CSCE 420 – Homework #5

due: Sat, Dec 6, 2025, 11:59 pm

Turn-in answers as a Word document (HW5.docx or .pdf) and commit/push it to your class github repo (in your HW5/ directory).

All homeworks must be typed, *not* hand-written and scanned.

1. Models in First-Order Logic.

Consider the following information about houses lining a street. Bob lives to the left of Alice. Carol lives to the left of Bob. David lives to the left of Bob. (Assume each person lives in a distinct house, and they all live on the same side of the street.) Furthermore, assume that the 'leftOf' relationship has the properties of a total order (antireflexive, symmetric, transitive). Write out axioms for 'leftOf' in FOL. Let the set of facts and axioms be S.

(To keep things simple, you can use terms like 'A' to refer to Alice's house, 'B' for Bob's house, etc. That way, you don't have to have separate objects for people and houses.)

1a. Draw a model depicting the scene described. Call it M1, such that sat(M1,S) or M1∈models(S).

leftOf axioms (S):

Antireflexive(an object can't be left of itself): ∀a ¬leftOf(a,a)

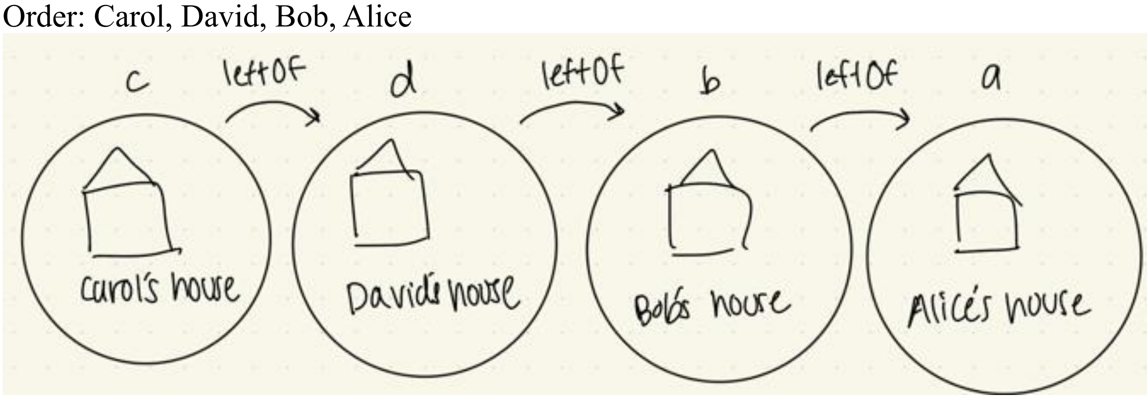
Antisymmetric(if a is the left of b then b can't be the left of a):

∀a∀b leftOf(a,b)-> ¬leftOf(b,a)

Transitive(If a is left of b and b is left of c then a is left of c):

∀a∀b∀c (leftOf(a,b) ^ leftOf(b,c)) -> leftOf(a,c)

Order: Carol, David, Bob, Alice



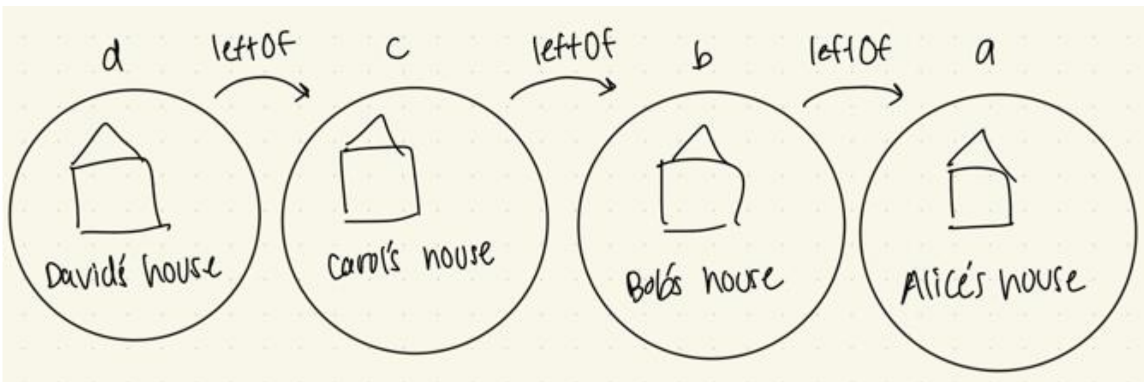
1b. Write M1 out formally in terms of <U,D,R>
U is the set of abstract objects in the universe: <A,B,C,D>
D is the denotations:

