Propositional Logic: Well-Formed Formulas

Recall our inductive definition of the set W of well-formed formulas in propositional logic. Given the alphabet $\{(,), \neg, \wedge, \vee, \rightarrow, \leftrightarrow, A_1, A_2, \ldots\}$,

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

Propositional Logic: Well-Formed Formulas

Recall our inductive definition of the set W of well-formed formulas in propositional logic. Given the alphabet $\{(,), \neg, \wedge, \vee, \rightarrow, \leftrightarrow, A_1, A_2, \ldots\}$,

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

Propositional Logic: Well-Formed Formulas

Recall our inductive definition of the set W of well-formed formulas in propositional logic. Given the alphabet $\{(,), \neg, \wedge, \vee, \rightarrow, \leftrightarrow, A_1, A_2, \ldots\}$,

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

Propositional Logic: Well-Formed Formulas

Recall our inductive definition of the set W of well-formed formulas in propositional logic. Given the alphabet $\{(,), \neg, \wedge, \vee, \rightarrow, \leftrightarrow, A_1, A_2, \ldots\}$,

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

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*.

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*.

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 $\mathbf{T}$ or $\mathbf{F}$) 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 $\mathbf{F}$ and $\mathbf{T}$ 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.

Truth Tables

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

α	$\neg\alpha$
$\mathbf{T}$	
$\mathbf{F}$	

α	β	$\alpha \wedge \beta$
$\mathbf{T}$	$\mathbf{T}$	
$\mathbf{T}$	$\mathbf{F}$	
$\mathbf{F}$	$\mathbf{T}$	
$\mathbf{F}$	$\mathbf{F}$	

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

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

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

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

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

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

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 $\overline{v}(\alpha) = \mathbf{T}$.

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 $\bigwedge(\Sigma) \rightarrow \alpha$ is valid.

Examples

- $(A \vee B) \wedge (\neg A \vee \neg B)$ is satisfiable, but not valid.
- $(A \vee B) \wedge (\neg A \vee \neg B) \wedge (A \leftrightarrow B)$ 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.
- $\{A, A \rightarrow B\} \models B \qquad (A \wedge (A \rightarrow B) \wedge (\neg B))$
- $\{A, \neg A\} \models (A \wedge \neg A) \quad (A \wedge (\neg A) \wedge \neg(A \wedge \neg A))$
- $\neg(A \wedge B)$ 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.
- $\{A, A \rightarrow B\} \models B \qquad (A \wedge (A \rightarrow B) \wedge (\neg B))$
- $\{A, \neg A\} \models (A \wedge \neg A) \quad (A \wedge (\neg A) \wedge \neg(A \wedge \neg A))$
- $\neg(A \wedge B)$ 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.
- $\{A, A \rightarrow B\} \models B \qquad (A \wedge (A \rightarrow B) \wedge (\neg B))$
- $\{A, \neg A\} \models (A \wedge \neg A) \quad (A \wedge (\neg A) \wedge \neg(A \wedge \neg A))$
- $\neg(A \wedge B)$ 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 $\mathbf{T}$ 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 $(\alpha_1 \wedge \ldots \wedge \alpha_k) \wedge (\neg\beta)$. 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

Example

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

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

Determining Satisfiability using Truth Tables

Example

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

A	B	C	A	$\wedge$	$((B$	$\vee$	$\neg A)$	$\wedge$	$(C$	$\vee$	$\neg B))$
F	**F**	**F**		**F**		**T**	**T**	**T**		**T**	**T**

Determining Satisfiability using Truth Tables

Example

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

A	B	C	A	$\wedge$	$((B$	$\vee$	$\neg A)$	$\wedge$	$(C$	$\vee$	$\neg B))$
F	F	F		F		T	T	T		T	T
F	F	T		F		T	T	T		T	T
F	T	F		F		T	T	F		F	F
F	T	T		F		T	T	T		T	F
T	F	F		F		F	F	F		T	T
T	F	T		F		F	F	F		T	T
T	T	F		F		T	F	F		F	F
T	T	T		T		T	F	T		T	F

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

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

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

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

Excluded Middle

- $A \vee \neg A$

Contradiction

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

Contraposition

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

Exportation

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

Propositional Connectives

We have five connectives: $\neg, \wedge, \vee, \to, \leftrightarrow$. 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?

