Logic and Computation:
Propositional Logic

Jonathan Buss

October 1, 2015

Based in part on materials prepared by B. Bonakdarpour from Huth & Ryan's text.

Additional thanks to D. Maftuleac, R. Trefler, and P. Van Beek.

Broad Outline of Propositional Logic

- Introduction to Propositional Logic
- Syntax
- Semantics
- Proof Systems

Logic: What and Why

Logic is the systematic study of the principles of reasoning and inference.

We use logic throughout computer science,

- To model the computer hardware, software and embedded systems we create or encounter, in order to *reason* about those objects in a mathematically precise and rigorous manner.
- To understand how to develop systems that can themselves apply reason and make inferences ("artificial intelligence").

Historically, logic and CS are closely linked.

- To define and build a "computer" required deep ideas from logic.
- Computer science gave the first real definition of "rigorous argument": an argument that may be checked by a machine.

An Example Argument

Consider this example.

> If the train arrives late and there are no taxis at the station, then John is late for his meeting.
>
> John is not late for his meeting.
>
> The train did arrive late.
>
> *Therefore*, there were taxis at the station.

Question. Is this argument *valid*? Why, or why not?

An Example Argument

Consider this example.

> If the train arrives late and there are no taxis at the station, then John is late for his meeting.
>
> John is not late for his meeting.
>
> The train did arrive late.
>
> *Therefore*, there were taxis at the station.

Question. Is this argument *valid*? Why, or why not?

Question. What is the structure of the argument?

An Example Argument

Consider this example.

> If the train arrives late and there are no taxis at the station, then John is late for his meeting.
>
> John is not late for his meeting.
>
> The train did arrive late.
>
> *Therefore*, there were taxis at the station.

Question. Is this argument *valid*? Why, or why not?

Question. What is the structure of the argument?

We can represent the structure symbolically as

> If p and not q, then r. Not r. p. Therefore q.

An Example Argument (2)

The argument in the previous example has the form

> If p and not q, then r. Not r. p. Therefore q.

where

p	stands for “the train arrives late.”
q	stands for “there are taxis at the station.”
r	stands for “John is late for his meeting.”

An Example Argument (2)

The argument in the previous example has the form

If p and not q, then r. Not r. p. Therefore q.

where

p stands for "the train arrives late."
q stands for "there are taxis at the station."
r stands for "John is late for his meeting."

What happens if we change our notion of p, q and r? Perhaps

p stands for "It rains."
q stands for "Jane takes her umbrella."
r stands for "Jane gets very wet."

Then the argument changes. . .

An Example Argument (3)

The essential argument: If p and not q, then r. Not r. p. Therefore q.

New choices for p, q and r:

- p stands for "It rains."
- q stands for "Jane takes her umbrella."
- r stands for "Jane gets very wet."

The new argument is

> If it rains, and Jane does not take her umbrella, then Jane gets very wet. Jane does not get very wet. It rains. Therefore, Jane takes her umbrella.

An equally valid argument!

What Is Logic?—Reprise

In the example argument,

- The factual content of the statements doesn't matter.
- The relationships among the statements govern the argument.

Logic concerns careful reasoning about the process of reasoning.

As part of this care, we need to know

- What, exactly, constitutes a "statement"?
- What, precisely, do the logical relationships mean?

We shall start with a basic form of logic, called *propositional logic*.

Propositions

A *proposition* is a declarative sentence that is either *true* or *false*.

In other words, we make the following defining assumption:

> For any particular proposition, in any particular situation (or "world"), either the proposition is true, or the proposition is false, and it is never the case that the proposition is both true and false.

Examples of Propositions

Each of the following is a proposition.

- The sum of 3 and 5 is 8.
- The sum of 3 and 5 is 35.
- $x \geq 5$.
- Program p terminates.
- If the input to program p is a non-negative integer x, then p outputs the value x^2.
- If Kathleen Wynne is a Liberal, then Stephen Harper is a Tory.
- Every even number greater than 2 is the sum of two prime numbers.
- Jane reacted violently to Jack's accusations.

In some of the cases, we may not know whether the statement is true or false, but it's one or the other—and not both.

Translating from English to propositional logic

English phrases for connectives

$\neg p$	not p; p does not hold; p is false; it is not the case that p
$p \wedge q$	p and q; p but q; not only p but q; p while q; p despite q, p yet q, p although q
$p \vee q$	p or q, p or q or both, p and/or q, p unless q
$p \rightarrow q$	if p then q; p implies q; q if p, p only if q, q when p, p is sufficient for q, q is necessary for p,
$p \leftrightarrow q$	p if and only if q (p iff q); p is equivalent to q; p exactly if q; p is necessary and sufficient for q,

Translating from English to propositional logic

Examples

1. She is clever and hard working.
2. He is clever but not hard working.
3. He didn't write the letter or the letter was lost.
4. If he does not study hard then he will fail.
5. He must study hard; otherwise he will fail.
6. He will fail unless he studies hard.
7. He will not fail only if he studies hard.

Translating from English to propositional logic

Examples

1. If it rains, he will be at home; otherwise he will go to the market or to school.
2. The sum of two numbers is even if and only if both numbers are even or both numbers are odd.
3. If y is an integer then z is not real, provided that x is rational.

Propositions

Some sentences are not propositions. For example, a sentence might be

Interrogative:	Where shall we go to eat?
Imperative:	Please pass the salt.
Ambiguous:	Time flies like an arrow.
Nonsense:	The mome raths outgrabe.
Otherwise problematic:	
	This sentence is false.

In the field of "artificial intelligence," one must deal with such sentences. For this course, however, we shall ignore them.

The Aspects of Logic

Propositional logic is a form of *symbolic* logic. That is, it uses strings of symbols to represent propositions and to build arguments.

A symbolic logic is formalized by the following.

Syntax: What statements do we consider?

Semantics: What does a statement mean?

Proof procedures: Given a statement, can we prove it true?

We shall treat each part in detail.

Syntax of Propositional Logic

Atomic and compound propositions

In propositional logic, simple *atomic* propositions are the basic building blocks.

We connect atomic propositions into *compound* propositions, and then analyze sets of interrelated propositions.

Typical questions to consider:

- Does a given sequence of propositions form a valid argument?
- Can all propositions in a given set be true simultaneously?

First, however, we must answer the question,

What, exactly, is a proposition?

Symbols and expressions

Propositions are represented by *formulas*.

A formula consists of a sequence of *symbols*. There are three kinds.

Propositional variables. Usually lowercase Latin letters;
e.g., p, q, r, etc., perhaps with subscripts (p_1, p_2, q_{27}, etc.),

Connectives. We shall use $\neg$, $\wedge$, $\vee$, $\rightarrow$ and $\leftrightarrow$.
(Others are possible.)

Punctuation. Only two such: '(' and ')'.
(We could avoid these, if we wished.)

Every formula is a sequence of symbols, but not every sequence of symbols is a formula.

We call an arbitrary finite sequence of symbols an *expression* (or string).

Example: expression

An *expression* is a finite sequence (or “string”) of symbols.

The *length* of an expression is its number of symbols.

For example, $(\neg)()\vee pq\rightarrow$ is an expression.

Questions:

> What is its length of this expression?
> Is it a formula?

We often use a letter that is not formally a symbol in order to name an expression. For example, we might denote the expression above by α.

This is an example of a “meta-symbol”. It is NOT a symbol!

Talking about expressions

Some terminology for expressions.

- Two expressions α and β are *equal*, written as $\alpha = \beta$, iff they are of the same length, say n, and if $n > 0$ then for all $i \in [1..n]$ the ith symbol of α is the same as the ith symbol of β.
- We write $\alpha\beta$ to mean the *concatenation* of two expressions α and β. For example, if
$$\alpha = (\neg)()$$
and
$$\beta = \vee pq{\rightarrow}$$
then
$$\alpha\beta = (\neg)()\vee pq{\rightarrow} \quad .$$

Concatenation, formally

Definition:

If α is an expression of length i and β is an expression of length j, then $\alpha\beta$ is an expression of length $i+j$. We have

$$\text{The } k\text{th symbol of } \alpha\beta \text{ is } \begin{cases} \text{the } k\text{th symbol of } \alpha & \text{if } k \leq i \\ \text{the } (k-i)\text{th symbol of } \beta & \text{if } k > i \end{cases}$$

Definition of "formula"

Let $\mathcal{P}$ be a set of propositional variables. We define the set of *formulas over $\mathcal{P}$* inductively as follows.

1. An expression consisting of a single symbol of $\mathcal{P}$ is a formula.
2. If φ is a formula, then $(\neg\varphi)$ is a formula.
3. If φ is a formula and η is a formula, then each of

   $$(\varphi \wedge \eta)\ , \quad (\varphi \vee \eta)\ , \quad (\varphi \rightarrow \eta)\ , \quad \text{and} \quad (\varphi \leftrightarrow \eta)$$

   is a formula.
4. Nothing else is a formula.

(Note the use of the meta-symbols 'φ' and 'η' to refer to formulas.)

Examples: well-formed formulas

Example: The following are well-formed formulas.

