ME 620: Fundamentals of Artificial Intelligence

Lecture 20: Answer Extraction

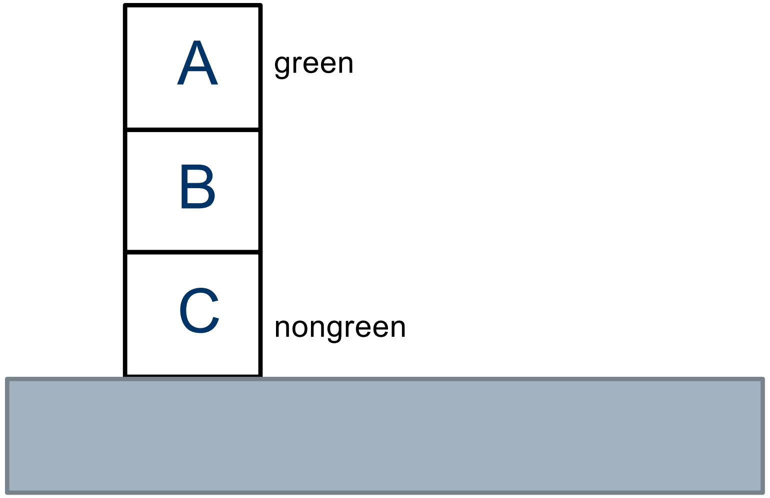
Shyamanta M Hazarika
Biomimetic Robotics and Artificial Intelligence Lab
Mechanical Engineering and M F School of Data Sc. & AI
IIT Guwahati

Answering a Query in FOL

Example

Suppose there are three coloured blocks stacked as shown, where the top one is `green' and the bottom is `not green'. Is there a `green' block on top of a `non-green' block?

on: holds if and only if block is immediately above the other.



Facts

1. on(A,B)
2. on(B,C)
3. green(A)
4. ¬ green(C)

Query

$$\exists x\, \exists y\ \mathbf{on}(x, y) \wedge \mathbf{green}(x) \wedge \neg \mathbf{green}(y)$$

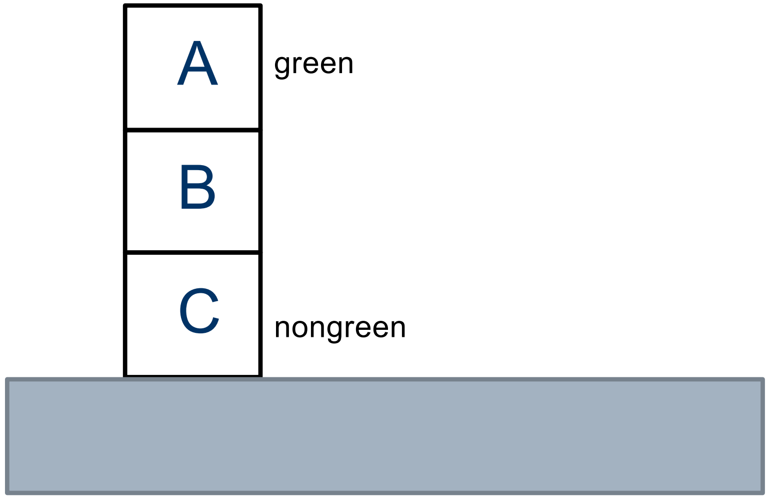
Answering a Query in FOL

Example

Negation of the Query

$$\neg\exists x\ \exists y\ (on(x,y) \land green(x) \land \neg green(y))$$

$$\neg\, on(x,y) \lor \neg\, green(x) \lor green(y)$$



Refutation Trace

1.	on(A,B)	C1
2.	on(B,C)	C2
3.	green(A)	C3
4.	¬ green(C)	C4
5.	¬ on(x,y) ∨ ¬ green(x) ∨ green(y)	C5
6.	¬ green(A) ∨ green(B)	1,5
7.	¬ green(B) ∨ green(C)	2,5
8.	green(B)	3,6
9.	¬ green(B)	4,7
10.	□.	8,9

In this problem the **KB entails that there is some block** which must be **green and on top of a nongreen** block.

