RESOLUTION

By

Ankit Shah

Professor Harper Langston

Discrete Mathematics

Summer 2007

RESOLUTION

Resolution is in itself a very wide topic and is now days very widely used in Computer Science and Artificial Intelligence. Resolution has its roots in Mathematical Logic. One of the main reason why I selected Resolution as my topic for presentation is that I wanted to have indepth knowledge about Resolution and wanted to clear my fundamentals about mathematical logic. It was a very knowledgeable experience to gather information about Resolution and cover such a vast topic in a 15 minute presentation.

Resolution was invented in the year 1965 by a Mathematician called Alan Robinson.

Definition: Resolution yields a complete inference algorithm when coupled with any complete search algorithm.
Resolution makes use of the inference rules. Resolution performs deductive inference. Resolution uses proof by contradiction.
One can perform Resolution from a Knowledge Base. A Knowledge Base is a collection of facts or one can even call it a database with all facts.

Resolution Principle:

$$\frac{l_1 \vee \ldots \vee l_k,\ m_1 \vee \ldots \vee m_n}{l_1 \vee \ldots \vee l_{i-1} \vee l_{i+1} \vee \ldots \vee l_k \vee m_1 \vee \ldots \vee m_{j-1} \vee m_{j-1} \vee \ldots \vee m_n}$$

l_i and m_j are complementary literals.
Resolution takes two clauses and produces a new clause containing all the literals of the two original clauses except the two complementary literals.

Resolution Algorithm

Resolution basically works by using the principle of proof by contradiction. To find the conclusion we should negate the conclusion. Then the resolution rule is applied to the resulting clauses. Each clause that contains complementary literals is resolved to produce a

new clause, which can be added to the set of facts (if it is not already present). This process continues until one of the two things happen:

- There are no new clauses that can be added
- An application of the resolution rule derives the empty clause

An empty clause shows that the negation of the conclusion is a complete contradiction, hence the negation of the conclusion is invalid or false or the assertion is completely valid or true.

Algorithm

function PL-RESOLUTION (KB, @) returns true or false

inputs: KB, the knowledge base, group of sentences/facts in propositional logic

@, the query, a sentence in propositional logic

clauses → the set of clauses in the CNF representation of KB ^ @

new → { }

loop do

for each C_i, C_j in clauses do

resolvents → PL-RESOLVE (C_i, C_j)

if resolvents contains the empty clause the return true

new → new union resolvents

if new is a subset of clauses then return false

clauses → clauses union true

Steps for Resolution

- Convert the given statements in Predicate/Propositional Logic
- Convert these statements into Conjunctive Normal Form
- Negate the Conclusion (Proof by Contradiction)
- Resolve using a Resolution Tree (Unification)

Steps to Convert to CNF (Conjunctive Normal Form)

Every sentence in Propositional Logic is logically equivalent to a conjunction of disjunctions of literals. A sentence expressed as a conjunction of disjunctions of literals is said to be in Conjunctive normal Form or CNF.

1. Eliminate implication '$\rightarrow$'

   $a \rightarrow b = \sim a \vee b$

   $\sim (a \wedge b) = \sim a \vee \sim b$ …….. DeMorgan's Law

   $\sim (a \vee b) = \sim a \wedge \sim b$ ……... DeMorgan's Law

   $\sim (\sim a) = a$

2. Eliminate Existential Quantifier '$\exists$'

   To eliminate an independent Existential Quantifier, replace the variable by a Skolem constant. This process is called as Skolemization.

   Example: $\exists y$: President (y)

   Here **'y'** is an independent quantifier so we can replace **'y'** by any name (say – George Bush).

   So, **$\exists y$: President (y)** becomes **President (George Bush)**.

   To eliminate a dependent Existential Quantifier we replace its variable by Skolem Function that accepts the value of 'x' and returns the corresponding value of 'y.'

   Example: $\forall x : \exists y$: father_of (x, y)

   Here **'y'** is dependent on **'x',** so we replace **'y'** by **S(x).**

   So, **$\forall x : \exists y$: father_of (x, y)** becomes **$\forall x : \exists y$: father_of (x, S(x)).**

3. Eliminate Universal Quantifier '$\forall$'

   To eliminate the Universal Quantifier, drop the prefix in PRENEX NORMAL FORM i.e. just drop $\forall$ and the sentence then becomes in PRENEX NORMAL FORM.

