CS 257: Advanced Topics in Formal Methods
Fall 2019

Lecture 2

Aleksandar Zeljić
(materials by Clark Barrett)
Stanford University

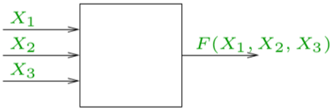
Outline

- Propositional Logic: Motivation
- Propositional Logic: Syntax
- Propositional Logic: Well-Formed Formulas
- Recognizing Well-Formed Formulas
- Propositional Logic: Semantics
- Truth Tables
- Satisfiability and Tautologies

Material is drawn from Chapter 1 of Enderton.

Propositional Logic: Motivation

Consider an electrical device having n inputs and one output. Assume that to each input we apply a signal that is either 1 or 0, and that this uniquely determines whether the output is 1 or 0.

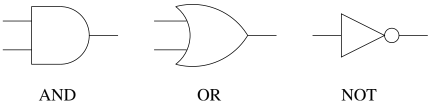


The behavior of such a device is described by a Boolean function:

$$F(X_1, \ldots, X_n) = \text{the output signal given the input signals } X_1, \ldots, X_n.$$

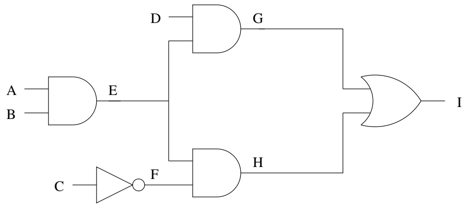
We call such a device a *Boolean gate*.

The most common Boolean gates are *AND*, *OR*, and *NOT* gates.



Propositional Logic: Motivation

The inputs and outputs of Boolean gates can be connected together to form a *combinational Boolean circuit*.



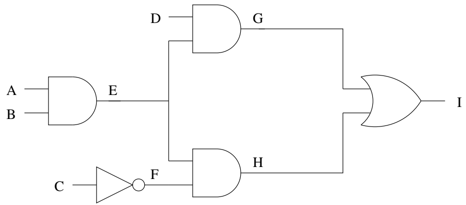
A combinational Boolean circuit corresponds to a *directed acyclic graph* (DAG) whose leaves are *inputs* and each of whose nodes is labeled with the name of a Boolean gate. One or more of the nodes may be identified as outputs.

A common question with Boolean circuits is whether it is possible to set an output to true (e.g. when the output represents an *error* signal).

Suppose your job was to find out if the output of a large Boolean circuit could ever be true. How would you do it?

Propositional Logic: Motivation

The inputs and outputs of Boolean gates can be connected together to form a *combinational Boolean circuit*.



A combinational Boolean circuit corresponds to a *directed acyclic graph* (DAG) whose leaves are *inputs* and each of whose nodes is labeled with the name of a Boolean gate. One or more of the nodes may be identified as outputs.

A common question with Boolean circuits is whether it is possible to set an output to true (e.g. when the output represents an *error* signal).

Suppose your job was to find out if the output of a large Boolean circuit could ever be true. How would you do it?

Propositional Logic provides the formalism to answer such questions.

Propositional Logic: Motivation

Propositional (or Sentential) logic is simple but extremely important in Computer Science

1. It is the basis for day-to-day reasoning (in programming, LSATs, etc.)
2. It is the theory behind digital circuits.
3. Many problems can be translated into propositional logic.
4. It is an important part of more complex logics (such as *first-order logic*, also called *predicate logic*, which we'll discuss later.)

What is Logic?

A formal logic is defined by its *syntax* and *semantics*.

Syntax

- An *alphabet* is a set of symbols.
- A finite sequence of these symbols is called an *expression*.
- A set of rules defines the *well-formed* expressions.

Semantics

- Gives meaning to well-formed expressions
- Formal notions of induction and recursion are required to provide a rigorous semantics.

Propositional Logic: Syntax

Alphabet

(	Left parenthesis	Begin group
)	Right parenthesis	End group
$\neg$	Negation symbol	English: not
$\wedge$	Conjunction symbol	English: and
$\vee$	Disjunction symbol	English: or (inclusive)
$\rightarrow$	Conditional symbol	English: if, then
$\leftrightarrow$	Bi-conditional symbol	English: if and only if
A_1	First propositional symbol	
A_2	Second propositional symbol	
$\ldots$		
A_n	nth propositional symbol	
$\ldots$		

Propositional Logic: Syntax

Alphabet

- *Propositional connective* symbols: $\neg$, $\wedge$, $\vee$, $\rightarrow$, $\leftrightarrow$.
- *Logical* symbols: $\neg$, $\wedge$, $\vee$, $\rightarrow$, $\leftrightarrow$, (,).
- *Parameters* or *nonlogical symbols*: A_1, A_2, A_3, $\ldots$

The meaning of logical symbols is always the same. The meaning of nonlogical symbols depends on the context.

Propositional Logic: Syntax

An *expression* is a sequence of symbols. A sequence is denoted explicitly by a comma separated list enclosed in angle brackets: $< a_1, \ldots, a_m >$.

Examples

$<(, A_1, \wedge, A_3,)>$
$<(, (, \neg, A_1,), \rightarrow, A_2,)>$
$<),), \leftrightarrow,), A_5>$

Propositional Logic: Syntax

An *expression* is a sequence of symbols. A sequence is denoted explicitly by a comma separated list enclosed in angle brackets: $<a_1, \ldots, a_m>$.

Examples

$$<(,\ A_1,\ \wedge,\ A_3,\ )> \qquad (A_1 \wedge A_3)$$
$$<(,\ (,\ \neg,\ A_1,\ ),\ \rightarrow,\ A_2,\ )> \qquad ((\neg A_1) \rightarrow A_2)$$
$$<),\ ),\ \leftrightarrow,\ ),\ A_5> \qquad)) \leftrightarrow) A_5$$

For convenience, we will write these sequences as a simple string of symbols, with the understanding that the *formal* structure represented is a sequence containing exactly the symbols in the string.

The formal meaning becomes important when trying to prove things about expressions.

Propositional Logic: Syntax

An *expression* is a sequence of symbols. A sequence is denoted explicitly by a comma separated list enclosed in angle brackets: $<a_1, \ldots, a_m>$.