However, it **does not make any commitment to any specific one**.

Answer-extraction deals with **providing instances for such variables**.

Extracting Answers from Refutation Proofs

- Many **applications for FOPC theorem-proving** systems involve
  - **proving formulas** that contains **existentially quantified variables.**
  - Finding values or **instances** for these variables.
  - Does $\exists x\, W(x)$ **logically follows from** Δ ?
    - If it does, we want an instance of the \`x'.
- **Producing the satisfying instance for \`x'** requires that the **proof method be \`constructive'.**
  - A constructive proof is a **proof that directly provides a specific example**, or which gives **an algorithm for producing an example**. Constructive proofs are also called demonstrative proofs.

Extracting Answers from Refutation Proofs

- **Prospect of producing satisfying instances** for existentially quantified variables **allows the possibility for posing quite general questions**.
- For Example

One needs to remember, though, that complex questions will require complex proofs; possibly so complex that our automatic procedures would not find it! Making these ideas not feasible in practice.

  - Does there exist a solution sequence to a certain 8-puzzle?
    - If a **constructive proof** could be found that a solution does exist, this **could mean that we can produce the desired solution as well**!
  - Whether there exist programs that perform desired computation?
    - From a constructive proof of a program's existence, **one could produce the desired program!**

A Simple Example

Mary had a little lamb
It's fleece was white as snow;
And everywhere that Mary went,
The lamb was sure to go.

The Lamb goes wherever Mary goes.
Mary is at school.

Where is the Lamb?

The problem specifies two FACTS and than asks a question; whose ANSWER can presumably be deduced from these facts.

Facts might be translated into the clause set; and prove the existence of a place where the Lamb is!

A Simple Example

The **Lamb goes wherever Mary goes**.

Mary is at school.

Where is the Lamb?

To answer this question, we FIRST prove that there is some place where the lamb is, i.e.,

$\exists x \; at(Lamb, x)$

A Simple Example

The **Lamb goes wherever Mary goes**. $\forall x\ [at(Mary,x) \rightarrow at(Lamb,x)]$.

Mary is at school. $at(Mary, School)$.

Where is the Lamb?

To answer this question, we FIRST prove that there is some place where the lamb is, i.e.,

$\exists x\ at(Lamb, x)$

A Simple Example – Key Idea

- Convert the question to a **goal well-formed formula containing an existential quantifier**.
  - The **existentially quantified variable represents an answer to the question**.
- If the question can be answered from Δ, the facts given, the **goal well-formed formula** created in this manner will **logically follow from** Δ.
- After obtaining the proof, we try to **extract an instance of the existentially quantified variable** to serve as an answer.

A Simple Example

The **Lamb goes wherever Mary goes**. $\forall x\ [at(Mary,x) \rightarrow at(Lamb,x)]$.

Mary is at school. $at(Mary, School)$.

Where is the Lamb?

To answer this question, we FIRST prove that there is some place where the lamb is, i.e.,

$\exists x\ at(Lamb, x)$

Negation of the goal statement 3. $\forall x \neg at(Lamb, x)$

A Simple Example

The **Lamb goes wherever Mary goes**. $\forall x\ [at(Mary,x) \rightarrow at(Lamb,x)]$.

Mary is at school. $at(Mary, School)$.

Where is the Lamb?

