

✓ **Resolution Principle** As another way of proving that an argument is correct, we use the resolution principle. This idea is used in the logic programming language Prolog.

A variable or negation of a variable is called a literal.

A disjunction of literals is called a sum and a conjunction of literals is called a product. A clause is a disjunction of literals, i.e., it is a sum.

For any two clauses C_1 and C_2 , if there is a literal L_1 in C_1 that is complementary to a literal L_2 in C_2 , then delete L_1 and L_2 from C_1 and C_2 respectively and construct the disjunction of the remaining clauses. The constructed clause is a resolvent of C_1 and C_2 .

Example 10 Suppose you have

$$C_1 = P \vee Q \vee R$$

$$C_2 = \neg P \vee \neg S \vee T$$

Then P and $\neg P$ are complementary to each other. The resolvent of C_1 and C_2 is obtained by deleting P and $\neg P$ from C_1 and C_2 respectively, and finding the disjunction of the remaining literals. In this case it is $Q \vee R \vee \neg S \vee T$. ◀

Theorem 1

Given two clauses C_1 and C_2 , a resolvent C of C_1 and C_2 is a logical consequence of C_1 and C_2 .

For example, consider the rules modus ponens and modus tollens.

Modus ponens rule is $P \wedge (P \rightarrow Q) \rightarrow Q$

Putting in clause form this amounts to

$$C_1 : P$$

$$C_2 : \neg P \vee Q$$

The resolvent of C_1 and C_2 is Q which is the logical consequence of C_1 and C_2 .

Similarly modus tollens rule is $(P \rightarrow Q) \wedge \neg Q \rightarrow \neg P$

Putting in clause form

$$C_1 : \neg P \vee Q$$

$$C_2 : \neg Q$$

The resolvent of C_1 and C_2 is $\neg P$ which is the logical consequence C_1 and C_2 .

Proof

Consider two clauses C_1 and C_2 .

C_1 contains L and C_2 contain $\neg L$

Writing C_1 as $L \vee C_1'$

and C_2 as $\neg L \vee C_2'$

resolvent of C_1 and C_2 is $C = C_1' \vee C_2'$

Now we want to show C is the logical consequence of C_1 and C_2 . In essence, we want to show that if C_1 and C_2 are true, C is true.

Assume C_1 and C_2 are true.

Now there are two possibilities.

Either L is true or L is false.

If L is true, $\neg L$ is false and in order that C_2 is true, C_2' is true.

Hence $C = C_1' \vee C_2'$ is true.

If L is false, in order that C_1 is true, C_1' is true.

Hence $C = C_1' \vee C_2'$ is true.

So whenever C_1 and C_2 are true C is true.

Hence C is the logical consequence of C_1 and C_2 .

✓ **The resolution principle** Given a set S of clauses, a (resolution) deduction of C from S is a finite sequence C_1, C_2, \dots, C_k of clauses such that each C_i either is a clause in S or a resolvent of clauses preceding C_i and $C_k = C$. A deduction of \square (empty clause) is called a refutation or a proof of S .

If you have an argument where P_1, P_2, \dots, P_n are the premises and C is the conclusion, to get a proof using resolution principle, put P_1, \dots, P_n in clause form and add to it $\neg C$ in clause form. From this sequence, if \square can be derived, the argument is valid.

For example, consider modus ponens P

$$\frac{P}{\therefore Q}$$

In the clause form $C_1 = P$

$$C_2 = \neg P \vee Q$$

Take as C_3 , the negation of the conclusion, i.e., $\neg Q$, $C_3 = \neg Q$

From C_1 and C_2 by resolution you can get Q and from this and C_3 by resolution \square is arrived at.

	C_1	P
	C_2	$\neg P \vee Q$
	C_3	$\neg Q$
from C_1 and C_2 by resolution	C_4	Q
from C_3 and C_4 by resolution	C_5	\square

✓ **Example 11** We show that the following argument is correct.

If today is Tuesday, I have a test in Mathematics or Economics. If my Economics Professor is sick, I will not have a test in Economics. Today is Tuesday and my Economics Professor is sick. Therefore, I have a test in Mathematics.

Converting to logical notation.