Examples

$<(,\ A_1,\ \wedge,\ A_3,\ )>$	$(A_1 \wedge A_3)$
$<(,\ (,\ \neg,\ A_1,\ ),\ \rightarrow,\ A_2,\ )>$	$((\neg A_1) \rightarrow A_2)$
$<),\ ),\ \leftrightarrow,\ ),\ A_5>$	$)) \leftrightarrow)A_5$

For convenience, we will write these sequences as a simple string of symbols, with the understanding that the *formal* structure represented is a sequence containing exactly the symbols in the string.

The formal meaning becomes important when trying to prove things about expressions.

Not all expressions make sense. Part of the job of defining a syntax is to *restrict* the kinds of expressions that will be allowed.

Propositional Logic: Syntax

We define the set W of *well-formed formulas* (*wffs*) as follows.

(a) Every expression consisting of a single propositional symbol is in W.

(b) If α and β are in W, so are $(\neg\alpha)$, $(\alpha \wedge \beta)$, $(\alpha \vee \beta)$, $(\alpha \rightarrow \beta)$, and $(\alpha \leftrightarrow \beta)$.

(c) No expression is in W unless forced by (a) or (b)

This definition is *inductive*: the set being defined is used as part of the definition.

Propositional Logic: Syntax

We define the set W of *well-formed formulas* (*wffs*) as follows.

(a) Every expression consisting of a single propositional symbol is in W.

(b) If α and β are in W, so are $(\neg\alpha)$, $(\alpha \wedge \beta)$, $(\alpha \vee \beta)$, $(\alpha \rightarrow \beta)$, and $(\alpha \leftrightarrow \beta)$.

(c) No expression is in W unless forced by (a) or (b)

This definition is *inductive*: the set being defined is used as part of the definition.

How would you use this definition to prove that $)) \leftrightarrow) A_5$ is not a *wff*?

Propositional Logic: Syntax

We define the set W of *well-formed formulas* (*wffs*) as follows.

(a) Every expression consisting of a single propositional symbol is in W.

(b) If α and β are in W, so are $(\neg\alpha)$, $(\alpha \wedge \beta)$, $(\alpha \vee \beta)$, $(\alpha \to \beta)$, and $(\alpha \leftrightarrow \beta)$.

(c) No expression is in W unless forced by (a) or (b)

This definition is *inductive*: the set being defined is used as part of the definition.

How would you use this definition to prove that $)) \leftrightarrow) A_5$ is not a *wff*?

Item (c) is too vague for our purposes. To make it more precise we use *induction*.

Propositional Logic: Well-Formed Formulas

We can use a formal inductive definition to define the set W of well-formed formulas in propositional logic.

- $U =$
- $B =$
- $F =$

Propositional Logic: Well-Formed Formulas

We can use a formal inductive definition to define the set W of well-formed formulas in propositional logic.

- $U =$ the set of all expressions.
- $B =$
- $F =$

Propositional Logic: Well-Formed Formulas

We can use a formal inductive definition to define the set W of well-formed formulas in propositional logic.

- $U =$ the set of all expressions.
- $B =$ the set of expressions consisting of a single propositional symbol.
- $F =$

Propositional Logic: Well-Formed Formulas

We can use a formal inductive definition to define the set W of well-formed formulas in propositional logic.

- $U =$ the set of all expressions.
- $B =$ the set of expressions consisting of a single propositional symbol.
- $F =$ the set of formula-building operations:
  - $\mathcal{E}_\neg(\alpha) = (\neg\alpha)$
  - $\mathcal{E}_\wedge(\alpha, \beta) = (\alpha \wedge \beta)$
  - $\mathcal{E}_\vee(\alpha, \beta) = (\alpha \vee \beta)$
  - $\mathcal{E}_\rightarrow(\alpha, \beta) = (\alpha \rightarrow \beta)$
  - $\mathcal{E}_\leftrightarrow(\alpha, \beta) = (\alpha \leftrightarrow \beta)$

Induction

We can call the set *generated from B by F* simply C.

Now, given any inductive definition of a set, we can prove things about that set using the following principle.

Induction Principle

If C is the set generated from B by F and S is a set which includes B and is closed under F (i.e. S is inductive), then $C \subseteq S$.

Proof

Since S is inductive, and C is the intersection of all inductive sets, it follows that $C \subseteq S$.

$\square$

We often use the induction principle to show that an inductive set C has a particular property. The argument looks like this: (i) Define S to be the subset of U with some property P; (ii) Show that S is inductive.

This proves that $C \subseteq S$ and thus all elements of C have property P.

Propositional Logic: Well-Formed Formulas

Given our inductive definition of well-formed formulas, we can use the induction principle to prove things about the set W of well-formed formulas.

Example

Prove that any *wff* has the same number of left parentheses and right parentheses.

Proof

Let $l(\alpha)$ be the number of left parentheses and $r(\alpha)$ the number of right parentheses in an expression α. Let S be the set of all expressions α such that $l(\alpha) = r(\alpha)$. We wish to show that $W \subseteq S$. This follows from the induction principle if we can show that S is inductive.

Base Case:

We must show that $B \subseteq S$. Recall that B is the set of expressions consisting of a single propositional symbol. It is clear that for such expressions, $l(\alpha) = r(\alpha) = 0$.

Propositional Logic: Well-Formed Formulas

Inductive Case:

We must show that S is closed under each formula-building operator in F.

Propositional Logic: Well-Formed Formulas

Inductive Case:

We must show that S is closed under each formula-building operator in F.

- $\mathcal{E}_\neg$
  Suppose $\alpha \in S$. We know that $\mathcal{E}_\neg(\alpha) = (\neg\alpha)$. It follows that $l(\mathcal{E}_\neg(\alpha)) = 1 + l(\alpha)$ and $r(\mathcal{E}_\neg(\alpha)) = 1 + r(\alpha)$.
  But because $\alpha \in S$, we know that $l(\alpha) = r(\alpha)$, so it follows that $l(\mathcal{E}_\neg(\alpha)) = r(\mathcal{E}_\neg(\alpha))$, and thus $\mathcal{E}_\neg(\alpha) \in S$.

Propositional Logic: Well-Formed Formulas

Inductive Case:

We must show that S is closed under each formula-building operator in F.

