Normal or Canonical forms for a Boolean Expression

R We have already studied the algebra of sets (denoted by $Sets$) and the algebra of propositions or propositional calculus (denoted by PC). Each of these is an example of a more abstract object called a Boolean algebra which we shall define using axioms. But before we do so, we list the relevant features in a systematic manner to make the analogies explicit. The three sets of rules, arranged side by side in columns, in the notes on Logic, exhibit the analogies explicitly. We use certain extra operations in each setup but these can be defined for each system. The systems have certain nullary operations (also known as distinguished elements) for example: $\{\ \}, \mathcal{U}$ in $Sets$ and 0,1 in PC, certain unary operations, for example: $(\ )^c$ a in $Sets$ and $\neg$ in PC and certain binary operations, for example: $\cup,\ \cap$ in $Sets$ and $\vee,\ \wedge$ in PC. Some binary operations, for example: $\setminus$ in $Sets$ and $\longrightarrow, \longleftrightarrow$ in PC are not regarded in this context as primary operations and these are not needed in the axiomatic treatment of Boolean Algebras. These can, however, be defined in Boolean Algebras. The three sets of rules have these definitions and the rules are stated in each of the three languages.

R In $Sets$ we choose $\left(\{\},\ \mathcal{U},\ (\ )^c,\ \cup,\ \cap\right)$ as the primary connectives

R In PC we choose $\left(\ \bot,\ \top, \neg,\ \vee\ ,\ \wedge\ \right)$ as the primary connectives

R In BA we choose $\left(\ 0, 1, (\ )'\ , +\ ,\ *\ \right)$ as the primary connectives

D A Boolean algebra is specified by a set B carrying two nullary operations or distinguished elements: 0 and 1, one unary operation: $(\ )'$ called complementation, and two binary operations addition $+$ and multiplication $*$, denoted as in the list in the and satisfying the following axioms:

$\forall x, y, z \in B$

$x + y = y + x$ $\qquad$ $x * y = y * x$

$x + (y * z) = (x + y) * (x + z)$ $\qquad$ $x * (y + z) = (x * y) + (x * z)$

$x + 0 = x$ $\qquad$ $x * 1 = x$

$x + x' = 1$ $\qquad$ $x * x' = 0$

N $\quad \forall x, y \in B \qquad xy \coloneqq x * y$

R We use $*$ when there is danger of confusion.

R The axioms allow us to prove the following theorems.

$\forall x, y, z \in B$

$x + x = x$ $\qquad$ $x * x = x$

$x + 1 = 1$ $\qquad$ $x * 0 = 0$

$x + (x * y) = x$ $\qquad$ $x * (x + y) = x$

$(x + y) + z = x + (y + z)$ $\qquad$ $(x * y) * z = x * (y * z)$

$0' = 1$ $\qquad$ $1' = 0$

$(x + y)' = x' * y'$ $\qquad$ $(x * y)' = x' + y'$

$(x')' = x$

$x + y = 1 \quad x * y = 0$

$\dfrac{}{y = x'}$

R An expression from any one of the systems: $Sets, PC$ or BAcan be translated into a corresponding expression into any of the other systems. We shall informally adopt a correspondence of unknowns as shown in the list of rules in three columns.

A 0 Given an expression in $Sets, PC$ or $BA,$ rewrite the expression making all connectives explicit.

00 In $PC,$

use $ab = a * b$ to replace ab

1 Rewrite every expression using only the allowed connectives from the corresponding system

In $Sets,$ use only: $\{\ \}, \mathcal{U}, (\ )^c, \cup, \cap$

In PC , use only: $\bot, \top, \neg, \vee, \wedge$

In BA, use only: $0, 1, (\quad)', +, *$

10 In $Sets$,

use $A \backslash B = A \cap (B^c)$ to replace $A \backslash B$

11 In PC,

use $p \longrightarrow q = \big((\neg p) \vee q\big)$ to replace $p \longrightarrow q$

use $p \longleftrightarrow q = \big((\neg p) \vee q\big) \wedge \big(p \vee (\neg q)\big)$ to replace $p \longleftrightarrow q$

2 Insert parentheses appropriately to make order of operations explicit.

R We use the above algorithm in a few examples below.

