$C_5 = \mathrm{Res}_p^{lock}(C_1, C_4) =_{(2)} p \vee_{(8)} \neg p$ and

$C_6 = \mathrm{Res}_p^{lock}(C_2, C_3) =_{(4)} q \vee_{(6)} \neg q$

If we apply also the deletion strategy, and we delete these tautologies, then we cannot continue the derivation process to obtain □.

Note that even if S is inconsistent, the empty clause (□) cannot be derived because there are too many restrictions imposed by lock resolution and the deletion strategy.

Therefore, *lock resolution + the deletion strategy is not complete.*

Continuing the lock resolution process but using also the resolvents C_5 and C_6 we obtain: $C_7 = \mathrm{Res}_p^{lock}(C_2, C_5) =_{(4)} q \vee_{(8)} \neg p$

The clauses C_7 and C_2 are similar but not identical, the literals order (provided by the indices) differs in these clauses: $\neg p$ is the literal to resolve upon from C_2, and q is the literal to resolve upon form C_7. Thus, the role of a tautology as a parent clause is to modify the literals order in the other parent clause.

$C_1 =_{(2)} p \vee_{(1)} q$, $C_2 =_{(3)} \neg p \vee_{(4)} q$, $C_3 =_{(5)} p \vee_{(6)} \neg q$, $C_4 =_{(8)} \neg p \vee_{(7)} \neg q$

$C_5 =_{(2)} p \vee_{(8)} \neg p$, $C_6 =_{(4)} q \vee_{(6)} \neg q$ $C_7 =_{(4)} q \vee_{(8)} \neg p$

$C_8 = \mathrm{Res}_q^{lock}(C_7, C_4) =_{(8)} \neg p$,

$C_9 = \mathrm{Res}_p^{lock}(C_3, C_8) =_{(6)} \neg q$,

$C_{10} = \mathrm{Res}_q^{lock}(C_1, C_9) =_{(2)} p$,

$C_{11} = \mathrm{Res}_p^{lock}(C_{10}, C_8) = \square$.

As a conclusion, in this second indexing the empty clause was derived from S with the help of tautologies.

Example 5.9.

In this example we shall prove that *lock resolution + the set-of-support strategy is not complete.*

Using lock resolution prove the deduction:

$$p \to (q \to r),\ r \wedge s \to t,\ u \to s \wedge \neg t \vdash p \wedge q \to \neg u.$$

The premises (hypotheses) of the deduction are:

$U_1 = p \to (q \to r)$, $U_2 = r \wedge s \to t$, $U_3 = u \to s \wedge \neg t$

The conclusion of the deduction is: $V = p \wedge q \to \neg u$

The resolution is applied to the set of clauses S, obtained from the conjunctive normal forms of the hypotheses and the negation of the conclusion.

$\text{CNF}(U_1) = \neg p \vee \neg q \vee r$, $C_1 = \neg p \vee \neg q \vee r$

$\text{CNF}(U_2) = \neg r \vee \neg s \vee t$, $C_2 = \neg r \vee \neg s \vee t$

$\text{CNF}(U_3) = (\neg u \vee s) \wedge (\neg u \vee \neg t)$, $C_3 = \neg u \vee s$, $C_4 = \neg u \vee \neg t$

$\text{CNF}(\neg V) = p \wedge q \wedge r$, $C_5 = p$, $C_6 = q$, $C_7 = u$

$S = \{C_1, C_2, C_3, C_4, C_5, C_6, C_7\}$

Using the set-of-support strategy we choose the support set $Y = \{C_5, C_6, C_7\}$ corresponding to the conclusion of the deduction.

$S \setminus Y = \{C_1, C_2, C_3, C_4\}$ is a consistent set of clauses corresponding to the premises (hypotheses) of the deduction.

We index the literals from the clauses as follows:

$C_1 = {}_{(3)}\neg p \vee {}_{(2)}\neg q \vee {}_{(1)} r$,

$C_2 = {}_{(6)}\neg r \vee {}_{(5)}\neg s \vee {}_{(4)} t$,

$C_3 = {}_{(8)}\neg u \vee {}_{(7)} s$,

$C_4 = {}_{(10)}\neg u \vee {}_{(9)}\neg t$,

$C_5 = {}_{(11)} p$, $C_6 = {}_{(12)} q$, $C_7 = {}_{(13)} u$

Note that the restriction imposed by lock resolution (the literals of lowest indices in their clauses resolve) combined with the set-of-support strategy (avoids to resolve two clauses belonging to the consistent set $S \setminus Y = \{C_1, C_2, C_3, C_4\}$) will block the resolution process and $\square$ cannot be derived.

