CS 357: Advanced Topics in Formal Methods

Fall 2019

Lecture 3

Aleksandar Zeljić
(materials by Clark Barrett)
Stanford University

Outline

- Recap
- Propositional connectives (cont.)
- Compactness
- CNF, Converting to CNF
- Modeling using propositional logic
- Computability and Decidability

Material is drawn from Chapter 1 of Enderton.

Truth Tables

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

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**

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.

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.

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.

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.

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

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

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

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

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

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

$\{\wedge, \to\}$ 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

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

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

Other Propositional Connectives

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

Why?

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.

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.

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 $\perp$ and $\top$ with the meaning that $\overline{v}(\perp) = \mathbf{F}$ 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: $\neg$.

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 **F**
	$\top$	constant **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 $\overline{v}(\alpha) = \mathbf{T}$.

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 $\alpha \in \Sigma$.

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

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.

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.

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?

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.

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
$$\begin{aligned} \Delta_0 &= \Sigma, \\ \Delta_{n+1} &= \begin{cases} \Delta_n \cup \{\alpha_{n+1}\} & \text{if this is finitely satisfiable,} \\ \Delta_n \cup \{\neg\alpha_{n+1}\} & \text{otherwise.} \end{cases} \end{aligned}$$

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
$$\begin{aligned} \Delta_0 &= \Sigma, \\ \Delta_{n+1} &= \begin{cases} \Delta_n \cup \{\alpha_{n+1}\} & \text{if this is finitely satisfiable,} \\ \Delta_n \cup \{\neg\alpha_{n+1}\} & \text{otherwise.} \end{cases} \end{aligned}$$

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.

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.

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 $\alpha \notin \Delta$. $\square$

Compactness

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 $\Sigma_0 \subseteq \Sigma$.

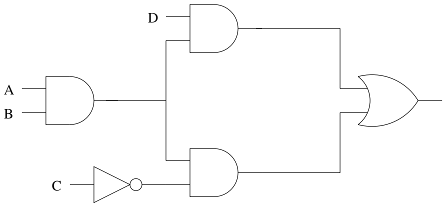
Then, $\Sigma_0 \cup \{\neg\alpha\}$ is satisfiable for every finite $\Sigma_0 \subseteq \Sigma$.

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

$\square$

Boolean Circuits

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



There is a natural correspondence between Boolean circuits and formulas of propositional logic. The formula corresponding to the above circuit is:

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

A satisfying assignment for this formula gives the values that must be applied to the inputs of the circuit in order to set the output of the circuit to true.

In this lecture, we will refer to propositional symbols such as A, B, etc. as *propositional variables*.

Sharing Sub-Expressions

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

This formula highlights an inefficiency in the logic representation as compared with the circuit representation: the formula $A \wedge B$ appears twice. For larger circuits, this kind of redundancy can result in an exponential blow-up in the size of the corresponding formula.

Sharing Sub-Expressions

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

This formula highlights an inefficiency in the logic representation as compared with the circuit representation: the formula $A \wedge B$ appears twice. For larger circuits, this kind of redundancy can result in an exponential blow-up in the size of the corresponding formula.

We can overcome this inefficiency by replacing the redundant sub-expression with a new place-holder variable. We then conjoin a new formula which says that the new variable is equivalent to the replaced expression:

$$((D \wedge E) \vee (E \wedge \neg C)) \wedge (E \leftrightarrow (A \wedge B))$$

Sharing Sub-Expressions

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

This formula highlights an inefficiency in the logic representation as compared with the circuit representation: the formula $A \wedge B$ appears twice. For larger circuits, this kind of redundancy can result in an exponential blow-up in the size of the corresponding formula.

We can overcome this inefficiency by replacing the redundant sub-expression with a new place-holder variable. We then conjoin a new formula which says that the new variable is equivalent to the replaced expression:

$$((D \wedge E) \vee (E \wedge \neg C)) \wedge (E \leftrightarrow (A \wedge B))$$

Note that the new formula is *not* tautologically equivalent to the original formula (why?).

Sharing Sub-Expressions

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

This formula highlights an inefficiency in the logic representation as compared with the circuit representation: the formula $A \wedge B$ appears twice. For larger circuits, this kind of redundancy can result in an exponential blow-up in the size of the corresponding formula.

We can overcome this inefficiency by replacing the redundant sub-expression with a new place-holder variable. We then conjoin a new formula which says that the new variable is equivalent to the replaced expression:

$$((D \wedge E) \vee (E \wedge \neg C)) \wedge (E \leftrightarrow (A \wedge B))$$

Note that the new formula is *not* tautologically equivalent to the original formula (why?).

