Digital Logic Design: a rigorous approach ©

Chapter 9: Representation of Boolean Functions by Formulas

Guy Even   Moti Medina

School of Electrical Engineering Tel-Aviv Univ.

April 10, 2012

Book Homepage:
`http://www.eng.tau.ac.il/~guy/Even-Medina`

Sum of Products

The first normal form we consider is called disjunctive normal form (DNF) or sum of products (SOP).

literals and products

We recall the definition of a literal.

Definition

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

Recall that the AND connective is associative. Thus we may apply it to multiple arguments without writing parenthesis. To simplify notation, we use the $\cdot$ notation for the AND connective so that

$$X \cdot Y \cdot Z$$

simply means (X AND Y AND Z). We often refer to such an AND as a product.

Definition

A formula that is the AND of literals is called a product term.

Appearance of variables in products

- a variable X appears in a product term p if either X or $\bar{X}$ is an argument of the AND in p.
- a variable might appear more than once in a term.
  For example, X appears three times in the product term $(X \cdot Y \cdot \bar{X} \cdot X)$.
- Recall that:
  1. $X \cdot \bar{X}$ is a contradiction
  2. $X \cdot X$ is logically equivalent to X
  3. $\bar{X} \cdot \bar{X}$ is logically equivalent to $\bar{X}$.

simple products

Definition

A product term p is simple if every variable appears at most once in p.

simple: $X_1 \cdot X_2 \cdot \bar{X}_3$
not simple: $X_1 \cdot X_2 \cdot \bar{X}_1$

Claim

Every product is a contradiction or logically equivalent to a simple product.

Example - 1

The following formulas are product terms.

1. $p_1 = X \cdot Y$,
2. $p_2 = \bar{A}$ AND B AND C,
3. $p_3 = L$,
4. $p_4 = G \wedge (\neg H) \wedge G$.

The variables A, B and C appear in p_2. The product term in p_4 is not simple, since the the variable G appears twice. On the other hand, the product term in p_1 is simple, since both X and Y appear once.

Notation.

- With each product term p, we associate the set of variables that appear in p.
- The set of variables that appear in p is denoted by $\mathit{vars}(p)$.
- Let $\mathit{vars}^+(p)$ denote the set of variables that appear in p that appear without negation.
- Let $\mathit{vars}^-(p)$ denote the set of variables that appear in p that with negation.
- Let $\mathit{literals}(p)$ denote the set of literals that appear in p.

Example

Let $p = X_1 \cdot \bar{X}_2 \cdot X_3$, then $\mathit{vars}(p) = \{X_1, X_2, X_3\}$, $\mathit{vars}^+(p) = \{X_1, X_3\}$ and $\mathit{vars}^-(p) = \{X_2\}$, and $\mathit{literals}(p) = \{X_1, \bar{X}_2, X_3\}$.

Minterms

Definition

A simple product term p is a minterm with respect to a set U of variables if $vars(p) = U$.

A minterm is a simple product term, and therefore, every variable in U appears exactly once in p.

lemma

A minterm p attains the truth value 1 for exactly one truth assignment.

sum-of-products (SOP)

Recall that the OR connective is also associative. We use the $+$ to denote the OR connective. The OR of multiple arguments is written as a "sum". For example,

$$X + Y + Z$$

simply means (X OR Y OR Z).
We often refer to such an OR as a sum. Substitution allows us to replace each occurrence of a variable by a product. This leads us to the terminology sum-of-products.

Definition

A Boolean formula is called a sum-of-products (SOP) if it is a constant or an OR of product terms.

Examples

Each of the following formulas is a sum-of-products.

1. $\varphi_1 = X \cdot Y + X \cdot Y$,
2. $\varphi_2 = (\bar{A} \text{ AND } B \text{ AND } C) \text{ OR } (A \text{ AND } \bar{B} \text{ AND } C) \text{ OR } \bar{D}$,
3. $\varphi_3 = L$.

Each of the following formulas is not a sum-of-products.

1. $(X + Y) \cdot Z$,
2. $(A \text{ OR } B) \text{ AND } (C \text{ OR } D)$.