To answer this question, we FIRST prove that there is some place where the lamb is, i.e.,

$\exists x\ at(Lamb, x)$

Negation of the goal statement

1. $\forall x\ [\neg\ at(Mary,x) \vee at(Lamb,x)]$.
2. $at(Mary, School)$
3. $\forall x\ \neg\ at(Lamb, x)$

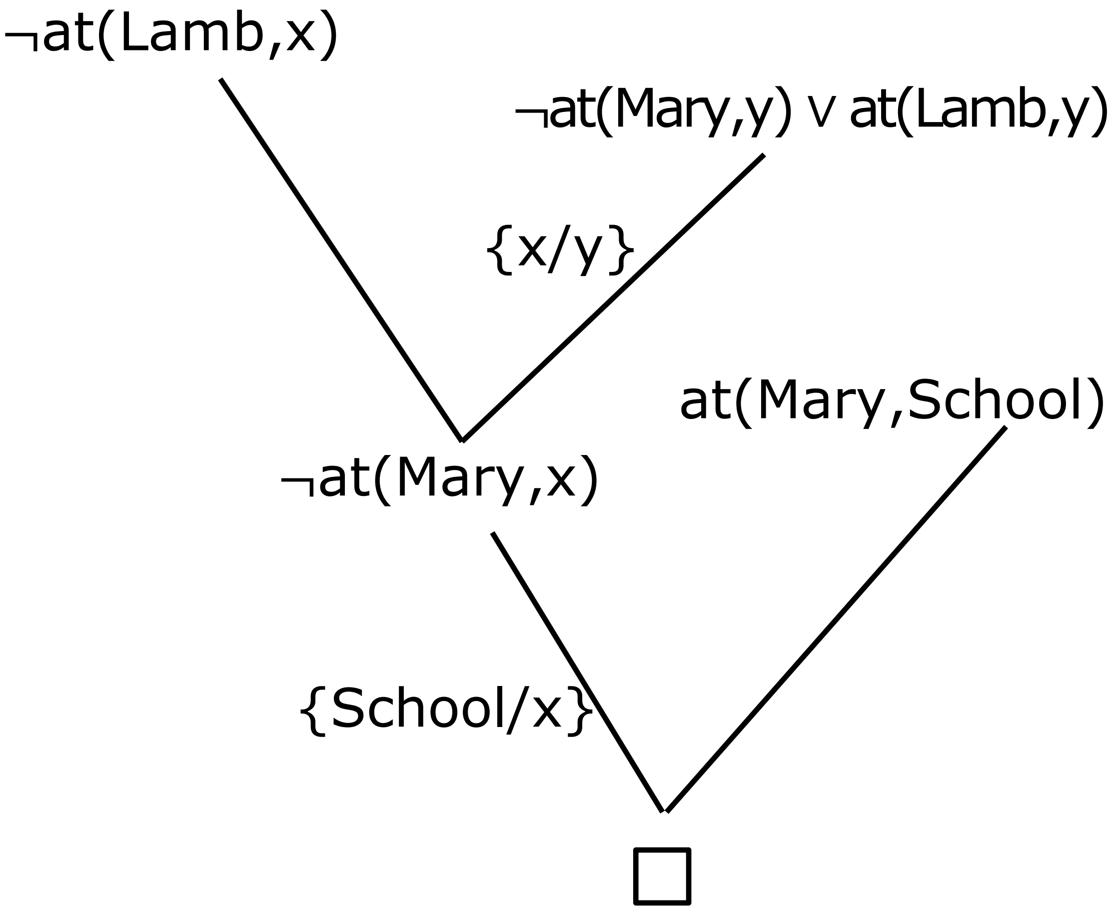
A Simple Example – Key Idea

Resolution Trace

1. [$\neg$ at(Mary,y) $\vee$ at(Lamb,y)]	C1
2. at(Mary, School)	C2
3. $\neg$ at(Lamb, x)	C3
4. $\neg$ at(Mary,x)	1,3
5. $\square$	

Resolution refutation is obtained in the usual manner.