Let T denote 'Today is Tuesday'

M denote 'I have a test in Mathematics'

E denote 'I have a test in Economics'

S denote 'My Economics Professor is sick'

So the premises are $T \rightarrow (M \vee E)$

$$S \rightarrow \neg E$$

$$T \wedge S$$

$$\therefore M$$

Putting in clause form $C_1 : \neg T \vee M \vee E$

$$C_2 : \neg S \vee \neg E$$

$$C_3 : T$$

$$C_4 : S$$

$$C_5 : \neg M \text{ (negation of conclusion)}$$

Now we have to check whether it is possible to derive \square from C_1, C_2, C_3, C_4, C_5 .

From C_1 and C_2 $C_6 : \neg T \vee M \vee \neg S$

from C_6 and C_3 $C_7 : M \vee \neg S$

from C_7 and C_4 $C_8 : M$
 from C_8 and C_5 $C_9 : \square$

Hence the argument is correct. ◀

✓ **Logic Programming** An important type of programming language is designed to reason using the rules of predicate logic. Prolog (from *Programming in Logic*), developed in the 1970s by computer scientists working in the area of artificial intelligence, is an example of such a language. Prolog programs include a set of declarations consisting of two types of statements, **Prolog facts** and **Prolog rules**. Prolog facts define predicates by specifying the elements that satisfy these predicates. Prolog rules are used to define new predicates using those already defined by Prolog facts. Example 28 illustrates these notions.

Links

Example 12 Consider a Prolog program given facts telling it the instructor of each class and in which classes students are enrolled. The program uses these facts to answer queries concerning the professors who teach particular students. Such a program could use the predicates $instructor(p, c)$ and $enrolled(s, c)$ to represent that professor p is the instructor of course c and that student s is enrolled in course c , respectively. For example, the Prolog facts in such a program might include:

```
instructor(chan, math273)
instructor(patel, ee222)
instructor(grossman, cs301)
enrolled(kevin, math273)
enrolled(juana, ee222)
enrolled(juana, cs301)
enrolled(kiko, math273)
enrolled(kiko, cs301)
```

(Lowercase letters have been used for entries because Prolog considers names beginning with an uppercase letter to be variables.)

A new predicate $teaches(p, s)$, representing that professor p teaches student s , can be defined using the Prolog rule

```
teaches(p, s) :- instructor(p, c), enrolled(s, c)
```

which means that $teaches(p, s)$ is true if there exists a class c such that professor p is the instructor of class c and student s is enrolled in class c . (Note that a comma is used to represent a conjunction of predicates in Prolog. Similarly, a semicolon is used to represent a disjunction of predicates.)

Prolog answers queries using the facts and rules it is given. For example, using the facts and rules listed, the query

```
?enrolled(kevin, math273)
```

produces the response

```
yes
```

because the fact $enrolled(kevin, math273)$ was provided as input. The query

```
?enrolled(X, math273)
```

produces the response

```
kevin
kiko
```

To produce this response, Prolog determines all possible values of X for which $enrolled(X, math273)$ has been included as a Prolog fact. Similarly, to find all the professors who are instructors in classes being taken by Juana, we use the query

```
?teaches(X, juana)
```

This query returns

patel
grossman

Now we shall consider the connection between resolution principle and Prolog.

A **Horn clause** is a clause with at most one non-negated literal. If all literals are negated, it is called a **headless Horn clause**. If one literal is non-negated, it is called a **headed Horn clause**.

Consider the following rule:

Aunt (x, y) :- Female (x), Sister (x, z), Parent (z, y) ✓ Rule (1)

This means if x is a Female and x is the Sister of z and z is a Parent of y , then x is the Aunt of y .

Suppose we want to conclude Sita is the Aunt of Mohan.

We have the following facts

Female (Sita) - F_1 ✓
Sister (Sita, Geetha) - F_2 ✓
Parent (Geetha, Mohan) - F_3 ✓

Instantiating the rule (1) we get

Aunt (Sita, Mohan) :- Female (Sita), Sister (Sita, Geetha), Parent (Geetha, Mohan).

Each F_1, F_2, F_3 is a headed Horn clause.