SOP representation

Definition

For a $v \in \{0,1\}^n$, define the minterm p_v to be $p_v \triangleq (\ell_1^v \cdot \ell_2^v \cdots \ell_n^v)$, where:

$$\ell_i^v \triangleq \begin{cases} X_i & \text{if } v_i = 1 \\ \bar{X}_i & \text{if } v_i = 0. \end{cases}$$

Definition

Let $f^{-1}(1)$ denote the set

$$f^{-1}(1) \triangleq \{v \in \{0,1\}^n \mid f(v) = 1\}.$$

.

SOP representation - cont

Definition

The set of minterms of f is defined by

$$M(f) \triangleq \{p_v \mid v \in f^{-1}(1)\}.$$

Theorem

Every Boolean function $f : \{0,1\}^n \rightarrow \{0,1\}$ that is not a constant zero is expressed by the sum of the minterms in $M(f)$.

A “bad” example

Consider the constant Boolean function $f : \{0,1\}^n \rightarrow \{0,1\}$ that is defined by $f(v) = 1$, for every v.
The sum-of-minterms that represents f is the sum of all the possible minterms over n variables. This sum contains 2^n minterms.
On the other hand, f can be represented by the constant 1.
The question of finding the shortest sum-of-products that represents a given Boolean formula is discussed in more detail later in this chapter.

Product of Sums

The second normal form we consider is called conjunctive normal form (CNF) or product of sums (POS) .

sum terms

Definition

A formula that is the OR of literals is called a sum term.

- a variable X appears in a sum term p if X or $\bar{X}$ is one of the arguments of the OR in p.
- A sum term is simple if every variable appears at most once in it.
- $\mathit{vars}(p) =$ the set of variables that appear in p.

The notation $\mathit{vars}^{+}(p)$ and $\mathit{vars}^{-}(p)$ is used as well.

examples

The following formulas are sum terms.

1. $$p_1 = X + Y,$$
2. $$p_2 = \bar{A} \text{ OR } B \text{ OR } C,$$
3. $$p_3 = L,$$
4. $$p_4 = G \vee (\neg H) \vee G.$$

The variables A, B and C appear in p_2. The sum term in p_4 is not simple, since the the variable G appears twice. On the other hand, the sum term in p_1 is simple, since both X and Y appear once.

maxterms

Definition

A simple sum term p is a maxterm with respect to a set U of variables if $vars(p) = U$.

As in the case of a minterm, each variable appears at most once in a maxterm since it is a simple sum term.

relation to De Morgan duality

Recall that $DM(\varphi)$ is the De Morgan dual of the formula φ.

observation

(1) If p is a minterm, then the formula $DM(p)$ is a maxterm.
(2) If p is a maxterm, then the formula $DM(p)$ is a minterm.

POS representation

Definition

A Boolean formula is called a product-of-sums if it is a constant or an AND of sum terms.

Observation

(1) If p is a sum-of-products, then the formula $DM(p)$ is a product-of-sums.
(2) If p is a product-of-sums, then the formula $DM(p)$ is a sum-of-products.

POS representation

Lemma

A maxterm p attains the truth value 0 *for exactly one truth assignment.*

Theorem

Every Boolean function $f : \{0,1\}^n \rightarrow \{0,1\}$ *that is not a constant one can be represented by a product of maxterms.*

Representation by polynomials

Definition

The Galois Field $GF(2)$ is defined as follows.

1. Elements: the elements of $GF(2)$ are $\{0, 1\}$. The zero is called the additive unity and one is called the multiplicative unity.
2. Operations:
    1. addition which is simply the XOR function, and
    2. multiplication which is simply the AND function.

In the context of $GF(2)$ we denote multiplication by $\cdot$ and addition by $\oplus$.

$GF(2)$ properties

We are used to infinite fields like the rationals (or reals) with regular addition and multiplication. In these fields, $1 + 1 \neq 0$. However, in $GF(2)$, $1 \oplus 1 = 0$.

Observation

$X \oplus X = 0$, *for every* $X \in \{0, 1\}$.

A minus sign in a field means the additive inverse.

Definition

The element $-X$ stands for the element Y such that $X \oplus Y = 0$.

