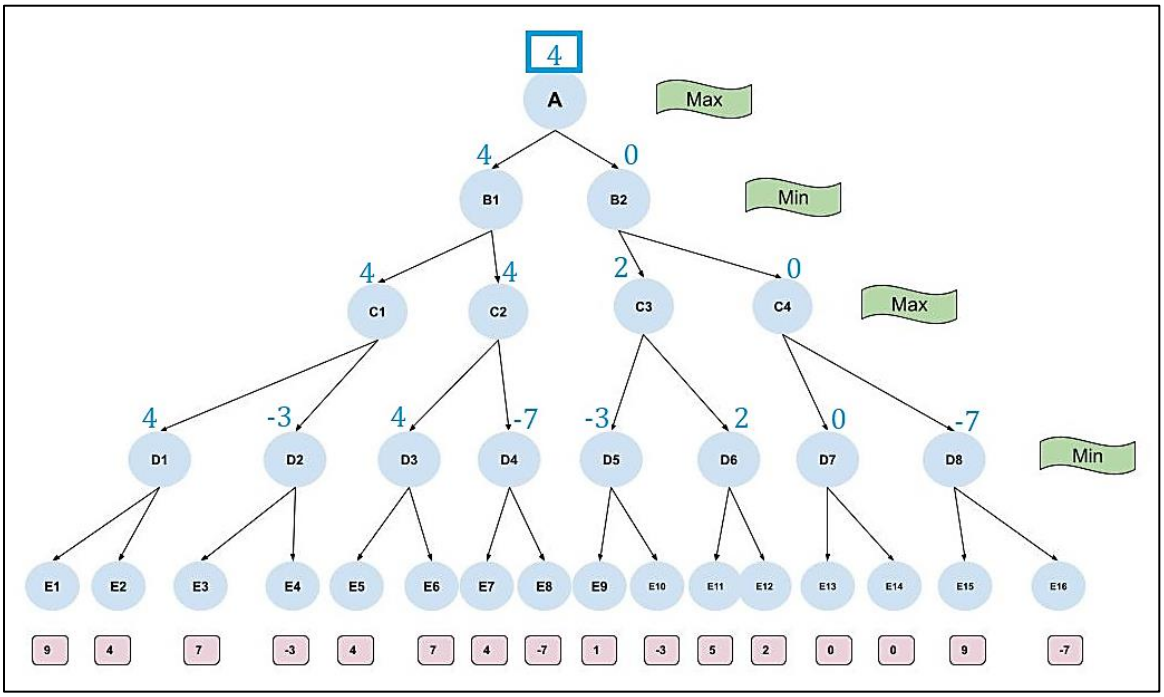
Homework 2

Section 1: Written Portion [72pts]

Problem 1: Adversarial Search [10pts]

a. [8pts] Apply minimax to the tree below. Please provide an annotated version of the tree. *You can use either a screenshot of the image provided below or save the image in the homework folder. Add this to your solution document.*



b. [2pts] If we use alpha-beta pruning (from left to right), which nodes could we omit? Please state ***every node*** that would be omitted.

D4, E7, E8, E10, C4, D7, E13, E14, D8, E15, E16

Problem 2: Entailment and Equivalence [10pts]

State whether each of the following statements is correct and provide a justification.
(*Hint: You may find it useful to construct a truth table for each part.)*

a. [2pts] $(A \vee B) \wedge C \models A \vee B$

Correct. We have the following truth table.

A	B	C	A ∨ B	(A ∨ B) ∧ C
T	T	T	**T**	**T**
T	T	F	T	F
T	F	T	**T**	**T**
T	F	F	T	F
F	T	T	**T**	**T**
F	T	F	T	F
F	F	T	F	F
F	F	F	F	F

$(A \vee B) \wedge C \models A \vee B$ because in every model in which $(A \vee B) \wedge C$ is true, $A \vee B$ is also true.

b. [2pts] $(A \wedge B) \models (A \Leftrightarrow B)$

Correct. We have the following truth table.