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

How do we show this formally?

Boolean Functions

For $k \geq 0$, a *k-place Boolean function* is a function from $\{\mathbf{F}, \mathbf{T}\}^k$ to $\{\mathbf{F}, \mathbf{T}\}$. A *Boolean function* then is anything which is a k-place Boolean function for some k.

Each *wff* α determines a corresponding Boolean function B_α. For example, if $\alpha = A_1 \wedge A_2$, then B_α is a 2-place Boolean function whose value is given by the following table.

X_1	X_2	$B_\alpha(X_1, X_2)$
T	**T**	**T**
T	**F**	**F**
F	**T**	**F**
F	**F**	**F**

Realizing Boolean Functions

In general, suppose that α is a *wff* whose propositional symbols are included in $A_1, \ldots, A_n$. We define an n-place Boolean function B^n_α, the Boolean function *realized* by α as

$B^n_\alpha(X_1, \ldots, X_n) =$ the truth value given to α when $A_1, \ldots, A_n$ are given the values $X_1, \ldots, X_n$.

In other words,

$B^n_\alpha(X_1, \ldots, X_n) = \overline{v}(\alpha)$ where $v(A_i) = X_i$.

Note that the function B^n_α is determined by *both* the formula α and the choice of n. In particular, α does not need to include all the symbols in $A_1, \ldots, A_n$.

Examples

- $I^n_i = B^n_{A_i}$
- $N = B^1_{\neg A_1}$
- $K = B^2_{A_1 \wedge A_2}$
- $A = B^2_{A_1 \vee A_2}$
- $C = B^2_{A_1 \rightarrow A_2}$
- $E = B^2_{A_1 \leftrightarrow A_2}$

From these functions, we can construct others by composition.

$$B^2_{\neg A_1 \vee \neg A_2}(X_1, X_2) = A(N(I^2_1(X_1, X_2)), N(I^2_2(X_1, X_2)))$$

Claim: Every Boolean function can be obtained as a composition of I, N, K, A, C, and E.

We will explain why this is true shortly.

Formulas and the Boolean Functions they Realize

Theorem

Let α and β be *wffs* whose sentence symbols are among $A_1, \ldots, A_n$.

(a) $\alpha \models \beta$ iff $B^n_\alpha(\vec{X}) \leq B^n_\beta(\vec{X})$ for all $\vec{X} \in \{\mathbf{F}, \mathbf{T}\}^n$.

(b) α is tautologically equivalent to β iff $B^n_\alpha = B^n_\beta$.

(c) $\models \beta$ iff the range of $B^n_\beta = \{\mathbf{T}\}$.

Proof

(a)

$\alpha \models \beta$ iff every truth assignment satisfying α also satisfies β
iff for every truth assignment v, $\overline{v}(\alpha) = \mathbf{T}$ implies $\overline{v}(\beta) = \mathbf{T}$
iff for all n-tuples $\vec{X}$, $B^n_\alpha(\vec{X}) = \mathbf{T}$ implies $B^n_\beta(\vec{X}) = \mathbf{T}$
iff for all n-tuples $\vec{X}$, $B^n_\alpha(\vec{X}) \leq B^n_\beta(\vec{X})$

(b) Follows from (a) and $X = Y$ iff $X \leq Y$ and $Y \leq X$.

(c) Follows from definition of tautology.

By shifting our focus from formulas to Boolean functions, tautologically equivalent *wffs* are identified.

Completeness of Propositional Connectives

Theorem (Post 1921)

Let G be an n-place Boolean function, $n \geq 1$. There exists a *wff* α such that $G = B^n_\alpha$, i.e., such that α realizes the function G.

