Resolution

Dr. Krishnendu Guha

Assistant Professor (Contractual)

National Institute of Technology (NIT), Jamshedpur

Email: krishnendu.ca@nitjsr.ac.in

Introduction

- Resolution methodology was invented in the year 1965 by a Mathematician called **Alan Robinson**

- Resolution method is an **inference rule** which is used in both Propositional as well as First-order Predicate Logic
- This method is basically used for **proving the satisfiability of a sentence**
- In resolution method, we use **Proof by Refutation** technique to prove the given statement
- The key idea for the resolution method is to use the knowledge base and negated goal to obtain null clause (which indicates contradiction).
- Resolution method is also called **Proof by Refutation**. Since the knowledge base itself is consistent, the contradiction must be introduced by a negated goal. As a result, we have to conclude that the original goal is true.

Alan Robinson

Unification Algorithm

- In propositional logic, it is easy to determine that two literals cannot both be true at the same time.
- Simply look for L and ¬L in predicate logic, this matching process is more complicated since the arguments of the predicates must be considered.
- For example, man(John) and ¬man(John) is a contradiction, while the man(John) and ¬man(Spot) is not.
- Thus, in order to determine contradictions, we need a matching procedure that compares two literals and discovers whether there exists a set of substitutions that makes them identical.
- There is a straightforward recursive procedure, called the unification algorithm, that does it

Algorithm: Unify(L1,L2)

```
1. If L1 or L2 are both variables or constants, then:
    1. If L1 and L2 are identical, then return NIL.
    2. Else if L1 is a variable, then if L1 occurs in L2 then return {FAIL}, else return (L2/L1).
    3. Also, Else if L2 is a variable, then if L2 occurs in L1 then return {FAIL}, else return (L1/L2). d. Else return {FAIL}.
2. If the initial predicate symbols in L1 and L2 are not identical, then return {FAIL}.
3. If LI and L2 have a different number of arguments, then return {FAIL}.
4. Set SUBST to NIL. (At the end of this procedure, SUBST will contain all the substitutions used to unify L1 and L2.)
5. For I ← 1 to the number of arguments in L1 :
    1. Call Unify with the ith argument of L1 and the ith argument of L2, putting the result in S.
    2. If S contains FAIL then return {FAIL}.
    3. If S is not equal to NIL then:
        1. Apply S to the remainder of both L1 and L2.
        2. SUBST: = APPEND(S, SUBST).
6. Return SUBST
```

Conjunctive Normal Form (CNF)

- In propositional logic, the resolution method is applied only to those clauses which are disjunction of literals.
- **There are following steps used to convert into CNF:**
- 1) Eliminate bi-conditional implication by replacing **A ⇔ B with (A → B) ∧ (B →A)**
- 2) Eliminate implication by replacing **A → B with ¬A V B.**
- 3) In CNF, negation(¬) appears only in literals, therefore we move it inwards as:
  - **¬ (¬A) ≡ A** (double-negation elimination
  - **¬ (A ∧ B) ≡ (¬A V ¬B)** (De Morgan)
  - **¬(A V B) ≡ (¬A ∧ ¬B)** (De Morgan)
- 4) Finally, using distributive law on the sentences, we can form the CNF as:
  - **$(A_1 \lor B_1) \land (A_2 \lor B_2) \land \ldots. \land (A_n \lor B_n)$.**
- **Note: CNF** can also be described as **AND of ORs**

Disjunctive Normal Form (DNF)

- This is a reverse approach of CNF
- The process is similar to CNF with the following difference:
  - **$(A_1 \wedge B_1) \vee (A_2 \wedge B_2) \vee \ldots \vee (A_n \wedge B_n)$**

- **In DNF, it is OR of ANDs, a sum of products, or a cluster concept, whereas, in CNF, it is ANDs of ORs**

