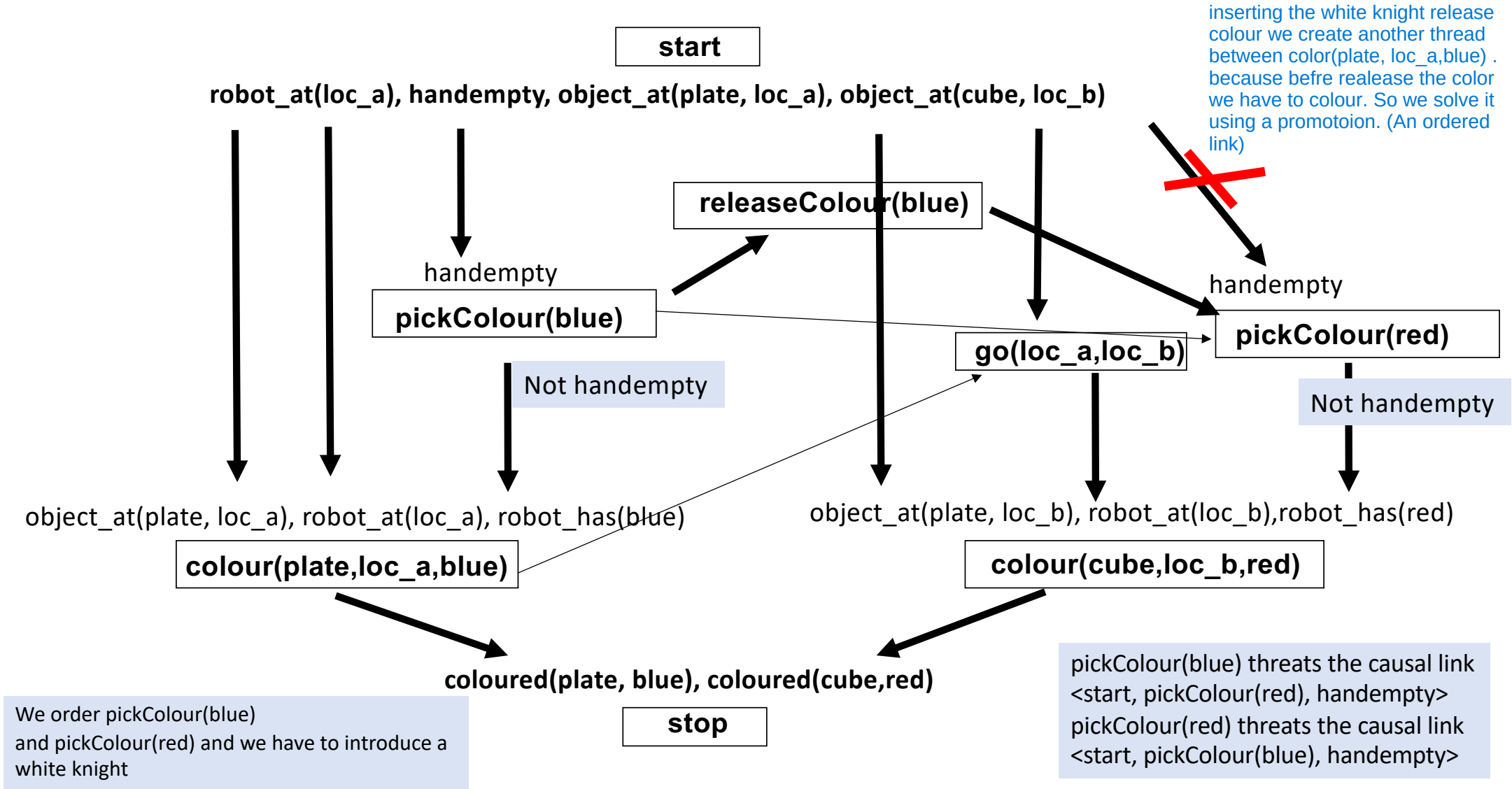
Until this moment we are fine

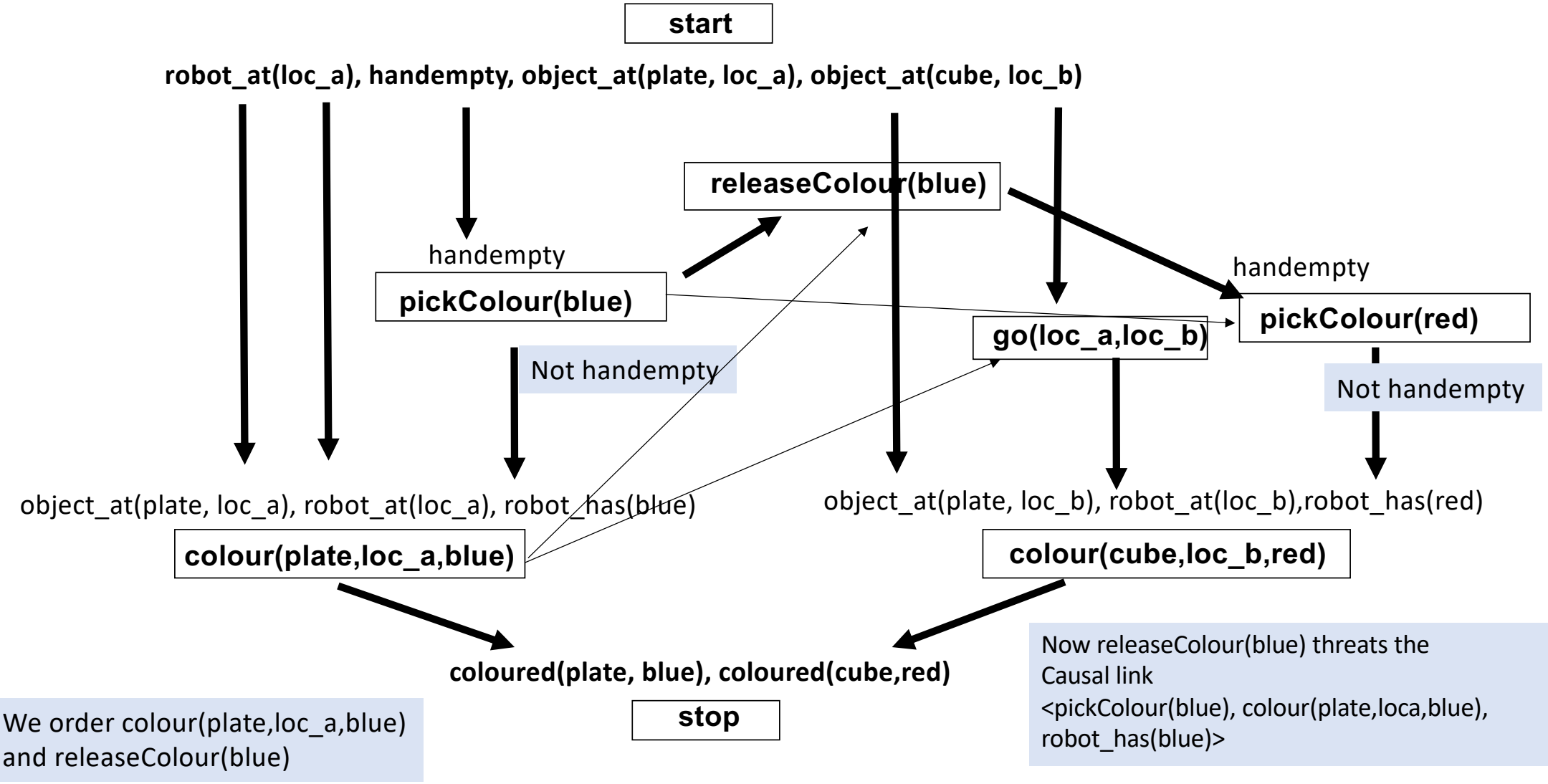

start
robot_at(loc_a), handempty, object_at(plate, loc_a), object_at(cube, loc_b)
L/loc_a
handempty
pickColour(blue)
L1/loc_b
object_at(plate, loc_a), robot_at(loc_a), robot_has(blue)
colour(plate,loc_a,blue)
object_at(cube, L1), robot_at(L1), robot_has(red)
colour(cube,L1,red)
coloured(plate, blue), coloured(cube,red)
stop


start
robot_at(loc_a), handempty, object_at(plate, loc_a), object_at(cube, loc_b)
L/loc_a
handempty
pickColour(blue)
L1/loc_b
X/loc_a
handempty
pickColour(red)
go(X,loc_b)
object_at(plate, loc_a), robot_at(loc_a), robot_has(blue)
object_at(cube, loc_b), robot_at(loc_b),robot_has(red)
colour(plate,loc_a,blue)
colour(cube,loc_b,red)
coloured(plate, blue), coloured(cube,red)
stop