Proof

If the range of G is just $\{\mathbf{F}\}$, then let $\alpha = A_1 \wedge \neg A_1$. Clearly, $B^n_\alpha = G$. Otherwise, $G = \mathbf{T}$ somewhere. Suppose there are k points where $G = \mathbf{T}$:

$$
\begin{aligned}
G(X_{11}, X_{12}, \ldots, X_{1n}) &= \mathbf{T} \\
G(X_{21}, X_{22}, \ldots, X_{2n}) &= \mathbf{T} \\
&\ldots \\
G(X_{k1}, X_{k2}, \ldots, X_{kn}) &= \mathbf{T}
\end{aligned}
$$

Let $\beta_{ij} = \begin{cases} A_j & \text{if } X_{ij} = \mathbf{T} \\ \neg A_j & \text{if } X_{ij} = \mathbf{F} \end{cases}$

$\gamma_i = \beta_{i1} \wedge \ldots \wedge \beta_{in}$

$\alpha = \gamma_1 \vee \gamma_2 \vee \ldots \vee \gamma_k$

Then α realizes G

Completeness of Propositional Connectives

Proof, continued

We know that $B^n_\alpha(\vec{X}) = \overline{v}(\alpha)$ where $v(A_i) = X_i$.

Since $\alpha = \gamma_1 \vee \gamma_2 \vee \ldots \vee \gamma_k$, it follows that $B^n_\alpha(\vec{X}) = \max(B^n_{\gamma_i}(\vec{X}))$.

But by construction, $B^n_{\gamma_i}(\vec{X}) = \mathbf{T}$ iff $\vec{X} = \langle X_{i1}, \ldots, X_{in} \rangle$.

Thus $B^n_\alpha(\vec{X}) = \mathbf{T}$ iff $\vec{X}$ is one of the points where G is $\mathbf{T}$.

This shows that every Boolean function can be realized by a *wff*. In fact, 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*.

The realizing formula is not unique. The formula built is in so-called *disjunctive normal form* (DNF). 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.

Thus, a corollary is that for every *wff*, there exists a tautologically equivalent *wff* in disjunctive normal form.

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.

Recall our definition of some basic Boolean functions:

- $I_i^n = B_{A_i}^n$
- $N = B_{\neg A_1}^1$
- $K = B_{A_1 \wedge A_2}^2$
- $A = B_{A_1 \vee A_2}^2$

Given that $\{\neg, \wedge, \vee\}$ is complete, it is not hard to see that any Boolean function can be constructed using only the Boolean functions I, N, K, and A.

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.

Incompleteness of Connectives

To prove that some set of connectives is incomplete, we find a property that is true of all *wffs* built using those connectives, but that is not true for some Boolean function.

Example

$\{\wedge, \rightarrow\}$ is not complete.

Proof

Let α be a *wff* which uses only these connectives, and let v be a truth assignment such that $v(A_i) = \mathbf{T}$ for all A_i. We prove by induction that $\overline{v}(\alpha) = \mathbf{T}$.

Base Case

$\overline{v}(A_i) = v(A_i) = \mathbf{T}$.

Inductive Case

$\overline{v}(\beta \wedge \gamma) = \min(\overline{v}(\beta), \overline{v}(\gamma)) = \min(\mathbf{T}, \mathbf{T}) = \mathbf{T}$
$\overline{v}(\beta \rightarrow \gamma) = \max(\mathbf{T} - \overline{v}(\alpha), \overline{v}(\beta)) = \max(\mathbf{F}, \mathbf{T}) = \mathbf{T}$

Thus, $\overline{v}(\alpha) = \mathbf{T}$ for all *wffs* α built from $\{\wedge, \rightarrow\}$. But $\overline{v}(\neg A_1) = \mathbf{F}$, so there is no such formula tautologically equivalent to $\neg A_1$.

Other Propositional Connectives

For each n, there are 2^{2^n} different n-place Boolean functions $B(X_1, \ldots, X_n)$

Why?

There are 2^n different input points and 2 possible output values for each input point. 2^{2^n} is also the number of possible n-ary propositional connectives.

0-ary connectives

There are two 0-place Boolean functions: the constants $\mathbf{F}$ and $\mathbf{T}$. We can construct corresponding 0-ary connectives $\bot$ and $\top$ with the meaning that $\overline{v}(\bot) = \mathbf{F}$ and $\overline{v}(\top) = \mathbf{T}$ regardless of the truth assignment v.

Unary connectives

There are four 1-place functions, but these include the two constant functions mentioned above and the identity function. Thus the only additional connective of interest is negation: $\neg$.

Binary connectives

There are sixteen 2-place Boolean functions. They are cataloged in the following table. Note that the first six correspond to 0-ary and unary connectives.