E Find expressions corresponding to $ab' + c$ in PC and $Sets$

$$ab' + c = \big(a * (b')\big) + c$$

Therefore, using correspondences in our three-column table we immediately obtain the corresponding expressions in PC and $Sets$.

$\big(p \wedge (\neg q)\big) \vee r$ in PC

$\big(A \cap (B^c)\big) \cup C$ in $Sets$

E Find expressions corresponding to $p \longrightarrow q$ in BA and $Sets$

$$p \longrightarrow q = \big((\neg p) \vee q\big)$$

Therefore, using correspondences in our three-column table we immediately obtain the corresponding expressions in BA and $Sets$.

$x + (y')$ in PC

$A \cup (B^c)$ in $Sets$

E Find expressions corresponding to $A \backslash B$ in BA and PC

$$A \backslash B = A \cap (B^c)$$

Therefore, using correspondences in our three-column table we immediately obtain the corresponding expressions in BA and PC.

$p \wedge (\neg q)$ in PC

$$x * (y') = x(y') = xy' \text{ in } Sets$$

R Although all intermediate steps must be shown, the final result may be in abbreviated form.

Q Translate twenty expressions and twenty equalities from the notes on Propositional Calculus and the notes on Sets into corresponding expressions and equalities in Boolean Algebras.

R For any identity in any system, we may employ any of the other systems as well, after appropriate translation. BA is set up to resemble ordinary algebra as closely as possible and may appear easier than the other two systems. But, once one has obtained a proof, one may readily translate back into the other systems line by line. One should, however strive to become proficient in each of the three systems.

E Prove that $(p \longrightarrow q) = \Big(\big(p \wedge (\neg q)\big) \longrightarrow \bot\Big)$ by translating each side into Boolean Algebras and then establishing equivalence/equality in Boolean Algebras.

Note that we established this equality in the notes on propositional calculus using truth tables.

We use the algorithm describes earlier.

Making all connectives explicit and using only allowed connectives, we have the following equivalence/equality.

$$\big((\neg p) \vee q\big) = \Big(\big(\neg\big(p \wedge (\neg q)\big)\big) \vee \bot\Big)$$

Please check that this is the case.

Upon being translated into BA the above reads:

$$\big((x') + y\big) = \Big(\big((x * (y'))'\big) + \ 0\Big)$$

We always start from the side that appears more complicated to us although one may start from either side. In this instance we start from the right side as it seems more intricate.

$$RS$$

$$
\begin{aligned}
&= && \Big(\big(x * (y')\big)'\Big) + 0 \\
&= && \big(x * (y')\big)' \\
&= && \big(x(y')\big)' && ((ab)' = a' + b') \\
&= && (ab)'\left\langle \begin{matrix} a \leftarrow x \\ b \leftarrow y' \end{matrix} \right\rangle && \left((ab)' = a' + b'\right) \\
&= && (a' + b')\left\langle \begin{matrix} a \leftarrow x \\ b \leftarrow y' \end{matrix} \right\rangle \\
&= && x' + (y')' \\
&= && x' + \left(((a')')\left\langle a \leftarrow y \right\rangle\right) && \left((a')' = a\right) \\
&= && x' + \left(a\left\langle a \leftarrow y \right\rangle\right) \\
&= && x' + y \\
&= && LS
\end{aligned}
$$

R We have written the rules on the side and shown the substitutions explicitly. You must also do the same on all work that you do including exams.

D Given a boolean expression E, we define:

$Ukn(E) :=$ The set of boolean unknowns that occur in E either in complemented or uncomplemented form, and

$SLtr(E) :=$ The set of boolean literals that occur in E

$MSLtr(E) :=$ The multiset of boolean literals that occur in E

D A literal is defined to be a complemented or an uncomplemented boolean unknown (variable) such as x or x'.

D We order the literals alphabetically by extending the alphanumerical order on unknowns and decreeing that for the same unknown, the uncomplemented

literal precedes the complemented literal. We do not take the time to make the above any more precise at this point but exhibit the following sequence to indicate what is intended.

$$a, a', b, b', c, c', \cdots, a_1, a_1', b_1, b_1', \cdots, a_2, a_2', b_2, b_2', \cdots$$

D A fundamental product (FP) is defined to be a product of literals in distinct unknowns.