A Simple Example – Key Idea



Refutation Tree

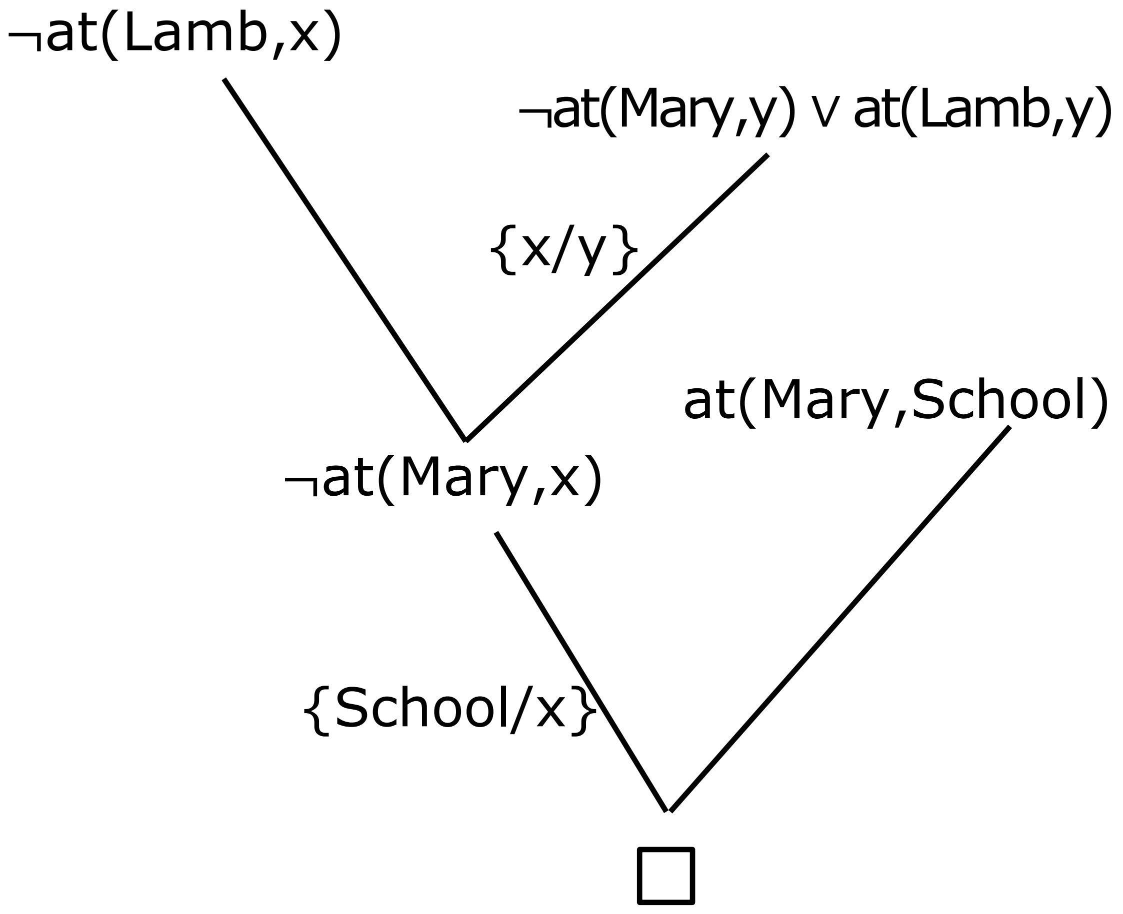
Resolution Trace

1.	[¬ at(Mary,y) ∨ at(Lamb,y)]	C1
2.	at(Mary, School)	C2
3.	¬ at(Lamb, x)	C3
4.	¬ at(Mary,x)	1,3
5.	□	

Resolution refutation is obtained in the usual manner.

Figure on the left is a Refutation Tree for the example.