A	B	A ∧ B	A ⇔ B
T	T	**T**	**T**
T	F	F	F
F	T	F	F
F	F	F	T

$(A \wedge B) \models (A \Leftrightarrow B)$ because in every model in which $(A \wedge B)$ true, $(A \Leftrightarrow B)$ is also true.

c. [2pts] False ⊨ True

Correct. If there are no models in which “False” is true, then there does not need to be any models in which “True” is also true.

d. [2pts] True ⊨ False

Incorrect. If every model makes “True” true but no models make “False” true, then True cannot entail False.

e. [2pts] $(C \vee (\neg A \wedge \neg B)) \equiv ((A \Rightarrow C) \wedge (B \Rightarrow C))$

Correct.

$C \vee (\neg A \wedge \neg B) \equiv (\neg A \vee C) \wedge (\neg B \vee C)$ Distributive Law

$\neg A \vee C \equiv A \Longrightarrow C$ Relation by Implication

$\neg B \vee C \equiv B \Longrightarrow C$

$\therefore \quad (\neg A \vee C) \wedge (\neg B \vee C) \equiv \underline{(A \Rightarrow C) \wedge (B \Rightarrow C)}$

We also have the following truth table.

A	B	C	$C \vee (\neg A \wedge \neg B)$	$(A \Longrightarrow C) \wedge (B \Longrightarrow C)$
T	T	T	**T**	**T**
T	T	F	**F**	**F**
T	F	T	**T**	**T**
T	F	F	**F**	**F**
F	T	T	**T**	**T**
F	T	F	**F**	**F**
F	F	T	**T**	**T**
F	F	F	**T**	**T**

Problem 3: Valid, Satisfiable, or Unsatisfiable [10pts]

State whether each of the following sentences is valid, satisfiable but not valid, or unsatisfiable. Provide a justification using truth tables, using a proof, and/or by simplifying the sentences using equivalence rules. If the statement is satisfiable but not valid, please support your reasoning with a counterexample.

a. [2pts] $\neg A \Rightarrow A$

Satifiable but not valid. If A is true, then $\neg A \Rightarrow A$ is true. If A is false, then $\neg A \Rightarrow A$ is false.

A	$\neg A$	$\neg A \Longrightarrow A$
T	**F**	**T**
F	**T**	**F**

b. [2pts] $(A \Rightarrow B) \Rightarrow (\neg B \Rightarrow \neg A)$

Valid. Consider the following truth table.

A	B	$A \Longrightarrow B$	$\neg B \Rightarrow \neg A$	$(A \Rightarrow B) \Rightarrow (\neg B \Rightarrow \neg A)$
T	T	T	T	**T**
T	F	F	F	**T**
F	T	T	T	**T**
F	F	T	T	**T**

c. [2pts] $\neg[(A \wedge B) \wedge \neg(A \Rightarrow B)]$

Valid. Consider the following truth table.

A	B	$A \wedge B$	$A \Longrightarrow B$	$\neg(A \Rightarrow B)$	$(A \wedge B) \wedge \neg(A \Rightarrow B)$	$\neg[(A \wedge B) \wedge \neg(A \Rightarrow B)]$
T	T	T	T	F	F	**T**
T	F	F	F	T	F	**T**
F	T	F	T	F	F	**T**
F	F	F	T	F	F	**T**

d. [2pts] $(\neg A \wedge B) \Rightarrow (A \Rightarrow B)$

Valid. Consider the following truth table.

A	B	$\neg A \wedge B$	$A \Rightarrow B$	$(\neg A \wedge B) \Rightarrow (A \Rightarrow B)$
T	T	F	T	**T**
T	F	F	F	**T**
F	T	T	T	**T**
F	F	F	T	**T**

e. [2pts] $[(\neg A \Rightarrow B) \wedge (\neg A \wedge B)] \leftrightarrow (A \vee \neg B)$

Unsatisfiable. Consider the following truth table.

