Lab-9

Date 19/12/2024
Page

Resolution

1. Convert all sentences to CNF

Steps to convert logical statement to CNF

(a) Eliminate biconditional & implications

i) eliminate $\Leftrightarrow$ representing $\alpha \Leftrightarrow \beta$

to $\alpha \Rightarrow \beta \wedge \beta \Rightarrow \alpha$

ii) Eliminate $\Rightarrow$ representing $\alpha \Rightarrow \beta$

with $\neg\alpha \vee \beta$

(b) move $\neg$ inwards

i) $\neg(\forall x\ p) \equiv \exists x \neg p$

ii) $\neg[\exists x\ p) \equiv \forall x \neg p$

iii) $\neg(\alpha \vee \beta) \equiv \neg\alpha \wedge \neg\beta$

iv) $\neg(\alpha \wedge \beta) \equiv \neg\alpha \vee \neg\beta$

v) $\neg\neg\alpha \equiv \alpha$

(c) Standardize variables apart by renaming them, each quantifiers should use a different variable

(d) Skolemize each existential variable is replaced by a skolem constant or skolem function of the enclosing universally quantified variable

ex. $\exists$ Rich(x) becomes Rich(G1), G1 is a new skolem constant

(e) Drop universal quantifier

$\forall x$ person(x) becomes person(x)

Date __/__/__
Page _____

(f) Distribute $\wedge$ over $\vee$

$(\alpha \wedge \beta) \vee \gamma \equiv (\alpha \vee \gamma) \wedge (\beta \vee \gamma)$

2. Negate conclusion S and convert result to CNF

3. Add negated conclusion S to the premise clause

4. Repeat until contradiction or no progress is made.
   i) Select 2 clauses
   ii) Resolve them together, performing all required unification
   iii) If resolvent is the empty clause, a contradiction has been found.
   iv) If not add resolved to the premise

Ex.

KB or premise

a)	John likes all kind of food	a) $\forall x$: food(x) $\rightarrow$ likes(John, x)
b)	Apple and vegetable are food	b) food(Apple) $\wedge$ food(vegetable)
c)	Anything anyone eats and not kills are food	c) $\forall x \forall y$: eats(x, y) $\wedge$ $\neg$ killed(x) $\rightarrow$ food
d)	Anil eats peanuts and still alive	d) eats(Anil, peanut) $\wedge$ alive (Anil)
e)	Harry eats everything that Anil eats	f) $\forall x$ eats(Anil, x) $\rightarrow$ alive(x)
f)	Anyone who is alive implies not killed	e) $\forall x$ eats(Anil, x) $\rightarrow$ eats(Harry, x)
g)	Anyone who is not killed implies alive	g) $\forall x$ alive(x) $\rightarrow$ $\neg$killed (x)

Date __/__/__
Page _____

prove by resolution that h) John likes peanut

(h) likes(John, peanut)

Eliminate implication

a) $\forall x \neg food(x) \vee likes(John, x)$

b) food(Apple) $\wedge$ food(vegetable)

c) $\forall x \forall y \neg(eats(x, y) \wedge \neg killed(x)) \vee food(y)$

d) eats(Anil, peanut) $\wedge$ alive(Anil)

e) $\forall x \neg eats(Anil, x) \vee eats(Harry, x)$

f) $\forall x \neg(\neg killed(x)) \vee alive(x)$

g) $\forall x \neg alive(x) \vee \neg killed(x)$

h) likes(John, peanut)

move $\neg$ inwards

$\forall x \forall y \neg eats(x, y) \vee killed(x) \vee food(y)$

Standardize variables

c) $\forall y \forall z \neg eats(y, z) \vee killed(y) \vee food(z)$

e) $\forall w \neg eats(Anil, w) \vee eats(Harry, w)$

f) $\forall g \; killed(g) \vee alive(g)$

g) $\forall k \; \neg alive(k) \vee \neg killed(k)$

drop universal quantifiers

a) $\neg food(x) \vee likes(John, x)$

b) food(Apple)

c) food(vegetable)

d) $\neg eats(y, z) \vee killed(y) \vee food(z)$

e) eats(Anil, Peanuts)

f) alive(Anil)