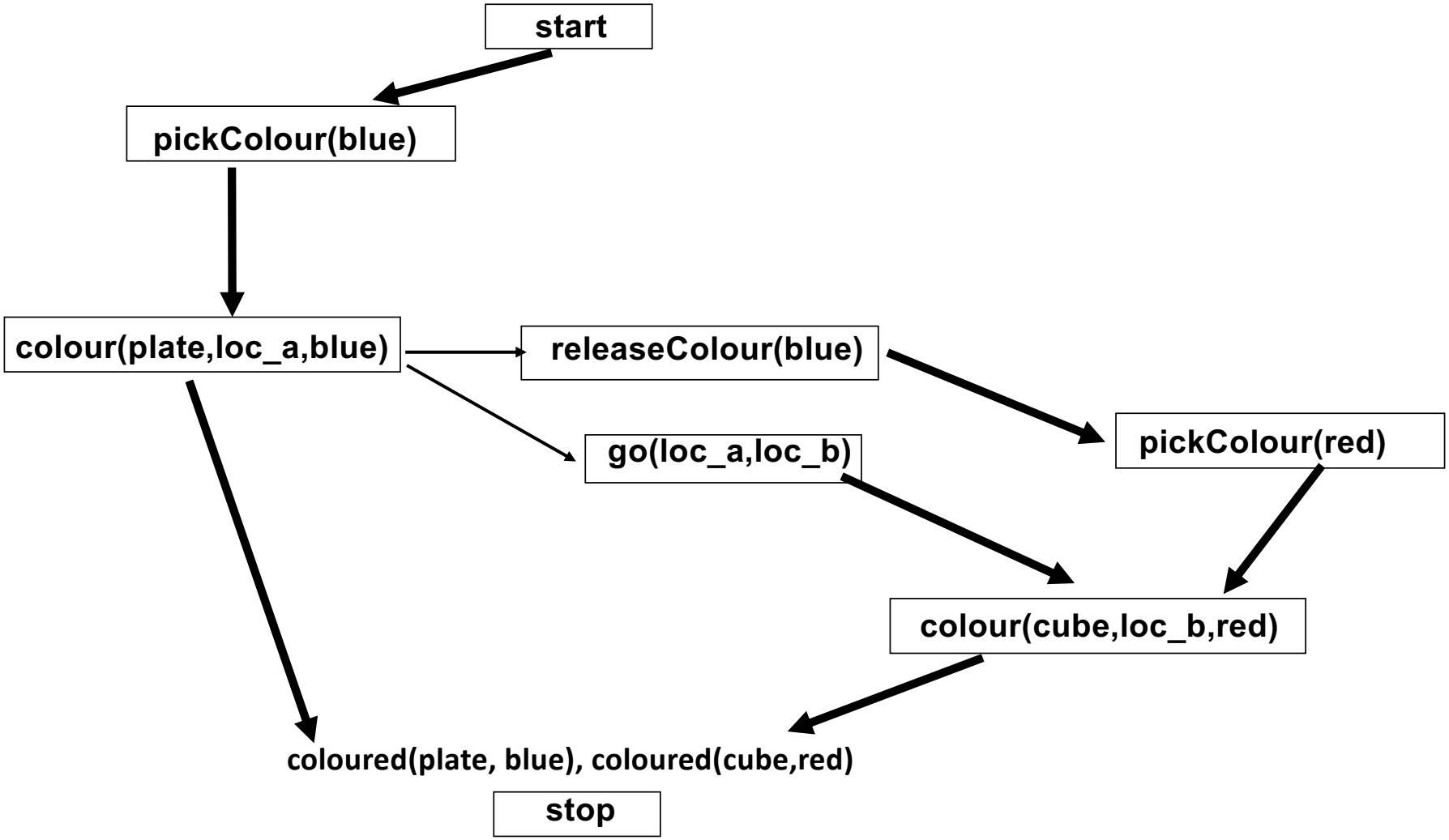

start
robot_at(loc_a), handempty, object_at(plate, loc_a), object_at(cube, loc_b)
handempty
pickColour(blue)
not handempty
go(loc_a,loc_b)
not robot_at(loc_a)
handempty
pickColour(red)
not handempty
object_at(plate, loc_a), robot_at(loc_a), robot_has(blue)
colour(plate,loc_a,blue)
object_at(plate, loc_b), robot_at(loc_b),robot_has(red)
colour(cube,loc_b,red)
coloured(plate, blue), coloured(cube,red)
stop
In the plan all open goals are achieved, but it contains threats
pickColour(blue) threats the causal link <start, pickColour(red), handempty>
pickColour(red) threats the causal link <start, pickColour(blue), handempty>
go(loc_a,loc_b) threats the causal link <start,colour(plate,loc_a,blue), robot_at(loc_a)


start
robot_at(loc_a), handempty, object_at(plate, loc_a), object_at(cube, loc_b)
L/loc_a
handempty
pickColour(blue)
not handempty
L/loc_b
X/loc_a
go(loc_a,loc_b)
handempty
pickColour(red)
not handempty
object_at(plate, loc_a), robot_at(loc_a), robot_has(blue)
colour(plate,loc_a,blue)
object_at(plate, loc_b), robot_at(loc_b),robot_has(red)
colour(cube,loc_b,red)
this is promotion, this is not a casual link
coloured(plate, blue), coloured(cube,red)
stop
We solve the third with an ordering constraint
pickColour(blue) threats the causal link <start, pickColour(red), handempty>
pickColour(red) threats the causal link <start, pickColour(blue), handempty>
go(loc_a,loc_b) threats the causal link <start,colour(plate,loc_a,blue), robot_at(loc_a)>


start
robot_at(loc_a), handempty, object_at(plate, loc_a), object_at(cube, loc_b)