$GF(2)$ properties - more

Observation

In $GF(2)$, the additive inverse of X is X itself, namely $-X = X$, for every $X \in \{0,1\}$.

Thus, we need not write minus signs, and adding an X is equivalent to subtracting an X.

The distributive law holds in $GF(2)$, namely:

Observation

$(X \oplus Y) \cdot Z = (X \cdot Z) \oplus (Y \cdot Z)$, *for every* $X, Y, Z \in \{0,1\}$.

$GF(2)$ properties - even more

Let X^k denote the product

$$X^k \triangleq \overbrace{X \cdot \cdots \cdot X}^{k \text{ times}}.$$

We define $X^0 = 1$, for every $X \in \{0,1\}$. The following observation proves that multiplication is idempotent.

Observation

$X^k = X$, for every $k \in \mathbb{N}^+$ and $X \in \{0,1\}$.

A field is a field

The structure of a field allows us to solve systems of equations. In fact, Gauss elimination works over any field. The definition of a vector space over $GF(2)$ is just like the definition of vector spaces over the reals. Definitions such as linear dependence, dimension of vector spaces, and even determinants apply also to vector spaces over $GF(2)$.

Examples

- $$X_1 \oplus X_2 = 0 \quad \Leftrightarrow \quad X_1 = X_2.$$
- We show how to solve a simple systems of equalities over $GF(2)$ using Gauss elimination. Consider the following system of equations

$$\begin{array}{ccccccc} X_1 & \oplus & X_2 & \oplus & X_3 & = 0\,, \\ X_1 & & & \oplus & X_3 & = 0\,, \\ & & X_2 & \oplus & X_3 & = 1\,. \end{array}$$

Polynomials over $GF(2)$

Definition

A monomial in $GF(2)$ over the variables in the set U is a finite product of the elements in $U \cup \{0, 1\}$.

Observation

Every monomial p in $GF(2)$ over the variables in U equals a constant or a product of variables in p.

- By commutativity: $X_1 \cdot X_2 \cdot X_3 \cdot X_1 = X_1^2 \cdot X_2 \cdot X_3$.
- Positive exponents can be reduced to one. For example, $X_1^2 \cdot X_2 \cdot X_3$ equals $X_1 \cdot X_2 \cdot X_3$.
- Constants can be eliminated. $X \cdot 0 = 0$, $X \cdot 1 = X$.

Polynomials

Definition

A polynomial in $GF(2)$ over the variables in the set U is a finite sum of monomials.

Example: $X_1 \cdot X_2 \oplus X_1 \cdot X_3 \oplus X_2 \cdot X_3$.

We denote the set of all polynomials in $GF(2)$ over the variables in U by $GF(2)[U]$. Just as multivariate polynomials over the reals can be added and multiplied, so can polynomials in $GF(2)[U]$.

representation by polynomials in $GF(2)[U]$

Every polynomial $p \in GF(2)[U]$ is a Boolean function $f_p : \{0,1\}^{|U|} \to \{0,1\}$. The converse is also true.

Theorem

Every Boolean function $f : \{0,1\}^n \to \{0,1\}$ can be represented by a polynomial in $GF(2)[U]$, where $U = \{X_1, \ldots, X_n\}$.

proof outline

- easy: f is constant.
- $f^{-1}(1) \triangleq \{v \in \{0,1\}^n \mid f(v) = 1\}$.
- For each $v \in f^{-1}(1)$, we define the product p_v. The polynomial $p \in GF(2)[U]$ is defined as follows.

$$p \triangleq \bigoplus_{v \in f^{-1}(1)} p_v.$$

Side effect

Corollary

The set of connectives $\{\text{XOR}, \text{AND}\}$ *is complete.*

Satisfiability

The problem of satisfiability of Boolean formulas is defined as follows.

Input: A Boolean formula φ.

Output: The output should equal "yes" if φ is satisfiable. If φ is not satisfiable, then the output should equal "no".

Note that the problem of satisfiability is quite different if the input is a truth table of a Boolean function. In this case, we simply need to check if there is an entry in which the function attains the value 1.