E $$E \coloneqq xy + x'z + yz$$

$$Ukn(E) = \{x, y, z\}$$

$$SLtr(E) = \{x, x', y, z\}$$

$$SLtr(E) = \{x\colon 1, x'\colon 1, y\colon 2, z\colon 2\}$$

D Given two fundamental products P_1 and P_2, P_1 is said to be included in P_2, and written:

$$P_1 \subseteq P_2 :\Leftrightarrow Ltr(P_1) \subseteq Ltr(P_2)$$

T $$\frac{P_1 \subseteq P_2}{P_1 + P_2 = P_1}$$

R This follows very easily from the absorption and commutativity and we do not record it here. The reader is invited to write it up.

E $$xz' + xyz' = xz' \qquad (xz' \subseteq xyz')$$

D A boolean expression E is said to be in sum-of-products (SOP) form, and written

$$SOPFrm(E) :\Leftrightarrow \left(E = \sum_{k=1}^{n} P_k\right) \quad \text{where}$$

$$(1)\ \forall k \epsilon 1..n, FP(\ P_k\ ) \qquad \text{and} \qquad (2)\ \ \frac{k \neq l}{P_k \nsubseteq P_l}$$

R The SOP form of a boolean expression E is not unique and any of the possible forms is denoted by $SOP(E)$.

D A Boolean expression E is said to be in complete-sum-of-products (SOP) form, and written

$$CSOPFrm(E) :\Leftrightarrow \Big(\big(SOPFrm(E)\big) \wedge\ (E = \textstyle\sum_{k=1}^{n} P_k)\Big) \qquad \text{and}$$

$$\forall k \epsilon 1..n, \quad Ukn(\ P_k\ ) = Ukn(\ E\ )$$

R The CSOP form of a boolean expression E is unique and is denoted by $CSOP(E)$ and since it is unique, it may be used as a canonical form to establish that two boolean expressions are equal.

T Every boolean expression E is equivalent to exactly one $CSOP(E)$.

R The following theorem is used to prove that two Boolean expressions are equivalent.

T $$\frac{CSOP(E_1) = CSOP(E_2)}{E_1 = E_2}$$

R The following algorithms will make repeated use of the following rules.

Identity	$x + 0 = x$	$x * 1 = x$
Complementation	$x + x' = 1$	$x * x' = 0$
De Morgan's	$(x + y)' = x' * y'$	$(x * y)' = x' + y'$
Involution	$(x')' = x$	

Algorithm for producing $SOP(E)$

The following algorithm takes a boolean expression E as input and returns some $SOP(E)$ as output.

A Input:

Boolean expression E

Output:

Sum of Products expression equivalent/equal to E.

(1) Use De Morgan's law and involution to move complementation into parentheses until complementation applies only to boolean unknowns and the expression is a product of factors, each of which is a sums of fundamental products).

(2) Use a tree to multiply out the factors and reduce the output at each branch of the tree using commutativity, idempotence, and complementation and order the literals alphabetically.

(3) Use absorption and identity to reduce the sum further.

(4) Within each term order the literals alphabetically.

(5) Within each summand order the literals alphabetically and order the summands lexicographically.

R Distinct summands may have distinct sets of unknowns.

Algorithm for producing $CSOP(E)$

The following algorithm takes a boolean expression E in sum-of-products form as input and returns some $CSOP(E)$ as output.

A Input:

Boolean expression E in sum-of-products form

Output:

Complete Sum of Products expression equivalent/equal to E where each summand of E contains exactly the same unknowns as in E.

(1) In each summand of $SOP(E)$ insert a 1 in the place of each missing unknown x_k.

(2) Replace each 1 with the corresponding $(x_k + x_k{}')$

(3) Use a tree to multiply out each summand.

(4) Retain only the first instance of each FP.