Rule (1) can be written in logical notation as

$\forall x \forall y \forall z [(Female(x) \wedge Sister(x, z) \wedge Parent(z, y)) \rightarrow Aunt(x, y)]$

Using instantiation we get

$(Female(Sita) \wedge Sister(Sita, Geetha) \wedge Parent(Geetha, Mohan)) \rightarrow Aunt(Sita, Mohan)$

Expressing as a clause this becomes

$\neg (Female(Sita) \wedge Sister(Sita, Geetha) \wedge Parent(Geetha, Mohan)) \vee Aunt(Sita, Mohan)$

Using DeMorgan's laws we get

$\neg Female(Sita) \vee \neg Sister(Sita, Geetha) \vee \neg Parent(Geetha, Mohan) \vee Aunt(Sita, Mohan) - IR_1$
(Instantiation of rule 1)

The conclusion we want to derive is Aunt (Sita, Mohan).

Negating the conclusion we get

$\neg Aunt(Sita, Mohan)$ - NC (negation of conclusion)


From F_1, F_2, F_3, IR_1 and NC , it can be seen that using resolution we can derive the empty clause. Hence the conclusion Aunt (Sita, Mohan) is correct. Note that in a rule in Prolog what occurs on the left hand side of :- is the headed portion and what occurs in the right hand portion is the headless portion. Hence a rule corresponds to a Horn clause. A fact consists of a single non-negated literal and hence a Horn clause. The conclusion derived is :- Aunt (Sita, Mohan) and for resolution we take the negation of the right hand side of :- . This is a headless Horn clause.

The question asked is ? Aunt (Sita, Mohan) and Prolog replies yes.

The goal is Aunt (Sita, Mohan) and Prolog tries to find rules and facts and using proper instantiation tries to see whether the goal is satisfied. This is called backward chaining.

Thus we see that the resolution principle is used in the logic programming language Prolog. It is also used in automatic theorem proving where a program or software package is used to prove theorems, given the axioms.

Fallacies Several common fallacies arise in incorrect arguments. These fallacies resemble rules of inference but are based on contingencies rather than tautologies. These are discussed here to show the distinction between correct and incorrect reasoning.

Links  The proposition $[(p \rightarrow q) \wedge q] \rightarrow p$ is not a tautology, because it is false when p is false and q is true. However, there are many incorrect arguments that treat this as a tautology. In other

words, they treat the argument with premises $p \rightarrow q$ and q and conclusion p as a valid argument form, which it is not. This type of incorrect reasoning is called the **fallacy of affirming the conclusion**.

Example 13 *Is the following argument valid?*

If you do every problem in this book, then you will learn discrete mathematics. You learned discrete mathematics.

Therefore, you did every problem in this book.

Solution Let p be the proposition "You did every problem in this book." Let q be the proposition "You learned discrete mathematics." Then this argument is of the form: if $p \rightarrow q$ and q , then p . This is an example of an incorrect argument using the fallacy of affirming the conclusion. Indeed, it is possible for you to learn discrete mathematics in some way other than by doing every problem in this book. (You may learn discrete mathematics by reading, listening to lectures, doing some, but not all, the problems in this book, and so on.)

The proposition $[(p \rightarrow q) \wedge q] \rightarrow p$ is not a tautology, because it is false when p is false and q is true. Many incorrect arguments use this incorrectly as a rule of inference. This type of incorrect reasoning is called the **fallacy of denying the hypothesis**.

Example 14 *Let p and q be as in Example 10. If the conditional statement $p \rightarrow q$ is true, and $\neg p$ is true, is it correct to conclude that $\neg q$ is true? In other words, is it correct to assume that you did not learn discrete mathematics if you did not do every problem in the book, assuming that if you do every problem in this book, then you will learn discrete mathematics?*

Solution It is possible that you learned discrete mathematics even if you did not do every problem in this book. This incorrect argument is of the form $p \rightarrow q$ and $\neg p$ imply $\neg q$, which is an example of the fallacy of denying the hypothesis.

Rules of Inference for Quantified Statements We have discussed rules of inference for propositions. We will now describe some important rules of inference for statements involving quantifiers. These rules of inference are used extensively in mathematical arguments, often without being explicitly mentioned.

Universal instantiation is the rule of inference used to conclude that $P(c)$ is true, where c is a particular member of the domain, given the premise $\forall x P(x)$. Universal instantiation is used when we conclude from the statement "All women are wise" that "Lisa is wise," where Lisa is a member of the domain of all women.