1. p, q, r, s	(rule 1)
2. $(\neg p)$	(rule 2, from #1)
3. $(r \wedge q)$	(rule 3, from #1)
4. $((\neg p) \rightarrow s)$	(rule 3, from #2 and #1)
5. $((r \wedge q) \vee ((\neg p) \rightarrow s))$	(rule 3, from #3 and #4)
6. $(\neg(r \wedge q))$	(rule 2, from #3)

The Six Kinds of Formulas

From the definition, we see that there are six kinds of formulas.

- A propositional variable is called an *atom*.
- A formula $(\neg\varphi)$ is called a *negation*.
- A formula $(\varphi \wedge \eta)$ is called a *conjunction*.
- A formula $(\varphi \vee \eta)$ is called a *disjunction*.
- A formula $(\varphi \to \eta)$ is called an *implication*.
- A formula $(\varphi \leftrightarrow \eta)$ is called an *equivalence*.

Question: Can a formula have two (or more) kinds? E.g., can it be both a conjunction and an implication? Or both a negation and a disjunction?

Question: Do we care?

Semantics of Propositional Logic

Semantics

The *semantics* of a logic describes how to interpret the well-formed formulas of the logic.

The semantics of propositional logic is “compositional”; in other words, the meaning of a whole formula derives from the meaning of its parts.

In propositional logic, we need to give meaning to atoms, connectives, and formulas.

For example, the interpretation of formula $(p \wedge q)$ depends on three things: the meaning of p, the meaning of q, and the meaning of $\wedge$.

Valuations: the status of atoms

Definition:

A *truth valuation* is a function with the set of all proposition symbols as domain and $\{\mathtt{F}, \mathtt{T}\}$ as range.

In other words, a truth valuation assigns a value to every propositional variable.

- If $t(p) = \mathtt{T}$, then we say/write, "t makes p true".
- If $t(p) = \mathtt{F}$, then we say/write, "t makes p false".

A propositional variable has no intrinsic meaning; it gets a meaning only via a valuation.

Compound formulas

Let φ and η be two formulas that express propositions $\mathcal{A}$ and $\mathcal{B}$. Intuitively, we give the following meanings to combinations.

$\neg\varphi$	Not $\mathcal{A}$
$\varphi \wedge \eta$	$\mathcal{A}$ and $\mathcal{B}$
$\varphi \vee \eta$	$\mathcal{A}$ or $\mathcal{B}$
$\varphi \rightarrow \eta$	If $\mathcal{A}$ then $\mathcal{B}$
$\varphi \leftrightarrow \eta$	$\mathcal{A}$ iff $\mathcal{B}$

The English, however, can be ambiguous. We want precise meanings for formulas.

Semantics of Connectives

Formally, a connective represents a function from truth values to truth values.

The connective $\neg$ is unary; it maps one value to one value. We can show its function in a picture, known as a *truth table*:

φ	$(\neg\varphi)$
T	F
F	T

The other connectives are binary; they map two values to one value. Thus their truth tables require four lines to cover the possibilities.

Truth tables for connectives

The binary connectives:

φ	η	$(\varphi \wedge \eta)$	$(\varphi \vee \eta)$	$(\varphi \to \eta)$	$(\varphi \leftrightarrow \eta)$
T	T	T	T	T	T
T	F	F	T	F	F
F	T	F	T	T	F
F	F	F	F	T	T

$\wedge$ is as expected: $(\varphi \wedge \eta)$ is true if and only if both φ and η are true.

The column for $\vee$ shows that it means "one or the other or both". (This is called "inclusive or".)

The column for $\to$ may look a bit strange.

“If-then”??

Some people find the meaning of $\rightarrow$ rather unintuitive. You may want to think of $\rightarrow$ as meaning “*truth is preserved*”.

- The meaning of $\mathtt{T} \rightarrow \mathtt{T}$ is $\mathtt{T}$ because truth is preserved.
- The meaning of $\mathtt{T} \rightarrow \mathtt{F}$ is $\mathtt{F}$ because truth is not preserved.
- The meaning of $\mathtt{F} \rightarrow \mathtt{T}$ and $\mathtt{F} \rightarrow \mathtt{F}$ are both $\mathtt{T}$, because there is no truth to preserve.

For example, the following sentence comes out true:

If everyone is a child, then the moon is made of green cheese.

Some people prefer to call that sentence non-sensical, rather than true. But propositional logic gives every formula a meaning.

Summary: value of a formula

Fix a truth valuation t. Every formula φ has a value under t, denoted φ^t, determined as follows.

1. $p^t = t(p)$.
2. $(\neg\varphi)^t = \begin{cases} \mathtt{T} & \text{if } \varphi^t = \mathtt{F} \\ \mathtt{F} & \text{if } \varphi^t = \mathtt{T} \end{cases}$
3. $(\varphi \wedge \eta)^t = \begin{cases} \mathtt{T} & \text{if } \varphi^t = \eta^t = \mathtt{T} \\ \mathtt{F} & \text{otherwise} \end{cases}$
4. $(\varphi \vee \eta)^t = \begin{cases} \mathtt{T} & \text{if } \varphi^t = \mathtt{T} \text{ or } \eta^t = \mathtt{T} \\ \mathtt{F} & \text{otherwise} \end{cases}$
5. $(\varphi \to \eta)^t = \begin{cases} \mathtt{T} & \text{if } \varphi^t = \mathtt{F} \text{ or } \eta^t = \mathtt{T} \\ \mathtt{F} & \text{otherwise} \end{cases}$
6. $(\varphi \leftrightarrow \eta)^t = \begin{cases} \mathtt{T} & \text{if } \varphi^t = \eta^t \\ \mathtt{F} & \text{otherwise} \end{cases}$

The value of a formula comes from the values of its variables, combined as given by its connectives.

The valuation t is necessary. Without a valuation, a formula has no value.

Unique Readability of Formulas

We have defined the semantics (meaning) of a formula from its syntax (the succession of symbols).

Is this well-defined? Or can a formula get two different meanings?

Theorem. Every formula has a unique derivation as a well-formed formula. That is, each formula has exactly one of the six forms:

> (1) an atom, (2) $(\neg\varphi)$, (3) $(\varphi \wedge \eta)$, (4) $(\varphi \vee \eta)$, (5) $(\varphi \rightarrow \eta)$, or (6) $(\varphi \leftrightarrow \eta)$.

In each case, it is of that form in exactly one way.

(Why) Is the Theorem True?

As an example, consider $\big((p \wedge q) \to r\big)$. It can be formed from the two formulas $(p \wedge q)$ and r using the connective $\to$.

If we tried to form it using $\wedge$, the two parts would need to be $(p$ and $q) \to r$. But neither of those is a formula!

How can we be sure the theorem holds for *every* formula?

Mathematical Induction

To prove the theorem, we will use mathematical induction.

Before doing the proof, we will review mathematical induction.

This starts with the natural numbers.

Natural Numbers

The "natural numbers" are the numbers we use to count things.

Before we start, we count zero; as we find things we count one, two, etc.

The natural numbers form an unbounded sequence

$$0, 1, 2, 3, 4, \ldots$$

Suppose P names a property. We write "$P(2)$" to mean "2 has property P", or "P holds for 2".

A statement "every natural number has property P" corresponds to a sequence of statements

$$P(0), P(1), P(2), P(3), P(4), \ldots$$

Mathematical Induction

Principle of mathematical induction:

Suppose we establish two things: that

- 0 has property P, and that
- whenever any number has property P, then the next number also has property P.

Then we may conclude that every natural number has property P.

Example: Show that $\sum_{x=0}^{n} x = \frac{n(n+1)}{2}$ for every natural number n.

Let P be the property; that is, let $P(n)$ be "$\sum_{x=0}^{n} x = \frac{n(n+1)}{2}$".

Proof for the example

Step 1 (basis): The property $P(0)$ is $\sum_{x=0}^{0} x = \frac{0(0+1)}{2}$.

The left side of the equation is just 0. Also the right side evaluates to 0. Thus 0 has property P.

Step 2 (inductive step): hypothesize that some number has property P; in other words, that

$$\sum_{x=0}^{\text{some number}} x = \frac{\text{some number (some number+1)}}{2}$$

Proof for the example

Step 1 (basis): The property $P(0)$ is $\sum_{x=0}^{0} x = \frac{0(0+1)}{2}$.

The left side of the equation is just 0. Also the right side evaluates to 0. Thus 0 has property P.

Step 2 (inductive step): hypothesize that some number has property P; in other words, that

$$\sum_{x=0}^{\text{some number}} x = \frac{\text{some number (some number}+1)}{2}$$

For simplicity, we give a name to the "some number". I choose k.

Thus the hypothesis becomes $\sum_{x=0}^{k} x = \frac{k(k+1)}{2}$.

Step 2 (inductive step), continued:

We hypothesize that k has property P; that is, $\sum_{x=0}^{k} x = \frac{k(k+1)}{2}$.