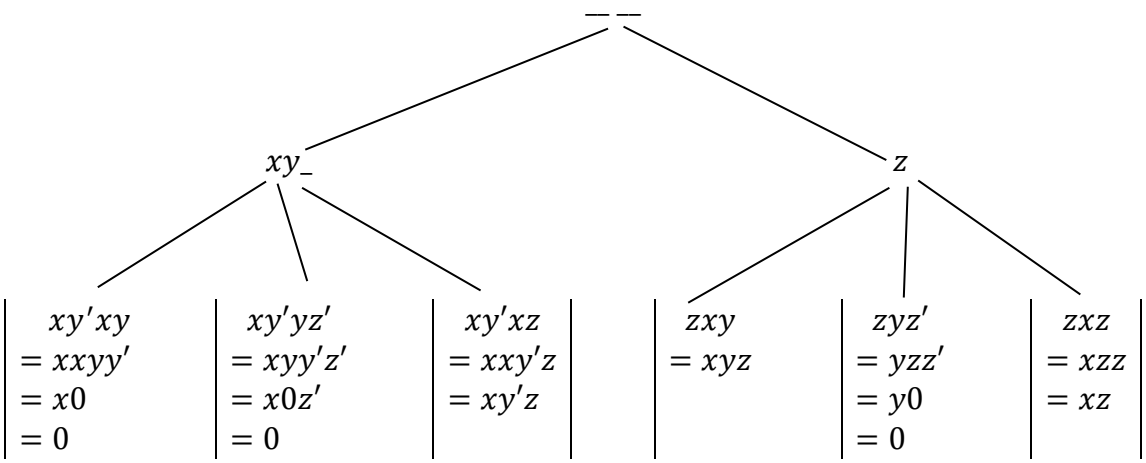
(5) Order the summands lexicographically.

R Every summand has exactly the same unknowns as $Ukn(E)$.

R For multiplying sums of products the work needs to be exhibited as follows:

E Multiply: $(xy' + z)(xy + yz' + xz)$

Tree



On the basis of the above, we get

$$(xy' + z)(xy + yz' + xz)$$

$$= \quad 0 + 0 + xy'z + xyz + 0 + xz$$

$$= \quad xy'z + xyz + xz$$

$$= \quad xyz + xy'z + xz$$

R Please note that in the above computation we have ordered the factors in each term using the extended alphabetical ordering that we discussed at in the preceding sections and we have ordered the summands lexicographically.

R All work on multiplication on all exams must be shown exactly in the above manner and the format must be exactly so. One will get no points otherwise. The number of blanks, the number of levels, and the number of branches will vary.

E Find $SOP(E)$ and $CSOP(E)$ where $E \coloneqq (xy' + z)(xy + yz' + xz)$

From the immediately preceding example, we note that:

$$\begin{aligned} & E \\ = \quad & (xy' + z)(xy + yz' + xz) \\ = \quad & xyz + xy'z + xz \qquad\qquad (xz \subseteq xyz,\ xz \subseteq xy'z) \\ = \quad & xz \end{aligned}$$

In the last step we have used absorption.

Therefore,

$$\begin{aligned} & E \\ = \quad & xz \\ = \quad & SOP(E) \end{aligned}$$

Note, however, that:

$Ukn(E) = \{x, y, z\}$ while

$Ukn\big(SOP(E)\big) = \{x, z\}$

Therefore, $SOP(E)$ does not contain unknown y.

Therefore $SOP(E)$ is not $CSOP(E)$ because $CSOP(E)$ must explicitly have the same unknowns as E, that is:

$$Ukn\big(CSOP(E)\big) = Ukn(E) = \{x, y, z\}$$

To find $CSOP(E)$ we use the corresponding algorithm and proceed as follows:

$$\begin{aligned} & E \\ = \quad & xz \\ = \quad & x(1)z \qquad (1 = y + y') \end{aligned}$$

$$= \quad x(y + y')z$$

$$= \quad xyz + xy'z \qquad (xyz \not\subseteq xy'z,\ xy'z \not\subseteq xyz\ )$$

$$= \quad CSOP(E)$$

Note that: $Ukn\big(CSOP(E)\big) = \{x, y, z\} = Ukn(E)$

R The above is a fairly simple situation. Our next example will be more intricate. The reader must verify all the steps thatType equation here. he or she is being asked to verify, without which it is not possible to understand.