Constants: {Alice's house->A, Bob's house->B, Carol's house->C, David's house-> D}

Functions: none

R is the relations for each predicate:
R_{leftOf} = {<B,A>,<D,A>, <C,A>, <C,B>, <D,B>, <C,D>}

1c. Draw a *different* model of S, M2.
Order:David Carol Bob Alice



1d. Write out M2 as <U,D.R>
U is the set of abstract objects in the universe: <A,B,C,D>
D is the denotations:

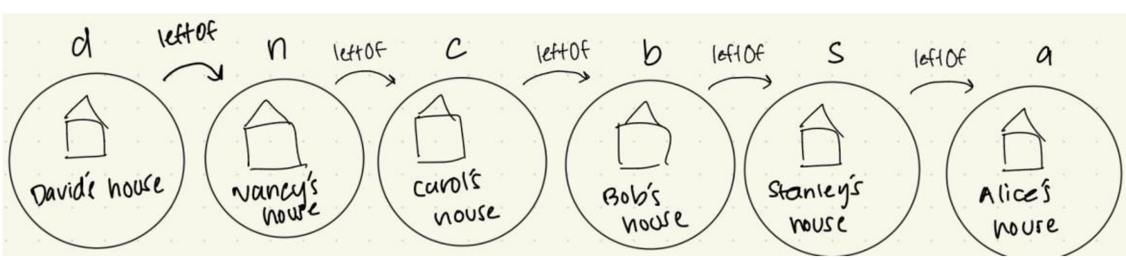
Constants: {Alice's house->A, Bob's house->B, Carol's house->C, David's house-> D}

Functions: none

R is the relations for each predicate:
R_{leftOf} = {<B,A>,<C,A>, <D,A>, <C,B>, <D,B>, <D,C>}

1e. Draw another model M3 in which there are at least 5 people living on the street.

Order: David Nancy Carol Bob Stanley Alice



1f. Write M3 as <U,D,R>
U is the set of abstract objects in the universe: <A,B,C,D>
D is the denotations:

Constants: {Alice's house->A, Bob's house->B, Carol's house->C, David's house-> D, Nancy's house->N, Stanley's House->S}

Functions: none

R is the relations for each predicate:
R_{leftOf} = {<S,A>,<B,A>, <C,A>, <N,A>, <D,A>, <B,S>, <C,S>, <N,S>, <D,S>, <C,B>, <N,B>, <D,B>, <N,C>, <D,C>, <D,N>}

1g. Suppose we added another axiom defining 'rightOf' as the *opposite* of 'leftOf', to create the theory (set of sentences) S'. Write the new axiom for 'rightOf'.

- **Does S' |= rightOf(A,D)?**
- **Does S' |= rightOf(C,D)?**
- **Does S' |= ∃x rightOf(x,A)?**

Use model theory arguments (reasoning about models) to explain your answers.
rightOf axioms (S'):

Antireflexive(an object can't be right of itself): ∀a ¬rightOf(a,a)
Antisymmetric(if a is the right of b then b can't be the right of a):
∀a∀b rightOf(a,b)-> ¬rightOf(b,a)
Transitive(If a is right of b and b is right of c then a is right of c):
∀a∀b∀c (rightOf(a,b) ^ rightOf(b,c)) -> rightOf(a,c)
Opposite of leftOf(if a is the right of b, then b is the left of a):
∀a∀b rightOf(a,b)↔leftOf(b,a)

Does S' |= rightOf(A,D)?
Yes. If we use the rule of transitivity, leftOf(D,B) and leftOf(B,A) derives leftOf(D,A). By using the biconditional, leftOf(D,A) is the same as rightOf(A,D).

Does S' |= rightOf(C,D)?
Here is the knowledge base given to us: Bob lives to the left of Alice. Carol lives to the left of Bob. David lives to the left of Bob.
There is no explicit sentence regarding the specific orientation of Carol and David. As you can see in M1 and M2 solutions previously, David and Carol can be on either side of each other and still satisfy the model. Therefore S' doesn't entail rightOf(C,D) as always true.

Does S' |= ∃x rightOf(x,A)?
Using the original 4 characters Alice, Bob, Carol, and David. Alice will always be the house on the very right.
If we use the rule of transitivity, leftOf(D,B) and leftOf(B,A) derives leftOf(D,A). By using the biconditional, leftOf(D,A) is the same as **rightOf(A,D).**
If we use the rule of transitivity, leftOf(C,B) and leftOf(B,A) derives leftOf(C,A). By using the biconditional, leftOf(D,A) is the same as **rightOf(A,C).**
By using the biconditional again, leftOf(B,A) is the same as **rightOf(A,B).**
In every derived rightOf predicate, A is always the right of every other person, so there will never exist a house on the right of A.