We need to demonstrate that $k+1$ has property P; that is,

$$\sum_{x=0}^{k+1} x = \frac{(k+1)((k+1)+1)}{2} = \frac{(k+1)(k+2)}{2} \quad .$$

We calculate:

$\sum_{x=0}^{k+1} x = \left(\sum_{x=0}^{k} x\right) + (k+1)$	definition of $\sum$
$= \frac{k(k+1)}{2} + (k+1)$	hypothesis
$= \left(\frac{k}{2} + 1\right)(k+1)$	"algebra"
$= \frac{(k+1)(k+2)}{2}$	DONE!

Step 2 (inductive step), continued:

We hypothesize that k has property P; that is, $\sum_{x=0}^{k} x = \frac{k(k+1)}{2}$.

We need to demonstrate that $k+1$ has property P; that is,

$$\sum_{x=0}^{k+1} x = \frac{(k+1)((k+1)+1)}{2} = \frac{(k+1)(k+2)}{2} \quad .$$

We calculate:

$$\begin{aligned}
\sum_{x=0}^{k+1} x &= \Big(\sum_{x=0}^{k} x\Big) + (k+1) && \text{definition of } \textstyle\sum \\
&= \tfrac{k(k+1)}{2} + (k+1) && \text{hypothesis} \\
&= \big(\tfrac{k}{2} + 1\big)(k+1) && \text{"algebra"} \\
&= \tfrac{(k+1)(k+2)}{2} && \text{DONE!}
\end{aligned}$$

Observations/Techniques

To talk about something, give it a name.
E.g., property P, number k, etc.

A formula is a textual object. In this text, we can substitute one symbol or expression for another. For example, we put "$k+1$" in place of "k".

The induction principle gives a "template" for a proof:

- The proof has two parts: the "basis" and the "inductive step".
- In the inductive step, hypothesize $P(k)$ and prove $P(k+1)$ from it.

But the induction principle does not say how to actually do either step. We must invent the method ourselves.

“Simple” Induction vs. “Strong” Induction

	Simple Induction	**Strong Induction** or **Course of Values**
Basis:	Show $P(0)$	Show $P(0)$
Ind. Hypothesis:	$P(k)$ holds	$P(m)$ holds, ***for every*** $m \leq k$
Ind. Step:	Show $P(k+1)$ holds	Show $P(k+1)$ holds
Conclusion:	$P(k)$ holds for every k	$P(k)$ holds for every k

What is the difference?

“Simple” Induction vs. “Strong” Induction

	Simple Induction	**Strong Induction** or **Course of Values**
Basis:	Show $P(0)$	Show $P(0)$
Ind. Hypothesis:	$P(k)$ holds	$P(m)$ holds, ***for every*** $m \leq k$
Ind. Step:	Show $P(k+1)$ holds	Show $P(k+1)$ holds
Conclusion:	$P(k)$ holds for every k	$P(k)$ holds for every k

What is the difference?

Define $Q(k)$ as the property “$P(m)$ holds, for every $m \leq k$”.

"Simple" Induction vs. "Strong" Induction

	Simple Induction	**Strong Induction** or **Course of Values**
Basis:	Show $P(0)$	Show $P(0)$
Ind. Hypothesis:	$P(k)$ holds	$P(m)$ holds, ***for every*** $m \leq k$
Ind. Step:	Show $P(k+1)$ holds	Show $P(k+1)$ holds
Conclusion:	$P(k)$ holds for every k	$P(k)$ holds for every k

What is the difference?

Define $Q(k)$ as the property "$P(m)$ holds, for every $m \leq k$".

- $Q(0)$ is equivalent to $P(0)$.

"Simple" Induction vs. "Strong" Induction

	Simple Induction	**Strong Induction** or **Course of Values**
Basis:	Show $Q(0)$	Show $P(0)$
Ind. Hypothesis:	$P(k)$ holds	$P(m)$ holds, ***for every*** $m \leq k$
Ind. Step:	Show $P(k+1)$ holds	Show $P(k+1)$ holds
Conclusion:	$P(k)$ holds for every k	$P(k)$ holds for every k

What is the difference?

Define $Q(k)$ as the property "$P(m)$ holds, for every $m \leq k$".

- $Q(0)$ is equivalent to $P(0)$.
- $Q(k+1)$ is equivalent to "$Q(k)$ and $P(k+1)$".
  To prove $Q(k+1)$ from $Q(k)$, need only to prove $P(k+1)$.

"Simple" Induction vs. "Strong" Induction

	Simple Induction	**Strong Induction** or **Course of Values**
Basis:	Show $Q(0)$	Show $P(0)$
Ind. Hypothesis:	$Q(k)$ holds	$P(m)$ holds, ***for every*** $m \leq k$
Ind. Step:	Show $Q(k+1)$ holds	Show $P(k+1)$ holds
Conclusion:	$P(k)$ holds for every k	$P(k)$ holds for every k

What is the difference?

Define $Q(k)$ as the property "$P(m)$ holds, for every $m \leq k$".

- $Q(0)$ is equivalent to $P(0)$.
- $Q(k+1)$ is equivalent to "$Q(k)$ and $P(k+1)$".
  To prove $Q(k+1)$ from $Q(k)$, need only to prove $P(k+1)$.
- "$Q(k)$ for every k" is equivalent to "$P(k)$ for every k".

"Simple" Induction vs. "Strong" Induction

	Simple Induction	**Strong Induction** or **Course of Values**
Basis:	Show $Q(0)$	Show $P(0)$
Ind. Hypothesis:	$Q(k)$ holds	$P(m)$ holds, ***for every*** $m \leq k$
Ind. Step:	Show $Q(k+1)$ holds	Show $P(k+1)$ holds
Conclusion:	$Q(k)$ holds for every k	$P(k)$ holds for every k

What is the difference? No difference!

Define $Q(k)$ as the property "$P(m)$ holds, for every $m \leq k$".

- $Q(0)$ is equivalent to $P(0)$.
- $Q(k+1)$ is equivalent to "$Q(k)$ and $P(k+1)$".
  To prove $Q(k+1)$ from $Q(k)$, need only to prove $P(k+1)$.
- "$Q(k)$ for every k" is equivalent to "$P(k)$ for every k".

Structural Induction

To prove: every formula has property P.

How to prove such a statement? Can we use induction?

A formula is not a natural number. . . .

Structural Induction

To prove: every formula has property P.

How to prove such a statement? Can we use induction?

A formula is not a natural number, but it suffices to prove any one of the following.

> For every natural number n, every formula with n or fewer symbols has property P.

OR

> For every natural number n, every formula with n or fewer connectives has property P.

OR

Structural Induction (2)

To prove: every formula has property P.

OR

For every natural number n, every formula whose parse tree has height n or less has property P.

OR

For every natural number n, every formula producible with n or fewer uses of the formation rules has property P.

In each of these formulations, the induction step requires showing that

If $P(\varphi)$ and $P(\eta)$, then $P\big((\neg\varphi)\big)$ and $P\big((\varphi \star \eta)\big)$.

Formulas φ and η have smaller n values than $(\neg\varphi)$ and $(\varphi \star \eta)$ do.

The principle of Structural Induction

Theorem. Let R be a property. Suppose that

1. for each atomic formula p, we have $R(p)$; and
2. for each formula φ, if $R(\varphi)$ then $R\big((\neg\varphi)\big)$; and
3. for each pair of formulas φ and η, and each connective $\star$, if $R(\varphi)$ and $R(\eta)$ then $R\big((\varphi \star \eta)\big)$.

Then $R(\varphi)$ for every formula φ.

Use of this principle is called *structural induction*.

Structural induction is a special case of mathematical induction.

Example: Parentheses in Formulas

To illustrate structural induction, we shall prove the following.

> *Lemma.* Every well-formed formula has an equal number of left and right parentheses.

Proof. We use structural induction. The property to prove is

> $R(\varphi)$: φ has an equal number of left and right parentheses

for every formula φ.

Base case: φ is an atom.

φ has no parentheses—only a propositional variable. Thus $R(\varphi)$ holds.

This completes the proof of the base case.

Example, cont'd — the Inductive Step

Inductive step:

Hypothesis: formulas φ and η both have property R.

To prove: each of the formulas $(\neg\varphi)$, $(\varphi \land \eta)$, $(\varphi \lor \eta)$, $(\varphi \to \eta)$ and $(\varphi \longleftrightarrow \eta)$ has property R.

Example, cont'd — the Inductive Step

Inductive step:

Hypothesis: formulas φ and η both have property R.

To prove: each of the formulas $(\neg\varphi)$, $(\varphi \wedge \eta)$, $(\varphi \vee \eta)$, $(\varphi \rightarrow \eta)$ and $(\varphi \longleftrightarrow \eta)$ has property R.

W.l.o.g, we consider $(\varphi \wedge \eta)$.

Notation: For any formula ζ, let $op(\zeta)$ denote the number of '(' in ζ, and let $cl(\zeta)$ denote the number of ')' in ζ.

We calculate $op\big((\varphi \wedge \eta)\big)$:

$$op\big((\varphi \wedge \eta)\big) = 1 + op(\varphi) + op(\eta) \qquad \text{inspection}$$

Example, cont'd — the Inductive Step

Inductive step:

Hypothesis: formulas φ and η both have property R.