A	B	A ⇒ B	¬(A ⇒ B)	¬A ∧ B	A ∨ ¬B	(¬A ⇒ B) ∧ (¬A ∧ B)	[(¬A ⇒ B) ∧ (¬A ∧ B)] ↔ (A ∨ ¬B)
T	T	T	F	F	T	**F**	**F**
T	F	F	T	F	T	**F**	**F**
F	T	T	F	T	F	**F**	**F**
F	F	T	F	F	T	**F**	**F**

Problem 4: First Order Logic [10pts]

Consider the following vocabulary:

1. *Cities(c): c is a city, valid constants are {New York, San Francisco, Seattle, Detroit}*
2. *Person(p): p is a person, valid constants are {Barrett, Haoyi}*
3. *Travels(p, c): person p travels to city c*
4. *Neighbor(p1, p2): person p1 follows person p2 on a social network*
5. *Friend(p1, p2): person p1 is a friend of person p2*

Translate the following sentences into first-order logic:

a. [2pts] Haoyi has a friend who travels to New York.

$$\exists p\ \boldsymbol{Friend(Haoyi, p) \wedge Travels(p, New\ York) \wedge Person(p)}$$

b. [2pts] Every person travels to a city.

$$\forall p[\exists c\ \boldsymbol{Person(p) \wedge\ Cities(c) \wedge Travels(p, c)}]$$

c. [2pts] Barrett does not have any friends who travel to Seattle.

$$\neg\exists p\ \boldsymbol{Friends(Barrett, p) \wedge\ Travels(p, Seattle) \wedge Person(p)}$$

d. [2pts] There is a person who travels to San Francisco whose followers all travel to Detroit.

$$\exists p_1 \Big[\boldsymbol{Travels(p_1, San\ Franciso) \wedge Person(p_1)} \wedge \big[\forall p_2\ [\boldsymbol{Person(p_2) \wedge Travels(p_2, Detroit) \wedge Neighbor(p_2, p_1)}] \big] \Big]$$

e. [2pts] At most one person travels to San Francisco. (*hint: if there are two variables which both represent someone who travels to* San Francisco*, what must be true about those two variables?*)

$$\neg\exists p_1 \Big[\boldsymbol{Travels(p_1, San\ Franciso) \wedge Person(p_1)} \wedge \big[\exists p_2\ [\boldsymbol{Travels(p_2, San\ Franciso) \wedge Person(p_2) \wedge p_1 \neq p_2}] \big] \Big]$$

Problem 5: Conjunctive Normal Form [12pts]

Convert the following first-order logic statements into Conjunctive Normal Form (CNF). Show your work accompanied with brief comments explaining each step.

a. [2pts] $\exists x\ A(x) \Rightarrow B(x)$

$\exists x \sim A(x) \vee B(x)$	Translate implication to disjunction.
$\mathbf{\sim A(X) \vee B(X)}$	Eliminate existentials via skolemization.

b. [4pts] $\forall x\ [\ [\ \exists y\ A(x, y) \wedge B(y)\ ] \Rightarrow C(x)\ ]$

$\forall x\ [\ \sim[\ \exists y\ A(x, y) \wedge B(y)\ ] \vee C(x)\ ]$	Translate implication to disjunction.
$\forall x\ [\ [\ \forall y \sim A(x, y) \wedge \sim B(y)\ ] \vee C(x)\ ]$	Move negation inwards.
$[\ [\sim A(x, y) \wedge \sim B(y)\ ] \vee C(x)\ ]$	Drop universal quantifiers.
$\mathbf{[\sim A(x, y) \vee C(x)] \wedge [\sim B(y) \vee C(x)]}$	Distribute $\vee$ over $\wedge$.

c. [6pts] $\exists x\ [\ [\ \exists y\ A(x, y) \wedge B(y)\ ] \Rightarrow [\ \forall y\ C(x, y)\ ]\ ]$

$\exists x\ [\ \sim[\ \exists y\ A(x, y) \wedge B(y)\ ] \vee [\ \forall y\ C(x, y)\ ]\ ]$	Translate implication to disjunction.
$\exists x\ [\ [\ \forall y \sim A(x, y) \vee \sim B(y)\ ] \vee [\ \forall y\ C(x, y)\ ]\ ]$	Move negation inwards.
$\exists x\ [\ [\ \forall y \sim A(x, y) \vee \sim B(y)\ ] \vee [\ \forall \mathrm{z}\ C(x, \mathrm{z})\ ]\ ]$	Standardize.
$[\ \forall y \sim A(\mathrm{X}, y) \vee \sim B(y)\ ] \vee [\ \forall \mathrm{z}\ C(\mathrm{X}, \mathrm{z})\ ]$	Skolemize.
$\mathbf{\sim A(X, y) \vee \sim B(y) \vee C(X, z)}$	Drop universal quantifiers.