Relation to P vs. NP

The main open problem in Computer Science since 1971 is whether $P = NP$. We will not define the classes P and NP, but we will phrase an equivalent question in this section.
Consider a Boolean formula φ. Given a truth assignment τ, it is easy to check if $\hat{\tau}(\varphi) = 1$. We showed how this can be done in Algorithm EVAL. In fact, the running time of the EVAL algorithm is linear in the length of φ.
On the other hand, can we find a satisfying truth assignment by ourselves (rather than check if τ is a satisfying assignment)? Clearly, we could try all possible truth assignments. However, if n variables appear in φ, then the number of truth assignments is 2^n.

Satisfiability and P vs. NP

We are ready to formulate a question that is equivalent to the question $P = NP$.

Satisfiability in polynomial time

Does there exist a constant $c > 0$ and an algorithm Alg such that:

1. Given a Boolean formula φ, algorithm Alg decides correctly whether φ is satisfiable.
2. The running time of Alg is $O(|\varphi|^c)$, where $|\varphi|$ denotes the length of φ.

This seemingly simple question turns out to be a very deep problem about what can be easily computed versus what can be easily proved. It is related to the question whether there is a real gap between checking that a proof is correct and finding a proof.

Minimization Heuristics

We consider the following minimization problem.

Input: A truth table of a Boolean function $f : \{0,1\}^n \rightarrow \{0,1\}$.

Output: An SOP Boolean formula ψ such that the Boolean function B_ψ defined by ψ satisfies: $f = B_\psi$.

Goal: Find a shortest SOP ψ such that $B_\psi = f$.

Don't expect an efficient algorithm!

Example

X	Y	Z	$f(X,Y,Z)$
0	0	0	1
1	0	0	0
0	1	0	1
1	1	0	0
0	0	1	0
1	0	1	1
0	1	1	1
1	1	1	1

$$\varphi_f = \bar{X}\cdot\bar{Y}\cdot\bar{Z} + \bar{X}\cdot Y\cdot\bar{Z} + X\cdot\bar{Y}\cdot Z + \bar{X}\cdot Y\cdot Z + X\cdot Y\cdot Z\,.$$

Example - cont

$$\varphi_f = \bar{X} \cdot \bar{Y} \cdot \bar{Z} + \bar{X} \cdot Y \cdot \bar{Z} + X \cdot \bar{Y} \cdot Z + \bar{X} \cdot Y \cdot Z + X \cdot Y \cdot Z\,.$$

There are two shortest SOP formulas φ_1, φ_2 that are logically equivalent to φ_f, as follows.

$$\begin{aligned}\varphi_1 &= \bar{X} \cdot \bar{Z} + X \cdot Z + \bar{X} \cdot Y\,,\\ \varphi_2 &= \bar{X} \cdot \bar{Z} + X \cdot Z + Y \cdot Z\,.\end{aligned}$$

Assumptions

- f denotes a Boolean function
- φ_f is a Boolean formula that represents f (i.e., φ_f is the sum of the minterms of f).
- the Boolean function f is not a constant function. Therefore, φ_f is satisfiable and not a tautology.

Implicants

Definition

A satisfiable product term p is an implicant of f if $(p \rightarrow \varphi_f)$ is a tautology.

We denote the set of implicants of f by $\mathcal{I}(f)$. Note that an implicant must be satisfiable, and hence an implicant cannot contain both a variable and its negation as literals.

Claim

Every minterm $p_v \in M(f)$ is a implicant of f, hence $M(f) \subseteq \mathcal{I}(f)$.

Claim

The sum (OR) of the implicants of f is logically equivalent to φ_f.

Implicants - properties

The following claim shows that $\mathcal{I}(f)$ is closed under "subsets" in the sense that removing part of the literals from an implicant keeps it an implicant.

Claim

Let $p \in \mathcal{I}(f)$. If q is a satisfiable product and $\mathit{literals}(p) \subseteq \mathit{literals}(q)$, then $q \in \mathcal{I}(f)$.

Claim

For every two satisfiable products p, q, the following holds:

$$(p \rightarrow q) \text{ is a tautology } \Leftrightarrow (\mathit{literals}(q) \subseteq \mathit{literals}(p)).$$

Prime Implicants

A prime implicant is an implicant that is minimal with respect to containment.