- $\mathcal{E}_\neg$
  Suppose $\alpha \in S$. We know that $\mathcal{E}_\neg(\alpha) = (\neg\alpha)$. It follows that $l(\mathcal{E}_\neg(\alpha)) = 1 + l(\alpha)$ and $r(\mathcal{E}_\neg(\alpha)) = 1 + r(\alpha)$.
  But because $\alpha \in S$, we know that $l(\alpha) = r(\alpha)$, so it follows that $l(\mathcal{E}_\neg(\alpha)) = r(\mathcal{E}_\neg(\alpha))$, and thus $\mathcal{E}_\neg(\alpha) \in S$.
- $\mathcal{E}_\wedge$
  Suppose $\alpha, \beta \in S$. We know that $\mathcal{E}_\wedge(\alpha, \beta) = (\alpha \wedge \beta)$. Thus $l(\mathcal{E}_\wedge(\alpha, \beta)) = 1 + l(\alpha) + l(\beta)$ and $r(\mathcal{E}_\wedge(\alpha, \beta)) = 1 + r(\alpha) + r(\beta)$.
  As before, it follows from the inductive hypothesis that $\mathcal{E}_\wedge(\alpha, \beta) \in S$

Propositional Logic: Well-Formed Formulas

Inductive Case:

We must show that S is closed under each formula-building operator in F.

- $\mathcal{E}_\neg$
  Suppose $\alpha \in S$. We know that $\mathcal{E}_\neg(\alpha) = (\neg\alpha)$. It follows that $l(\mathcal{E}_\neg(\alpha)) = 1 + l(\alpha)$ and $r(\mathcal{E}_\neg(\alpha)) = 1 + r(\alpha)$.
  But because $\alpha \in S$, we know that $l(\alpha) = r(\alpha)$, so it follows that $l(\mathcal{E}_\neg(\alpha)) = r(\mathcal{E}_\neg(\alpha))$, and thus $\mathcal{E}_\neg(\alpha) \in S$.
- $\mathcal{E}_\wedge$
  Suppose $\alpha, \beta \in S$. We know that $\mathcal{E}_\wedge(\alpha, \beta) = (\alpha \wedge \beta)$. Thus $l(\mathcal{E}_\wedge(\alpha, \beta)) = 1 + l(\alpha) + l(\beta)$ and $r(\mathcal{E}_\wedge(\alpha, \beta)) = 1 + r(\alpha) + r(\beta)$.
  As before, it follows from the inductive hypothesis that $\mathcal{E}_\wedge(\alpha, \beta) \in S$
- The arguments for $\mathcal{E}_\vee$, $\mathcal{E}_\rightarrow$, and $\mathcal{E}_\leftrightarrow$ are exactly analogous to the one for $\mathcal{E}_\wedge$.

$\square$

Propositional Logic: Well-Formed Formulas

Inductive Case:

We must show that S is closed under each formula-building operator in F.

- $\mathcal{E}_\neg$
  Suppose $\alpha \in S$. We know that $\mathcal{E}_\neg(\alpha) = (\neg\alpha)$. It follows that $l(\mathcal{E}_\neg(\alpha)) = 1 + l(\alpha)$ and $r(\mathcal{E}_\neg(\alpha)) = 1 + r(\alpha)$.
  But because $\alpha \in S$, we know that $l(\alpha) = r(\alpha)$, so it follows that $l(\mathcal{E}_\neg(\alpha)) = r(\mathcal{E}_\neg(\alpha))$, and thus $\mathcal{E}_\neg(\alpha) \in S$.
- $\mathcal{E}_\wedge$
  Suppose $\alpha, \beta \in S$. We know that $\mathcal{E}_\wedge(\alpha, \beta) = (\alpha \wedge \beta)$. Thus $l(\mathcal{E}_\wedge(\alpha, \beta)) = 1 + l(\alpha) + l(\beta)$ and $r(\mathcal{E}_\wedge(\alpha, \beta)) = 1 + r(\alpha) + r(\beta)$.
  As before, it follows from the inductive hypothesis that $\mathcal{E}_\wedge(\alpha, \beta) \in S$
- The arguments for $\mathcal{E}_\vee$, $\mathcal{E}_\rightarrow$, and $\mathcal{E}_\leftrightarrow$ are exactly analogous to the one for $\mathcal{E}_\wedge$.

$\square$

Since S includes B and is closed under the operations in F, it is inductive. It follows by the induction principle that $W \subseteq S$.

Propositional Logic: Well-Formed Formulas

Now we can return to the question of how to prove that an expression is not a *wff*.

How do we know that $)) \leftrightarrow)A_5$ is not a *wff*?

Propositional Logic: Well-Formed Formulas

Now we can return to the question of how to prove that an expression is not a *wff*.

How do we know that $)) \leftrightarrow) A_5$ is not a *wff*?

It does not have the same number of left and right parentheses.

It follows from the theorem we just proved that $)) \leftrightarrow) A_5$ is not a *wff*.

An Algorithm for Recognizing WFFs

Lemma

Let α be a *wff*. Then exactly one of the following is true.

- α is a propositional symbol.
- $\alpha = (\neg\beta)$ where β is a *wff*.
- $\alpha = (\beta \odot \gamma)$ where $\odot$ is one of $\{\wedge, \vee, \to, \leftrightarrow\}$, β is the first parentheses-balanced initial segment of the result of dropping the first $($ from α, and β and γ are *wffs*.

An Algorithm for Recognizing WFFs

Lemma

Let α be a *wff*. Then exactly one of the following is true.

- α is a propositional symbol.
- $\alpha = (\neg\beta)$ where β is a *wff*.
- $\alpha = (\beta \odot \gamma)$ where $\odot$ is one of $\{\wedge, \vee, \rightarrow, \leftrightarrow\}$, β is the first parentheses-balanced initial segment of the result of dropping the first $($ from α, and β and γ are *wffs*.

How would you prove this?

An Algorithm for Recognizing WFFs

Lemma

Let α be a *wff*. Then exactly one of the following is true.

- α is a propositional symbol.
- $\alpha = (\neg\beta)$ where β is a *wff*.
- $\alpha = (\beta \odot \gamma)$ where $\odot$ is one of $\{\wedge, \vee, \rightarrow, \leftrightarrow\}$, β is the first parentheses-balanced initial segment of the result of dropping the first $($ from α, and β and γ are *wffs*.

How would you prove this?

Induction, of course!

An Algorithm for Recognizing WFFs

Input: expression α Output: *true* or *false* (indicating whether α is a *wff*).