Resolution in Propositional Logic

- In propositional logic, resolution method is the only **inference rule which gives a new clause when two or more clauses are coupled togetther**
- Using propositional resolution, it becomes easy to make a theorem prover sound and complete for all

- **The process followed to convert the propositional logic into resolution method contains the below steps:**

- Convert the given axiom into clausal form, i.e., disjunction form.
- Apply and proof the given goal using negation rule.
- Use those literals which are needed to prove.
- Solve the clauses together and achieve the goal

Example of Propositional Resolution

- Consider the following Knowledge Base:

1. The humidity is high or the sky is cloudy.
2. If the sky is cloudy, then it will rain.
3. If the humidity is high, then it is hot.
4. It is not hot.

- **Goal:** It will rain.
- Use propositional logic and apply resolution method to prove that the goal is derivable from the given knowledge base

- **Solution:** Let's construct propositions of the given sentences one by one:

1. Let, P: Humidity is high.

   Q: Sky is cloudy.

- It will be represented as **P V Q.**

2) Q: Sky is cloudy. ...**from(1)**

   Let, R: It will rain.

- It will be represented as b**Q** → **R.**

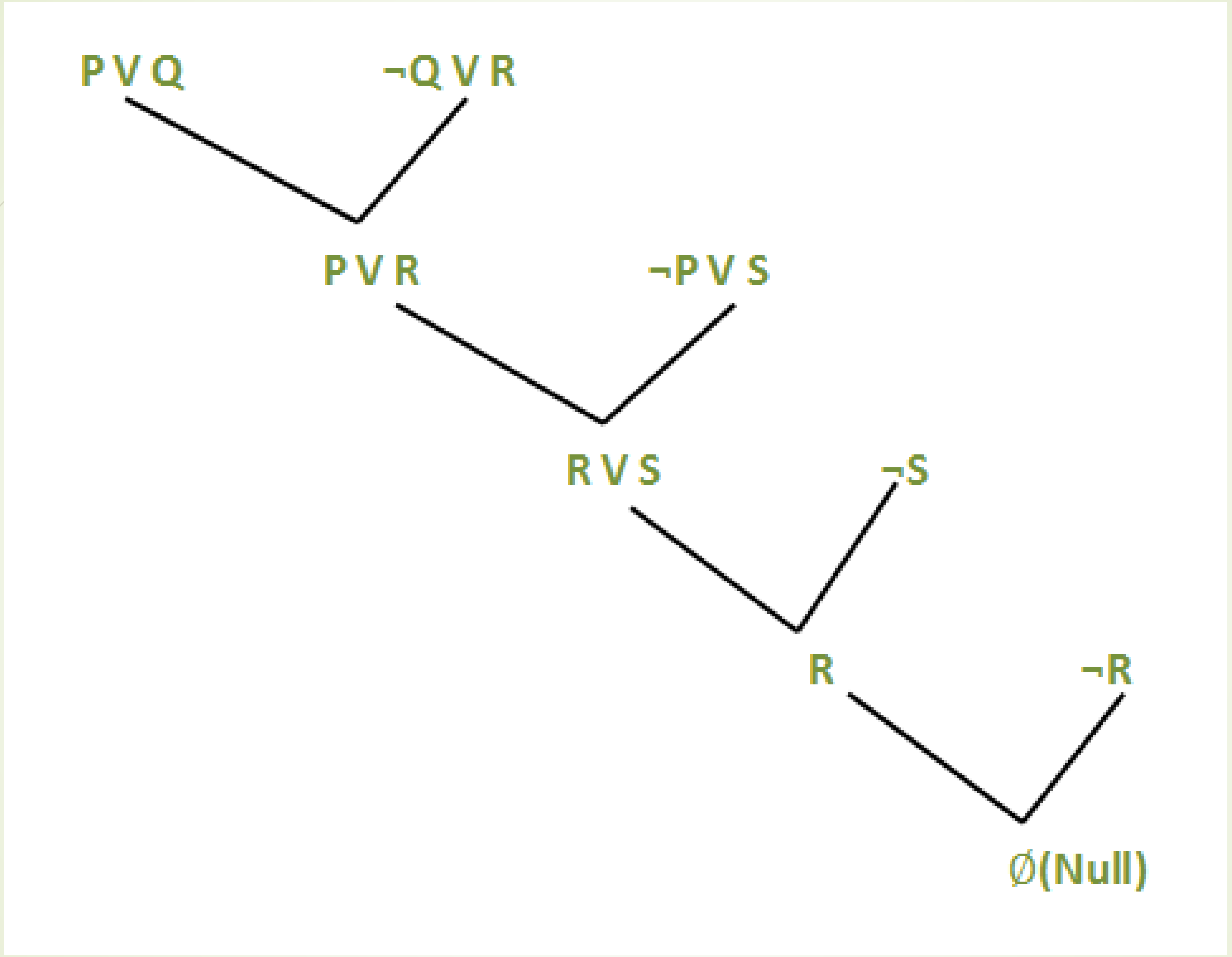
3) P: Humidity is high. ...**from(1)**

   Let, S: It is hot.

- It will be represented as **P** → **S.**

4) ¬S: It is not hot.

- **Applying resolution method:**
- In (2), Q → R will be converted as (¬Q V R)
- In (3), P → S will be converted as (¬P V S)