Universal generalization is the rule of inference that states that $\forall x P(x)$ is true, given the premise that $P(c)$ is true for all elements c in the domain. Universal generalization is used when we show that $\forall x P(x)$ is true by taking an arbitrary element c from the domain and showing that $P(c)$ is true. The element c that we select must be an arbitrary, and not a specific, element of the domain. That is, when we assert from $\forall x P(x)$ the existence of an element c in the domain, we have no control over c and cannot make any other assumptions about c other than it comes from the domain. Universal generalization is used implicitly in many proofs in mathematics and is seldom mentioned explicitly. However, the error of adding unwarranted assumptions about the arbitrary element c when universal generalization is used is all too common in incorrect reasoning.

Existential instantiation is the rule that allows us to conclude that there is an element c in the domain for which $P(c)$ is true if we know that $\exists x P(x)$ is true. We cannot select an arbitrary value of c here, but rather it must be a c for which $P(c)$ is true. Usually we have no knowledge of what c is, only that it exists. Because it exists, we may give it a name (c) and continue our argument.

Existential generalization is the rule of inference that is used to conclude that $\exists x P(x)$ is true when a particular element c with $P(c)$ true is known. That is, if we know one element c in the domain for which $P(c)$ is true, then we know that $\exists x P(x)$ is true.

We su
for quanti

Exempl
computer
in compu

Solution

Extr
Example

Step

1. $\forall x$

2. $D()$

3. $D()$

4. $C()$

Exam
in this c
the book

Solution

The pre
can be u

Step

1. \exists

2. C

3. C

4. \forall

5. C

6. P

7. $-$

8. F

9. \exists

We summarize these rules of inference in Table 2. We will illustrate how one of these rules of inference for quantified statements is used in Example 15.

Example 15 Show that the premises "Everyone in this discrete mathematics class has taken a course in computer science" and "Marla is a student in this class" imply the conclusion "Marla has taken a course in computer science."

Table 2 Rules of Inference for quantified statements.

Rule of Inference	Name
$\frac{\forall x P(x)}{\therefore P(c)}$	Universal instantiation
$\frac{P(c) \text{ for an arbitrary } c}{\therefore \forall x P(x)}$	Universal generalization
$\frac{\exists x P(x)}{\therefore P(c) \text{ for some element } c}$	Existential instantiation
$\frac{P(c) \text{ for some element } c}{\therefore \exists x P(x)}$	Existential generalization

Solution Let $D(x)$ denote "x is in this discrete mathematics class," and let $C(x)$ denote "x has taken a course in computer science." Then the premises are $\forall x(D(x) \rightarrow C(x))$ and $D(\text{Marla})$. The conclusion is $C(\text{Marla})$.

Extra Examples The following steps can be used to establish the conclusion from the premises.

Step	Reason
1. $\forall x(D(x) \rightarrow C(x))$	Premise
2. $D(\text{Marla}) \rightarrow C(\text{Marla})$	Universal instantiation from (1)
3. $D(\text{Marla})$	Premise
4. $C(\text{Marla})$	Modus ponens from (2) and (3)

Example 16 Show that the premises "A student in this class has not read the book," and "Everyone in this class passed the first exam" imply the conclusion "Someone who passed the first exam has not read the book."

Solution Let $C(x)$ be "x is in this class," $B(x)$ be "x has read the book," and $P(x)$ be "x passed the first exam." The premises are $\exists x(C(x) \wedge \neg B(x))$ and $\forall x(C(x) \rightarrow P(x))$. The conclusion is $\exists x(P(x) \wedge \neg B(x))$. These steps can be used to establish the conclusion from the premises.

Step	Reason
1. $\exists x(C(x) \wedge \neg B(x))$	Premise
2. $C(a) \wedge \neg B(a)$	Existential instantiation from (1)
3. $C(a)$	Simplification from (2)
4. $\forall x(C(x) \rightarrow P(x))$	Premise
5. $C(a) \rightarrow P(a)$	Universal instantiation from (4)
6. $P(a)$	Modus ponens from (3) and (5)
7. $\neg B(a)$	Simplification from (2)
8. $P(a) \wedge \neg B(a)$	Conjunction from (6) and (7)
9. $\exists x(P(x) \wedge \neg B(x))$	Existential generalization from (8)

71A