Problem 6: Unification [6pts]

Determine whether or not the following pairs can be unified. If they can be unified, present the most general unifier and show the result. If they cannot be unified, explain why. Use the convention that variables are lowercase letters while constants are capital letters. Functions are represented as either capital letters or as “ManufacturerOf” or “EmployerOf” or “FriendOf” or “GameOf”. Use the unique-names assumption.

a. P(x, y, y) P(A, B, C)
Cannot be unified. Cannot substitute y with two different constants at the same time.

b. P(B, ManufacturerOf(x)) P(B, EmployerOf(x))
Cannot be unified. There are no variables to replace.

c. C(B,FriendOf(B), z) C(B, FriendOf(B), GameOf(FriendOf(B)))
Can be unified. [z/GameOf(FriendOf(B))]

d. P(x, y) P(A, FriendOf(B))
Can be unified. [x/A, y/FriendOf(B)]

e. Q(B, A) R(x, FriendOf(y))
Cannot be unified. Cannot unify because predicates are not the same.

f. Q(B, A) Q(x, FriendOf(B))
Cannot be unified. Cannot replace constant A with function FriendOf(B).

Problem 7: Resolution and Refutation [14pts]

Consider the following sentences in our knowledge base:

1. ***Coco is a puppy.***
2. ***Cristina is a dog lover.***
3. ***All puppies are dogs.***
4. ***If any puppy follows directions, that puppy is good.***
5. ***There is a dog that follows directions.***
6. ***If any dog follows directions, Coco follows directions.***
7. ***Whenever a dog lover finds a dog, if the dog is good, the dog lover adopts the dog.***
8. ***Cristina finds Coco.***

a. [4pts] Translate the above knowledge base into Conjunctive Normal Form, and give each sentence a number from 1 to 8.
Use the following conventions:
1. *Puppy(x) – x is a puppy.*
2. *Dog(x) – x is a dog.*
3. *Good(x) – x is good.*
4. *DogLover(x) – x is a dog lover.*
5. *FollowsDirections(x) – x follows directions.*
6. *Finds(x, y) – x finds y*
7. *Adopts(x, y) – x adopts y*

1. Puppy(Coco)

2. DogLover(Cristina)

3. ∀p Puppy(p) ⟹ Dog(p)
∀p ~Puppy(p) ∨ Dog(p)
~Puppy(p) ∨ Dog(p)

4. ∀p [Puppy(p) ∧ FollowsDirections(p) ⟹ Good(p)]
∀p [~[Puppy(p) ∧ FollowsDirections(p)] ∨ Good(p)]
∀p [[~Puppy(p) ∨ ~FollowsDirections(p)] ∨ Good(p)]
[~Puppy(p) ∨ ~FollowsDirections(p)] ∨ Good(p)
~Puppy(p) ∨ ~FollowsDirections(p) ∨ Good(p)

5. ∃d Dog(d) ∧ FollowsDirections(d)
Dog(*D*) ∧ FollowsDirections(*D*)
a. Dog(*D*)
b. FollowsDirections(*D*)

6. [∃d Dog(d) ∧ FollowsDirections(d)] ⟹ FollowsDirections(Coco)
~[∃d Dog(d) ∧ FollowsDirections(d)] ∨ FollowsDirections(Coco)
[∀d ~[Dog(d) ∧ FollowsDirections(d)]] ∨ FollowsDirections(Coco)
[∀d [~Dog(d) ∨ ~FollowsDirections(d)]] ∨ FollowsDirections(Coco)
[~Dog(d) ∨ ~FollowsDirections(d)] ∨ FollowsDirections(Coco)
~Dog(d) ∨ ~FollowsDirections(d) ∨ FollowsDirections(Coco)

