

AI-II Solutions

1. Propositional Logic

- (1) true: every sentence is true in all models where false is true (none!).
- (2) false: false is not true in any model, let alone every model.
- (3) true: $A \Leftrightarrow B$ is true when both A and B are true.
- (4) true: A true and B false together satisfy the sentence.

2. First-Order Logic I

- (a)
- | | |
|---------------------|---|
| Apartment(x): | x is an apartment |
| In(x, y): | x is in y |
| Rent(x): | Function returning the rent for apartment x |
| Cat(x): | x is a cat |
| Likes (x, y): | x likes y |
| At (x, y): | x is at the location y |
| CaughtMice (x): | function returning the number of mice caught by x |
| Milk: | constant representing milk |
| Cardiff: | constant representing the city Cardiff |
| London: | constant representing the city London |

A: $\exists x \text{ Cat}(x) \wedge \text{Likes}(x, \text{Milk})$

B: $\forall x \text{ Cat}(x) \Rightarrow \text{CaughtMice}(x) > 0$

C: $\exists x \forall y \text{ Cat}(x) \wedge \text{At}(x, \text{Cardiff}) \wedge \text{Cat}(y) \wedge \text{At}(y, \text{London})$
 $\wedge \text{CaughtMice}(x) > \text{CaughtMice}(y)$

D: $\forall x \text{ Apartment}(x) \wedge \text{In}(x, \text{Cardiff})$
 $\Rightarrow (\exists y \text{ Apartment}(y) \wedge \text{In}(y, \text{London}) \wedge \text{Rent}(y) > \text{Rent}(x))$

E: $\exists x \text{ Apartment}(x) \wedge \text{In}(x, \text{London}) \wedge \text{Rent}(x) < 1000 \wedge$
 $(\forall y \text{ Apartment}(y) \wedge \text{In}(y, \text{London}) \wedge \text{Rent}(y) < 1000 \Rightarrow x = y)$

F: $\forall x \text{ Apartment}(x) \wedge (\forall y \text{ Apartment}(y) \wedge \text{In}(y, \text{Cardiff}) \Rightarrow \text{Rent}(x) > \text{Rent}(y))$
 $\Rightarrow \text{In}(x, \text{London})$

- (b)
 - i. There are at least two mail carriers in Cardiff.
 - ii. All mail carriers in Cardiff know each other.
 - iii. There is a dog which hates every cat which eats birds.

3. First-Order Logic II

- (a)
- **A:** $\forall x \forall y (\neg(x = y) \Rightarrow \neg(\text{DNA}(x) = \text{DNA}(y)))$
 $\wedge \text{DerivedFrom}(\text{DNA}(x), \text{DNA}(\text{Mother}(x)), \text{DNA}(\text{Father}(x)))$
 - **B:** $\exists x \text{ Mouse}(x) \wedge \text{Trapped}(x) \wedge \forall y \text{ Mouse}(y) \wedge \text{Trapped}(y) \Rightarrow x = y$
 - **C:** $\forall x \forall y \text{ Cat}(x) \wedge \text{Cat}(y) \wedge \text{FellInto}(x, \text{Pit}) \wedge \text{FellInto}(y, \text{Pit}) \Rightarrow x = y$
 or (for any pit)
 $\forall x \forall y \text{ Cat}(x) \wedge \text{Cat}(y) \wedge (\exists z \text{ Pit}(z) \wedge \text{FellInto}(x, z)) \wedge (\exists z \text{ Pit}(z) \wedge \text{FellInto}(y, z)) \Rightarrow$
 $x = y$
- (b)
- Modus tollens is sound, because:
 $P \Rightarrow Q \wedge \neg Q \equiv (\neg P \vee Q) \wedge \neg Q \equiv (\neg P \wedge \neg Q) \vee (Q \wedge \neg Q) \equiv (\neg P \wedge \neg Q)$, i.e. $\neg P$ is true (indirect proof, enumeration, etc. also possible).
 - Abduction is unsound, because $P \Rightarrow Q$ is true if P is false independent of the value of Q (counter example).
 - Abduction becomes sound under the condition that P is the only possible cause for Q . More generally if P is a likely cause for Q it may be applied (but then it might be better to use probabilistic reasoning).