- **Negation of Goal (¬R):** It will not rain



After applying Proof by Refutation (Contradiction) on the goal, the problem is solved, and it has terminated with a **Null clause (∅).** Hence, the goal is achieved. Thus, It is not raining

Resolution Method in FOPL/ Predicate Logic

- **In FOPL, the process to apply the resolution method is as follows:**
- Convert the given axiom into CNF, i.e., a conjunction of clauses. Each clause should be dis-junction of literals.
- Apply negation on the goal given.
- Use literals which are required and prove it.
- Unlike propositional logic, FOPL literals are complementary if one unifies with the negation of other literal.

Conjunctive Normal Form (cont.)

- **There are following steps used to convert into CNF:**
- Eliminate the implications as:
- **∀x: A(x) → B(x) with {¬ x: ¬A(∀x) V B(x)}**
- Move negation (¬) inwards as:
- **¬∀x: A becomes ∃x: ¬A and,**
- **¬∃x: A becomes ∀x: ¬A**
- It means that the universal quantifier becomes existential quantifier and vice-versa**.**
- **Standardize variables:** If two sentences use same variable, it is required to change the name of one variable. This step is taken so as to remove the confusion when the quantifiers will be dropped.
- **For example**: { ∀x: A(x) V ∀x: B(x)}
- **Skolemize:** It is the process of removing existential quantifier through elimination.
- **Drop universal quantifiers:** If we are on this step, it means all remaining variables must be universally quantified. Drop the quantifier.
- **Distribute V over ∧:** Here, the nested conjunction and disjunction are flattened.