2. Bayesian Inference

Consider two factors that influence whether a student passes a given test: a) being smart, and b) studying. Suppose 30% of students believe they are intrinsically smart. But since students do not know a priori whether they are smart enough to pass a test, suppose 40% of will study for it anyway. (assume Smart and Study are independent). The causal relationship of these variables on the probability of actually passing the test can be expressed in a conditional probability table (CPT) as follows:

P(pass|Smart,Study)	¬smart	smart
¬study	0.2	0.7
study	0.6	0.95

prior probabilities: P(smart)=0.3, P(study)=0.4

a) Write out the equation for calculating joint probabilities, P(Smart,Study,Pass).
P(Smart,Study,Pass) equation can be written as P(Smart)*P(Study)*P(Pass|Smart, Study). In plain English, the joint probability of all 3 happening is the product of the probability of a student being smart, a student studying, and a student passing given that they studied and are smart.

b) Calculate all the entries in the full joint probability table (JPT) [a 4x2 matrix, like Fig 12.3 in the textbook; [Note: names of variables are capitalized, lower-case indicates truth value, e.g. 'pass' means Pass=T, and '-pass' means Pass=F.]

	Pass		-Pass	
	-smart	smart	smart	-smart
-study	0.084	0.126	0.054	0.336
study	0.168	0.114	0.006	0.112

Calculations:
P(Pass, -Smart, -Study) = P(-Smart)*P(-Study)*P(Pass|-Smart, -Study).
=0.7*0.6*0.2 = 0.084
P(Pass, -smart, study)= P(-smart)*P(study)*P(Pass|-smart, study)
=0.7*0.4*0.6=0.168
P(Pass, smart, -study)= P(smart)*P(-study)*P(Pass|smart, -study)
=0.3*0.6*0.7=0.126
P(Pass, smart, study)= P(smart)*P(study)*P(Pass|smart,study)
=0.3*0.4*0.95=0.114

P(-Pass,-Smart,-Study)=P(-smart)*P(-study)*P(-pass|-smart,-study)
=0.7*0.6*(1-0.2)=0.336
P(-Pass, -smart, study)= P(-smart)*P(study)*P(-Pass|-smart, study)
=0.7*0.4*(1-0.6)=0.112
P(-Pass, smart, -study)= P(smart)*P(-study)*P(-Pass|smart, -study)
=0.3*0.6*(1-0.7)=0.054
P(-Pass, smart, study)= P(smart)*P(study)*P(-Pass|smart,study)

=0.3*0.4*(1-0.95)=0.006

c) From the JPT, compute the probability that a student is smart, given that they pass the test but did not study.
P(Smart|pass,-study)= P(Smart,pass,-study)/(P(smart,pass,-study)+P(-smart,pass,-study))
=(0.126)/(0.126+0.084) = 0.6 or 60%

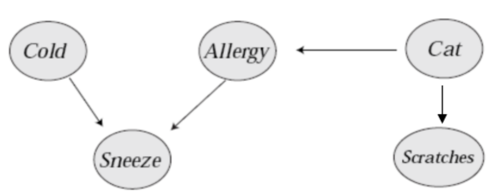
d) From the JPT, compute the probability that a student did not study, given that they are smart but did not pass the test.
P(-study|smart,-pass)= P(-study,smart,-pass)/(P(-study,smart,-pass)+P(study,smart,-pass)
=(.054)/(.054+.006)=0.9 or 90%

e) Compute the marginal probability that a student will pass the test given that they are smart.
Equation = (P(Smart,study,pass)+P(smart,-study,pass))/P(smart)
=(.126+.114)/(0.3)=.8 or 80%

f) Compute the marginal probability that a student will pass the test given that they study.
Equation = (P(Smart,study,pass)+P(-smart,study,pass))/P(study)
=(.168+.114)/(0.4)=.705 or 70.5%

3. Bayesian Networks.

Here is a probabilistic model that describes what it might mean when a person sneezes, e.g. depending on whether they have a cold, or whether a cat is present and they are allergic. Scratches on the furniture would be evidence that a cat had been present.



Cold = False	Cold = True
.95	.05

Cat Presence = False	Cat Presence = True
.98	.02

Cat Presence	Allergic Reaction = False	Allergic Reaction = True
False	.95	.05
True	.25	.75

Cold	Allergic Reaction	Sneeze = False	Sneeze = True
False	False	.99	.01
False	True	.3	.7
True	False	.2	.8
True	True	.1	.9

Cat Presence	Scratches = False	Scratches = True
False	.95	.05
True	.5	.5

a) Using Equation 13.2 in the textbook (p. 415), write out the expression for the joint probability for any state (i.e. combination of truth values for the 5 variables in this problem). [Note: Use capital letters for names of variables, and lower-case to indicate truth value, e.g. 'cold' means Cold=T, and '-cold' means Cold=F.]