Definition

An implicant $p \in \mathcal{I}(f)$ is a prime implicant of f if the following holds:

$$\forall q \in \mathcal{I}(f) : \mathit{literals}(q) \subseteq \mathit{literals}(p) \Rightarrow (\mathit{literals}(q) = \mathit{literals}(p)).$$

We denote the set of prime implicants of f by $\mathcal{I}'(f)$.

immediate predecessor

Definition

Let $p, q \in \mathcal{I}(f)$. We say that p is an immediate predecessor of q if:

- $\mathit{literals}(q) \subseteq \mathit{literals}(p)$, and
- $\mathit{literals}(p) \setminus \mathit{literals}(q)$ contains a single literal.

The Implicants' Graph

One usually defines a partial order over the implicants of a Boolean function by containment of the set of literals. We represent this partial order by a directed graph which we call the implicants' graph.

Definition

The implicants' graph $G_f = (V, E)$ of a Boolean function f is a directed graph defined as follows.

1. $V \triangleq \mathcal{I}(f)$.
2. $E \triangleq \{(p, q) \in V \times V \mid p \text{ is an immediate predecessor of } q\}$.

Claim

The implicants' graph is a acyclic.

Prime implicants are sinks in the implicants' graph

Definition

The implicants' graph $G_f = (V, E)$ of a Boolean function f is a directed graph defined as follows.

1. $V \triangleq \mathcal{I}(f)$.
2. $E \triangleq \{(p, q) \in V \times V \mid \ p \text{ is an immediate predecessor of } q\}$.

Lemma

An implicant $p \in \mathcal{I}(f)$ is a prime implicant iff it is a sink in G_f.

What about implicants that are not prime?

Claim

If $p \in \mathcal{I}(f) \setminus \mathcal{I}'(f)$, then the following two statements hold:

- *There exists an implicant $q \in \mathcal{I}(f)$ such that p is an immediate predecessor of q.*
- *There exists a prime implicant $q \in \mathcal{I}'(f)$ such that $\mathit{literals}(q) \subset \mathit{literals}(p)$.*

Proof: p is not a sink so consider a maximal path from p.

Essential Prime Implicants - preliminaries

We now define a covering relation between minterms and prime implicants. Recall that $M(f)$ denotes the set of minterms of f, and $\mathcal{I}'(f)$ denotes the set of prime implicants of f.

Definition

The covering relation $C_f \subseteq M(f) \times \mathcal{I}'(f)$ is the set

$$C_f \triangleq \{(r,p) \in M(f) \times \mathcal{I}'(f) \mid r \rightarrow p \text{ is a tautology}\}.$$

Observation

Let $(r,p) \in M(f) \times \mathcal{I}'(f)$. Then, $(r,p) \in C_f$ iff there exists a path from r to p in the implicants' graph G_f.

essential prime implicants

$$C_f \triangleq \{(r,p) \in M(f) \times \mathcal{I}'(f) \mid r \rightarrow p \text{ is a tautology}\}.$$

Definition

A prime implicant $p \in \mathcal{I}'(f)$ is an essential prime implicant if there exists minterm r such that p is the only prime implicant that covers r.

We denote the set of essential prime implicants of φ by $\mathcal{I}^e(f)$.

Observation

A prime implicant $p \in \mathcal{I}'(f)$ is an essential prime implicant iff there exists a minterm r such that every path in G_f from r to a prime implicant ends in p.

Essential prime implicants - property

Definition

A prime implicant $p \in \mathcal{I}'(f)$ is an essential prime implicant if there exists minterm r such that p is the only prime implicant that covers r.

Claim

A prime implicant $p \in \mathcal{I}'(f)$ is an essential prime implicant iff there exists a truth assignment τ such that

- $\hat{\tau}(p) = 1$, *and*
- $\hat{\tau}(q) = 0$, *for every* $q \in \mathcal{I}'(f) \setminus \{p\}$.

Proof: correspondence between τ and the minterm r that is only satisfied by τ.