4. Eliminate AND '^'

   a ^ b splits the entire clause into two separate clauses i.e. a and b

   (a v b) ^ c splits the entire clause into two separate clauses a v b and c

   (a ^ b) v c splits the clause into two clauses i.e. a v c and b v c

To eliminate '^' break the clause into two, if you cannot break the clause, distribute the OR 'v' and then break the clause.

EXAMPLE

Now let us see an example which uses resolution.

Problem Statement:

1. Ravi likes all kind of food.
2. Apples and chicken are food
3. Anything anyone eats and is not killed is food
4. Ajay eats peanuts and is still alive
5. Rita eats everything that Ajay eats

Prove by resolution that Ravi likes peanuts using resolution.

Solution:

Step 1: Converting the given statements into Predicate/Propositional Logic

i. $\forall x : food(x) \rightarrow likes(Ravi, x)$
ii. $food(Apple) \wedge food(chicken)$
iii. $\forall a : \forall b: eats(a, b) \wedge killed(a) \rightarrow food(b)$
iv. $eats(Ajay, Peanuts) \wedge alive(Ajay)$
v. $\forall c : eats(Ajay, c) \rightarrow eats(Rita, c)$
vi. $\forall d : alive(d) \rightarrow \sim killed(d)$
vii. $\forall e: \sim killed(e) \rightarrow alive(e)$

Conclusion: likes (Ravi, Peanuts)

Step 2: Convert into CNF

i. $\sim food(x) \vee likes(Ravi, x)$
ii. Food (apple)
iii. Food (chicken)
iv. $\sim eats(a, b) \vee killed(a) \vee food(b)$

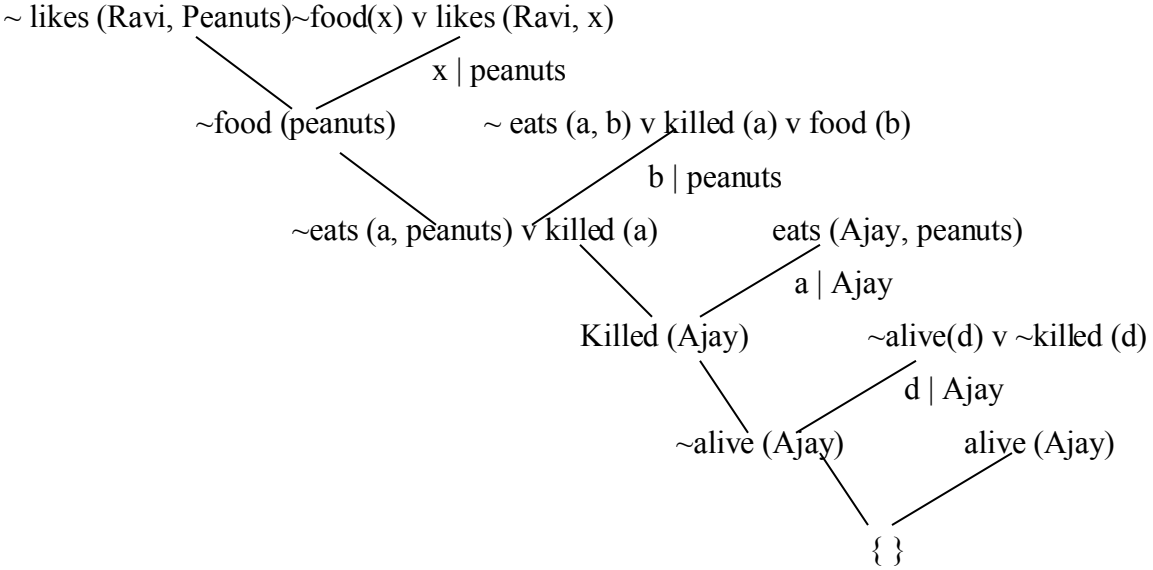
v. Eats (Ajay, Peanuts)
vi. Alive (Ajay)
vii. ~eats (Ajay, c) V eats (Rita, c)
viii. ~alive (d) v ~ killed (d)
ix. Killed (e) v alive (e)

Conclusion: likes (Ravi, Peanuts)

Step 3: Negate the conclusion

~ likes (Ravi, Peanuts)

Step 4: Resolve using a resolution tree



Hence we see that the negation of the conclusion has been proved as a complete contradiction with the given set of facts.

Hence the negation is completely invalid or false or the assertion is completely valid or true.