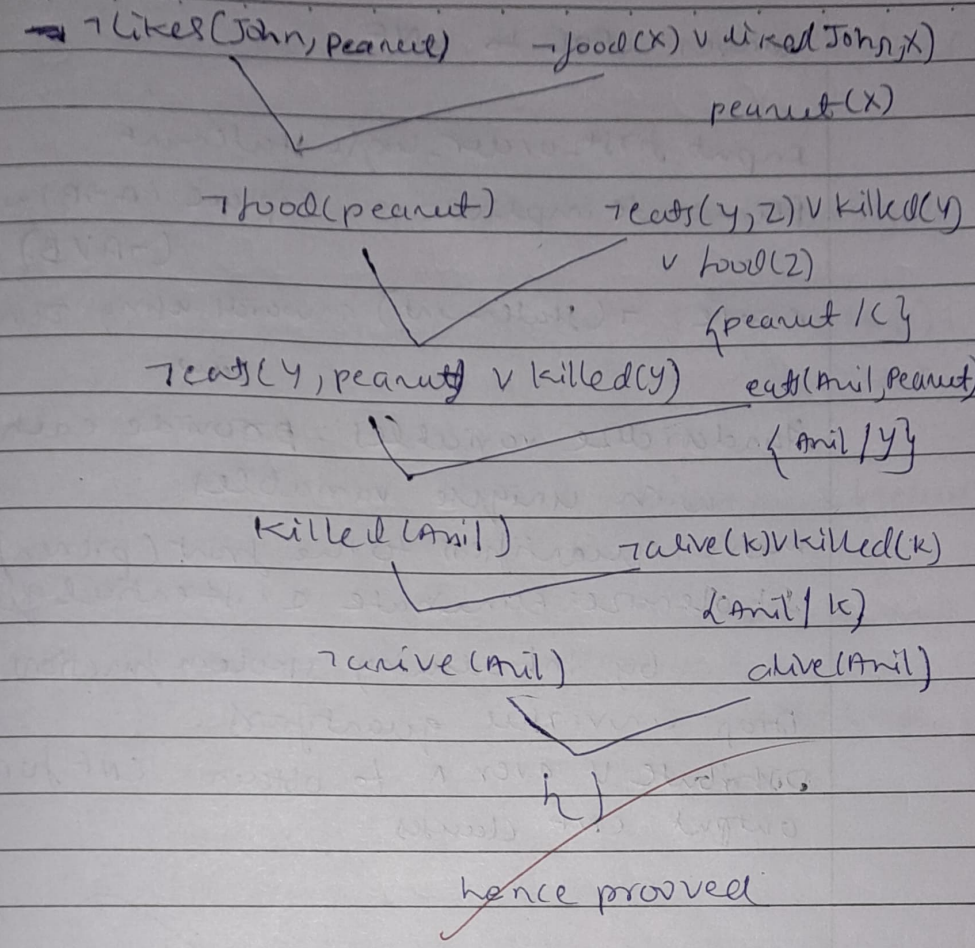
g) $\neg eats(Anil, w) \vee eats(Harry, w)$

h) killed(g) $\vee$ alive(g)

i) $\neg alive(k) \vee \neg killed(k)$

j) likes(John, peanut)

Date __/__/__
Page _____


¬likes(John, Peanut)
¬food(x) ∨ likes(John, x)
peanut(x)
¬food(peanut)
¬eats(y, z) ∨ killed(y) ∨ food(z)
{peanut / z}
¬eats(y, peanut) ∨ killed(y)
eats(Anil, Peanut)
{Anil / y}
killed(Anil)
¬alive(k) ∨ killed(k)
{Anil / k}
¬alive(Anil)
alive(Anil)
hence proved

mnaz7lpj4

December 21, 2024

[4]:
```
print("Name:Sudarshan Komar", "USN:1BM22CS291", sep="\n")

from sympy import symbols, Or, And, Not
from sympy.logic.inference import satisfiable

def resolution_proof():
    # Define predicates as symbols
    Eats_Anil_Peanuts = symbols('Eats_Anil_Peanuts')
    NotKilled_Anil = symbols('NotKilled_Anil')
    Food_Peanuts = symbols('Food_Peanuts')
    Likes_John_Peanuts = symbols('Likes_John_Peanuts')

    # Step 1: Encode the statements as clauses
    # Fact 1: Anything anyone eats and is not killed is food
    clause1 = Or(Not(Eats_Anil_Peanuts), Not(NotKilled_Anil), Food_Peanuts)

    # Fact 2: Anil eats peanuts and is still alive
    clause2 = And(Eats_Anil_Peanuts, NotKilled_Anil)

    # Fact 3: John likes all kinds of food
    clause3 = Or(Not(Food_Peanuts), Likes_John_Peanuts)

    # Negate the goal: John does NOT like peanuts
    negated_goal = Not(Likes_John_Peanuts)

    # Step 2: Perform resolution
    # Resolve clause1 and clause2 to prove Food_Peanuts
    resolved_clause1 = clause1.subs({Eats_Anil_Peanuts: True, NotKilled_Anil:␣
 ↪True})
    print(f"Resolved Clause 1: {resolved_clause1}")

    # Resolve resolved_clause1 to prove Food_Peanuts
    if resolved_clause1 == Food_Peanuts:
        print("Food(Peanuts) is proven.")

    # Resolve Food_Peanuts with clause3 to prove Likes_John_Peanuts
    resolved_clause2 = clause3.subs({Food_Peanuts: True})
```

```
    print(f"Resolved Clause 2: {resolved_clause2}")

    # Final check
    result = satisfiable(And(resolved_clause2, negated_goal))
    if result is False:
        print("The goal 'John likes peanuts' is proven using resolution.")
    else:
        print("The goal 'John likes peanuts' cannot be proven.")

# Run the proof
resolution_proof()
```

```
Name:Sudarshan Komar
USN:1BM22CS291
Resolved Clause 1: Food_Peanuts
Food(Peanuts) is proven.
Resolved Clause 2: Likes_John_Peanuts
The goal 'John likes peanuts' is proven using resolution.
```