To prove: each of the formulas $(\neg\varphi)$, $(\varphi \wedge \eta)$, $(\varphi \vee \eta)$, $(\varphi \to \eta)$ and $(\varphi \leftrightarrow \eta)$ has property R.

W.l.o.g, we consider $(\varphi \wedge \eta)$.

> Notation: For any formula ζ, let $op(\zeta)$ denote the number of '(' in ζ, and let $cl(\zeta)$ denote the number of ')' in ζ.

We calculate $op\big((\varphi \wedge \eta)\big)$:

$$\begin{aligned} op\big((\varphi \wedge \eta)\big) &= 1 + op(\varphi) + op(\eta) && \text{inspection} \\ &= 1 + cl(\varphi) + cl(\eta) && R(\varphi) \text{ and } R(\eta) \end{aligned}$$

Example, cont'd — the Inductive Step

Inductive step:

Hypothesis: formulas φ and η both have property R.

To prove: each of the formulas $(\neg\varphi)$, $(\varphi \wedge \eta)$, $(\varphi \vee \eta)$, $(\varphi \rightarrow \eta)$ and $(\varphi \longleftrightarrow \eta)$ has property R.

W.l.o.g, we consider $(\varphi \wedge \eta)$.

> Notation: For any formula ζ, let $op(\zeta)$ denote the number of '(' in ζ, and let $cl(\zeta)$ denote the number of ')' in ζ.

We calculate $op\big((\varphi \wedge \eta)\big)$:

$$\begin{aligned} op\big((\varphi \wedge \eta)\big) &= 1 + op(\varphi) + op(\eta) && \text{inspection} \\ &= 1 + cl(\varphi) + cl(\eta) && R(\varphi) \text{ and } R(\eta) \\ &= cl\big((\varphi \wedge \eta)\big) && \text{inspection.} \end{aligned}$$

Back to the Unique Readability Theorem

Theorem. Every formula is exactly one of an atom, $(\neg\varphi)$, $(\varphi \wedge \eta)$, $(\varphi \vee \eta)$, $(\varphi \to \eta)$ or $(\varphi \leftrightarrow \eta)$; and in each case it is of that form in exactly one way.

We want to prove this using structural induction. How will it go?

Back to the Unique Readability Theorem

Theorem. Every formula is exactly one of an atom, $(\neg\varphi)$, $(\varphi \wedge \eta)$, $(\varphi \vee \eta)$, $(\varphi \rightarrow \eta)$ or $(\varphi \leftrightarrow \eta)$; and in each case it is of that form in exactly one way.

We want to prove this using structural induction. How will it go?

The proof will consider formulas of the form $(\varphi \rightarrow \eta)$. One such is our previous example $((p \wedge q) \rightarrow r)$, which has $(p \wedge q)$ for φ and r for η.

Back to the Unique Readability Theorem

Theorem. Every formula is exactly one of an atom, $(\neg\varphi)$, $(\varphi \wedge \eta)$, $(\varphi \vee \eta)$, $(\varphi \to \eta)$ or $(\varphi \leftrightarrow \eta)$; and in each case it is of that form in exactly one way.

We want to prove this using structural induction. How will it go?

The proof will consider formulas of the form $(\varphi \to \eta)$. One such is our previous example $((p \wedge q) \to r)$, which has $(p \wedge q)$ for φ and r for η.

Is this the only way to write the formula $((p \wedge q) \to r)$?

Back to the Unique Readability Theorem

Theorem. Every formula is exactly one of an atom, $(\neg\varphi)$, $(\varphi \wedge \eta)$, $(\varphi \vee \eta)$, $(\varphi \to \eta)$ or $(\varphi \leftrightarrow \eta)$; and in each case it is of that form in exactly one way.

We want to prove this using structural induction. How will it go?

The proof will consider formulas of the form $(\varphi \to \eta)$. One such is our previous example $((p \wedge q) \to r)$, which has $(p \wedge q)$ for φ and r for η.

Is this the only way to write the formula $((p \wedge q) \to r)$? What about

$$((p \wedge q) \to r) = (\varphi' \wedge \eta') \quad ,$$

where φ' is the expression '$(p$' and η' is the expression '$q) \to r$'?

Back to the Unique Readability Theorem

Theorem. Every formula is exactly one of an atom, $(\neg\varphi)$, $(\varphi \wedge \eta)$, $(\varphi \vee \eta)$, $(\varphi \to \eta)$ or $(\varphi \leftrightarrow \eta)$; and in each case it is of that form in exactly one way.

We want to prove this using structural induction. How will it go?

The proof will consider formulas of the form $(\varphi \to \eta)$. One such is our previous example $((p \wedge q) \to r)$, which has $(p \wedge q)$ for φ and r for η.

Is this the only way to write the formula $((p \wedge q) \to r)$? What about

$$((p \wedge q) \to r) = (\varphi' \wedge \eta') \ ,$$

where φ' is the expression '$(p$' and η' is the expression '$q) \to r$'?

Fortunately, neither φ' nor η' is a formula. (Why?)

Does It Always Work?

The theorem worked out for one example.

How can we make sure the inductive step works for *every* formula?
That is, if $\varphi' \wedge \eta'$ is the same expression as $\varphi \to \eta$, how can we argue that neither φ' nor η' can be a formula?
(Unless, of course, $\varphi = \varphi'$ and $\eta = \eta'$.)

Can φ' (or η') have an equal number of left and right parentheses?
If not, why not?

Does It Always Work?

The theorem worked out for one example.

How can we make sure the inductive step works for *every* formula?
That is, if $\varphi' \wedge \eta'$ is the same expression as $\varphi \to \eta$, how can we argue that neither φ' nor η' can be a formula?
(Unless, of course, $\varphi = \varphi'$ and $\eta = \eta'$.)

Can φ' (or η') have an equal number of left and right parentheses?
If not, why not?

To do the proof, we actually need to know more about formulas.

This illustrates a common feature of inductive proofs: they often prove more than just the statement given in the theorem.

Proving Unique Readability

Property $P(n)$:

Every formula φ containing at most n connectives satisfies all three of the following.

A: The first symbol of φ is either '(' or a variable.

B: φ has an equal number of '(' and ')', and each proper prefix of φ has more '(' than ')'.

C: φ has a unique construction as a formula.

(A *proper prefix* of φ is a non-empty expression x such that φ is xy for some non-empty expression y.)

We prove property P for all n by induction.

A Key Case of the Inductive Step

Inductive hypothesis: $P(k)$ holds for some natural number k.

To show $P(k+1)$, let formula φ have $k+1$ connectives.

A key case: φ is $(\eta \star \zeta)$. For property C, we must show

If φ is $(\eta' \star' \zeta')$ for **formulas** η' and ζ', then $\eta = \eta'$, $\star = \star'$ and $\zeta = \zeta'$.

If η' has the same length as η, then they must be the same string (both start at the second symbol of φ).

Otherwise, either η' is a proper prefix of η or η is a proper prefix of η'.
But since η and η' are formulas with at most k connectives, the inductive hypothesis applies to them.
In particular, each has property B, and thus neither can be a proper prefix of the other.

Thus φ has a unique derivation, as required by property C.

Commentary

The "goal" of the proof is property C—unique formation.

However, properties A and B are required in order to prove C.

There are actually two equally good options for a proof:

- Prove A, B and C simultaneously, as a single "compound" property. (As done here.)
- Prove them separately: first A, then B, and finally C. (The text uses this method.)

Two fundamental techniques:

- If a proof doesn't work, go back and fix it— as often as necessary.
- Start from the end and work backwards.

Two consequences of unique formation

We will define the semantics (meaning) of a formula from its syntax. Thus unique formation ensures unambiguous formulas.

Given a formula, determine its sub-formulas by counting parentheses.

$$\begin{array}{ccccc} 1 & 2 \qquad \dots \qquad m & m+1 & m+2 \;\dots\; n-1 & n \\ (& \underbrace{(\langle\texttt{rest of subformula}\rangle)}_{\textit{To determine } m\textit{: count excess of '(' over ')'}} & \star & \langle\texttt{subformula 2}\rangle &) \end{array}$$

When the count returns to zero, the subformula has ended.

(For efficient parsing of more-complicated formulas/programs, see CS 241.)

Working with Formulas

Evaluating formulas

Recall that propositional logic is *compositional*. The value of two subformulas, determines the value of their composition using a propositional connective. Given a valuation t:

$$p^t = t(p)$$

$$(\neg\alpha)^t = \begin{cases} \mathrm{T} & \text{if } \alpha^t = \mathrm{F} \\ \mathrm{F} & \text{if } \alpha^t = \mathrm{T} \end{cases}$$

$$(\alpha \wedge \beta)^t = \begin{cases} \mathrm{T} & \text{if } \alpha^t = \beta^t = \mathrm{T} \\ \mathrm{F} & \text{otherwise} \end{cases}$$

$$(\alpha \vee \beta)^t = \begin{cases} \mathrm{T} & \text{if } \alpha^t = \mathrm{T} \text{ or } \beta^t = \mathrm{T} \\ \mathrm{F} & \text{otherwise} \end{cases}$$

$$(\alpha \rightarrow \beta)^t = \begin{cases} \mathrm{T} & \text{if } \alpha^t = \mathrm{F} \text{ or } \beta^t = \mathrm{T} \\ \mathrm{F} & \text{otherwise} \end{cases}$$

$$(\alpha \leftrightarrow \beta)^t = \begin{cases} \mathrm{T} & \text{if } \alpha^t = \beta^t \\ \mathrm{F} & \text{otherwise} \end{cases}$$

Using these rules, we can build a *truth table* considering all combinations.