But it *is* equisatisfiable (i.e. the original formula is satisfiable iff the new formula is satisfiable). Since we are only concerned with the *satisfiability* of the formula, this is sufficient.

Converting to CNF

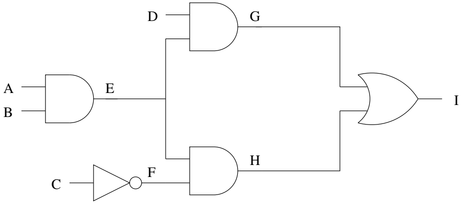
This same idea is behind a simple algorithm for converting any propositional formula (or an associated Boolean circuit) into an equisatisfiable formula in conjunctive normal form (CNF) in linear time and space. We will view the formula or circuit as a DAG.

1. Label each non-leaf node of the DAG with a new propositional variable.

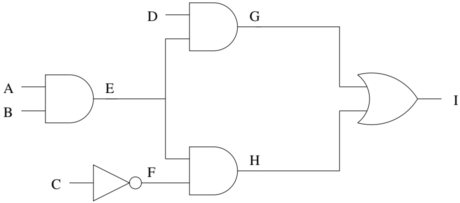
2. Construct a conjunction of disjunctive clauses which relate the inputs of that node to its output (the new propositional variable)

3. The conjunction of all of these clauses together with a single clause consisting of the variable for the root node is satisfiable iff the original formula is satisfiable.

Converting to CNF: Example

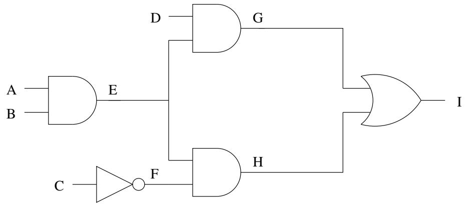


Converting to CNF: Example



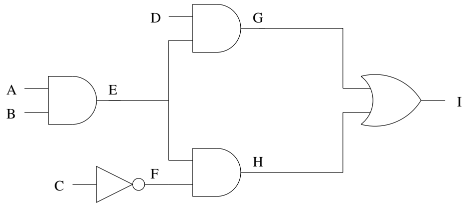
$(A \land B) \leftrightarrow E$

Converting to CNF: Example



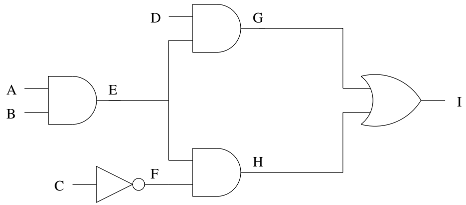
$(A \wedge B) \leftrightarrow E$
$((A \wedge B) \rightarrow E) \wedge (E \rightarrow (A \wedge B))$

Converting to CNF: Example



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

Converting to CNF: Example



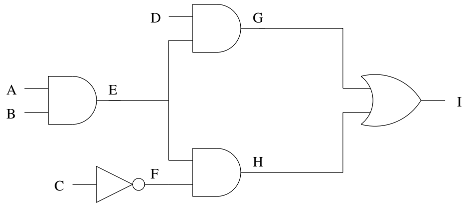
$(A \wedge B) \leftrightarrow E$

$((A \wedge B) \rightarrow E) \wedge (E \rightarrow (A \wedge B))$

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

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

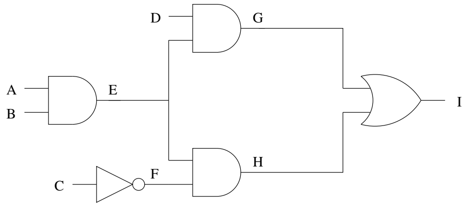
Converting to CNF: Example



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

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

Converting to CNF: Example



$(A \wedge B) \leftrightarrow E$

$((A \wedge B) \rightarrow E) \wedge (E \rightarrow (A \wedge B))$

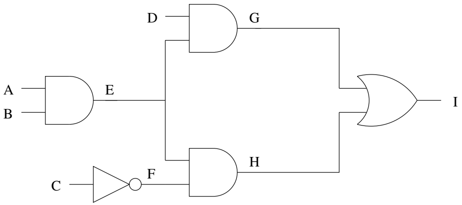
$(\neg(A \wedge B) \vee E) \wedge (\neg E \vee (A \wedge B))$