P(Cold,Sneeze,Allergic,Scratches,Cat) =
P(Cat)*P(Scratches|Cat)*P(Allergy|Cat)*P(Sneeze|Allergy,Cold)*P(Cold)

This equation is derived from following the order of nodes in the Bayesian network. Cat and Cold are independent variables so the probability is just one variable. The other intermediate nodes are dependent on other variables so they are written in P(A|B) format.

b) Use the equation above to calculate the *joint probability* that the person sneezes, but does not have a cold, has a cat, is allergic, and there are scratches on the furniture:

P(-Cold,Sneeze,Allergic,Scratches,Cat) =
P(Cat)*P(Scratches|Cat)*P(Allergy|Cat)*P(Sneeze|Allergy,-Cold)*P(-Cold)
= .02*.5*.75*.7*.95
=.0049875

This equation mirrors the equation in part A but replaces all probabilities that contained Cold with -Cold.

c) Use *normalization* to calculate the conditional probability that a person has cat, given that they sneeze and are allergic to cats, but do not have a cold, and there are scratches on the furniture.

P(cat|-cold,sneeze,allergic,scratches) =
P(cat,-cold,sneeze,allergic,scratches)/ (P(cat,-cold,sneeze,allergic,scratches)+ P(-cat, -cold,sneeze,allergic,scratches))
= .0049875(calculated in part b)/(.0049875+(.98*.95*.7*.05*.05))
=0.7545

By using the normalization equation in our PPT notes. P(X|Y,Z) = P(X,Y,Z)/P(Y,Z). The denominator is marginalized over X to get P(Y,Z) = P(X,Y,Z)+P(¬X,Y,Z). In this case, I've written the equation to fit 5 variables instead of 3 but it follows the same rules.

d) Use Bayes' Rule to re-write the expression for P(cat|scratches). Look up the values for the numerator in the table above.

P(cat|scratches) =
(P(scratches|cat)*P(cat))/P(scratches)
The values for the numerators are P(Scratches|cat)=0.5 and P(cat)=0.02

e) The denominator in the answer for (d) would require *marginalization* over how many joint probabilities? Write out the expressions for these (i.e. expand the denominator, but you don't have to calculate the actual values).

The denominator requires marginalization over 2 joint probabilities to consider the presence of scratches WITH a cat and scratches WITHOUT a cat.
P(scratches)= P(Scratches|cat)+P(Scratches|-cat)

4. **PDDL and Situation Calculus**

To start a car, you have to be at the car and have the key, and the car has to have a charged battery and the tank has to have gas. Afterwards, the car will be running, and you will still be at the car and have the key after starting the engine.

a. Write a <u>PDDL operator</u> to describe this action.
(note: you can express this ego-centrically – you don't have to refer explicitly to the person starting the car; but the operator should take the car being started as an argument)
PDDF operators for problem 4 were written referencing the class PPT notes.
Init(Person(a)^Car(b)^Key(c)^Battery(d)^Tank(e))
Action(StartACar(a,b,c,d,e))
Precondition: At(a,b) ^ hasKey(a,c) ^ hasBattery(b,d) ^ hasGas(b,e)
Effect: Running(c) ^ At(a,b) ^ hasKey(a,c)

Note* hasBattery(b,d) means car b has battery d.
Note* hasGas(b,e) means car b has gas in tank e.
b. Describe the same operator using <u>Situation Calculus</u> (remember to add a situation argument to your predicates).
∀a,b,c,d,e StartACar(a,b,c,d,e,s)-> At(a,b,s) ^ hasKey(a,c,s) ^ hasBattery(b,d,s) ^ hasGas(b,e,s)
s represents the current state/situation we are at in this point in time.

c. Add a <u>Frame Axiom</u> that says that starting this car will not change whether any other car is out of gas (tank empty).
Frame axioms for problem 4 were written referencing the class PPT notes.
Assuming f is any other car and g is the car's gas tank.
∀a,b,c,d,e,f,g At(a,b,s) ^ hasKey(a,c,s) ^ hasBattery(b,d,s) ^ hasGas(b,s)→ [hasGas(f,g,s) ↔ hasGas(f,StartACar(a,b,c,d,e),s))]

The state of car g having gas is the same as the state of car g having gas after we have started car b with the preconditions.