Hence Proved

Explanation of Resolution Tree (Unification)

In the first step of the resolution tree, **~ likes (Ravi, Peanuts)** and **likes (Ravi, x)** get resolved (cancelled). So we are only left with ~food (peanuts). In this **'x**' is replaced by peanuts i.e. **'x'** is bound to peanuts.

In the second step of the resolution tree, **~food(peanuts)** and **food (b)** get resolved, so we are left with **~eats (a, peanuts) v killed(a)**. In this '**b'** is bound to peanuts thus we replace every instance of '**b'** by **Peanuts** in that particular clause. Thus now we are left with **~eats (a, peanuts) v killed (a).**

In the third step of the resolution tree, **~eats (a, peanuts)** and eats (Ajay, peanuts) gets resolved. In this **'a'** is bound to **Ajay**. So we replace every instance of **'a'** by **Ajay**. Thus now we are left with **killed (Ajay).**

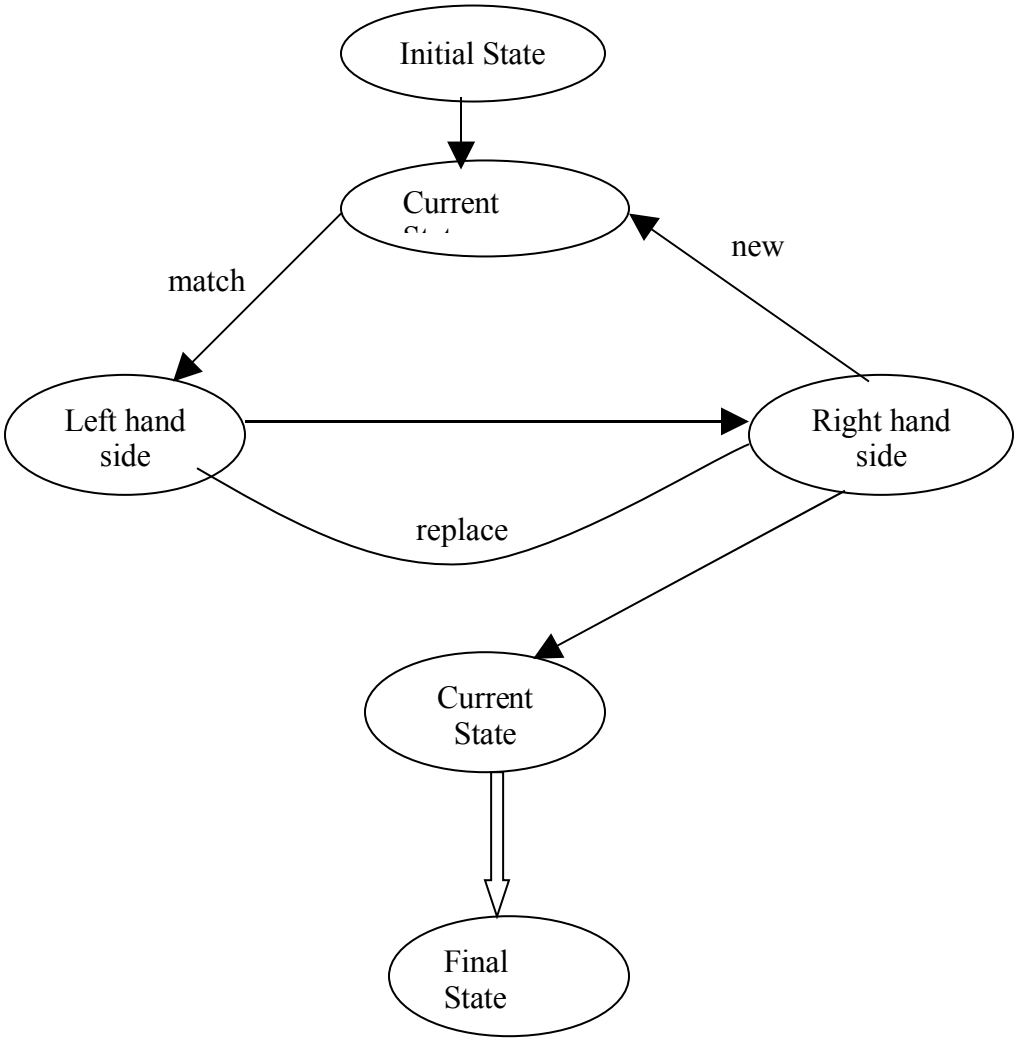
In the forth step of the resolution tree, **killed (Ajay)** and **~killed (d)** get resolved. In this **'d**' is bound to Ajay, thus ever instance of **'d**' is replaced by **Ajay**. Now we are left with **~alive(Ajay)**.