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

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

$(\neg C \vee F) \wedge (\neg F \vee C) \wedge$

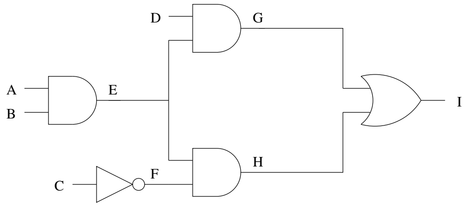
Converting to CNF: Example



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

$(\neg A \vee \neg B \vee E) \wedge (\neg E \vee A) \wedge (\neg E \vee B) \wedge$
$(\neg C \vee F) \wedge (\neg F \vee C) \wedge$
$(\neg D \vee \neg E \vee G) \wedge (\neg G \vee D) \wedge (\neg G \vee E) \wedge$
$(\neg E \vee \neg F \vee H) \wedge (\neg H \vee E) \wedge (\neg H \vee F) \wedge$

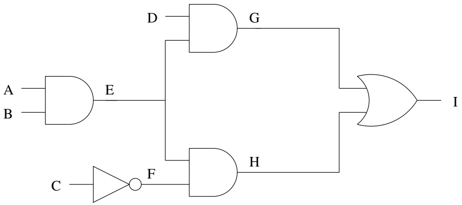
Converting to CNF: Example



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

$(\neg A \vee \neg B \vee E) \wedge (\neg E \vee A) \wedge (\neg E \vee B) \wedge$
$(\neg C \vee F) \wedge (\neg F \vee C) \wedge$
$(\neg D \vee \neg E \vee G) \wedge (\neg G \vee D) \wedge (\neg G \vee E) \wedge$
$(\neg E \vee \neg F \vee H) \wedge (\neg H \vee E) \wedge (\neg H \vee F) \wedge$
$(G \vee H \vee \neg I) \wedge (I \vee \neg G) \wedge (I \vee \neg H) \wedge$

Converting to CNF: Example



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

$(\neg A \vee \neg B \vee E) \wedge (\neg E \vee A) \wedge (\neg E \vee B) \wedge$
$(\neg C \vee F) \wedge (\neg F \vee C) \wedge$
$(\neg D \vee \neg E \vee G) \wedge (\neg G \vee D) \wedge (\neg G \vee E) \wedge$
$(\neg E \vee \neg F \vee H) \wedge (\neg H \vee E) \wedge (\neg H \vee F) \wedge$
$(G \vee H \vee \neg I) \wedge (I \vee \neg G) \wedge (I \vee \neg H) \wedge$
(I)

CNF Representation

$$(\neg A \vee \neg B \vee E) \wedge (\neg E \vee A) \wedge (\neg E \vee B) \wedge$$
$$(\neg C \vee F) \wedge (\neg F \vee C) \wedge$$
$$(\neg D \vee \neg E \vee G) \wedge (\neg G \vee D) \wedge (\neg G \vee E) \wedge$$
$$(\neg E \vee \neg F \vee H) \wedge (\neg H \vee E) \wedge (\neg H \vee F) \wedge$$
$$(G \vee H \vee \neg I) \wedge (I \vee \neg G) \wedge (I \vee \neg H) \wedge$$
$$(I)$$

$$(A' + B' + E)(E' + A)(E' + B)$$
$$(C' + F)(F' + C)$$
$$(D' + E' + G)(G' + D)(G' + E)$$
$$(E' + F' + H)(H' + E)(H' + F)$$
$$(G + H + I')(I + G')(I + H')$$
$$(I)$$

Standard Representation

Each variable is represented by a positive integer. A negative integer refers to the negation of the variable. Clauses are given as sequences of integers separated by spaces. A 0 terminates the clause.

$(A' + B' + E)(E' + A)(E' + B)$	-1 -2 5 0	-5 1 0	-5 2 0
$(C' + F)(F' + C)$	-3 6 0	-6 3 0	
$(D' + E' + G)(G' + D)(G' + E)$	-4 -5 7 0	-7 4 0	-7 5 0
$(E' + F' + H)(H' + E)(H' + F)$	-5 -6 8 0	-8 5 0	-8 6 0
$(G + H + I')(I + G')(I + H')$	7 8 -9 0	9 -7 0	9 -8 0
(I)	9 0		

Boolean Satisfiability (SAT)

We have seen that there is a natural correspondence between checking Boolean circuits and satisfiability of propositional formulas.

It turns out that Boolean satisfiability or *SAT* is widely useful for a variety of problems.