A Simple Example – Key Idea



Refutation Tree

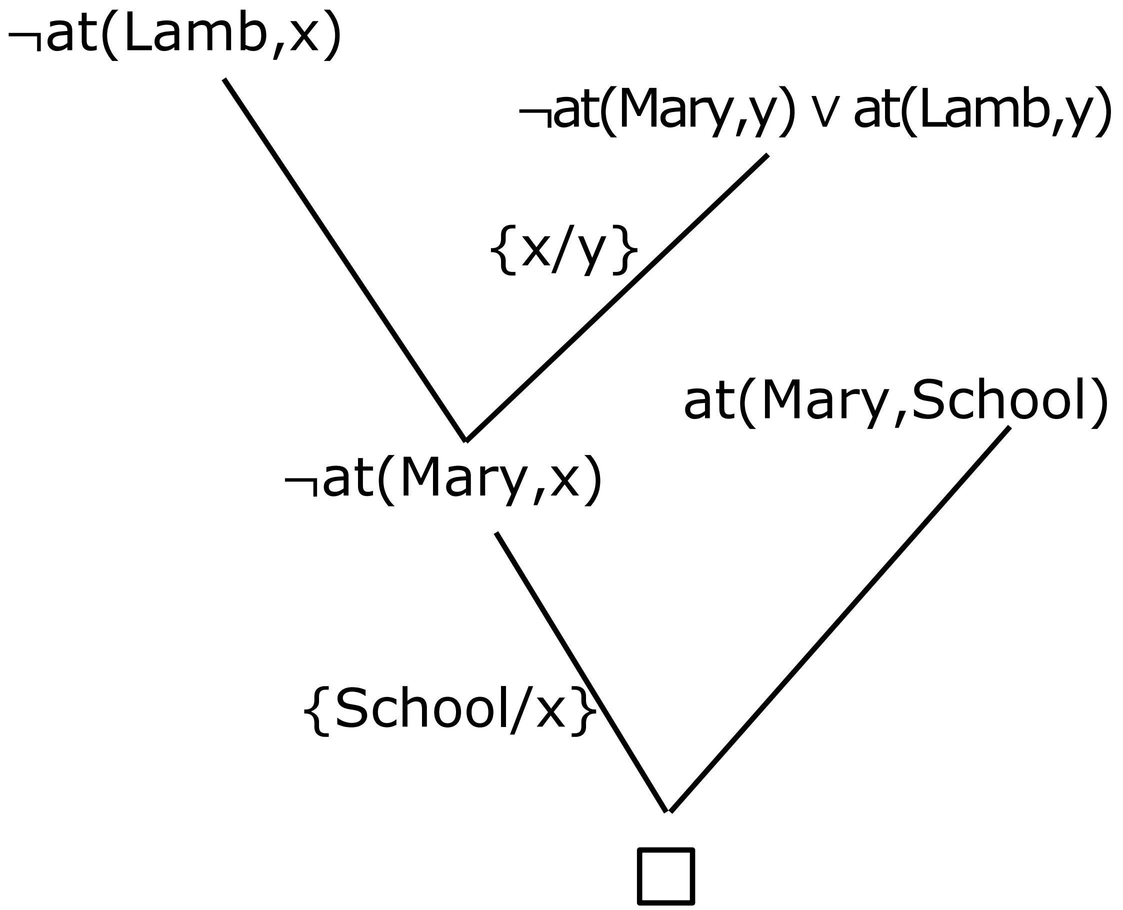
Process for extracting the answer for the question

Where is the Lamb?