inserting the white knight release colour we create another thread between color(plate, loc_a,blue) . because befre realease the color we have to colour. So we solve it using a promotoion. (An ordered link)
releaseColour(blue)
handempty
pickColour(blue)
handempty
pickColour(red)
go(loc_a,loc_b)
Not handempty
Not handempty
object_at(plate, loc_a), robot_at(loc_a), robot_has(blue)
object_at(plate, loc_b), robot_at(loc_b),robot_has(red)
colour(plate,loc_a,blue)
colour(cube,loc_b,red)
coloured(plate, blue), coloured(cube,red)
stop
We order pickColour(blue)
and pickColour(red) and we have to introduce a white knight
pickColour(blue) threats the causal link <start, pickColour(red), handempty>
pickColour(red) threats the causal link <start, pickColour(blue), handempty>


start
robot_at(loc_a), handempty, object_at(plate, loc_a), object_at(cube, loc_b)
releaseColour(blue)
handempty
pickColour(blue)
handempty
pickColour(red)
go(loc_a,loc_b)
Not handempty
Not handempty
object_at(plate, loc_a), robot_at(loc_a), robot_has(blue)
object_at(plate, loc_b), robot_at(loc_b),robot_has(red)
colour(plate,loc_a,blue)
colour(cube,loc_b,red)
coloured(plate, blue), coloured(cube,red)
stop
We order colour(plate,loc_a,blue) and releaseColour(blue)
Now releaseColour(blue) threats the Causal link <pickColour(blue), colour(plate,loca,blue), robot_has(blue)>

To sum up


start
pickColour(blue)
colour(plate,loc_a,blue)
releaseColour(blue)
pickColour(red)
go(loc_a,loc_b)
colour(cube,loc_b,red)
coloured(plate, blue), coloured(cube,red)
stop

GRAPHPLAN: EXERCISE

Given the following initial state **robot_at(loc_a), handempty, object_at(plate, loc_a), object_at(cube, loc_b),**

We have to reach the goal: **coloured(plate, blue), coloured(cube,red)**

Actions are modelled as follows:

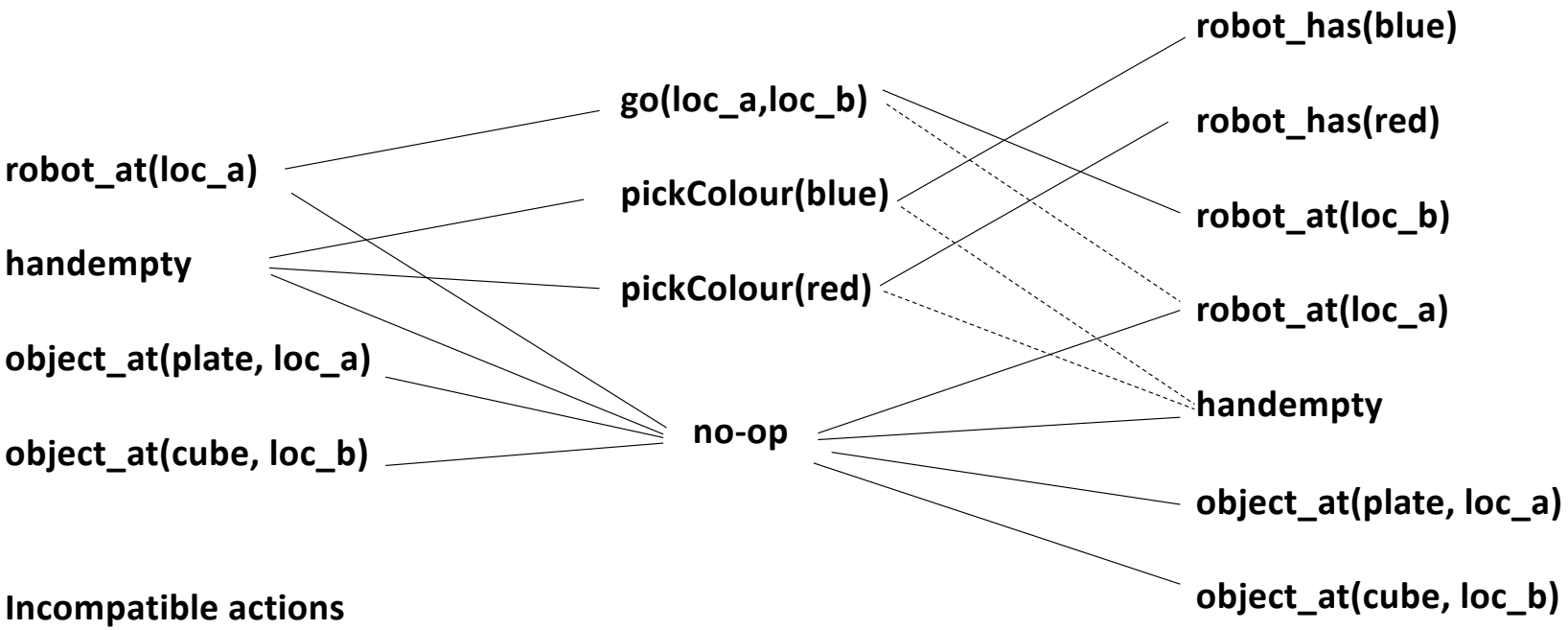
colour(Object, Location, Colour)
PRECOND: object_at(Object, Location), robot_at(Location), robot_has(Colour)
ADD: coloured(Object, Colour)

go(X,Y)
PRECOND: robot_at(X)
DELETE: robot_at(X)
ADD: robot_at(Y)

pickColour(C)
PRECOND: handempty
DELETE: handempty
ADD: robot_has(C)

releaseColour(C)
PRECOND: robot_has(C)
DELETE: robot_has(C)
ADD: handempty

Show two levels of graphplan for this problem showing incompatible actions and incompatible propositions



Incompatible actions

Interference:
pickColour(blue) and pickColour(red)
pickColour(blue) and no op on handempty
pickColour(red) and no op on handempty
go(loc_a,loc_b) and no op on robot_at(loc_a)

Incompatible propositions

robot_at(loc_a) and robot_at(loc_b)
robot_has(blue) and handempty
robot_has(red) and handempty