E Find $SOP(E)$ and $CSOP(E)$ where

$$E \coloneqq \Big((xy)'z\Big)'\Big((x' + z)(y' + z')\Big)'$$

We repeatedly use the rules: $(ab)' = a' + b'$ $\quad (a + b)' = a'b'$

First, we note the following:

$$\Big((xy)'z\Big)'$$

$$= \quad (ab)\left\langle \begin{matrix} a \leftarrow (xy)' \\ b \leftarrow z \end{matrix} \right\rangle$$

$$= \quad (a' + b')\left\langle \begin{matrix} a \leftarrow (xy)' \\ b \leftarrow z \end{matrix} \right\rangle$$

$$= \quad \Big((xy)'\Big)' + z' \qquad ((a')' = a)$$

$$= \quad \Big(\big((a')'\big)\big\langle a \leftarrow xy\big\rangle\Big) + z'$$

$$= \quad \Big(\big(\ a\ \big)\big\langle a \leftarrow xy\big\rangle\Big) + z'$$

$$= \quad (xy) + z'$$

$$= \quad xy + z'$$

Secondly, we note the following:

$$\Big((x'+z)(y'+z')\Big)'$$

$$= \quad (ab)'\left\langle \begin{matrix} a \longleftarrow (x'+z) \\ b \longleftarrow (y'+z') \end{matrix} \right\rangle$$

$$= \quad (a'+b')\left\langle \begin{matrix} a \longleftarrow (x'+z) \\ b \longleftarrow (y'+z') \end{matrix} \right\rangle$$

$$= \quad ((x'+z)'+(y'+z')')$$

$$= \quad \Big(((x')')z'+\big((y')'\big)\big((z')'\big)\Big) \qquad \text{(Show explicit substitution)}$$

$$= \quad xz'+yz$$

Hence, using the previous two results, we have

$$E$$

$$= \quad \Big((xy)'z\Big)'\Big((x'+z)(y'+z')\Big)'$$

$$= \quad \Big(xy+z'\Big)\Big(xz'+yz\Big) \qquad \text{(Use a tree to multiply)}$$

$$= \quad xyz'+xyz+xz'$$

$$= \quad xyz+xyz'+xz' \qquad \text{(The terms are now ordered)}$$

R The reader must carry out the multiplication $\Big(xy+z'\Big)\Big(xz'+yz\Big)$ using a tree as indicated in a previous example to obtain the last expression. One must practise these steps oneself in order to learn the material.

We further note that: $xyz \not\subseteq xyz'$ $xyz' \not\subseteq xyz$

$xyz \not\subseteq xz'$ $xz' \not\subseteq xyz$

$xyz' \not\subseteq xz'$ $xz' \not\subseteq xyz'$

Hence $SOP(E) = xyz+xyz'+xz'$

R Since $Ukn(E) = \{x, y, z\}$ while the summand xz' in $SOP(E)$ does not contain y $SOP(E)$ is not in complete sum of products form. To put it in complete sum of products form we carry out the corresponding algorithm from the preceding as follows.

R You need to exhibit trees for all multiplications. These have been omitted here.

$$
\begin{array}{cccccccccc}
 & SOP(E) & & & & & & & & \\
= & xyz & = & +\ x(1)z' & = & xyz & = & xyz & = & xyz \quad = CSOP(E) \\
| & +\ xyz' & | & +\ xyz' & | & +\ xyz' & | & +\ xyz'(1) & | & +\ xyz' \\
| & +\ xz' & | & & | & +\ x(y+y')z' & | & +\ \cancel{xyz'}(2) & | & \\
| & & & & & & & +xy'z' & & +xy'z'
\end{array}
$$

Therefore, we have

$$E := (xy' + z)(xy + yz'xz)$$

$$SOP(E) = xyz + xyz' + xz'$$

$$CSOP(E) = xyz + xyz' + xy'z'$$

R Note that after expanding all terms so that each has exactly all the unknowns of E, we retain only the first instance of a summand if it is related, since, un Boolean Aldebras, $x + x = x$

Prime Implicants and Minimal Sum of Products

D Given a boolean expression E in SOP form, we define:

(0) $Smnd(E) \coloneqq$ The set of summands in E

(1) $E_L \coloneqq \nu\big(MSLtr(E)\big)$

(2) $E_S \coloneqq \nu\big(Smnd(E)\big)$

R Note that the number of summands is precisely 1 larger than the number of +'s.