Example of FOPL Resolution

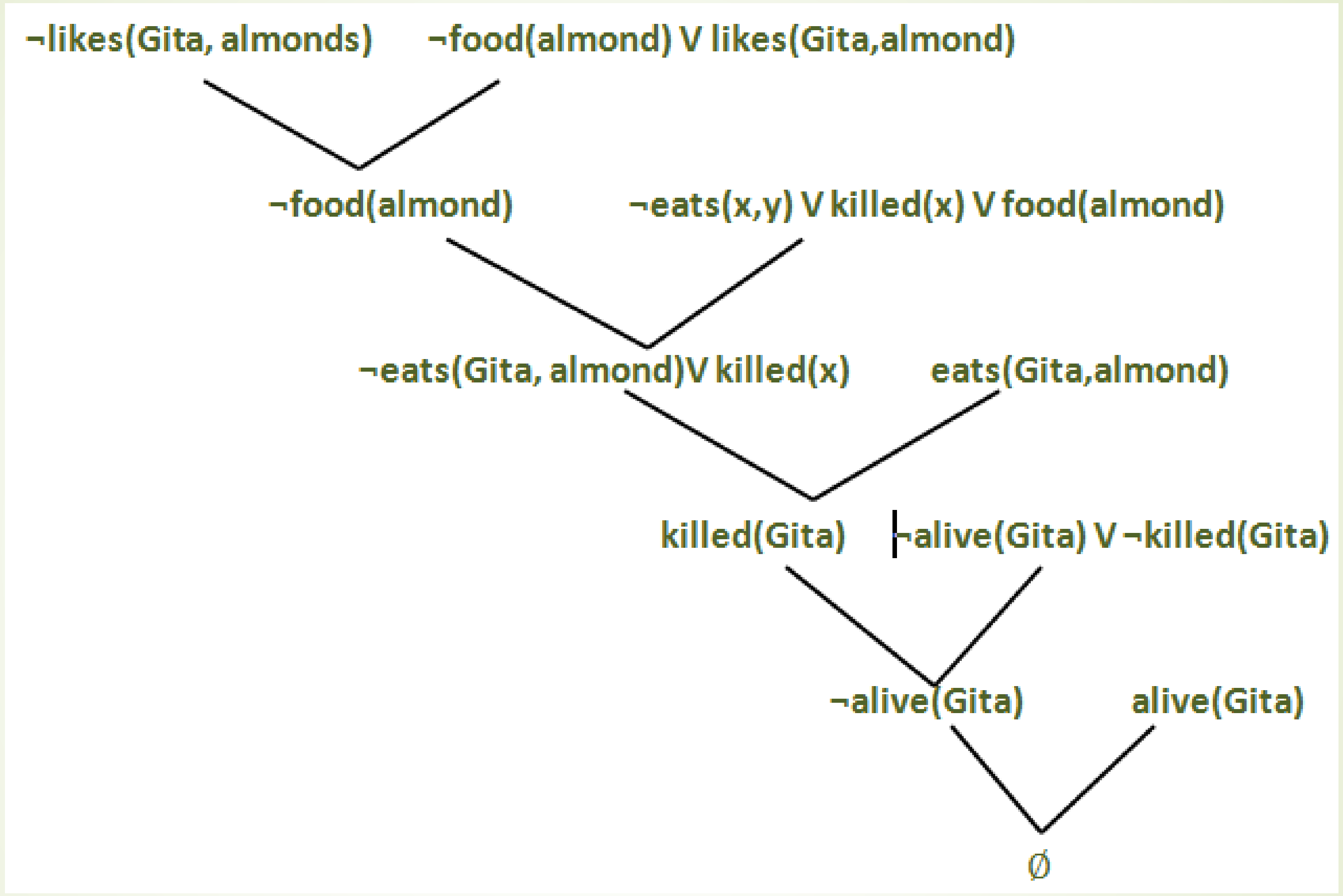
- Consider the following knowledge base:
- Gita likes all kinds of food.
- Mango and chapati are food.
- Gita eats almond and is still alive.
- Anything eaten by anyone and is still alive is food.
- **Goal:** Gita likes almond.

- **Solution:** Convert the given sentences into FOPL as:
- **Let, x be the light sleeper.**

1. ∀x: food(x) → likes(Gita,x)
2. food(Mango),food(chapati)
3. ∀x∀y: eats(x,y) Λ ¬ killed(x → food(y)
4. eats(Gita, almonds) Λ alive(Gita)
5. ∀x: ¬killed(x) → alive(x)
6. ∀x: alive(x) → ¬killed(x)

- **Goal:** likes(Gita, almond)
- **Negated goal**: ¬likes(Gita, almond)
- **Now, rewrite in CNF form:**

1. ¬food(x) V likes(Gita, x)
2. food(Mango),food(chapati)
3. ¬eats(x,y) V killed(x) V food(y)
4. eats(Gita, almonds), alive(Gita)
5. killed(x) V alive(x)
6. ¬alive(x) V ¬killed(x)


