AI Planning

Our agent performs planning to move cargoes between cities. The agent can load a cargo at a city into an empty cargo space in an airplane that is at the same city. The agent can also unload a cargo from an airplane. The agent can fly an airplane from a city to another city. The airplanes *Plane1* and *Plane2* are at *Melbourne* and the airplane *Plane3* is at *Sydney*. *Plane1* has two cargo spaces *CS11* and *CS12*. *Plane2* has three cargo spaces *CS21*, *CS22* and *CS23*. *Plane3* has two cargo spaces *CS31* and *CS32*. Cargo *C1* is currently occupying cargo space *CS12* in *Plane1*. Cargo spaces *CS11*, *CS21*, *CS22*, *CS23*, *CS31* and *CS32* are currently empty. Cargoes *C2* and *C3* are currently at *Melbourne*. Cargoes *C4* and *C5* are currently at *Sydney*. The goal is to get the cargoes *C1*, *C2*, and *C3* to *Sydney* and to get the cargoes *C4* and *C5* to *Melbourne*.

1. Write down the initial state description and the agent's goals.
2. Write down STRIPS-style definitions of the three actions.
3. Write down a consistent partial-order plan (POP) with no open preconditions for this problem.

ANSWER:
1.
INITIAL STATE:

Plane(Plane1), Plane(Plane2), Plane(Plane3), City(Melbourne), City(Sydney),

Has(Plane1, CS11), Has(Plane1, CS12),
Has(Plane2, CS21), Has(Plane2, CS22), Has(Plane2, CS23),
Has(Plane3, CS31), Has(Plane3, CS32),

At(Plane1, Melbourne),
At(Plane2, Melbourne),
At(Plane3, Sydney),

Occupy(C1, CS12), Empty(CS11),
Empty(CS21), Empty(CS22), Empty(CS23),
Empty(CS31), Empty(CS32),

Cargo(C1), Cargo(C2), Cargo(C3), Cargo(C4), Cargo(C5),
At(C2, Melbourne),
At(C3, Melbourne),
At(C4, Sydney),
At(C5, Sydney),

Loadable(C1),
Loadable(C2),
Loadable(C3),
Loadable(C4),
Loadable(C5)

GOAL:
At(C1, Sydney),
At(C2, Sydney),
At(C3, Sydney),
At(C4, Melbourne),

At(C5, Melbourne)

2.
Action(Load(c, p, cs, loc):
PRECONDS: At(c, loc) ∧ At(p, loc) ∧ Has(p, cs) ∧ Empty(cs) ∧ Loadable(c).
EFFECTS: ¬Empty(cs) ∧ ¬At(c, loc) ∧ Occupy(c, cs))

Action(Unload(c, p, cs, loc):
PRECONDS: At(p, loc) ∧ Has(p, cs) ∧ Occupy(c, cs) ∧ Loadable(c).
EFFECTS: Empty(cs) ∧ At(c, loc) ∧ ¬Occupy(c, cs))

Action(Fly(p, from, to),
PRECOND: At(p,from) ∧ Plane(p) ∧ City(from) ∧ City(to)
EFFECT: ¬AT(p,from) ∧ At(p,to))

3.
Actions = {Start, Finish, Load(C2, Plane1, CS12, Melbourne), Fly(Plane1, Melbourne, Sydney), Unload(C1, Plane1, CS11, Sydney), Unload(C2, Plane1, CS12, Sydney), Load(C3, Plane2, CS21, Melbourne), Fly(Plane2, Melbourne, Sydney), Unload(C3, Plane2, CS21, Sydney), Load(C4, Plane3, CS31, Sydney), Load(C5, Plane3, CS32, Sydney), Fly(Plane3, Sydney, Melbourne), Unload(C4, Plane3, CS31, Melbourne), Unload(C5, Plane3, CS32, Melbourne) }

<<<The following are just part of the plan (to allow the Open Precond At(C2, Sydney) to be closed. See if you can close the rest of the Open Preconds yourself. >>>

Causal Links = { **Start** – [At(C2, Melbourne), At(Plane1, Melbourne), Has(Plane1, CS12), Empty(CS12), Loadable(C2)] -> *Load(C2, Plane1, CS12, Melbourne)*,

Start – [At(Plane1, Melbourne), Plane(Plane1), City(Melbourne), City(Sydney)] -> Fly(Plane1, Melbourne, Sydney),

Start –[Has(Plane1, CS12), Loadable(C2)]-> Unload(C2, Plane1, CS12, Sydney)
Fly(Plane1, Melbourne, Sydney)–[At(Plane1, Sydney)]-> Unload(C2, Plane1, CS12, Sydney),
Load(C2, Plane1, CS12, Melbourne)–[Occupy(C2, CS12)]-> Unload(C2, Plane1, CS12, Sydney)

Unload(C2, Plane1, CS12, Sydney) –[***At(C2, Sydney)***]-> **Finish**
……
}

Ordering Constraints = { Load(C2, Plane1, CS12, Melbourne) **<** Fly(Plane1, Melbourne, Sydney), … **}**

(Question? Why do we need the above O.C?)

Open Preconditions = { ***At(C2, Sydney)***, At(C1, Sydney), At(C3, Sydney), At(C4, Melbourne), At(C5, Melbourne) }

Uncertain reasoning

Kangaroo Electronics is an electronics manufacturer that uses an AI system FPD to detect faulty products. The FPD system classifies a product into one of two bags: Good and Bad. When a faulty product is examined by FPD, it is classified as Bad by FPD with a probability of 0.98. When a non-faulty product is examined by FPD, it is classified as Bad by FPD with a probability of 0.01. Statistics from Kangaroo Electronics shows that, on average, there is 1 in 200 products is faulty.

1. What is the probability that the next product is classified as Bad by FPD?

2. What is the probability that the next product is both faulty and classified as Bad by FPD?

3. What is the probability that the next product is non-faulty and classified as Bad by FPD?

4. A product is classified as Bad by FPD, what is the probability that it is actually faulty?

ANSWER:
FP – Product is faulty
FSBad – FPD System classifies BAD product

$P(FSBad \mid FP) = 0.98$
$P(FSBad \mid \neg FP) = 0.01$
$P(FP) = 1/200 = 0.005$

Thus,
$P(\neg FP) = 1 - P(FP) = 0.995$

1. Use Conditioning:
   $P(FSBad) = P(FSBad \mid FP)*P(FP) + P(FSBad \mid \neg FP)*P(\neg FP) =$
   $0.98*0.005 + 0.01*0.995 = 0.0049 + 0.00995 = 0.01485$ (= 1.485%)

2. $P(FP \wedge FSBad) = P(FSBad \mid FP)*P(FP) = 0.98*0.005 = 0.0049$ (= 0.49%)
3. $P(\neg FP \wedge FSBad) = P(FSBad \mid \neg FP)*P(\neg FP) = 0.01*0.995 = 0.00995$
4. $P(FP \mid FSBad) = P(FSBad \mid FP)*P(FP)/P(FSBad) = 0.0049/0.01485 = 0.329966$ (~ 33%)

Machine Learning:

After training your linear regression model, you observe a training error of 10% but a test error of 45%. What can you infer about this linear regression model?

ANSWER:
As the training error is much smaller compared to the test error, it is an indication that the model is overfitted (to the training data). That is, the model is not generalizable well to unseen data. Thus, there is a significant risk in deploying the model in a production environment as it is likely to return wrong predictions for new input instances.