Symbol	Equivalent	Description
	$\bot$	constant $\mathbf{F}$
	$\top$	constant $\mathbf{T}$
	A	projection of first argument
	B	projection of second argument
	$\neg A$	negation of first argument
	$\neg B$	negation of second argument
$\wedge$	$A \wedge B$	and
$\vee$	$A \vee B$	or
$\rightarrow$	$A \rightarrow B$	conditional
$\leftrightarrow$	$A \leftrightarrow B$	bi-conditional
$\leftarrow$	$B \rightarrow A$	reverse conditional
$\oplus$	$(A \wedge \neg B) \vee (\neg A \wedge B)$	exclusive or
$\downarrow$	$\neg(A \vee B)$	nor (or Nicod stroke)
$\mid$	$\neg(A \wedge B)$	nand (or Sheffer stroke)
$<$	$\neg A \wedge B$	less than
$>$	$A \wedge \neg B$	greater than

Compactness

Recall that a *wff* α is satisfiable if there exists a truth assignment v such that $\overline{v}(\alpha) = \mathbf{T}$.

A set Σ of *wffs* is satisfiable if there exists a truth assignment v such that $\overline{v}(\alpha) = \mathbf{T}$ for each $\alpha \in \Sigma$.

A set Σ is *finitely satisfiable* iff every finite subset of Σ is satisfiable.

Compactness Theorem

A set of *wffs* is satisfiable iff it is finitely satisfiable.

Proof

The *only if* direction is trivial since any subset of a satisfiable set is clearly satisfiable.

To prove the other direction, assume that Σ is a set which is finitely satisfiable. We must show that Σ is satisfiable.

Compactness

Let Σ be finitely satisfiable. We extend Σ to form a *maximal* finitely satisfiable set Δ as follows.

Let $\alpha_1, \ldots, \alpha_n, \ldots$ be a fixed enumeration of all *wffs*.

Why is this possible?

The set of all sequences of a countable set is countable.

Then, let $\Delta_0 = \Sigma$,

$$\Delta_{n+1} = \begin{cases} \Delta_n \cup \{\alpha_{n+1}\} & \text{if this is finitely satisfiable,} \\ \Delta_n \cup \{\neg\alpha_{n+1}\} & \text{otherwise.} \end{cases}$$

It is not hard to show that each Δ_n is finitely satisfiable.

Let $\Delta = \bigcup_n \Delta_n$. It is then clear that

1. $\Sigma \subseteq \Delta$
2. $\alpha \in \Delta$ or $\neg\alpha \in \Delta$ for any *wff* α, and
3. Δ is finitely satisfiable.

Compactness

Now we show that Δ is satisfiable (and thus $\Sigma \subseteq \Delta$ is also satisfiable).

Define a truth assignment v as follows. For each propositional symbol A_i,

$$v(A_i) = \mathbf{T} \text{ iff } A_i \in \Delta.$$

We claim that for any *wff* α, v satisfies α iff $\alpha \in \Delta$. The proof is by induction on well-formed formulas.

Base Case

Follows directly from the definition of v.

Induction Case

We will just consider one case. Suppose $\alpha = \beta \wedge \gamma$. Then

$\overline{v}(\alpha) = \mathbf{T}$ iff both $\overline{v}(\beta) = \mathbf{T}$ and $\overline{v}(\gamma) = \mathbf{T}$ iff both $\beta \in \Delta$ and $\gamma \in \Delta$.

Now, if both β and γ are in Δ, then since $\{\beta, \gamma, \neg\alpha\}$ is not satisfiable, we must have $\alpha \in \Delta$.

Similarly, if one of β or γ is not in Δ, then its negation must be in Δ, so $\alpha \notin \Delta$.

Corollary

If $\Sigma \models \alpha$ then there is a finite $\Sigma_0 \subseteq \Sigma$ such that $\Sigma_0 \models \alpha$.

Proof

Suppose that $\Sigma_0 \not\models \alpha$ for every finite $\Sigma_0 \subseteq \Sigma$.

Then, $\Sigma_0 \cup \{\neg\alpha\}$ is satisfiable for every finite $\Sigma_0 \subseteq \Sigma$.

So, by compactness, $\Sigma \cup \{\neg\alpha\}$ is satisfiable which contradicts the fact that $\Sigma \models \alpha$.