0. Begin with an initial construction tree T containing a single node labeled with α.
1. If all leaves of T are labeled with propositional symbols, return *true*.
2. Select a leaf labeled with an expression α_1 which is not a propositional symbol.
3. If α_1 does not begin with $($ return *false*.
4. If $\alpha_1 = (\neg\beta)$, then add a child to the leaf labeled by α_1, label it with β, and goto 1.
5. Scan α_1 until first reaching $(\beta$, where β is a nonempty expression having the same number of left and right parentheses. If there is no such β, return *false*.
6. If $\alpha_1 = (\beta \odot \gamma)$ where $\odot$ is one of $\{\wedge, \vee, \rightarrow, \leftrightarrow\}$, then add two children to the leaf labeled by α_1, label them with β and γ, and goto 1.
7. Return *false*.

An Algorithm for Recognizing WFFs

Termination

How do we prove termination of this algorithm?

An Algorithm for Recognizing WFFs

Termination

How do we prove termination of this algorithm?

We can show that the sum of the lengths of all the expressions labeling leaves decreases on each iteration of the loop.

An Algorithm for Recognizing WFFs

Termination

How do we prove termination of this algorithm?

We can show that the sum of the lengths of all the expressions labeling leaves decreases on each iteration of the loop.

Soundness

If the algorithm returns *true* when given input α, then α is a *wff*.

The proof is by induction on the tree T generated by the algorithm from the leaves up to the root.

An Algorithm for Recognizing WFFs

Termination

How do we prove termination of this algorithm?

We can show that the sum of the lengths of all the expressions labeling leaves decreases on each iteration of the loop.

Soundness

If the algorithm returns *true* when given input α, then α is a *wff*.

The proof is by induction on the tree T generated by the algorithm from the leaves up to the root.

Completeness

If α is a *wff*, then the algorithm will return *true*.

Proof using the induction principle for the set of *wffs*.

Notational Conventions

- Larger variety of propositional symbols: A, B, C, D, p, q, r, etc.
- Outermost parentheses can be omitted: $A \wedge B$ instead of $(A \wedge B)$.
- Negation symbol binds stronger than binary connectives and its scope is as small as possible: $\neg A \wedge B$ means $((\neg A) \wedge B)$.
- $\{\wedge, \vee\}$ bind stronger than $\{\rightarrow, \leftrightarrow\}$: $A \wedge B \rightarrow \neg C \vee D$ is $((A \wedge B) \rightarrow ((\neg C) \vee D))$
- When one symbol is used repeatedly, grouping is to the right: $A \wedge B \wedge C$ is $(A \wedge (B \wedge C))$

Note that conventions are only unambiguous for *wffs*, not for arbitrary expressions.

Propositional Logic: Semantics

Intuitively, given a *wff* α and a value (either **T** or **F**) for each propositional symbol in α, we should be able to determine the value of α.

How do we make this precise?

Let v be a function from B to $\{\mathbf{F}, \mathbf{T}\}$. We call this function a *truth assignment*.

Now, we define $\overline{v}$, a function from W to $\{\mathbf{F}, \mathbf{T}\}$ as follows (we compute with **F** and **T** as if they were 0 and 1 respectively).

- For each propositional symbol A_i, $\overline{v}(A_i) = v(A_i)$.
- $\overline{v}(\mathcal{E}_{\neg}(\alpha)) = \mathbf{T} - \overline{v}(\alpha)$
- $\overline{v}(\mathcal{E}_{\wedge}(\alpha, \beta)) = \min(\overline{v}(\alpha), \overline{v}(\beta))$
- $\overline{v}(\mathcal{E}_{\vee}(\alpha, \beta)) = \max(\overline{v}(\alpha), \overline{v}(\beta))$
- $\overline{v}(\mathcal{E}_{\rightarrow}(\alpha, \beta)) = \max(\mathbf{T} - \overline{v}(\alpha), \overline{v}(\beta))$
- $\overline{v}(\mathcal{E}_{\leftrightarrow}(\alpha, \beta)) = \mathbf{T} - |\overline{v}(\alpha) - \overline{v}(\beta)|$

The recursion theorem and the unique readability theorem guarantee that $\overline{v}$ is well-defined. (see Enderton)

Truth Tables

There are other ways to present the semantics which are less formal but perhaps more intuitive.

α	$\neg\alpha$
T	
F	

α	β	$\alpha \wedge \beta$
T	**T**	
T	**F**	
F	**T**	
F	**F**	

α	β	$\alpha \vee \beta$
T	**T**	
T	**F**	
F	**T**	
F	**F**	

α	β	$\alpha \rightarrow \beta$
T	**T**	
T	**F**	
F	**T**	
F	**F**	

α	β	$\alpha \leftrightarrow \beta$
T	**T**	
T	**F**	
F	**T**	
F	**F**	

Truth Tables

There are other ways to present the semantics which are less formal but perhaps more intuitive.

α	$\neg\alpha$
T	**F**
F	**T**

α	β	$\alpha \wedge \beta$
T	**T**	
T	**F**	
F	**T**	
F	**F**	

α	β	$\alpha \vee \beta$
T	**T**	
T	**F**	
F	**T**	
F	**F**	

α	β	$\alpha \rightarrow \beta$
T	**T**	
T	**F**	
F	**T**	
F	**F**	

α	β	$\alpha \leftrightarrow \beta$
T	**T**	
T	**F**	
F	**T**	
F	**F**	

Truth Tables

There are other ways to present the semantics which are less formal but perhaps more intuitive.

α	$\neg\alpha$
T	**F**
F	**T**

α	β	$\alpha \wedge \beta$
T	**T**	**T**
T	**F**	**F**
F	**T**	**F**
F	**F**	**F**

α	β	$\alpha \vee \beta$
T	**T**	
T	**F**	
F	**T**	
F	**F**	

α	β	$\alpha \rightarrow \beta$
T	**T**	
T	**F**	
F	**T**	
F	**F**	

α	β	$\alpha \leftrightarrow \beta$
T	**T**	
T	**F**	
F	**T**	
F	**F**	

Truth Tables

There are other ways to present the semantics which are less formal but perhaps more intuitive.