4. Cats and Fish

Knowledge base:

- (1) $\forall x \text{ Cat}(x) \Rightarrow \text{Likes}(x, \text{Fish})$
- (2) $\forall x \forall y \text{ Cat}(x) \wedge \text{Likes}(x, y) \Rightarrow \text{Eats}(x, y)$
- (3) $\text{Cat}(\text{Ziggy})$

(a) With forward chaining:

- (1), (3) give: (4) $\text{Likes}(\text{Ziggy}, \text{Fish})$
- (3), (4), (2) give: (5) $\text{Eats}(\text{Ziggy}, \text{Fish})$
- Hence, the answer is yes.

(b) With backward chaining:

- Goal unifies with conclusion of (2), meaning we get two new sub-goals (a) $\text{Cat}(\text{Ziggy})$ and (b) $\text{Likes}(\text{Ziggy}, \text{Fish})$
- (a) matches (3) without yielding further sub-goals
- (b) unifies with conclusion of (1), so we get the new sub-goal (c) $\text{Cat}(\text{Ziggy})$
- (c) matches (3)
- No further sub-goals are left, so the answer is yes

5. Resolution in First-Order Logic

- Convert KB into CNF:
 - A:** $\forall x \forall y \text{ LostATrial}(x) \wedge \text{Client}(y, x) \Rightarrow \neg \text{Happy}(y)$.
 $\neg(\text{LostATrial}(x) \wedge \text{Client}(y, x)) \vee \neg \text{Happy}(y)$.
 $\neg \text{LostATrial}(x) \vee \neg \text{Client}(y, x) \vee \neg \text{Happy}(y)$.
 - B:** $\forall x \text{ SuccessfulLawyer}(x) \Rightarrow \exists y \text{ Client}(y, x) \wedge \text{Happy}(y)$.
 (introduce skolem function $f(x)$ for y)
 $\text{SuccessfulLawyer}(x) \Rightarrow \text{Client}(f(x), x) \wedge \text{Happy}(f(x))$.
 $\neg \text{SuccessfulLawyer}(x) \vee (\text{Client}(f(x), x) \wedge \text{Happy}(f(x)))$.
 - B1:** $\neg \text{SuccessfulLawyer}(x) \vee \text{Client}(f(x), x)$.
 - B2:** $\neg \text{SuccessfulLawyer}(x) \vee \text{Happy}(f(x))$.
 - Negate C:
 - C':** $\neg(\forall x \text{ SuccessfulLawyer}(x) \Rightarrow \neg \text{LostATrial}(x))$.
 - Convert C' into CNF:
 - C':** $\exists x \neg(\text{SuccessfulLawyer}(x) \Rightarrow \neg \text{LostATrial}(x))$.
 (introduce skolem constant H for x)
 $\neg(\text{SuccessfulLawyer}(H) \Rightarrow \neg \text{LostATrial}(H))$.
 $\neg(\neg \text{SuccessfulLawyer}(H) \vee \neg \text{LostATrial}(H))$.
 $\text{SuccessfulLawyer}(H) \wedge \text{LostATrial}(H)$.
 - C1':** $\text{SuccessfulLawyer}(H)$.
 - C2':** $\text{LostATrial}(H)$.
 - Resolution proof:
 - Resolve **C1'** and **B1** with substitution x/H :
D: $\text{Client}(f(H), H)$.
 - Resolve **C2'** and **A** with substitution x/H :
E: $\neg \text{Client}(y, H) \vee \neg \text{Happy}(y)$.
 - Resolve **D** and **E** with substitution $y/f(H)$:
F: $\neg \text{Happy}(f(H))$.
 - Resolve **F** and **B2** with substitution x/H :
G: $\neg \text{SuccessfulLawyer}(H)$.
- Resolving **G** and **C1'** gives empty clause. Thus, we have a contradiction, which proves that **C** is true.