E $G \coloneqq xyz' + xy' + x'yz' + x'y'z'$

$$MSLtr(G) \coloneqq \Big\{x\colon 2, x'\colon 2, y\colon 2, y'\colon 2, z'\colon 3\Big\}$$

Therefore, $G_L \coloneqq \nu\big(MSLtr(E)\big) = 2 + 2 + 2 + 2 + 3 = 11$

Also $G_S \coloneqq \nu\big(Smnd(E)\big) = 4$

E $H \coloneqq xy + z$

$$MSLtr(H) \coloneqq \Big\{x\colon 1,\ y\colon 1, z\colon 1\Big\}$$

Therefore, $H_L \coloneqq \nu\big(MSLtr(H)\big) = 1 + 1 + 1 = 3$

Also $H_S \coloneqq \nu\big(Smnd(H)\big) = 2$

D Given two equivalent boolean expression E and F both in SOP form, we define: a relation as follows:

$$Smplr(E,F) \coloneqq \Bigg(\Big(E_L < F_L\Big) \ \wedge \ \Big(E_S \leq F_L\Big)\Bigg) \vee \Bigg(\Big(E_L \leq F_L\Big) \ \wedge \ \Big(E_S < F_L\Big)\Bigg)$$

E Given the two equivalent expressions $G \coloneqq x$ and $H \coloneqq x + xy$, determine which one, if any, is simpler.

$G \coloneqq x$

$MSLtr(G) \coloneqq \{x: 1\}$

Therefore, $G_L \coloneqq \nu\big(MSLtr(E)\big) = 1$

Also $G_S \coloneqq \nu\big(Smnd(E)\big) = 1$

$H \coloneqq x + xy$

$MSLtr(H) \coloneqq \left\{x: 2,\ y: 1\right\}$

Therefore, $H_L \coloneqq \nu\big(MSLtr(H)\big) = 2 + 1 = 3$

Also $H_S \coloneqq \nu\big(Smnd(H)\big) = 3$

$$\frac{\left(G_L = 1 < 3 = H_L\right) \ \wedge \ \left(G_S = 1 \leq 3 = H_L\right)}{Smplr(G, H)}$$

D A boolean expression E in SOP form is said to be minimal, and written

$$Mnml(E): \Leftrightarrow \ \neg\big(\exists\, F\ (SOPFrm(F))\big) \wedge (E \Leftrightarrows F) \wedge \big(Smplr(E, F)\big)$$

D A fundamental product P is called a prime implicant of a boolean expression E, and written:

$$PI(P, E): \Leftrightarrow (P + E = E) \wedge \left(\frac{FP(Q) \quad Q \subseteq P}{Q + E \neq E}\right)$$

D $PI(E) \coloneqq \left\{P \in BA \,\middle|\, PI(P, E)\right\}$ denotes the set of prime implicants of E.

D A Boolean expression E is said to be in minimal sum-of-products ($MSOP$) form, and written

$MSOPFrm(E) :\Leftrightarrow E = \sum_{k=1}^{n} P_k$ where

(1) $\forall k \epsilon 1..n, PI(P_k, E)$ and

(2) $\frac{k \neq l}{P_k \neq P_l}$ and (3) $\forall k \epsilon 1..n \left(E \neq \sum_{l \epsilon ((1..n) \setminus \{k\})}^{(\ )} P_l\right)$

R A minimal sum-of-products of for a boolean expression E is a sum of irredundant prime implicants of E.

R The $MSOP$ form of a boolean expression E is not unique and any of the possible forms is denoted by $MSOP(E)$.