For a formula with n variables, the full truth table has 2^n lines.

Example: evaluating a formula

Example. Consider $((p \vee q) \rightarrow (q \wedge r))$.

It has the following truth table.

p	q	r	$(p \vee q)$	$(q \wedge r)$	$((p \vee q) \rightarrow (q \wedge r))$
F	F	F	F	F	T
F	F	T	F	F	T
F	T	F	T	F	F
F	T	T	T	T	T
T	F	F	T	F	F
T	F	T	T	F	F
T	T	F	T	F	F
T	T	T	T	T	T

Exercise

Build the truth table of $\big((p \to \neg q) \to (q \vee \neg p)\big)$.

Tautology, Satisfaction, Contradiction

Definition.

> A formula α is a *tautology* if and only if for every truth valuation t, $\alpha^t = \mathtt{T}$.
>
> A formula α is a *contradiction* if and only if for every truth valuation t, $\alpha^t = \mathtt{F}$.
>
> A formula α is *satisfiable* if and only if there is some truth valuation t such that $\alpha^t = \mathtt{T}$.

Note: a formula is satisfiable if and only if it is not a contradiction.

Example. The formula $p \wedge \neg p$ is a contradiction.

Example. The formula $(((p \wedge q) \to r) \wedge (p \to q)) \to (p \to r)$ is satisfiable. (Consider setting each variable to $\mathtt{T}$.)

A Tautology Example

Example. Is $(((p \wedge q) \rightarrow r) \wedge (p \rightarrow q)) \rightarrow (p \rightarrow r)$ a tautology?

One method: Fill out a truth table.
Every line will end up with `T` in the final column.

For larger formulas, with more variables, this can take a long time.

Can we do better? Is there some other method?

“Short-Cutting” a Truth Table

Rather than fill out an entire truth table, we can analyze what would happen if we did.

Let A be a truth value; that is, $A \in \{\mathrm{F}, \mathrm{T}\}$. We can combine it with other truth values as follows.

$\neg\mathrm{T}$	F
$\neg\mathrm{F}$	T

$A \wedge \mathrm{T}$	A
$A \wedge \mathrm{F}$	F
$\mathrm{T} \wedge A$	A
$\mathrm{F} \wedge A$	F
$A \wedge A$	A

$A \vee \mathrm{T}$	T
$A \vee \mathrm{F}$	A
$\mathrm{T} \vee A$	T
$\mathrm{F} \vee A$	A
$A \vee A$	A

$A \rightarrow \mathrm{T}$	T
$A \rightarrow \mathrm{F}$	$\neg A$
$\mathrm{T} \rightarrow A$	A
$\mathrm{F} \rightarrow A$	T
$A \rightarrow A$	T

$A \leftrightarrow \mathrm{T}$	A
$A \leftrightarrow \mathrm{F}$	$\neg A$
$\mathrm{T} \leftrightarrow A$	A
$\mathrm{F} \leftrightarrow A$	$\neg A$
$A \leftrightarrow A$	T

We can use these rules to evaluate a formula, by using a *valuation tree*. A valuation tree may sometimes be much smaller than the corresponding truth table.

Example: Valuation trees

Example. Show that $\big((p \wedge q) \to r) \wedge (p \to q)\big) \to (p \to r)$ is a tautology.

Example: Valuation trees

Example. Show that $\big((p \wedge q) \to r) \wedge (p \to q)\big) \to (p \to r)$ is a tautology.

In valuations with $t(p) = \mathtt{T}$, we put $\mathtt{T}$ in for p:

$$\Big(\big((\mathtt{T} \wedge q) \to r\big) \wedge (\mathtt{T} \to q)\Big) \to (\mathtt{T} \to r) \ .$$

From the previous table, this becomes $\big((q \to r) \wedge q\big) \to r$.

Example: Valuation trees

Example. Show that $\big((p \wedge q) \to r) \wedge (p \to q)\big) \to (p \to r)$ is a tautology.

In valuations with $t(p) = \mathtt{T}$, we put $\mathtt{T}$ in for p:

$$\Big(\big((\mathtt{T} \wedge q) \to r\big) \wedge (\mathtt{T} \to q)\Big) \to (\mathtt{T} \to r) \ .$$

From the previous table, this becomes $\big((q \to r) \wedge q\big) \to r$.

If $t(q) = \mathtt{T}$, this yields $r \to r$ and then $\mathtt{T}$. (Check!).
If $t(q) = \mathtt{F}$, it yields $\mathtt{F} \to r$ and then $\mathtt{T}$. (Check!).

Example: Valuation trees

Example. Show that $\big((p \wedge q) \to r) \wedge (p \to q)\big) \to (p \to r)$ is a tautology.

In valuations with $t(p) = \mathtt{T}$, we put $\mathtt{T}$ in for p:

$$\big(((\mathtt{T} \wedge q) \to r\big) \wedge (\mathtt{T} \to q)\big) \to (\mathtt{T} \to r) \ .$$

From the previous table, this becomes $\big((q \to r) \wedge q\big) \to r$.

If $t(q) = \mathtt{T}$, this yields $r \to r$ and then $\mathtt{T}$. (Check!).
If $t(q) = \mathtt{F}$, it yields $\mathtt{F} \to r$ and then $\mathtt{T}$. (Check!).

On the other hand, in valuations with $t(p) = \mathtt{F}$, we get

$$\big(((\mathtt{F} \wedge q) \to r) \wedge (\mathtt{F} \to q)\big) \to (\mathtt{F} \to r) \ ,$$

Simplification yields $\big((\mathtt{F} \to r) \wedge \mathtt{T}\big) \to \mathtt{T}$ and eventually $\mathtt{T}$.

Thus every valuation makes the formula true, as required.

Equivalence of Formulas

Suppose that a formula $\alpha \leftrightarrow \beta$ is a tautology.

Then α and β must have the same final column in their truth tables—they have the same value under any valuation.

In symbols: $\alpha^t = \beta^t$, for every valuation t.

Such formulas are called *equivalent* formulas. We use the notation

$$\alpha \equiv \beta$$

to mean that α and β are equivalent.

Equivalent is Equivalent

Equivalent formulas are equivalent in any context.

Lemma. Suppose that $\alpha \equiv \beta$. Then for any formula γ, and any connective $\star$, the formulas $\alpha \star \gamma$ and $\beta \star \gamma$ are equivalent:

$$\alpha \star \gamma \equiv \beta \star \gamma \quad .$$

Proof idea: a value $(\alpha \star \gamma)^t$ depends only on the values α^t and γ^t, and the identity of $\star$.

Example: Since $(\neg p \to p) \equiv p$ [check this!], we get that $\big((\neg p \to p) \wedge q\big) \equiv (p \wedge q)$.

Algebra of Formulas

Many equivalences of formulas look much like rules of ordinary arithmetic and/or algebra.

Commutativity

$$\alpha \wedge \beta \equiv \beta \wedge \alpha$$
$$\alpha \vee \beta \equiv \beta \vee \alpha$$
$$\alpha \leftrightarrow \beta \equiv \beta \leftrightarrow \alpha$$

Associativity

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

Distributivity

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

Idempotence

$$\alpha \vee \alpha \equiv \alpha$$
$$\alpha \wedge \alpha \equiv \alpha$$

Double Negation

$$\neg(\neg\alpha) \equiv \alpha$$

De Morgan's Laws

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

Algebra of Formulas, cont'd

Simplification I (Absorbtion)

$$\alpha \land \mathtt{T} \equiv \alpha$$
$$\alpha \lor \mathtt{T} \equiv \mathtt{T}$$
$$\alpha \land \mathtt{F} \equiv \mathtt{F}$$
$$\alpha \lor \mathtt{F} \equiv \alpha$$

Simplification II

$$\alpha \lor (\alpha \land \beta) \equiv \alpha$$
$$\alpha \land (\alpha \lor \beta) \equiv \alpha$$

Implication

$$\alpha \to \beta \equiv \neg\alpha \lor \beta$$

Contrapositive

$$\alpha \to \beta \equiv \neg\beta \to \neg\alpha$$

Equivalence

$$\alpha \leftrightarrow \beta \equiv (\alpha \to \beta) \land (\beta \to \alpha)$$

Excluded Middle

$$\alpha \lor \neg\alpha \equiv \mathtt{T}$$

Contradiction

$$\alpha \land \neg\alpha \equiv \mathtt{F}$$

("T" and "F" aren't really formulas, but we'll pretend.)

Examples of Using Identities

Prove or disprove each of the following.