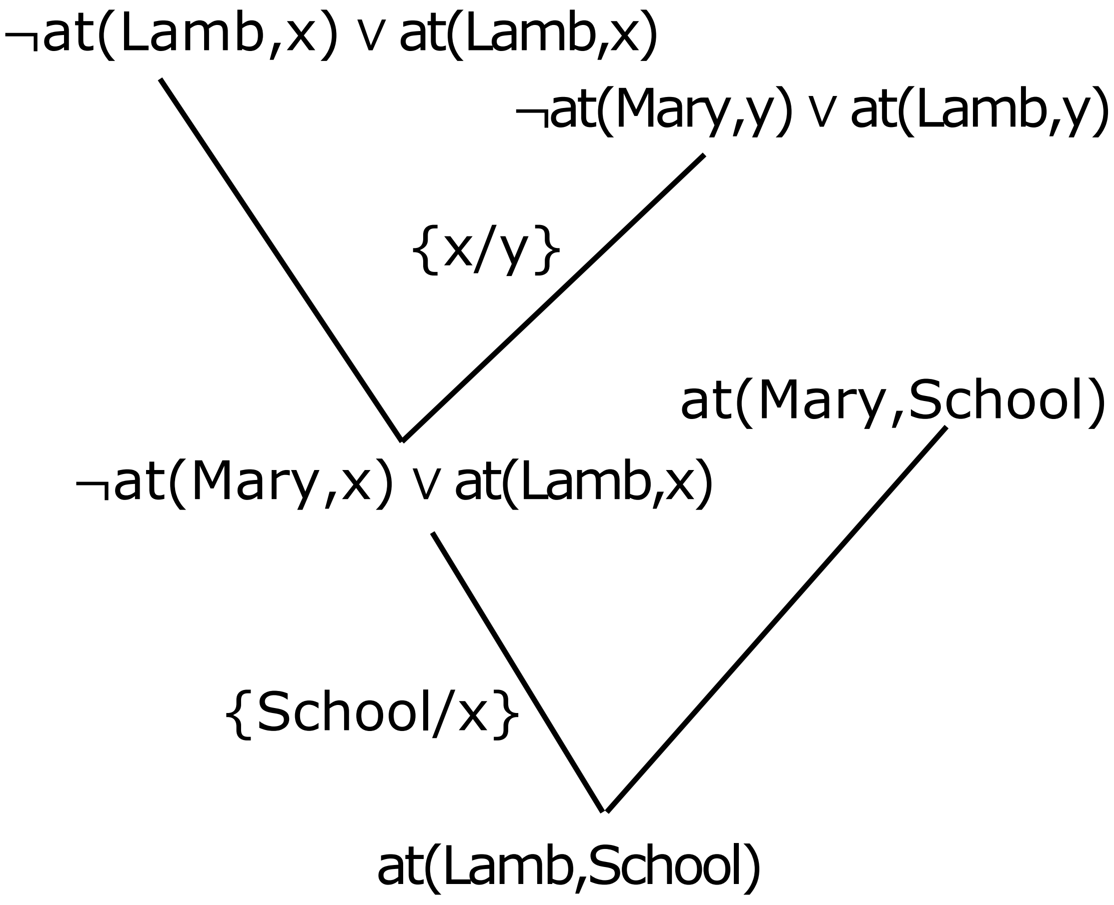
is as follow:

1. **Append** each clause arising from the **negation of the goal to its own negation**.
2. Follow the structure of the refutation tree; performing same resolutions.
3. Use **clause at the root**.

A Simple Example – Key Idea



Refutation Tree



Modified Proof Tree

The answer statement has a form similar to that of the goal; only difference is a constant (the answer) in place of the existentially quantified variable in the goal.

The Answer Extraction Process

- Answer extraction **involves converting a refutation tree to a proof tree** with statement at the root that can be used as an answer.
  - **Convert** every clause arising from the negation of the goal well-formed formula **into a tautology**.
  - The **statement at the root of the Modified Proof Tree logically** follows **from the axioms and the tautologies.**
    - Hence it follows from the axioms alone!
    - **Modifies Proof tree justifies answer extraction**.

Fill-in-the-Blank

- Answer extraction : answering a fill-in-the-blank question.
- A fill-in-the-blank question is a **predicate calculus sentence with free variables** specifying the blanks to be filled in.
  - Goal is to find bindings for the free variables such that the database logically implies the sentence obtained by substituting the bindings into the original question.
- **Answer literal** for a fill-in-the-blank question ϕ is a **term of the form ans(v_1, …., v_n)** where v_1, …. ,v_n are the free variables in ϕ.
  - To answer ϕ, we form a disjunction Γ of the negation of ϕ and its answer literal and convert to clausal form.

Another Example

Consider the following

1. Aryan is the father of Jeevan.
2. Bhuban is the father of Kavya.
3. Fathers are parents.

Who is Jeevan’s parent?

$\exists x$ parent(x, Jeevan)

Another Example

Resolution Trace

Rather than the empty clause, the procedure halts as soon as it derives a clause consisting of only answer literals.