D Consensus of fundamental products

Given two fundamental products $P$and Q such that:

(1) at least one is of length strictly greater than 1, and

(2) exactly one boolean unknown occurs uncomplemented in the one and complemented in the other.

we define the consensus of $P$and Q, written $Cns(P, Q)$as follows:

$Cns(P, Q)$

$\coloneqq$ the reduced form of the word obtained by juxtaposing P and Q and deleting x and x'

L $P + Q + Cns(P, Q) = P + Q$

E $P \coloneqq wxyz'$ $Q \coloneqq xy'z$

Since the length of each strictly exceeds 1, and exactly one unknown y occurs complemented in the one and not complemented in the other $CNS(P, Q) = wxz$

E $P \coloneqq x'y \qquad Q \coloneqq x$

Since the length of $x'y$ strictly exceeds 1, and exactlly one unknown x occurs complemented in the one and not complemented in the other, $CNS(P,Q) = y$

E $P \coloneqq wx'y \qquad Q \coloneqq x'yz$

Since the number of unknowns that occur complemented in the one and uncomplemented in the other is 0, $CNS(P,Q)$ does not exist; we denote this state of affairs by $\cancel{CNS}(P,Q)$.

E $P \coloneqq x'yz \qquad Q \coloneqq xyz'$

Since the number of unknowns that occur complemented in the one and uncomplemented in the other is 2 (these are x and z), $CNS(P,Q)$ does not exist; we denote this state of affairs by $\cancel{CNS}(P,Q)$.

Consensus method for finding all prime implicants of a boolean expression E

The input is $E = \sum_{k=1}^{n} F_k$ expressed as a sum of fundamental products. The algorithm ends with E expressed as a sum of all its prime implicants.

A Input: A boolean expression E expressed as a sum of fundamental products

Output: E expressed as a sum of all its prime implicants

(1) Delete any fundamental product P that includes any other fundamental product Q

(2) Add the consensus $Cns(P_k, P_l)$ provided that $Cns(P_k, P_l)$ does not include of the other summands Q.

(3) Repeat (1) and (2) until neither can be applied any further.

Algorithm for producing $MSOP(E)$

The following algorithm may be used to produce a $MSOPFrm(E)$. The input is the result of the previous algorithm, namely, the boolean expression E expressed as the sum of all prime implicants.

A Input: Boolean expression E expressed as the sum of all its prime implicants

Output: E expressed as a minimal sum of its prime implicants

(1) Express each prime implicant P as $CSOP(P)$.

(2) Delete one by one every prime implicant that is redundant, that is whose summands occur among the summands of the other (remaining) prime implicants.

R $MSOPFrm(E)$ is not unique.

E Use the consensus method to find all prime implicants of E and $MSOP(E)$, where

$$E \coloneqq xyz + xyz' + x'yz' + x'y'z + x'z'$$

R We order E before applying the algorithm. We scan from left to right and apply a suitable step of the algorithm to the candidates first encountered.

	E	
$=$	$xyz + xyz' + \underline{x'yz'} + x'y'z + \underline{x'z'}$	$(x'z' \subseteq x'yz')$
$=$	$\underline{xyz} + \underline{xyz'} + x'y'z + x'z'$	$(Cns(xyz, xyz') = xy)$
$=$	$\underline{xyz} + \underline{xyz'} + x'y'z + x'z' + \underline{xy}$	$(xy \subseteq xyz, xy \subseteq xyz')$
$=$	$\underline{x'y'z} + x'z' + \underline{xy}$	$(Cns(x'y'z, x'z') = x'y')$
$=$	$\underline{x'y'z} + x'z' + xy + \underline{x'y'}$	$(x'y' \subseteq x'y'z)$
$=$	$x'z' + xy + x'y'$	$(Cns(x'z', xy) = yz')$
$=$	$x'z' + xy + x'y' + yz'$	(Order terms)
$=$	$xy + x'y' + x'z' + yz'$	

The algorithm has stopped but we have to prove that this is the case.