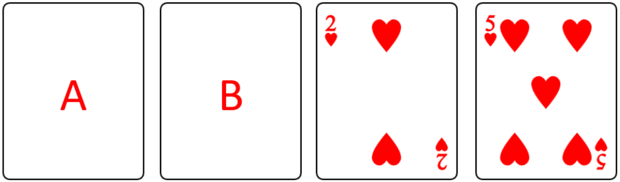
1. $(p \wedge q) \vee (q \wedge r) \equiv q \wedge (p \vee r)$
2. $(p \vee r) \wedge (q \vee s) \equiv (p \wedge q) \vee (p \wedge s) \vee (r \wedge q) \vee (r \wedge s)$
3. $p \vee (p \wedge q) \equiv p$ *(without using Simplification II)*
4. $\neg(\neg p \vee \neg(r \vee s)) \equiv (p \wedge r) \vee (p \wedge s)$
5. $\neg(\neg(p \wedge q) \vee p) \equiv \mathtt{F}$
6. $p \equiv p \wedge (q \to p)$
7. $p \equiv p \wedge (\neg(\neg q \wedge \neg p) \vee p)$
8. $p \wedge (\neg(\neg q \wedge \neg p) \vee p) \equiv q$

Note: Apply only one rule per line of your proof (but you may apply the rule multiple times).

"Application": Cards

We are given a deck of cards with a letter of the alphabet on one side and a natural number on the other side.

Claim: Each card that has a vowel on one side, has an even number on the other side.



Is the claim true?

How many cards must you turn over, in order to determine whether or not the claim is true? Which ones?

Satisfiability of Sets of Formulas

The notion of satisfiability extends to sets of formulas.

Let Σ denote a set of formulas and t a valuation. Define

$$\Sigma^t = \begin{cases} \mathtt{T} & \text{if for each } \beta \in \Sigma, \beta^t = \mathtt{T} \\ \mathtt{F} & \text{otherwise} \end{cases}$$

When $\Sigma^t = \mathtt{T}$, we say that t *satisfies* Σ.

A set Σ is *satisfiable* iff there is some valuation t such that $\Sigma^t = \mathtt{T}$.

Example. The set $\{((p \to q) \vee r), (p \vee q \vee s)\}$ is satisfiable.

Logical Consequence, a.k.a. Entailment

Let Σ be a set of formulas, and let α be a formula. We say that

- α is a *logical consequence* of Σ, or
- Σ *(semantically) entails* α, or
- in symbols, $\Sigma \models \alpha$,

if and only if for any truth valuation t,

$$\text{if } \Sigma^t = \mathtt{T} \text{ then also } \alpha^t = \mathtt{T}.$$

We write $\Sigma \not\models \alpha$ for "not $\Sigma \models \alpha$". That is,
there exists a truth valuation t such that $\Sigma^t = \mathtt{T}$ and $\alpha^t = \mathtt{F}$.

Examples: Entailment

Example.

$$\{(p \to q), (q \to r)\} \models (p \to r) \quad .$$

Examples: Entailment

Example.

$$\{(p \rightarrow q),\ (q \rightarrow r)\} \models (p \rightarrow r) \ .$$

Example.

$$\big\{((p \rightarrow \neg q) \vee r),\ (q \wedge \neg r),\ (p \leftrightarrow r)\big\} \not\models (p \wedge (q \rightarrow r)) \ .$$

Examples: Entailment

Example.

$$\{(p \to q),\ (q \to r)\} \models (p \to r)\ .$$

Example.

$$\big\{((p \to \neg q) \vee r),\ (q \wedge \neg r),\ (p \leftrightarrow r)\big\} \not\models (p \wedge (q \to r))\ .$$

Example. $\emptyset \models \alpha$ means that α is a tautology. Why?

Examples: Entailment

Example.

$$\{(p \rightarrow q),\ (q \rightarrow r)\} \models (p \rightarrow r)\ .$$

Example.

$$\Big\{((p \rightarrow \neg q) \vee r),\ (q \wedge \neg r),\ (p \leftrightarrow r)\Big\} \not\models (p \wedge (q \rightarrow r))\ .$$

Example. $\emptyset \models \alpha$ means that α is a tautology. Why?

Example. $\{\alpha, \neg\alpha\} \models \beta$ is always true, whatever α and β are. Why?

Equivalence and Entailment

Equivalence can be expressed using the notion of entailment.

Lemma. $\alpha \equiv \beta$ if and only if both $\{\alpha\} \models \beta$ and $\{\beta\} \models \alpha$.

A Code Example

```
if ( (input > 0) OR NOT output ) {
    if ( NOT (output AND (queuelength < 100) ) ) {
        $P_1$
    } else if ( output AND NOT (queuelength < 100) ) {
        $P_2$
    } else { $P_3$ }
} else { $P_4$ }
```

When does each piece of code get executed?

Let i: `input > 0`,
u: `output`,
q: `queuelength < 100`.

A Code Example, cont'd

```
if ( $i \lor \neg u$ ) {
    if ( $\neg(u \land q)$ ) {
        $P_1$
    }
    else if ( $u \land \neg q$ ) {
        $P_2$
    }
    else {
        $P_3$
    }
} else {
    $P_4$
}
```

i	u	q	$i \lor \neg u$	$\neg(u \land q)$	$u \land \neg q$	Action
T	T	T	T			
T	T	F	T			
T	F	T	T			
T	F	F	T			
F	T	T	F			P_4
F	T	F	F			P_4
F	F	T	T			
F	F	F	T			

A Code Example, cont'd

```
if ( $i \lor \neg u$ ) {
    if ( $\neg(u \land q)$ ) {
        $P_1$
    }
    else if ( $u \land \neg q$ ) {
        $P_2$
    }
    else {
        $P_3$
    }
} else {
    $P_4$
}
```

i	u	q	$i \lor \neg u$	$\neg(u \land q)$	$u \land \neg q$	Action
T	T	T	T	F	F	P_3
T	T	F	T	T		P_1
T	F	T	T	T		P_1
T	F	F	T	T		P_1
F	T	T	F			P_4
F	T	F	F			P_4
F	F	T	T	T		P_1
F	F	F	T	T		P_1

A Code Example, cont'd

```
if ( $i \vee \neg u$ ) {
    if ( $\neg(u \wedge q)$ ) {
        $P_1$
    }
    else if ( $u \wedge \neg q$ ) {
        $P_2$
    }
    else {
        $P_3$
    }
} else {
    $P_4$
}
```

i	u	q	$i \vee \neg u$	$\neg(u \wedge q)$	$u \wedge \neg q$	Action
T	T	T	T	F	F	P_3
T	T	F	T	T		P_1
T	F	T	T	T		P_1
T	F	F	T	T		P_1
F	T	T	F			P_4
F	T	F	F			P_4
F	F	T	T	T		P_1
F	F	F	T	T		P_1

P_2 is never executed.

Finding Dead Code

Prove that P_2 is dead code.

The conditions leading to P_2 can never be true.

$$
\begin{array}{lll}
\multicolumn{3}{l}{(i \vee \neg u) \wedge \neg\neg(u \wedge q) \wedge (u \wedge \neg q)} \\
\equiv & (i \vee \neg u) \wedge (u \wedge q) \wedge (u \wedge \neg q) & \text{Negation} \\
\equiv & (i \vee \neg u) \wedge (u \wedge q) \wedge (\neg q \wedge u) & \text{Commutativity} \\
\equiv & (i \vee \neg u) \wedge u \wedge (q \wedge \neg q) \wedge u & \text{Associativity} \\
\equiv & (i \vee \neg u) \wedge u \wedge \mathtt{F} \wedge u & \text{Contradiction} \\
\equiv & \mathtt{F} & \text{Simplification I}
\end{array}
$$

Finding Live Code

Prove that P_3 is live code.

The conditions leading to P_3 can be true.

P_3 is executed when the formula

$$(i \vee \neg u) \wedge \neg\neg(u \wedge q) \wedge \neg(u \wedge \neg q)$$

is true.

Find a satisfying truth valuation for this formula.

For example: $t(i) = \mathtt{T}, t(u) = \mathtt{T}, t(q) = \mathtt{T}$.

Simplifying Code

Consider these two fragments of code. Are they equivalent?

Fragment 1:

```
if ( $i \lor \neg u$ ) {
    if ( $\neg(u \land q)$ ) {
        $P_1$
    }
    else if ( $u \land \neg q$ ) {
        $P_2$
    }
    else {
    $P_3$
    }
}
else {
$P_4$
}
```

Fragment 2:

```
if ( $i \land u \land q$ ) {
    $P_3$
}
else if ( $\neg i \land u$ ) {
    $P_4$
}
else {
    $P_1$
}
```

Simplifying Code

To prove that the two fragments are equivalent, show that each block of code P_1, P_2, P_3, and P_4 is executed under equivalent conditions.

Block	Fragment 1	Fragment 2
P_1	$(i \vee \neg u) \wedge \neg(u \wedge q)$	$\neg(i \wedge u \wedge q) \wedge \neg(\neg i \wedge u)$
P_2	$(i \vee \neg u) \wedge \neg\neg(u \wedge q) \wedge (u \wedge \neg q)$	F
P_3	$(i \vee \neg u) \wedge \neg\neg(u \wedge q) \wedge \neg(u \wedge \neg q)$	$(i \wedge u \wedge q)$
P_4	$\neg(i \vee \neg u)$	$\neg(i \wedge u \wedge q) \wedge (\neg i \wedge u)$

Definability of Connectives

Formulas $\alpha \to \beta$ and $\neg\alpha \vee \beta$ are equivalent.