7. ∀x [∀d [[DogLover(x) ∧ Finds(x, d) ∧ Dog(d)] ⟹ [Good(d) ⟹ Adopts(x, d)]]]
∀x [∀d [~[DogLover(x) ∧ Finds(x, d) ∧ Dog(d)] ∨ [~Good(d) ∨ Adopts(x, d)]]]
∀x [∀d [[~DogLover(x) ∨ ~Finds(x, d) ∨ ~Dog(d)] ∨ [~Good(d) ∨ Adopts(x, d)]]]
~DogLover(x) ∨ ~Finds(x, d) ∨ ~Dog(d) ∨ ~Good(d) ∨ Adopts(x, d)

8. Finds(Cristina, Coco)

b. [10pts] Use resolution-refutation to **prove that Cristina adopts Coco** by filling in the rest of the table below. In the Number column, write the number of the sentence in the KB. In the Sentence column, write the resulting sentence you are adding to the KB. In the Resolution and Substitution column, write which two sentences you are resolving together, and what you are substituting for the variables in the two sentences, if necessary.

Number	Sentence	Resolution and Substitution
9	¬Adopts(Cristina, Coco)	Assume negation.
10	~DogLover(Cristina) ∨ ~Finds(Cristina, Coco) ∨ ~Dog(Coco) ∨ ~Good(Coco) ∨ Adopts(Cristina, Coco)	Resolve 7 and 9; substitute {x/Cristina, d/Coco}
11	~Finds(Cristina, Coco) ∨ ~Dog(Coco) ∨ ~Good(Coco) ∨ Adopts(Cristina, Coco)	Resolve 2 and 10.
12	~Dog(Coco) ∨ ~Good(Coco) ∨ Adopts(Cristina, Coco)	Resolve 8 and 11.
13	~Dog(Coco) ∨ ~Good(Coco)	Resolve 9 and 12.
--	--	--
14	Dog(Coco)	Resolve 1 and 3; substitute {p/Coco}
15	~Good(Coco)	Resolve 13 and 14.
--	--	--
16	~FollowsDirections(Coco) ∨ Good(Coco)	Resolve 1 and 4; substitute {p/Coco}

17	Good(Coco) v ~Dog(d) v ~FollowsDirections(d)	Resolve 6 and 16.
18	Good(Coco)	Resolve 5 and 17; subtitute {d/D}
--	--	--
19	{}	Resolve 15 and 18.

Section 2: Coding Portion [28pts]

Constraint Satisfaction

1. [1pt] Consider the image above on the left. There are two types of variables, those with numbers listed and those without. Assume that variables with the number shown on the left can only have that value in their domain.
   List all of the variables (e.g., G4) in the bottom center box (the one that has a red border) and their corresponding domains after **only unary constraints** have been enforced. Do not enforce arc constraints yet.
   G4: {6}
   G5: {1, 2, 3, 4, 5, 6, 7, 8, 9}
   G6: {9}
   H4: {2}
   H5: {1, 2, 3, 4, 5, 6, 7, 8, 9}
   H6: {3}
   I4: {1, 2, 3, 4, 5, 6, 7, 8, 9}
   I5: {1}
   I6: {1, 2, 3, 4, 5, 6, 7, 8, 9}

2. [1pt] Reduce the domain for variables in this box by enforcing the arc consistency for these domains with the entire puzzle. Then, perform AC-3 over arcs within the red box, only adding arcs that are within the red box. List the new domains for the variables in this box.
   G4: {6}, G5: {8}, G6: {9}
   H4: {2}, H5: {5}, H6: {3}
   I4: {4}, I5: {1}, I6: {7}

Reflection:

4. [1pt] Run sudoku.py on the puzzle in sudoku_comparison.txt without using any additional inference (so, the unmodified sudoku.py file) and using AC-3. How many backtracks were made without using AC-3? How many were made when using AC-3?
   Backtracks without AC-3: **12441**
   Backtracks with AC-3: **4**

5. [1pt] What is the impact of this difference?
   Integrating AC-3 into our backtracking search significantly improved its efficiency.