We note that:

$$xy \not\subseteq x'y' \qquad x'y' \not\subseteq xy$$

$xy \not\subseteq x'z'$ $\quad x'z' \not\subseteq xy$

$x'y' \not\subseteq yz'$ $\quad yz' \not\subseteq x'y'$

$x'y' \not\subseteq x'z'$ $\quad x'z' \not\subseteq x'y'$

$x'y' \not\subseteq yz'$ $\quad yz' \not\subseteq x'y'$

$x'z' \not\subseteq yz'$ $\quad yz' \not\subseteq x'z'$

so that the first step cannot be applied.

We further note that:

~~Cns~~$(xy, x'y')$,

$Cns(xy, x'z') = yz'$ which is already present

~~Cns~~(xy, yz')

~~Cns~~$(\, x'y', x'z')$

$Cns(x'y', yz') = xz'$ which is already present

~~Cns~~$(x'z', yz')$

so that the second step cannot be applied.

Therefore, the algorithm has indeed stopped.

The reader will need to prove that the algorithm has stopped as indicated here in all submitted work.

Hence, the set of prime implicants of E is the following:

$$PI(E) = \{xy, x'y', x'z', yz'\}$$

Now we have to find all subsets of irredundant prime implicants.

First, using the algorithm for $CSOP$, we note that:

$xy = xy(1) = xy(z + z') = xyz + xyz'$

$x'y' = x'y'(1) = x'y'(z + z') = x'y'z + x'y'z'$

$x'z' = x'(1)z' = x'(y + y')z' = x'yz' + x'y'z'$

$yz' = (1)yz' = (x + x')yz' = xyz' + x'yz'$

Hence

$$
\begin{array}{cccccccc}
 & E & & & & & & \\
= & xy & = & xyz + xyz'(11) & = & xyz + xyz'(11) & = & xy \\
| & +x'y' & | & +x'y'z + x'y'z'(21) & | & +x'y'z + x'y'z'(21) & | & +x'y' \\
| & +\cancel{x'z'} & | & +x'yz'(31) + x'y'z'(22) & | & +\cancel{x'yz'}(31) + \cancel{x'y'z'}(22) & | & \\
| & +yz' & | & +xyz'(12) + x'yz'(32) & | & +xyz'(12) + x'yz'(32) & | & +yz'
\end{array}
$$

Hence

$$MSOP_1(E) \quad = \quad xy + x'y' + yz'$$

The bookkeeping scheme keeps track of terms that are repeated. The indices in parentheses (mn) denotes the nth occurrence of the mth term. (32) denotes the second occurrence of the third term that is repeated. (11) is immediately the right of the 1st term that is repeated. Here the goal is to determine if some prime implicant is redundant. The first two rows are essential as one of the summands does not repeat in other rows. The third row is redundant because each of the summands can be found in some other row. Hence $x'z'$ is a redundant prime implicant as long as we retain the other three. The fourth row is also redundant beccause each of the summands can be found in some other row. Hence yz' is also a redundant prime implicant as long as we retain the other three; this possibility is exhibited below.

$$
\begin{array}{cccccccc}
 & E & & & & & & \\
= & xy & = & xyz + xyz'(11) & = & xyz + xyz'(11) & = & xy \\
| & +x'y' & | & +x'y'z + x'y'z'(21) & | & +x'y'z + x'y'z'(21) & | & +x'y' \\
| & +x'z' & | & +x'yz'(31) + x'y'z'(22) & | & +x'yz'(31) + x'y'z'(22) & | & +x'z' \\
| & +\cancel{yz'} & | & +xyz'(12) + x'yz'(32) & | & +\cancel{xyz'}(12) + \cancel{x'yz'}(32) & | &
\end{array}
$$

Hence

$$MSOP_2(E) \quad = \quad xy + x'y' + yz'$$

Thus we get two distinct minimal sum-of-products forms.

R For any submitted work one must follow exactly the same format. All multiplication must be shown using trees. No detail may be omitted. Any departure from correct format will lead to a score of 0.

Q Find $SOP(E), CSOP(E), PI(E)$, and all possible $MSOP(E)$ where

0 $E = (x + y)(x + y')$

1 $E = (x' + y)(x + y')$

3 $E = (x' + y)(y' + z)(x + z')$

4 $E = (x + y' + z)(x' + y + z')$

5 $E = (x + y)(x + z')(y' + z)$