In the fifth step of the resolution tree, ~Alive(Ajay) and Alive(Ajay) get resolved and we are only left with a null set.

More on Resolution

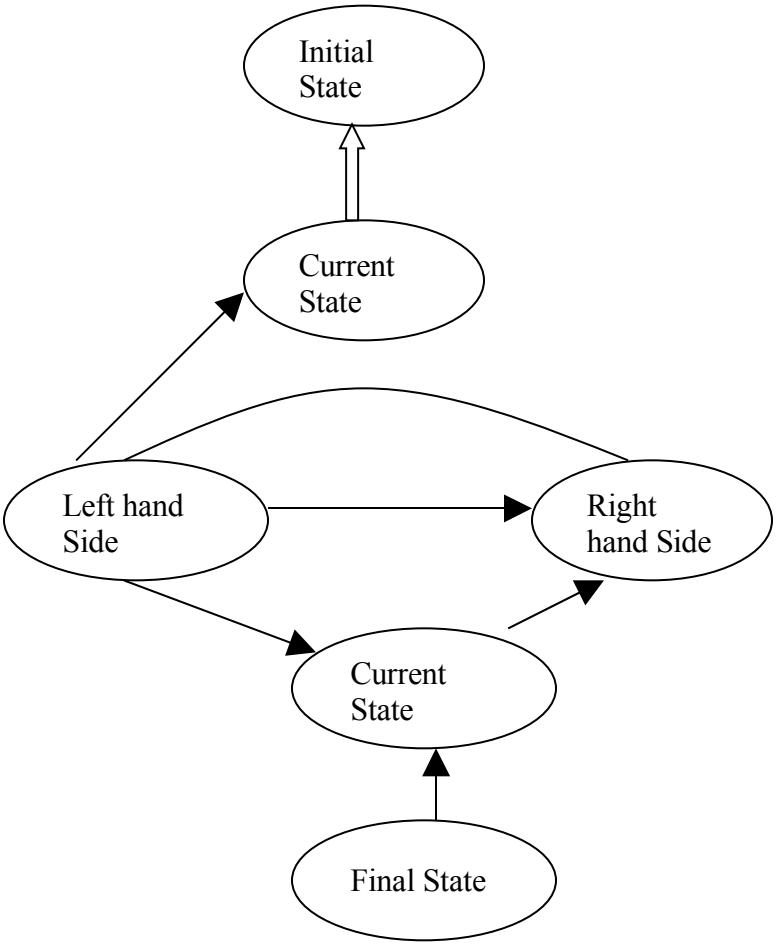
Resolution even helps us in Forward and Backward Reasoning.

Forward Reasoning



Diagrammatic Representation of Forward Reasoning

Backward Reasoning



Diagrammatic Representation of Backward Reasoning

- For theorem proving Backward Reasoning is more suitable than Forward Reasoning.
- If we have one Initial State and many Final States then Forward Reasoning is suitable.
- If we have one Final State and many Initial States then Backward Reasoning is suitable.

Uses of Resolution in Today's World

- Helps in the development of computer programs to automate reasoning and theorem proving
- Used widely in AI.
- Helps in Forward and Backward Reasoning
- Resolution is a proof by contradiction which is even used in Math problems.

Sources of Information

- www.csgeek.net/cs5804/files/Note-6.ppt
- http://mathworld.wolfram.com/Resolution.html
- http://mathworld.wolfram.com/Unification.html
- http://mathworld.wolfram.com/ResolutionPrinciple.html
- Artificial Intelligence: A Modern Approach by Stuart Russell and Peter Norvig, Secong Edition, Published 2003 Prentice Hall
- Discrete Mathematics and its Applications by Kenneth H. Rosen
- http://www.sci.brooklyn.cuny.edu/~parsons/courses/32-fall-2003/notes/lect18.pdf

Acknowledgements

I would like to thank Professor Harper for his help and guidance. He has been a wonderful professor and made Discrete Math very interesting. It was a good learning experience. I would even like to thank Prof. Davi Geiger who guided me through the topic of RESOLUTION and cleared my doubts.