¬likes(Gita, almonds)
¬food(almond) V likes(Gita,almond)
¬food(almond)
¬eats(x,y) V killed(x) V food(almond)
¬eats(Gita, almond)V killed(x)
eats(Gita,almond)
killed(Gita)
¬alive(Gita) V ¬killed(Gita)
¬alive(Gita)
alive(Gita)
Ø

Hence, we have achieved the given goal with the help of Proof by Contradiction. Thus, it is proved that Gita likes almond

Forward Chaining in AI

- Forward Chaining is the process which works on the basis of available data to make certain decisions.
- In forward chaining, we start with the available data and use inference rules to extract data until the goal is not reached.
  - OR From the available data, expand until a decision is not made

- **In artificial intelligence, we have two different methods to use forward chaining.**
  - Forward Chaining in Propositional Logic
  - Forward Chaining in Predicate Logic/(FOPL)

Forward Chaining in Propositional Logic

- In propositional logic, forward chaining starts its journey from the given knowledge base
- If all the premises of the implication are known, then its conclusion will be added to the set of known facts

- Example

1. If D barks and D eats bone, then D is a dog.
2. If V is cold and V is sweet, then V is ice-cream.
3. If D is a dog, then D is black.
4. If V is ice-cream, then it is Vanilla.

- **Derive forward chaining using the given known facts to prove Tomy is black.**
  - Tomy barks.
  - Tomy eats bone.
- **Solution:** Given Tomy barks.

- From (1), it is clear:
- If Tomy barks and Tomy eats bone, then Tomy is a dog.
- From (3), it is clear:
- If Tomy is a dog, then Tomy is black.
- Hence, it is proved that **Tomy is black**.

- **Note:** There is an advantage of forward chaining that is, we can draw new inference from the given facts.

Forward Chaining in Predicate Logic

- Forward Chaining in Predicate Logic is different from forward chaining in Propositional Logic. In FOPL, forward chaining is efficiently implemented on first-order clauses. First-order clauses are the disjunction of literals of which exactly one is positive.
- **For example,** Crow(x) ∧ Thrust(x)→Water(x).
- Crow(x).
- Thrust(x).
- Therefore, a definite clause is either atomic or an implication whose predecessor is a conjunction of positive literals. As a result, it results in a single positive literal.