Prime implicants suffice

Claim

The sum (i.e., OR) of the prime implicants of f is logically equivalent to φ_f.

Replace non-prime implicants by prime implicants

Suppose that f is represented by an SOP that contains an implicant that is not prime. Can this SOP be shortened? The following claim shows that we can substituting a non-prime implicant by a prime implicant (that covers the non-prime implicant) to make the SOP shorter.

Claim

Let $p \in \mathcal{I}(f) \setminus \mathcal{I}'(f)$. Let $\varphi \in \mathcal{BF}$, such that $(\varphi \vee p)$ is equivalent to φ_f. Then, there exists $q \in \mathcal{I}'(f)$ such that:

- *$literals(q) \subsetneq literals(p)$, and*
- *$(\varphi \vee q)$ is equivalent to φ_f.*

Corollary

If ψ is a shortest SOP formula that is logically equivalent to φ_f, then every product term in ψ is a prime implicant of f.

Essential prime implicants are essential

Claim

Suppose that

- ψ *is the sum of a subset of the prime implicants of* f*, and*
- ψ *is logically equivalent to* φ_f*.*

Then, every essential prime implicant $p \in \mathcal{I}'(f)$ *appears as a product term in* ψ*.*

We remark that there exist Boolean functions f such that f is not logically equivalent to the sum of the essential prime implicants of f.

For example, consider the function f represented by the Boolean formula $\varphi_f(X, Y, Z) = \bar{X} \cdot Z + Y \cdot Z + X \cdot Y + X \cdot \bar{Z} + \bar{Y} \cdot \bar{Z}$.
Only $\bar{X} \cdot Z$ and $\bar{Y} \cdot \bar{Z}$ are essential.

Heuristic for Min SOP

Heuristic for finding a shortest SOP ψ that represents f.

1. Compute $\mathcal{I}'(f)$ and $\mathcal{I}^e(f)$.
2. Add every product in $\mathcal{I}^e(f)$ to ψ.
3. Find a shortest subset $A \subseteq \mathcal{I}'(f) \setminus \mathcal{I}^e(f)$ such that adding the products in A to ψ makes ψ logically equivalent to φ_f.

Last step uses exhaustive search.

The Quine-McCluskey Heuristic

Algorithm for computing the prime implicants and the essential prime implicants of formula φ. The algorithm simply constructs the implicants' graph of f. The specification of the algorithm is as follows.

Input: A truth table T_f of a nonconstant Boolean function $f : \{0,1\}^n \rightarrow \{0,1\}$.

Output: The sets $\mathcal{I}'(f)$ and $\mathcal{I}^e(f)$ where $\mathcal{I}'(f)$ and $\mathcal{I}^e(f)$ are the sets of prime implicants and essential prime implicants of f, respectively.

Terminology

Definition

The symmetric difference of two sets A, B is the set $(A \setminus B) \cup (B \setminus A)$.

We denote the symmetric difference by $A \triangle B$.

Definition

Let p and q denote two satisfiable product terms.

1. The product term $p \cap q$ is the product of the literals in $\mathit{literals}(p) \cap \mathit{literals}(q)$.
2. If $\mathit{vars}(p) = \mathit{vars}(q)$, then the distance between p and q is defined by

$$\mathit{dist}(p,q) \triangleq \left|\{i : \{X_i, \bar{X}_i\} \subseteq \mathit{literals}(p) \triangle \mathit{literals}(q)\}\right| .$$

If $\mathit{vars}(p) \neq \mathit{vars}(q)$, then define $\mathit{dist}(p,q) \triangleq \infty$.