SAT was the first problem ever shown to be $\mathcal{NP}$-complete:

> *S. A. Cook. The Complexity of Theorem Proving Procedures.* Proceedings of the Third Annual ACM Symposium on the Theory of Computing, *151-158, 1971.*

This means that:

- Unless $\mathcal{P} = \mathcal{NP}$, we will never find a polynomial algorithm to solve SAT.
- If we can nonetheless improve algorithms for SAT, there are many other problems that could benefit.

Worst Case Upper Bounds for SAT

A weakly exponential upper bound is a bound of the form $p(n)c^n$ where $c < 2$ is a constant, n is the number of variables, and p is a polynomial. A k-SAT solver solves SAT instances in which no clause has length greater than k. Some interesting best-known bounds are as follows.

- General SAT: $p(n)2^n$
- k-SAT: $p(n)(2 - \frac{2}{k+1})^n$
- 3-SAT: $p(n)1.481^n$
- 3-SAT formula with exactly one satisfying assignment: $p(n)1.308^n$

Solving General Search Problems with SAT

Modeling

- Define a finite set of possibilities called *states*.
- Model states using (vectors of) propositional variables.
- Use propositional formulas to describe legal and illegal states.

Solving

- Construct a propositional formula describing the desired state.
- Translate the formula into an equisatisfiable CNF formula.
- If the formula is satisfiable, the satisfying assignment gives the desired state.
- If the formula is not satisfiable, the desired state does not exist.

Example

Recall that a *graph* consists of a set V of vertices and a set E of edges, where each edge is an *unordered* pair of *distinct* vertices.

A *complete graph* on n vertices is a graph with $|V| = n$ such that E contains all possible pairs of vertices.

Example

Recall that a *graph* consists of a set V of vertices and a set E of edges, where each edge is an *unordered* pair of *distinct* vertices.

A *complete graph* on n vertices is a graph with $|V| = n$ such that E contains all possible pairs of vertices.

How many edges are in a complete graph?

Example

Recall that a *graph* consists of a set V of vertices and a set E of edges, where each edge is an *unordered* pair of *distinct* vertices.

A *complete graph* on n vertices is a graph with $|V| = n$ such that E contains all possible pairs of vertices.

How many edges are in a complete graph? $\frac{n(n-1)}{2}$

Example

Recall that a *graph* consists of a set V of vertices and a set E of edges, where each edge is an *unordered* pair of *distinct* vertices.

A *complete graph* on n vertices is a graph with $|V| = n$ such that E contains all possible pairs of vertices.

How many edges are in a complete graph? $\frac{n(n-1)}{2}$

Problems involving *graph coloring* are important in both theoretical and applied computer science.

Suppose we wish to color each edge of a complete graph without creating any triangles in which all the edges have the same color.

What is the largest complete graph for which this is possible? The answer depends on the number of colors we are allowed to use.

What if you are only allowed one color?

Example

Recall that a *graph* consists of a set V of vertices and a set E of edges, where each edge is an *unordered* pair of *distinct* vertices.

A *complete graph* on n vertices is a graph with $|V| = n$ such that E contains all possible pairs of vertices.

How many edges are in a complete graph? $\frac{n(n-1)}{2}$

Problems involving *graph coloring* are important in both theoretical and applied computer science.

Suppose we wish to color each edge of a complete graph without creating any triangles in which all the edges have the same color.

What is the largest complete graph for which this is possible? The answer depends on the number of colors we are allowed to use.

What if you are only allowed one color? **Answer:** $n = 2$

Example

Recall that a *graph* consists of a set V of vertices and a set E of edges, where each edge is an *unordered* pair of *distinct* vertices.

A *complete graph* on n vertices is a graph with $|V| = n$ such that E contains all possible pairs of vertices.

How many edges are in a complete graph? $\frac{n(n-1)}{2}$

Problems involving *graph coloring* are important in both theoretical and applied computer science.

Suppose we wish to color each edge of a complete graph without creating any triangles in which all the edges have the same color.

What is the largest complete graph for which this is possible? The answer depends on the number of colors we are allowed to use.

What if you are only allowed one color? **Answer:** $n = 2$

What if the number of colors is *2*?

Example

Recall that a *graph* consists of a set V of vertices and a set E of edges, where each edge is an *unordered* pair of *distinct* vertices.

A *complete graph* on n vertices is a graph with $|V| = n$ such that E contains all possible pairs of vertices.

How many edges are in a complete graph? $\frac{n(n-1)}{2}$

Problems involving *graph coloring* are important in both theoretical and applied computer science.