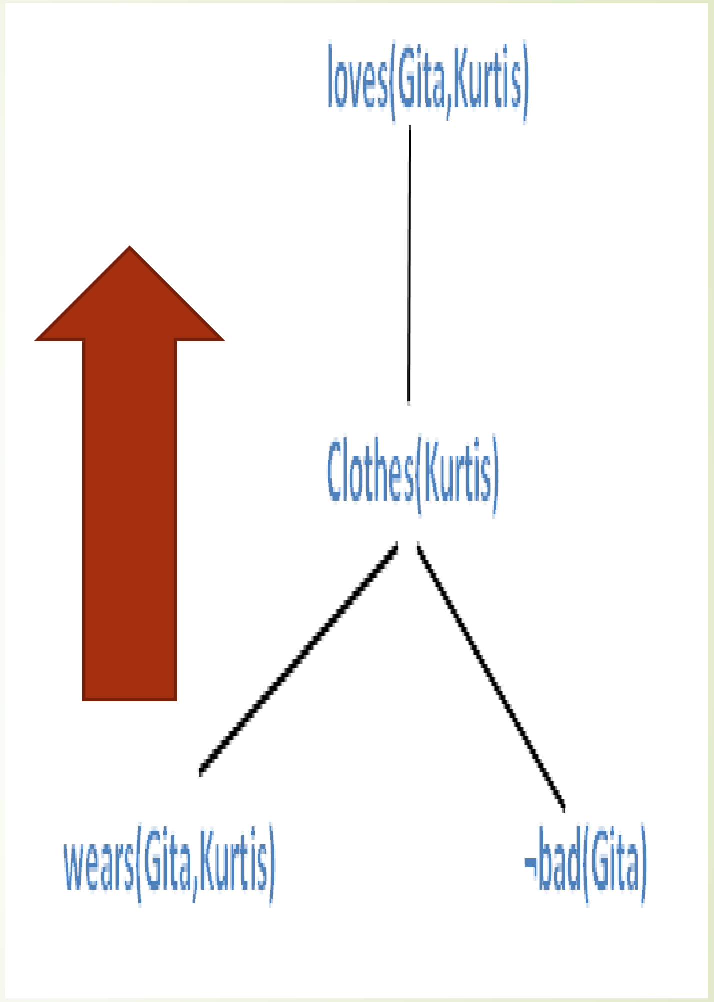
Example

1. Gita loves all types of clothes.
2. Suits are clothes.
3. Jackets are clothes.
4. Anything any wear and isn't bad is clothes.
5. Sita wears skirt and is good.
6. Renu wears anything Sita wears.

- **Solution:** Convert the given axioms into FOPL as:

1. x: clothes(x)→loves(Gita, x).
2. Suits(x)→Clothes(x).
3. Jackets(x)→Clothes(x).
4. wears(y,x)∧ ¬bad(y)→Clothes(x)
5. wears(Sita,skirt) ∧ ¬good(Sita)
6. wears(Sita,x)→wears(Renu,x)

- **To prove:** Gita loves Kurtis.
- **FOPL:** loves(Gita, Kurtis).



Thus, it is proved that Gita loves Kurtis

Backward Chaining in AI

- Backward Chaining is a backward approach which works in the backward direction. It begins its journey from the back of the goal.
- Like, forward chaining, we have backward chaining for **Propositional logic** as well as **Predicate logic** followed by their respective algorithms

- History
- The first algorithm given is known as the **Davis-Putnam algorithm**. It was proposed by **Martin Davis and Hilary Putnam in 1960.**
- In **1962, Davis, Logemann and Loveland** presented a new version of the Davis-Putnam algorithm. It was named under the initials of all four authors as **DPLL.**

Backward Chaining in Propositional Logic

- In propositional logic, backward chaining begins from the goal and using the given propositions, it proves the asked goal.
- **Example:**

1. If D barks and D eats bone, then D is a dog.
2. If V is cold and V is sweet, then V is ice-cream.
3. If D is a dog, then D is black.
4. If V is ice-cream, then it is Vanilla.

- **Derive backward chaining using the given known facts to prove Tomy is black.**

- Tomy barks.
- Tomy eats bone.

- **Solution:**

1. **On replacing D with Tomy in (3), it becomes:**

- If Tomy is a dog, then Tomy is black.
- Thus, the goal is matched with the above axiom.
- **Now, we have to prove Tomy is a dog. ...(new goal)**
- Replace D with Tomy in (1), it will become:
- If Tomy barks and Tomy eats bone, then Tomy is a dog. **...(new goal)**
- Again, the goal is achieved.
- **Now, we have to prove that Tomy barks and Tomy eats bone. ...(new goal)**
- As we can see, the goal is a combination of two sentences which can be further divided as:
- Tomy barks.
- Tomy eats bone.
- From (1), it is clear that Tomy is a dog.
- Hence, Tomy is black.
- **Note:** Statement (2) and (4) are not used in proving the given axiom. So, it is clear that goal never matches the negated versions of the axioms.