Quine-McCluskey Heuristic

Algorithm 1 QM(T_f) - An algorithm for computing the prime implicants of $f : \{0,1\}^n \rightarrow \{0,1\}$ given its truth table T_f.

```
1  Construct the implicants' graph $G_f$ over implicants of $f$
    1  $I_n \leftarrow \{p \mid p \text{ is a minterm of } f\}$.
    2  For $k = n$ downto 2 do:
        1  $I_{k-1} \leftarrow \emptyset$.
        2  For each pair of implicants $p, q \in I_k$ such that $dist(p,q) = 1$
           do
                $I_{k-1} \leftarrow I_{k-1} \cup \{p \cap q\}$
                add $p \longrightarrow (p \cap q)$ and $q \longrightarrow (p \cap q)$ to $G$.
2  Return $\{p \mid p \text{ is a sink in } G\}$.
```

Quine-McCluskey Heuristic - correctness

Claim

If $p, q \in \mathcal{I}(f)$ *and* $\text{dist}(p, q) = 1$*, then* $p \cap q \in \mathcal{I}(f)$*.*

Theorem

Each set I_k *constructed by algorithm* $QM(T_f)$ *equals the set of implicants of* f *that contain* k *literals.*

Claim

Algorithm $QM(T_f)$ *constructs the implicants' graph* G_f*.*

Finding the essential prime implicants

Algorithm $QM(T_f)$ computes the implicants' graph. The essential prime implicants can be computed as follows.

1. For each minterm r, compute the set of sinks in G_f that are reachable from r.
2. If this set contains a single sink p, then add p to $\mathcal{I}^e(f)$.
3. After all minterms have been scanned, return $\mathcal{I}^e(f)$.

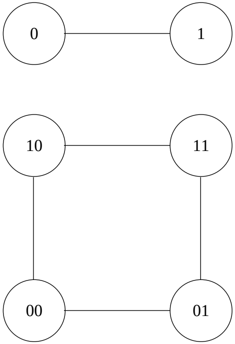
The hypercube

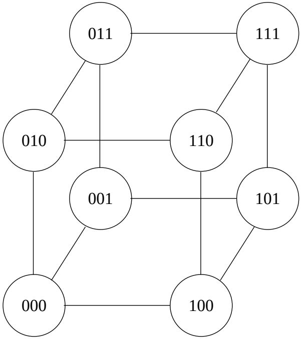
The Hamming distance between $u, v \in \{0,1\}^n$ is defined by

$$dist(u,v) = |\{i \mid u_i \neq v_i\}|.$$

The hypercube is the graph $H_n = (\{0,1\}^n, E_n)$, where

$$E_n \triangleq \{(u,v) \mid dist(u,v) = 1\}.$$

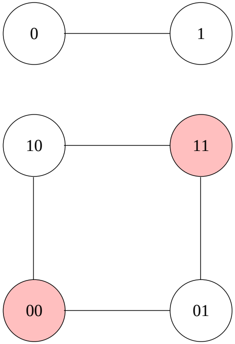


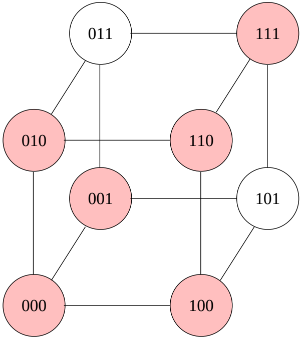


Minterms and implicants in the hypercube

Mark the vertices v for which $f(v) = 1$.

- marked vertices = minterms
- edge with both vertices marked = implicant with 2 literals
- face with all vertices marked = implicant with 1 literal





Karnaugh Maps

A tabular method to obtain the prime implicants and the essential prime implicants is called Karnaugh Maps. This method works reasonably well for Boolean functions $f : \{0,1\}^n \rightarrow \{0,1\}$ where $n \leq 4$. The idea is as follows:

1. Write the multiplication table of f. It useful to order the columns and rows in a Gray code order.
2. Identify $a \times b$ "generalized" maximal rectangles of all-ones in the table where both a and b are powers of 2.
3. Each such maximal rectangle corresponds to a prime implicant.
4. If a "1" is covered only by one such rectangle, then this rectangle corresponds to an essential prime implicant.

Karnaugh Maps - example

$\frac{YZ}{X}$	00	01	11	10
0	1	0	1	1
1	1	1	0	1

$\frac{YZ}{X}$	00	01	11	10
0	1	0	1	1
1	1	1	0	1

$\frac{YZ}{X}$	00	01	11	10
0	1	0	1	1
1	1	1	0	1

$\frac{YZ}{X}$	00	01	11	10
0	1	0	1	1
1	1	1	0	1