1.	father(Aryan, Jeevan)	C1
2.	father(Bhuban, Kavya)	C2
3.	[¬father(x,y) ∨ parent(x,y)]	C3
4.	[¬ parent(z, Jeevan) ∨ ans(z)]	Γ
5.	parent(Aryan, Jeevan)	1,3
6.	parent(Bhuban, Kavya)	2,3
7.	[¬father(x, Jeevan) ∨ ans(x)]	3,4
8.	ans(Aryan)	4,5
9.	ans(Aryan)	1,7

If this procedure produces only one answer literal, the terms it contains constitute the only answer to the questions.

Another Example

Suppose, the **database had BOTH** the **father and mother** of Jeevan. And we asked the same question.

Who is the parent of Jeevan?

Another Example

Suppose, the **database had BOTH** the **father and mother** of Jeevan. And we asked the same question.

Who is the parent of Jeevan?

The resolution trace shows that we can **derive two answers to this question.**

However, we have no way of knowing **whether or not the answer statement** from the given refutation **exhausts all possibilities**.

Resolution Trace

1.	father(Aryan, Jeevan)	C1
2.	mother(Annie, Jeevan)	C2
3.	[¬father(x,y) ∨ parent(x,y)]	C3
4.	[¬mother(x,y) ∨ parent(x,y)]	C4
5.	[¬ parent(z, Jeevan) ∨ ans(z)]	Γ
6.	parent(Aryan, Jeevan)	1,3
7.	parent(Annie, Jeevan)	2,4
8.	[¬father(s, Jeevan) ∨ ans(s)]	3,5
9.	[¬mother(t, Jeevan) ∨ ans(t)]	4,5
10.	ans(Aryan)	5,6
11.	ans(Annie)	5,7
12.	ans(Aryan)	1,8
13.	ans(Annie)	2,9

Another Example

Some cases the procedure can **yield a clause** containing **more than one answer literal**.

Significance of this is that **no one answer is guaranteed to work**; but one of the answers must be correct.

Database in such cases is a disjunction; we get a second clause after arriving at a answer!

The disjunction in the database asserting that either Aryan or Bhabesh is Jeevan’s father.

Another Example

Some cases the procedure can **yield a clause** containing **more than one answer literal**.

Significance of this is that **no one answer is guaranteed to work**; but one of the answers must be correct.

Database in such cases is a disjunction; we get a second clause after arriving at a answer!

Resolution Trace

1.	[father(Aryan, Jeevan ∨ father(Bhabesh, Jeevan)]	C1
2.	[¬ father(z, Jeevan) ∨ ans(z)]	Γ
3.	father(Bhabesh,Jeevan) ∨ ans(Aryan)	1,2
4.	[ans(Aryan) ∨ ans(Bhabesh)]	2,3

The disjunction in the database asserting that either Aryan or Bhabesh is Jeevan's father.

We can continue searching in hope of finding a more specific answer; However, given the **undecidability of logical implication, we can never know in general whether we can stop** and say no more specific answer exists.

Answer Extraction Process

The **steps of the answer extraction process** can be summarized as follows:

1. **Resolution-refutation tree is found**. Unification subsets of the clauses in this tree are marked.
2. **New variables are substituted for any Skolem functions** in the **clauses from the negation of the goal.**
3. Clauses resulting from **negation of the goal are converted into tautologies.**
4. **Modified Proof Tree is produced** replicating the structure of the original Refutation Tree; **use a unification set determined by the original unification set** used by corresponding resolution.
5. Clause at the **root of the Modified Proof Tree is the answer**.

Answer Extraction Process

- The **final statement of the Modified Proof Tree** i.e., the answer statement **depends upon the refutation which is replicated**.
  - Several different refutations may exist for the same problem; **from each an answer statement could be extracted.**
  - Some answer statements may be identical; some more general than the others.
- No way to know **whether the extracted answer is the most general** than the others.
  - One could, of course, continue to search for proofs until one found one producing a sufficiently general answer
  - **Undecidability of predicate calculus**, though would mean, we would not always know whether we had found all possible proofs.