α	$\neg\alpha$
T	**F**
F	**T**

α	β	$\alpha \wedge \beta$
T	**T**	**T**
T	**F**	**F**
F	**T**	**F**
F	**F**	**F**

α	β	$\alpha \vee \beta$
T	**T**	**T**
T	**F**	**T**
F	**T**	**T**
F	**F**	**F**

α	β	$\alpha \rightarrow \beta$
T	**T**	**T**
T	**F**	**F**
F	**T**	**T**
F	**F**	**T**

α	β	$\alpha \leftrightarrow \beta$
T	**T**	**T**
T	**F**	**F**
F	**T**	**F**
F	**F**	**T**

Complex truth tables

Truth tables can also be used to calculate all possible values of $\overline{v}$ for a given *wff*: We associate a column with each propositional symbol and a column with each propositional connective. There is a row for each possible truth assignment to the propositional connectives.

A_1	A_2	A_3	$(A_1$	$\vee$	$(A_2$	$\wedge$	$\neg A_3))$
T	**T**	**T**	**T**		**T**		
T	**T**	**F**	**T**		**T**		
T	**F**	**T**	**T**		**F**		
T	**F**	**F**	**T**		**F**		
F	**T**	**T**	**F**		**T**		
F	**T**	**F**	**F**		**T**		
F	**F**	**T**	**F**		**F**		
F	**F**	**F**	**F**		**F**		

Complex truth tables

Truth tables can also be used to calculate all possible values of $\overline{v}$ for a given *wff*: We associate a column with each propositional symbol and a column with each propositional connective. There is a row for each possible truth assignment to the propositional connectives.

A_1	A_2	A_3	$(A_1$	$\vee$	$(A_2$	$\wedge$	$\neg A_3))$
T	T	T	T		T		F
T	T	F	T		T		T
T	F	T	T		F		F
T	F	F	T		F		T
F	T	T	F		T		F
F	T	F	F		T		T
F	F	T	F		F		F
F	F	F	F		F		T

Complex truth tables

Truth tables can also be used to calculate all possible values of $\overline{v}$ for a given *wff*: We associate a column with each propositional symbol and a column with each propositional connective. There is a row for each possible truth assignment to the propositional connectives.

A_1	A_2	A_3	$(A_1$	$\vee$	$(A_2$	$\wedge$	$\neg A_3))$
T	**T**	**T**	**T**		**T**	**F**	**F**
T	**T**	**F**	**T**		**T**	**T**	**T**
T	**F**	**T**	**T**		**F**	**F**	**F**
T	**F**	**F**	**T**		**F**	**F**	**T**
F	**T**	**T**	**F**		**T**	**F**	**F**
F	**T**	**F**	**F**		**T**	**T**	**T**
F	**F**	**T**	**F**		**F**	**F**	**F**
F	**F**	**F**	**F**		**F**	**F**	**T**

Complex truth tables

Truth tables can also be used to calculate all possible values of $\overline{v}$ for a given *wff*: We associate a column with each propositional symbol and a column with each propositional connective. There is a row for each possible truth assignment to the propositional connectives.

A_1	A_2	A_3	$(A_1$	$\vee$	$(A_2$	$\wedge$	$\neg A_3))$
T	**T**	**T**	**T**	**T**	**T**	**F**	**F**
T	**T**	**F**	**T**	**T**	**T**	**T**	**T**
T	**F**	**T**	**T**	**T**	**F**	**F**	**F**
T	**F**	**F**	**T**	**T**	**F**	**F**	**T**
F	**T**	**T**	**F**	**F**	**T**	**F**	**F**
F	**T**	**F**	**F**	**T**	**T**	**T**	**T**
F	**F**	**T**	**F**	**F**	**F**	**F**	**F**
F	**F**	**F**	**F**	**F**	**F**	**F**	**T**

Definitions

If α is a *wff*, then a truth assignment v *satisfies* α if $\overline{v}(\alpha) = \mathbf{T}$.

Definitions

If α is a *wff*, then a truth assignment v *satisfies* α if $\overline{v}(\alpha) = \mathbf{T}$.

A *wff* α is *satisfiable* if there exists some truth assignment v which satisfies α.

Definitions

If α is a *wff*, then a truth assignment v *satisfies* α if $\overline{v}(\alpha) = \mathbf{T}$.

A *wff* α is *satisfiable* if there exists some truth assignment v which satisfies α.

Suppose Σ is a set of *wffs*. Then Σ *tautologically implies* α, $\Sigma \models \alpha$, if every truth assignment which satisfies each formula in Σ also satisfies α.

Definitions

If α is a *wff*, then a truth assignment v *satisfies* α if $\overline{v}(\alpha) = \mathbf{T}$.

A *wff* α is *satisfiable* if there exists some truth assignment v which satisfies α.

Suppose Σ is a set of *wffs*. Then Σ *tautologically implies* α, $\Sigma \models \alpha$, if every truth assignment which satisfies each formula in Σ also satisfies α.

Particular cases:

- If $\emptyset \models \alpha$, then we say α is a *tautology* or α is *valid* and write $\models \alpha$.

Definitions

If α is a *wff*, then a truth assignment v *satisfies* α if $\overline{v}(\alpha) = \mathbf{T}$.

A *wff* α is *satisfiable* if there exists some truth assignment v which satisfies α.

Suppose Σ is a set of *wffs*. Then Σ *tautologically implies* α, $\Sigma \models \alpha$, if every truth assignment which satisfies each formula in Σ also satisfies α.

Particular cases:

- If $\emptyset \models \alpha$, then we say α is a *tautology* or α is *valid* and write $\models \alpha$.
- If Σ is *unsatisfiable*, then $\Sigma \models \alpha$ for every *wff* α.

Definitions

If α is a *wff*, then a truth assignment v *satisfies* α if $\overline{v}(\alpha) = \mathbf{T}$.

A *wff* α is *satisfiable* if there exists some truth assignment v which satisfies α.

Suppose Σ is a set of *wffs*. Then Σ *tautologically implies* α, $\Sigma \models \alpha$, if every truth assignment which satisfies each formula in Σ also satisfies α.

Particular cases:

- If $\emptyset \models \alpha$, then we say α is a *tautology* or α is *valid* and write $\models \alpha$.
- If Σ is *unsatisfiable*, then $\Sigma \models \alpha$ for every *wff* α.
- If $\alpha \models \beta$ (shorthand for $\{\alpha\} \models \beta$) and $\beta \models \alpha$, then α and β are *tautologically equivalent*.

Definitions

If α is a *wff*, then a truth assignment v *satisfies* α if $\overline{v}(\alpha) = \mathbf{T}$.

A *wff* α is *satisfiable* if there exists some truth assignment v which satisfies α.