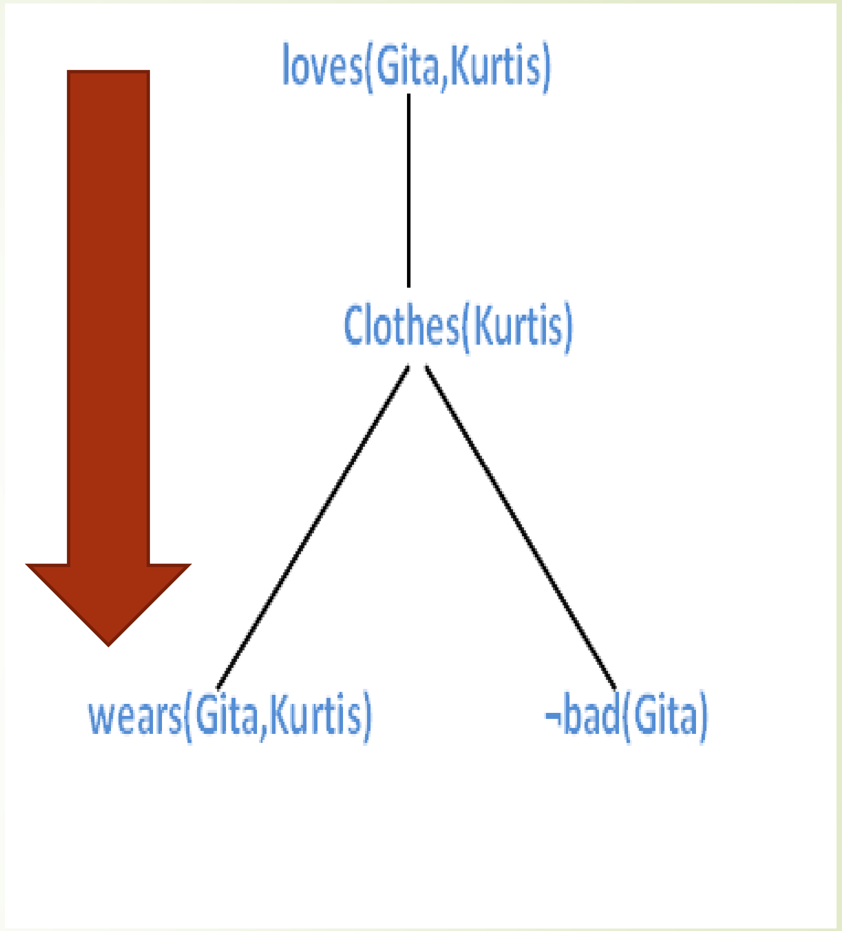
Backward Chaining in Predicate Logic

- In FOPL, backward chaining works from the backward direction of the goal, apply the rules on the known facts which could support the proof.
- Backward Chaining is a type of **AND/OR** search because we can prove the goal by applying any rule in the knowledge base
- **Example :**
- 1) Gita loves all types of clothes.
- 2) Suits are clothes.
- 3) Jackets are clothes.
- 4)Anything any wear and isn't bad is clothes.
- 5) Sita wears skirt and is good.
- 6) Renu wears anything Sita wears.
- **Apply backward chaining and prove that Gita loves Kurtis.**

- **Solution:** Convert the given axioms into FOPL as:

1. x: clothes(x)→loves(Gita, x).
2. Suits(x)→Clothes(x).
3. Jackets(x)→Clothes(x).
4. wears(y,x)→∧ ¬bad(y)→Clothes(x)
5. wears(Sita,skirt)∧ ¬good(Sita)
6. wears(Sita,x)→wears(Renu,x)

- **To prove:** Gita loves Kurtis.
- **FOPL:** loves(Gita, Kurtis).



It is clear from the above graph Gita wears Kurtis and does not look bad. Hence, **Gita loves Kurtis.**
Note: We have seen that the graph of forward and backward chaining is same.
It means that forward chaining follows the **bottom-up approach** and backward chaining follows the **top-down approach**

Thank
You
Obrigado
Merci
Multumesc
Spasibo
Asante
Grazie
Arigato
Matondo
Dank Je
Kiitos
Welalin
Juspaxar
Vinaka
Maake
Kia Ora
Mamana