Thus $\to$ is said to be *definable* in terms of $\neg$ and $\vee$.

There are actually sixteen possible binary connectives. (Why?)

Of these, two are essentially nullary (they ignore the values they connect).

Four others are essentially unary (they ignore one value but not the other).

Adequate Sets

A set of connectives is said to be *adequate* iff any n-ary $(n \geq 1)$ connective can be defined in terms of the ones in the set.

Lemma. $\{\wedge, \vee, \neg\}$ is an adequate set of connectives.

Proof: see the equivalence rules "Implication" and "Equivalence".

Lemma. Each of the sets $\{\wedge, \neg\}$, $\{\vee, \neg\}$, and $\{\rightarrow, \neg\}$ is adequate.

Proof: For the first two, use De Morgan's laws. For the third, $\ldots$?

Theorem. The set $\{\wedge, \vee\}$ is *not* an adequate set of connectives.

Some Questions about Adequate Sets

Question: Is there a binary connective $*$ such that the singleton set $\{*\}$ is adequate?

Question: Are there binary connectives c_1, c_2 and c_3 such that $\{c_1, c_2, c_3\}$ is adequate, but none of $\{c_1, c_2\}$, $\{c_1, c_3\}$, or $\{c_2, c_3\}$ is adequate?
(Such a set is called a *minimal* adequate set.)

Question: Find all minimal adequate sets containing only binary, unary and nullary connectives.

Proofs in Propositional Logic: Resolution

What Is a “Proof”?

A *proof* is a formal demonstration that a statement is true.

- It must be mechanically checkable. A reader need not apply any intuition or insight to verify that it is correct.
- In fact, a computer could verify its correctness.

A proof is generally syntactic, rather than semantic.

- Syntactic rules permit mechanical checking.
- The rules are chosen for semantic reasons, but their use remains purely syntactic.

What Makes a Proof?

Generically, a proof consists of a list of formulas.

- The assumptions, if any, are listed first.
- Each subsequent formula must be a valid *inference* from preceding formulas.
  That is, there is an *inference rule* (defined by the proof system) that justifies the formula, based on the previous ones.
- The final formula is the conclusion.

The key here is the set of inference rules. A set of inference rules defines a *proof system*.

We notate "there is a proof with assumptions Σ and conclusion φ" by

$$\Sigma \vdash \varphi \quad .$$

Inference Rules

In general, an inference rule is written as

$$\frac{\alpha_1 \quad \alpha_2 \quad \dots \quad \alpha_i}{\beta} \quad .$$

This means,

> Suppose that each of the formulas α_1, α_2, …, α_i already appears in the proof (either assumed or previously inferred).
>
> Then one may infer the formula β.

Examples of possible rules:

$\dfrac{\alpha \quad \beta}{\alpha \wedge \beta}$ A kind of definition of $\wedge$. $\dfrac{\alpha \wedge \beta}{\alpha \vee \beta}$ Rules need not be equivalences.

Approaches to Proofs

Direct proofs:

To establish $\Sigma \models \varphi$, give a proof with $\alpha_1, \alpha_2, \ldots, \alpha_n$ as assumptions, and obtain φ as the conclusion.

Refutations (a.k.a. indirect proofs, or proofs by contradiction):

To establish $\Sigma \models \varphi$, take $\neg\varphi$ as an assumption, in addition to $\alpha_1, \alpha_2, \ldots, \alpha_n$. Obtain a definitive contradiction (denoted $\perp$) as a conclusion.
(In other words, give a direct proof of $\Sigma \cup \{\neg\varphi\} \models \perp$.)

Why does the refutation approach work?

If $\Sigma \cup \{\neg\varphi\}$ is a contradiction, then any valuation t that makes Σ true must make $\neg\varphi$ false and thus make φ true. Therefore, $\Sigma \models \varphi$.

Proofs and Entailment

We have outlined the following plan.

Goal: Show that $\Sigma \models \varphi$.
Method: Show that $\Sigma \vdash \varphi$ (i.e., give a proof).

To justify this, we need that

$$\Sigma \vdash \varphi \text{ implies } \Sigma \models \varphi.$$

Of course, this depends on what the proof system is!

The “Resolution” System and Rule

Resolution is a refutation system, with the following inference rule:

$$\frac{\alpha \vee p \qquad \neg p \vee \beta}{\alpha \vee \beta}$$

for any variable p and formulas α and β.

We consider the following as special cases:

Unit resolution:

$$\frac{\alpha \vee p \qquad \neg p}{\alpha}$$

Contradiction:

$$\frac{p \qquad \neg p}{\bot}$$

Resolution is a refutation system; a proof is complete when one derives a contradiction $\bot$.

In this case, the original assumptions are refuted.

Example of Using Resolution

To prove: $\{p,\ q\} \vdash_{Res} p \wedge q$.

Our aim: derive a contradiction from the assumptions $\{p,\ q,\ \neg(p \wedge q)\}$.

As a preliminary step, re-write the third formula as $\neg p \vee \neg q$.

We start the actual proof with the three assumptions.

1.	p	assumption
2.	q	assumption
3.	$\neg p \vee \neg q$	assumption (from negated goal)

Now, we recall the inference rule: $\dfrac{\alpha \vee p \quad \neg p \vee \beta}{\alpha \vee \beta}$.

Consider lines 1 and 3. . . .

Example of Using Resolution, cont'd

The proof so far:

1.	p	assumption
2.	q	assumption
3.	$\neg p \vee \neg q$	assumption (from negated goal)

We have the formulas (1) p and (3) $\neg p \vee \neg q$.
Apply unit resolution, yielding the formula $\neg q$.

Example of Using Resolution, cont'd

1.	p	assumption
2.	q	assumption
3.	$\neg p \vee \neg q$	assumption (from negated goal)
4.	$\neg q$	1, 3

We have the formulas (1) p and (3) $\neg p \vee \neg q$.
Apply unit resolution, yielding the formula $\neg q$.

We have the formulas (2) q and (4) $\neg q$.
Apply the contradiction rule, yielding $\perp$.

Example of Using Resolution, cont'd

1.	p	assumption
2.	q	assumption
3.	$\neg p \vee \neg q$	assumption (from negated goal)
4.	$\neg q$	1, 3
5.	$\bot$	2, 4

We have the formulas (1) p and (3) $\neg p \vee \neg q$.
Apply unit resolution, yielding the formula $\neg q$.

We have the formulas (2) q and (4) $\neg q$.
Apply the contradiction rule, yielding $\bot$.

Done!

Conjunctive Normal Form

The Resolution rule can only be used successfully on formulas of a restricted form.

Conjunctive normal form (CNF):

- A *literal* is a (propositional) variable or the negation of a variable.
- A *clause* is a disjunction of literals.
- A formula is in *conjunctive normal form* if it is a conjunction of clauses.

In other words, a formula is in CNF if and only if

- its only connectives are $\neg$, $\vee$ and/or $\wedge$,
- $\neg$ applies only to variables, and
- $\vee$ applies only to subformulas with no occurrence of $\wedge$.

Converting to CNF

1. Eliminate implication and equivalence.
   Replace $(\alpha \to \beta)$ by $(\neg\alpha \lor \beta)$
   Replace $(\alpha \leftrightarrow \beta)$ by $(\neg\alpha \lor \beta) \land (\alpha \lor \neg\beta)$.
   (*Now only* $\land$, $\lor$ *and* $\neg$ *appear as connectives.*)
2. Apply De Morgan's and double-negation laws as often as possible.
   Replace $\neg(\alpha \lor \beta)$ by $\neg\alpha \land \neg\beta$.
   Replace $\neg(\alpha \land \beta)$ by $\neg\alpha \lor \neg\beta$.
   Replace $\neg\neg\alpha$ by α.
   (*Now negation only occurs in literals.*)
3. Transform into a conjunction of clauses using distributivity.
   Replace $\big(\alpha \lor (\beta \land \gamma)\big)$ by $\big((\alpha \lor \beta) \land (\alpha \lor \gamma)\big)$.
   (*One could stop here, but. . . .*)
4. Simplify using idempotence, contradiction, excluded middle and Simplification I & II.

The Resolution Proof Procedure

To prove φ from Σ, via a Resolution refutation:

1. Convert each formula in Σ to CNF.
2. Convert $\neg\varphi$ to CNF.
3. Split the CNF formulas at the $\wedge$s, yielding a set of clauses.
4. From the resulting set of clauses, keep applying the resolution inference rule until either:
   - The empty clause $\perp$ results.
     In this case, φ is a theorem.
   - The rule can no longer be applied to give a new formula.
     In this case, φ is not a theorem.

Example: Resolution

To show: $\{(p \to q), (q \to r)\} \models (p \to r)$.

Convert each assumption formula to CNF.

We get $(\neg p \vee q)$ and $(\neg q \vee r)$.