Suppose Σ is a set of *wffs*. Then Σ *tautologically implies* α, $\Sigma \models \alpha$, if every truth assignment which satisfies each formula in Σ also satisfies α.

Particular cases:

- If $\emptyset \models \alpha$, then we say α is a *tautology* or α is *valid* and write $\models \alpha$.
- If Σ is *unsatisfiable*, then $\Sigma \models \alpha$ for every *wff* α.
- If $\alpha \models \beta$ (shorthand for $\{\alpha\} \models \beta$) and $\beta \models \alpha$, then α and β are *tautologically equivalent*.
- $\Sigma \models \alpha$ if and only if $\bigwedge(\Sigma) \to \alpha$ is valid.

Examples

- $(A \vee B) \wedge (\neg A \vee \neg B)$

Examples

- $(A \vee B) \wedge (\neg A \vee \neg B)$ is satisfiable, but not valid.

Examples

- $(A \vee B) \wedge (\neg A \vee \neg B)$ is satisfiable, but not valid.
- $(A \vee B) \wedge (\neg A \vee \neg B) \wedge (A \leftrightarrow B)$

Examples

- $(A \vee B) \wedge (\neg A \vee \neg B)$ is satisfiable, but not valid.
- $(A \vee B) \wedge (\neg A \vee \neg B) \wedge (A \leftrightarrow B)$ is unsatisfiable.

Examples

- $(A \vee B) \wedge (\neg A \vee \neg B)$ is satisfiable, but not valid.
- $(A \vee B) \wedge (\neg A \vee \neg B) \wedge (A \leftrightarrow B)$ is unsatisfiable.
- $\{A, A \rightarrow B\} \models B$
- $\{A, \neg A\} \models (A \wedge \neg A)$

Examples

- $(A \lor B) \land (\neg A \lor \neg B)$ is satisfiable, but not valid.
- $(A \lor B) \land (\neg A \lor \neg B) \land (A \leftrightarrow B)$ is unsatisfiable.
- $\{A, A \rightarrow B\} \models B$
- $\{A, \neg A\} \models (A \land \neg A)$
- $\neg(A \land B)$ is tautologically equivalent to $\neg A \lor \neg B$

Examples

- $(A \vee B) \wedge (\neg A \vee \neg B)$ is satisfiable, but not valid.
- $(A \vee B) \wedge (\neg A \vee \neg B) \wedge (A \leftrightarrow B)$ is unsatisfiable.
- $\{A, A \rightarrow B\} \models B$
- $\{A, \neg A\} \models (A \wedge \neg A)$
- $\neg(A \wedge B)$ is tautologically equivalent to $\neg A \vee \neg B$

Suppose you had an algorithm *SAT* which would take a *wff* α as input and return *true* if α is satisfiable and *false* otherwise. How would you use this algorithm to verify each of the claims made above?

Examples

- $(A \vee B) \wedge (\neg A \vee \neg B)$ is satisfiable, but not valid.
- $(A \vee B) \wedge (\neg A \vee \neg B) \wedge (A \leftrightarrow B)$ is unsatisfiable.
- $\{A, A \rightarrow B\} \models B$ $\qquad$ $(A \wedge (A \rightarrow B) \wedge (\neg B))$
- $\{A, \neg A\} \models (A \wedge \neg A)$
- $\neg(A \wedge B)$ is tautologically equivalent to $\neg A \vee \neg B$

Suppose you had an algorithm *SAT* which would take a *wff* α as input and return *true* if α is satisfiable and *false* otherwise. How would you use this algorithm to verify each of the claims made above?

Examples

- $(A \vee B) \wedge (\neg A \vee \neg B)$ is satisfiable, but not valid.
- $(A \vee B) \wedge (\neg A \vee \neg B) \wedge (A \leftrightarrow B)$ is unsatisfiable.
- $\{A, A \rightarrow B\} \models B$ $\quad (A \wedge (A \rightarrow B) \wedge (\neg B))$
- $\{A, \neg A\} \models (A \wedge \neg A)$ $\quad (A \wedge (\neg A) \wedge \neg(A \wedge \neg A))$
- $\neg(A \wedge B)$ is tautologically equivalent to $\neg A \vee \neg B$

Suppose you had an algorithm *SAT* which would take a *wff* α as input and return *true* if α is satisfiable and *false* otherwise. How would you use this algorithm to verify each of the claims made above?

Examples

- $(A \vee B) \wedge (\neg A \vee \neg B)$ is satisfiable, but not valid.
- $(A \vee B) \wedge (\neg A \vee \neg B) \wedge (A \leftrightarrow B)$ is unsatisfiable.
- $\{A, A \rightarrow B\} \models B$ $\quad (A \wedge (A \rightarrow B) \wedge (\neg B))$
- $\{A, \neg A\} \models (A \wedge \neg A)$ $\quad (A \wedge (\neg A) \wedge \neg(A \wedge \neg A))$
- $\neg(A \wedge B)$ is tautologically equivalent to $\neg A \vee \neg B$
  $\neg(\neg(A \wedge B) \leftrightarrow (\neg A \vee \neg B))$

Suppose you had an algorithm *SAT* which would take a *wff* α as input and return *true* if α is satisfiable and *false* otherwise. How would you use this algorithm to verify each of the claims made above?

Examples

- $(A \vee B) \wedge (\neg A \vee \neg B)$ is satisfiable, but not valid.
- $(A \vee B) \wedge (\neg A \vee \neg B) \wedge (A \leftrightarrow B)$ is unsatisfiable.
- $\{A, A \rightarrow B\} \models B$ $\quad (A \wedge (A \rightarrow B) \wedge (\neg B))$
- $\{A, \neg A\} \models (A \wedge \neg A)$ $\quad (A \wedge (\neg A) \wedge \neg(A \wedge \neg A))$
- $\neg(A \wedge B)$ is tautologically equivalent to $\neg A \vee \neg B$
  $\neg(\neg(A \wedge B) \leftrightarrow (\neg A \vee \neg B))$

Now suppose you had an algorithm *CHECKVALID* which returns *true* when α is valid and *false* otherwise. How would you verify the claims given this algorithm?