Suppose we wish to color each edge of a complete graph without creating any triangles in which all the edges have the same color.

What is the largest complete graph for which this is possible? The answer depends on the number of colors we are allowed to use.

What if you are only allowed one color? **Answer:** $n = 2$

What if the number of colors is *2*? **Answer:** $n = 5$

Example

Recall that a *graph* consists of a set V of vertices and a set E of edges, where each edge is an *unordered* pair of *distinct* vertices.

A *complete graph* on n vertices is a graph with $|V| = n$ such that E contains all possible pairs of vertices.

How many edges are in a complete graph? $\frac{n(n-1)}{2}$

Problems involving *graph coloring* are important in both theoretical and applied computer science.

Suppose we wish to color each edge of a complete graph without creating any triangles in which all the edges have the same color.

What is the largest complete graph for which this is possible? The answer depends on the number of colors we are allowed to use.

What if you are only allowed one color? **Answer:** $n = 2$

What if the number of colors is *2*? **Answer:** $n = 5$

What if the number of colors is *3*?

Example

Recall that a *graph* consists of a set V of vertices and a set E of edges, where each edge is an *unordered* pair of *distinct* vertices.

A *complete graph* on n vertices is a graph with $|V| = n$ such that E contains all possible pairs of vertices.

How many edges are in a complete graph? $\frac{n(n-1)}{2}$

Problems involving *graph coloring* are important in both theoretical and applied computer science.

Suppose we wish to color each edge of a complete graph without creating any triangles in which all the edges have the same color.

What is the largest complete graph for which this is possible? The answer depends on the number of colors we are allowed to use.

What if you are only allowed one color? **Answer:** $n = 2$

What if the number of colors is *2*? **Answer:** $n = 5$

What if the number of colors is *3*? **This is a job for SAT**

Example

- *Define a finite set of possibilities called states.*
- *Model states using (vectors of) propositional variables.*
- *Use propositional formulas to describe legal and illegal states.*

Example

- *Define a finite set of possibilities called states.*
  For this problem, each possible coloring is a state. There are $3^{|E|}$ possible states.
- *Model states using (vectors of) propositional variables.*
- *Use propositional formulas to describe legal and illegal states.*

Example

- *Define a finite set of possibilities called states.*
  For this problem, each possible coloring is a state. There are $3^{|E|}$ possible states.
- *Model states using (vectors of) propositional variables.*
  A simple encoding uses two propositional variables for each edge. Since there are 4 possible combinations of values of two variables, this gives us a state space of $4^{|E|}$, which is larger than we need, but keeps the encoding simple.
- *Use propositional formulas to describe legal and illegal states.*

Example

- *Define a finite set of possibilities called states.*
  For this problem, each possible coloring is a state. There are $3^{|E|}$ possible states.
- *Model states using (vectors of) propositional variables.*
  A simple encoding uses two propositional variables for each edge. Since there are 4 possible combinations of values of two variables, this gives us a state space of $4^{|E|}$, which is larger than we need, but keeps the encoding simple.
- *Use propositional formulas to describe legal and illegal states.*
  Since the color of each edge is modeled with 2 variables, there are 4 possible colors. We can write a set of formulas which disallow the fourth color.
  For example, if e_1 and e_2 are the variables for edge e, we simply require $\neg(e_1 \wedge e_2)$.

Example

- *Construct a propositional formula describing the desired state.*
- *Translate the formula into an equisatisfiable CNF formula.*
- *If the formula is satisfiable, the satisfying assignment gives the desired state.*
- *If the formula is not satisfiable, the desired state does not exist.*

Example

- *Construct a propositional formula describing the desired state.*
  The desired state is one in which there are no triangles of the same color.
  For each triangle made up of edges e, f, g, we require:
  $\neg((e_1 \leftrightarrow f_1) \wedge (f_1 \leftrightarrow g_1) \wedge (e_2 \leftrightarrow f_2) \wedge (f_2 \leftrightarrow g_2))$.
- *Translate the formula into an equisatisfiable CNF formula.*
- *If the formula is satisfiable, the satisfying assignment gives the desired state.*
- *If the formula is not satisfiable, the desired state does not exist.*

Example

- *Construct a propositional formula describing the desired state.*
  The desired state is one in which there are no triangles of the same color.
  For each triangle made up of edges e, f, g, we require:
  $\neg((e_1 \leftrightarrow f_1) \wedge (f_1 \leftrightarrow g_1) \wedge (e_2 \leftrightarrow f_2) \wedge (f_2 \leftrightarrow g_2))$.
- *Translate the formula into an equisatisfiable CNF formula.*
  This can be done using the CNF conversion algorithm we described earlier.
- *If the formula is satisfiable, the satisfying assignment gives the desired state.*
- *If the formula is not satisfiable, the desired state does not exist.*