Convert the **negation** of the goal formula to CNF:

Replacing the $\to$ yields $\neg(\neg p \vee r)$; then
De Morgan yields $(p \wedge \neg r)$.

Splitting the $\wedge$ yields four clauses: $(\neg p \vee q)$, $(\neg q \vee r)$, p and $\neg r$.

Example, cont'd

Now we can make inferences, starting from our assumptions.

1.	$\neg p \vee q$	assumption
2.	$\neg q \vee r$	assumption
3.	p	assumption (from negated conclusion)
4.	$\neg r$	assumption (from negated conclusion)

Example, cont'd

Now we can make inferences, starting from our assumptions.

1.	$\neg p \vee q$	assumption
2.	$\neg q \vee r$	assumption
3.	p	assumption (from negated conclusion)
4.	$\neg r$	assumption (from negated conclusion)
5.	q	1, 3 (variable p)

Example, cont'd

Now we can make inferences, starting from our assumptions.

1.	$\neg p \vee q$	assumption
2.	$\neg q \vee r$	assumption
3.	p	assumption (from negated conclusion)
4.	$\neg r$	assumption (from negated conclusion)
5.	q	1, 3 (variable p)
6.	r	2, 5 (variable q)

Example, cont'd

Now we can make inferences, starting from our assumptions.

1.	$\neg p \vee q$	assumption
2.	$\neg q \vee r$	assumption
3.	p	assumption (from negated conclusion)
4.	$\neg r$	assumption (from negated conclusion)
5.	q	1, 3 (variable p)
6.	r	2, 5 (variable q)
7.	$\bot$	4, 6 (variable r)

Refutation complete!

Thinking About Consistency

Suppose I have a set of sentences Σ in propositional logic and an additional sentence φ.

In each of the following cases, what can I conclude?

- If $\Sigma \wedge \neg\varphi$ is consistent, then ...
- If $\Sigma \wedge \neg\varphi$ is inconsistent, then ...
- If $\Sigma \wedge \varphi$ is consistent, then...
- If $\Sigma \wedge \varphi$ is inconsistent, then ...

Resolution Is Sound

For resolution to be meaningful, we need the following.

Theorem. Suppose that $\{\alpha_1, \ldots, \alpha_n\} \vdash_{Res} \bot$; that is, there is a resolution refutation with assumptions $\alpha_1, \ldots, \alpha_n$ and conclusion $\bot$. Then the set $\{\alpha_1, \ldots, \alpha_n\}$ is unsatisfiable (contradictory).

That is, if $\Sigma \cup \{\neg\varphi\} \vdash_{Res} \bot$, then $\Sigma \cup \{\neg\varphi\}$ is a contradiction. Therefore, $\Sigma \models \varphi$.

In other words, the Resolution proof system is sound.
(If we prove something, it is true.)

We prove the theorem by induction on the length of the refutation.

Soundness: The central argument

Claim: Suppose that a set $\Gamma = \{\beta_1, \ldots, \beta_k\}$ is satisfiable. Let β_{k+1} be a formula obtained from Γ by one use of the resolution inference rule. Then the set $\Gamma \cup \{\beta_{k+1}\}$ is satisfiable.

Proof: Let valuation v satisfy Γ; that is, $\beta_i^v = \mathtt{T}$ for each i.

Let β_{k+1} be $\gamma_1 \vee \gamma_2$, obtained by resolving $\beta_i = p \vee \gamma_1$ and $\beta_j = \neg p \vee \gamma_2$.

Case I: $v(p) = \mathtt{F}$. Since $\beta_i^v = \mathtt{T}$, we must have $\gamma_1^v = \mathtt{T}$. Thus $\beta_{k+1}^v = \mathtt{T}$.
Case II: $v(p) = \mathtt{T}$. Since $\beta_j^v = \mathtt{T}$, we must have $\gamma_2^v = \mathtt{T}$. Thus $\beta_{k+1}^v = \mathtt{T}$.

In either of the two possible cases, we have $\beta_{k+1}^v = \mathtt{T}$, as claimed.

The Claim Implies the Theorem

Using induction on n, the previous claim implies

> *Claim II*: Suppose that the set $\Gamma = \{\beta_1, \ldots, \beta_k\}$ is satisfiable. Let α be a formula obtained from Γ by n uses of the resolution inference rule. Then the set $\Gamma \cup \{\alpha\}$ is satisfiable.

(The previous claim is the inductive step of this one.)

Therefore, if a set of assumptions leads to $\perp$ after any number n of resolution steps, the set must be unsatisfiable—since any set containing $\perp$ is unsatisfiable.

Thus Resolution is a sound refutation system, as required.

Can Resolution Fail?

In some cases, there may be no way to obtain $\bot$, using any number of resolution steps. What then?

Definition. A proof system S is *complete* if every entailment has a proof; that is, if

$$\Sigma \models \alpha \quad \text{implies} \quad \Sigma \vdash_S \alpha \ .$$

Theorem. Resolution is a complete refutation system for CNF formulas. That is, if there is no proof of $\bot$ from a set Σ of assumptions in CNF, then Σ is satisfiable.

Resolution Is Complete (Outline)

Claim. Suppose that a resolution proof "reaches a dead end"—that is, no new clause can be obtained, and yet $\bot$ has not been derived. Then the entire set of formulas (including the assumptions!) is satisfiable.

Proof (outline): We use induction again. However, it is not an induction on the length of the proof, nor on the number of formulas. Instead, we use induction on the number of variables present in the formulas.

Basis: only one variable occurs, say p.
After conversion to CNF and simplification, the only possible clauses are p and $\neg p$. If both occured, $\bot$ would be derivable. Thus at most one does; we can satisfy it.

Completeness Proof, part II

Inductive hypothesis: The claim holds for sets having at most k variables.

Consider a set of clauses using $k+1$ variables, from which no additional clause can be derived via the resolution rule. Suppose that it does not contain $\perp$. Select any one variable, say p, and separate the clauses into three sets:

S_p:	the clauses that contain the literal p.
$S_{\neg p}$:	the clauses that contain the literal $\neg p$.
R:	the remaining clauses, which do not contain variable p at all.

The "remainder" set R has at most k variables.
Thus the hypothesis applies: it has a satisfying valuation v.

Completeness Proof, part III

We have a valuation v, on the variables other than p, that satisfies set R. We now must satisfy the sets S_p and $S_{\neg p}$.

Case I: Every clause in S_p, of the form $p \vee \alpha$, has $\alpha^v = \mathtt{T}$.

In this case, the set S_p is already satisfied. Define $v(p) = \mathtt{F}$, which additionally makes every clause in $S_{\neg p}$ true.

Case II: S_p has some clause $p \vee \alpha$ with $\alpha^v = \mathtt{F}$.

In this case, set $v(p) = \mathtt{T}$; this satisfies every formula in S_p.

What about a clause $\neg p \vee \beta$ in $S_{\neg p}$?
Consider the formula $\alpha \vee \beta$, obtained by resolution from $p \vee \alpha$ and $\neg p \vee \beta$. It must lie in R; thus $\beta^v = \mathtt{T}$. Thus also $(\neg p \vee \beta)^v = \mathtt{T}$, as required.

Done!

Resolution Provides an Algorithm

The resolution method yields an algorithm to determine whether a given formula, or set of formulas, is satisfiable or contradictory.

- Convert to CNF. (A well-specified series of steps.)
- Form resolvents, until either $\perp$ is derived, or no more derivations are possible.
- If $\perp$ is derived, the original formula/set is contradictory. Otherwise, the preceding proof describes how to find a satisfying valuation.

The Algorithm Can Be Very Slow

The algorithm can be “souped up” in many ways.

- Choosing a good order of doing resolution steps. (It matters!)
- Sophisticated data structures, to handle large numbers of clauses.
- Additional techniques: setting variables, “learning”, etc.

However, it still has limitations.

Theorem (Haken, 1985): There is a number $c > 1$ such that
For every n, there is an unsatisfiable formula on n variables (and about $n^{1.5}$ total literals) whose smallest resolution refutation contains more than c^n steps.

Resolution is an exponential-time algorithm!
(And you thought quadratic was bad. . . .)

Resolution in Practice: Satisfiability (SAT) solvers

Determining the satisfiability of a set of propositional formulas is a fundamental problem in computer science.

Examples:

- software and hardware verification
- automatic test pattern generation
- planning
- scheduling

. . . many problems of practical importance can be formulated as determining the satisfiability of a set of formulas.

Resolution in practice: "SAT Solvers"

Modern SAT solvers can often solve hard real-world instances with over a million propositional variables and several million clauses.

Annual SAT competitions:

http://www.satcompetition.org/

Many are open source systems.

Best SAT solvers are based on backtracking search.

Satisfiability in Theory

If a formula is satisfiable, then there is a short demonstration of that: simply give the valuation. Anyone can easily check that it is correct.

The class of problems with this property is known as NP.

The class of problems for which one can find a solution efficiently is known as P.

(For a precise definition, we need to define "efficiently." We won't, here.)

A Fundamental Question: Is $P = NP$?

A partial answer: If SAT is in P (by any algorithm), then $P = NP$.