Examples

- $(A \vee B) \wedge (\neg A \vee \neg B)$ is satisfiable, but not valid.
- $(A \vee B) \wedge (\neg A \vee \neg B) \wedge (A \leftrightarrow B)$ is unsatisfiable.
- $\{A, A \rightarrow B\} \models B$ $\quad (A \wedge (A \rightarrow B) \wedge (\neg B))$
- $\{A, \neg A\} \models (A \wedge \neg A)$ $\quad (A \wedge (\neg A) \wedge \neg(A \wedge \neg A))$
- $\neg(A \wedge B)$ is tautologically equivalent to $\neg A \vee \neg B$
  $\neg(\neg(A \wedge B) \leftrightarrow (\neg A \vee \neg B))$

Now suppose you had an algorithm *CHECKVALID* which returns *true* when α is valid and *false* otherwise. How would you verify the claims given this algorithm?

Satisfiability and validity are dual notions: α is unsatisfiable if and only if $\neg\alpha$ is valid.

Determining Satisfiability using Truth Tables

An Algorithm for Satisfiability

To check whether α is satisfiable, form the truth table for α. If there is a row in which **T** appears as the value for α, then α is satisfiable. Otherwise, α is unsatisfiable.

Determining Satisfiability using Truth Tables

An Algorithm for Satisfiability

To check whether α is satisfiable, form the truth table for α. If there is a row in which **T** appears as the value for α, then α is satisfiable. Otherwise, α is unsatisfiable.

An Algorithm for Tautological Implication

To check whether $\{\alpha_1, \ldots, \alpha_k\} \models \beta$, check the satisfiability of $(\alpha_1 \wedge \cdots \wedge \alpha_k) \wedge (\neg\beta)$. If it is unsatisfiable, then $\{\alpha_1, \ldots, \alpha_k\} \models \beta$, otherwise $\{\alpha_1, \ldots, \alpha_k\} \not\models \beta$.

Determining Satisfiability using Truth Tables

What is the complexity of this algorithm?

Determining Satisfiability using Truth Tables

What is the complexity of this algorithm?

2^n where n is the number of propositional symbols.

Determining Satisfiability using Truth Tables

What is the complexity of this algorithm?

2^n where n is the number of propositional symbols.

Can you think of a way to speed up these algorithms?

Determining Satisfiability using Truth Tables

What is the complexity of this algorithm?

2^n where n is the number of propositional symbols.

Can you think of a way to speed up these algorithms?

In an upcoming lecture, we will discuss some of the applications and best-known techniques for the *SAT* algorithm.

Some tautologies

Associative and Commutative laws for $\wedge, \vee, \leftrightarrow$

Some tautologies

Associative and Commutative laws for $\wedge, \vee, \leftrightarrow$

Distributive Laws

- $(A \wedge (B \vee C)) \leftrightarrow ((A \wedge B) \vee (A \wedge C))$.
- $(A \vee (B \wedge C)) \leftrightarrow ((A \vee B) \wedge (A \vee C))$.

Some tautologies

Associative and Commutative laws for $\wedge, \vee, \leftrightarrow$

Distributive Laws

- $(A \wedge (B \vee C)) \leftrightarrow ((A \wedge B) \vee (A \wedge C))$.
- $(A \vee (B \wedge C)) \leftrightarrow ((A \vee B) \wedge (A \vee C))$.

Negation

- $\neg\neg A \leftrightarrow A$
- $\neg(A \to B) \leftrightarrow (A \wedge \neg B)$
- $\neg(A \leftrightarrow B) \leftrightarrow ((A \wedge \neg B) \vee (\neg A \wedge B))$

Some tautologies

Associative and Commutative laws for $\wedge, \vee, \leftrightarrow$

Distributive Laws

- $(A \wedge (B \vee C)) \leftrightarrow ((A \wedge B) \vee (A \wedge C))$.
- $(A \vee (B \wedge C)) \leftrightarrow ((A \vee B) \wedge (A \vee C))$.

Negation

- $\neg\neg A \leftrightarrow A$
- $\neg(A \rightarrow B) \leftrightarrow (A \wedge \neg B)$
- $\neg(A \leftrightarrow B) \leftrightarrow ((A \wedge \neg B) \vee (\neg A \wedge B))$

De Morgan's Laws

- $\neg(A \wedge B) \leftrightarrow (\neg A \vee \neg B)$
- $\neg(A \vee B) \leftrightarrow (\neg A \wedge \neg B)$

More Tautologies

Implication

- $(A \to B) \leftrightarrow (\neg A \vee B)$

More Tautologies

Implication

- $(A \to B) \leftrightarrow (\neg A \lor B)$

Excluded Middle

- $A \lor \neg A$

More Tautologies

Implication

- $(A \to B) \leftrightarrow (\neg A \vee B)$

Excluded Middle

- $A \vee \neg A$

Contradiction

- $\neg(A \wedge \neg A)$

More Tautologies

Implication

- $(A \rightarrow B) \leftrightarrow (\neg A \vee B)$

Excluded Middle

- $A \vee \neg A$

Contradiction

- $\neg(A \wedge \neg A)$

Contraposition

- $(A \rightarrow B) \leftrightarrow (\neg B \rightarrow \neg A)$

More Tautologies

Implication

- $(A \to B) \leftrightarrow (\neg A \lor B)$

Excluded Middle

- $A \lor \neg A$

Contradiction

- $\neg(A \land \neg A)$

Contraposition

- $(A \to B) \leftrightarrow (\neg B \to \neg A)$

Exportation

- $((A \land B) \to C) \leftrightarrow (A \to (B \to C))$

Propositional Connectives

We have five connectives: $\neg, \wedge, \vee, \rightarrow, \leftrightarrow$. Would we gain anything by having more? Would we lose anything by having fewer?

Propositional Connectives

We have five connectives: $\neg, \wedge, \vee, \rightarrow, \leftrightarrow$. Would we gain anything by having more? Would we lose anything by having fewer?

Example: Ternary Majority Connective $\#$

$\mathcal{E}_{\#}(\alpha, \beta, \gamma) = (\#\alpha\beta\gamma)$

$\overline{v}((\#\alpha\beta\gamma)) = \mathbf{T}$ iff the majority of $\overline{v}(\alpha)$, $\overline{v}(\beta)$, and $\overline{v}(\gamma)$ are $\mathbf{T}$.

Propositional Connectives

We have five connectives: $\neg, \wedge, \vee, \rightarrow, \leftrightarrow$. Would we gain anything by having more? Would we lose anything by having fewer?