Example

- *Construct a propositional formula describing the desired state.*
  The desired state is one in which there are no triangles of the same color.
  For each triangle made up of edges e, f, g, we require:
  $\neg((e_1 \leftrightarrow f_1) \wedge (f_1 \leftrightarrow g_1) \wedge (e_2 \leftrightarrow f_2) \wedge (f_2 \leftrightarrow g_2))$.
- *Translate the formula into an equisatisfiable CNF formula.*
  This can be done using the CNF conversion algorithm we described earlier.
- *If the formula is satisfiable, the satisfying assignment gives the desired state.*
  An actual coloring can be constructed by looking at the values of each variable given by the satisfying assignment.
- *If the formula is not satisfiable, the desired state does not exist.*

Example

- *Construct a propositional formula describing the desired state.*
  The desired state is one in which there are no triangles of the same color. For each triangle made up of edges e, f, g, we require:
  $\neg((e_1 \leftrightarrow f_1) \wedge (f_1 \leftrightarrow g_1) \wedge (e_2 \leftrightarrow f_2) \wedge (f_2 \leftrightarrow g_2))$.
- *Translate the formula into an equisatisfiable CNF formula.*
  This can be done using the CNF conversion algorithm we described earlier.
- *If the formula is satisfiable, the satisfying assignment gives the desired state.*
  An actual coloring can be constructed by looking at the values of each variable given by the satisfying assignment.
- *If the formula is not satisfiable, the desired state does not exist.*
  If the formula can be shown to be unsatisfiable, this is essentially a proof that there is no coloring.

Example

- *Construct a propositional formula describing the desired state.*
  The desired state is one in which there are no triangles of the same color.
  For each triangle made up of edges e, f, g, we require:
  $\neg((e_1 \leftrightarrow f_1) \wedge (f_1 \leftrightarrow g_1) \wedge (e_2 \leftrightarrow f_2) \wedge (f_2 \leftrightarrow g_2))$.
- *Translate the formula into an equisatisfiable CNF formula.*
  This can be done using the CNF conversion algorithm we described earlier.
- *If the formula is satisfiable, the satisfying assignment gives the desired state.*
  An actual coloring can be constructed by looking at the values of each variable given by the satisfying assignment.
- *If the formula is not satisfiable, the desired state does not exist.*
  If the formula can be shown to be unsatisfiable, this is essentially a proof that there is no coloring.

What if the number of colors is *3*?

Example

- *Construct a propositional formula describing the desired state.*
  The desired state is one in which there are no triangles of the same color.
  For each triangle made up of edges e, f, g, we require:
  $\neg((e_1 \leftrightarrow f_1) \wedge (f_1 \leftrightarrow g_1) \wedge (e_2 \leftrightarrow f_2) \wedge (f_2 \leftrightarrow g_2))$.
- *Translate the formula into an equisatisfiable CNF formula.*
  This can be done using the CNF conversion algorithm we described earlier.
- *If the formula is satisfiable, the satisfying assignment gives the desired state.*
  An actual coloring can be constructed by looking at the values of each variable given by the satisfying assignment.
- *If the formula is not satisfiable, the desired state does not exist.*
  If the formula can be shown to be unsatisfiable, this is essentially a proof that there is no coloring.

What if the number of colors is *3*? **Answer:** $n = 16$

Example

- *Construct a propositional formula describing the desired state.*
  The desired state is one in which there are no triangles of the same color. For each triangle made up of edges e, f, g, we require:
  $\neg((e_1 \leftrightarrow f_1) \wedge (f_1 \leftrightarrow g_1) \wedge (e_2 \leftrightarrow f_2) \wedge (f_2 \leftrightarrow g_2))$.
- *Translate the formula into an equisatisfiable CNF formula.*
  This can be done using the CNF conversion algorithm we described earlier.
- *If the formula is satisfiable, the satisfying assignment gives the desired state.*
  An actual coloring can be constructed by looking at the values of each variable given by the satisfying assignment.
- *If the formula is not satisfiable, the desired state does not exist.*
  If the formula can be shown to be unsatisfiable, this is essentially a proof that there is no coloring.

What if the number of colors is *3*? **Answer:** $n = 16$

These and similar questions are studied in *Ramsey theory*.