6. Marcus hates Cæsar

- (a) **A:** Human(Marcus)
B: Pompeian(Marcus)
C: $\forall x$ Pompeian(x) \Rightarrow Roman(x)
D: Ruler(Cæsar)
E: $\forall x$ Roman(x) \Rightarrow LoyalTo(x , Cæsar) \vee Hates(x , Cæsar)
F: $\forall x \exists y$ LoyalTo(x , y)
G: $\forall x \forall y$ Human(x) \wedge Ruler(y) \wedge TryAssassinate(x , y) \Rightarrow \neg LoyalTo(x , y)
H: TryAssassinate(Marcus, Cæsar)
- (b) • Convert to CNF:
A: Human(Marcus)
B: Pompeian(Marcus)
C: \neg Pompeian(x_1) \vee Roman(x_1)
D: Ruler(Cæsar)
E: \neg Roman(x_2) \vee LoyalTo(x_2 , Cæsar) \vee Hates(x_2 , Cæsar)
F: LoyalTo(x_3 , $S_1(x_3)$)
G: \neg Human(x_4) \vee \neg Ruler(y_1) \vee \neg TryAssassinate(x_4 , y_1) \vee \neg LoyalTo(x_4 , y_1)
H: TryAssassinate(Marcus, Cæsar)
- Negate Hates(Marcus, Cæsar) and convert to CNF:
I: \neg Hates(Marcus, Cæsar)
- Resolution proof:
– I and E with Marcus/ x_2 :
R1: \neg Roman(Marcus) \vee LoyalTo(Marcus, Cæsar)
– R1 and C with Marcus/ x_1 :
R2: LoyalTo(Marcus, Cæsar) \vee \neg Pompeian(Marcus)
– R2 and B:
R3: LoyalTo(Marcus, Cæsar)
– R3 and G with Marcus/ x_4 , Cæsar/ y_1 :
R4: \neg Human(Marcus) \vee \neg Ruler(Cæsar) \vee \neg TryAssassinate(Marcus, Cæsar)
– R4 with A:
R5: \neg Ruler(Cæsar) \vee \neg TryAssassinate(Marcus, Cæsar)
– R5 with D:
R6: \neg TryAssassinate(Marcus, Cæsar)
– R6 with H:
Empty (Contradiction)

7. Towers of Hanoi

- (a) Vocabulary: OnPeg(x , y) Disc x is on Peg y
OnDisc(x , y) Disc x is on Disc y
ClearPeg(x) There is nothing on peg x
ClearDisc(x) There is nothing on disc x
Smaller(x , y) Disc x is smaller than disc y
- Initial state: Smaller(A , B), ClearDisc(A), OnDisc(A , B), OnPeg(B , 1), ClearPeg(2), ClearPeg(3)
Goal state: OnDisc(A , B), OnPeg(B , 3)
- (b) Action(move-d-d-p(d , e , p), PRECONDITION: ClearDisc(d), OnDisc(d , e), ClearPeg(p)
EFFECT: \neg ClearPeg(p), OnPeg(d , p), \neg OnDisc(d , e), ClearDisc(e))
- Action(move-d-p-p(d , q , p), PRECONDITION: ClearDisc(d), OnPeg(d , q), ClearPeg(p)
EFFECT: \neg ClearPeg(p), OnPeg(d , p), \neg OnPeg(d , q), ClearPeg(q))
- Action(move-d-p-d(d , p , e), PRECONDITION: ClearDisc(d), OnPeg(d , p), ClearDisc(e), Smaller(d , e)
EFFECT: \neg ClearDisc(e), OnDisc(d , e), \neg OnPeg(d , p), ClearPeg(p))