Example: Ternary Majority Connective $\#$

$\mathcal{E}_{\#}(\alpha, \beta, \gamma) = (\# \alpha \beta \gamma)$

$\overline{v}((\# \alpha \beta \gamma)) =$ **T** iff the majority of $\overline{v}(\alpha)$, $\overline{v}(\beta)$, and $\overline{v}(\gamma)$ are **T**.

What does this new connective do for us?

Propositional Connectives

We have five connectives: $\neg, \wedge, \vee, \rightarrow, \leftrightarrow$. Would we gain anything by having more? Would we lose anything by having fewer?

Example: Ternary Majority Connective $\#$

$\mathcal{E}_{\#}(\alpha, \beta, \gamma) = (\#\alpha\beta\gamma)$

$\overline{v}((\#\alpha\beta\gamma)) =$ **T** iff the majority of $\overline{v}(\alpha)$, $\overline{v}(\beta)$, and $\overline{v}(\gamma)$ are **T**.

What does this new connective do for us?

The extended language obtained by allowing this new symbol has the same expressive power as the original language.

Propositional Connectives

We have five connectives: $\neg, \wedge, \vee, \rightarrow, \leftrightarrow$. Would we gain anything by having more? Would we lose anything by having fewer?

Example: Ternary Majority Connective $\#$

$\mathcal{E}_{\#}(\alpha, \beta, \gamma) = (\#\alpha\beta\gamma)$

$\overline{v}((\#\alpha\beta\gamma)) = \mathbf{T}$ iff the majority of $\overline{v}(\alpha)$, $\overline{v}(\beta)$, and $\overline{v}(\gamma)$ are $\mathbf{T}$.

What does this new connective do for us?

The extended language obtained by allowing this new symbol has the same expressive power as the original language.

Every Boolean function can be realized by a *wff* which uses only the connectives $\{\neg, \wedge, \vee\}$. We say that this set of connectives is *complete*.

In fact, we can do better. It turns out that $\{\neg, \wedge\}$ and $\{\neg, \vee\}$ are complete as well.

A formula is in DNF if it is a disjunction of formulas, each of which is a conjunction of *literals*, where a literal is either a propositional symbol or its negation.

In fact, we can do better. It turns out that $\{\neg, \wedge\}$ and $\{\neg, \vee\}$ are complete as well.

Why?

A formula is in DNF if it is a disjunction of formulas, each of which is a conjunction of *literals*, where a literal is either a propositional symbol or its negation.

In fact, we can do better. It turns out that $\{\neg, \wedge\}$ and $\{\neg, \vee\}$ are complete as well.

Why?

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

Using these identities, the completeness can be easily proved by induction.

A formula is in DNF if it is a disjunction of formulas, each of which is a conjunction of *literals*, where a literal is either a propositional symbol or its negation.

Completeness of Propositional Connectives

Example

Let G be a 3-place Boolean function defined as follows:

$G(\mathbf{F}, \mathbf{F}, \mathbf{F}) = \mathbf{F}$
$G(\mathbf{F}, \mathbf{F}, \mathbf{T}) = \mathbf{T}$
$G(\mathbf{F}, \mathbf{T}, \mathbf{F}) = \mathbf{T}$
$G(\mathbf{F}, \mathbf{T}, \mathbf{T}) = \mathbf{F}$
$G(\mathbf{T}, \mathbf{F}, \mathbf{F}) = \mathbf{T}$
$G(\mathbf{T}, \mathbf{F}, \mathbf{T}) = \mathbf{F}$
$G(\mathbf{T}, \mathbf{T}, \mathbf{F}) = \mathbf{F}$
$G(\mathbf{T}, \mathbf{T}, \mathbf{T}) = \mathbf{T}$

Completeness of Propositional Connectives

Example

Let G be a 3-place Boolean function defined as follows:

$G(\mathbf{F}, \mathbf{F}, \mathbf{F}) = \mathbf{F}$
$G(\mathbf{F}, \mathbf{F}, \mathbf{T}) = \mathbf{T}$
$G(\mathbf{F}, \mathbf{T}, \mathbf{F}) = \mathbf{T}$
$G(\mathbf{F}, \mathbf{T}, \mathbf{T}) = \mathbf{F}$
$G(\mathbf{T}, \mathbf{F}, \mathbf{F}) = \mathbf{T}$
$G(\mathbf{T}, \mathbf{F}, \mathbf{T}) = \mathbf{F}$
$G(\mathbf{T}, \mathbf{T}, \mathbf{F}) = \mathbf{F}$
$G(\mathbf{T}, \mathbf{T}, \mathbf{T}) = \mathbf{T}$
There are four points at which G is true, so a DNF formula which realizes G is

$$(\neg A_1 \wedge \neg A_2 \wedge A_3) \vee (\neg A_1 \wedge A_2 \wedge \neg A_3) \vee (A_1 \wedge \neg A_2 \wedge \neg A_3) \vee (A_1 \wedge A_2 \wedge A_3).$$

Completeness of Propositional Connectives

Example

Let G be a 3-place Boolean function defined as follows:

$G(\mathbf{F}, \mathbf{F}, \mathbf{F}) = \mathbf{F}$
$G(\mathbf{F}, \mathbf{F}, \mathbf{T}) = \mathbf{T}$
$G(\mathbf{F}, \mathbf{T}, \mathbf{F}) = \mathbf{T}$
$G(\mathbf{F}, \mathbf{T}, \mathbf{T}) = \mathbf{F}$
$G(\mathbf{T}, \mathbf{F}, \mathbf{F}) = \mathbf{T}$
$G(\mathbf{T}, \mathbf{F}, \mathbf{T}) = \mathbf{F}$
$G(\mathbf{T}, \mathbf{T}, \mathbf{F}) = \mathbf{F}$
$G(\mathbf{T}, \mathbf{T}, \mathbf{T}) = \mathbf{T}$

There are four points at which G is true, so a DNF formula which realizes G is

$$(\neg A_1 \wedge \neg A_2 \wedge A_3) \vee (\neg A_1 \wedge A_2 \wedge \neg A_3) \vee (A_1 \wedge \neg A_2 \wedge \neg A_3) \vee (A_1 \wedge A_2 \wedge A_3).$$

Note that another formula which realizes G is $A_1 \leftrightarrow A_2 \leftrightarrow A_3$. Thus, adding additional connectives to a complete set may allow a function to be realized more concisely.