If we do not use the set-of-support strategy we can derive $\square$.

$C_8 = \text{Res}_t^{lock}(C_2, C_4) = {}_{(6)}\neg r \vee {}_{(5)}\neg s \vee {}_{(10)}\neg u$

$C_9 = \text{Res}_s^{lock}(C_3, C_8) = {}_{(6)}\neg r \vee {}_{(8)}\neg u$

$C_{10} = \text{Res}_r^{lock}(C_1, C_9) = {}_{(3)}\neg p \vee {}_{(2)}\neg q \vee {}_{(8)}\neg u$

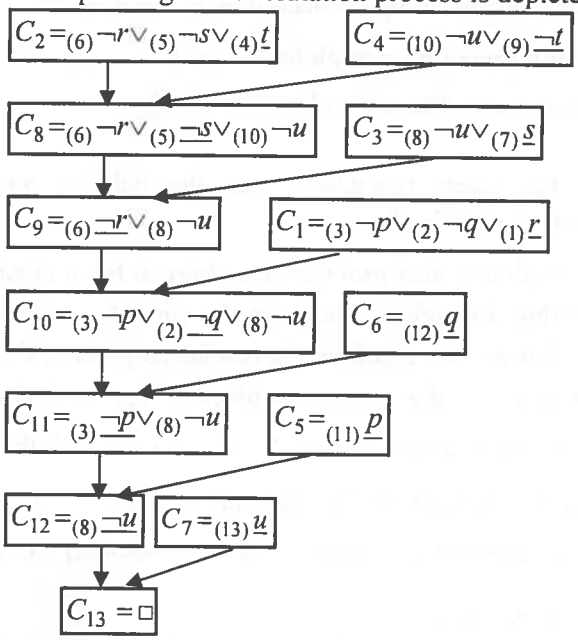
$C_{11} = \text{Res}_q^{lock}(C_6, C_{10}) = {}_{(3)}\neg p \vee {}_{(8)}\neg u$

$C_{12} = \text{Res}_p^{lock}(C_5, C_{11}) = {}_{(8)}\neg u$

$C_{13} = \text{Res}_u^{lock}(C_7, C_{12}) = \square$

We conclude that the set $S = \{C_1, C_2, C_3, C_4, C_5, C_6, C_7\}$ is inconsistent and *the deduction* $p \to (q \to r)$, $r \wedge s \to t$, $u \to s \wedge \neg t \vdash p \wedge q \to \neg u$ *holds.*

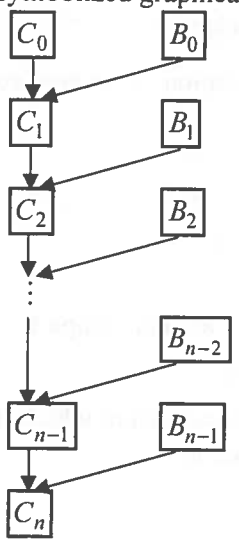
The binary tree corresponding to the refutation process is depicted below.



5.4. Linear resolution

Proposed by D.W. Loveland in 1970 [30], linear resolution is a very efficient refinement of resolution. The resolution process is a linear one: at each step one of the parent clauses is the resolvent derived at the previous step.

For a set S of clauses, a ***linear deduction*** of C_n from S with $C_0 \in S$ as the top clause is symbolized graphically as follows:



C_0 is the ***top clause***

$C_1, C_2, \ldots C_{n-1}, C_n$ are ***central clauses***

$B_0, B_1, \ldots, B_{n-1}$ are ***side clauses***

$\forall i = 1,\ldots,n : C_i = \mathrm{Re}s(C_{i-1}, B_{i-1})$

$\forall i = 0,\ldots,n-1 : B_i \in S \cup \{C_0, C_1, \ldots, C_{i-2}\}$

Linear resolution is sound and complete according to the theorem:

Theorem 5.9. Soundness and completeness [30]
The set S of clauses is inconsistent if and only if $S \vdash_{\text{Res}}^{lin} \square$.

We can combine the linear resolution with the deletion strategy and the completeness property is preserved.

This refinement of resolution also provides a strategy at the implementation level: **backtracking** algorithm. In each iteration, for the current central clause there are more possible side clauses. We continue the resolution process choosing one side clause. If in the iteration i the process is blocked (the central clause C_i is a tautology or it is an existing central clause: $C_i = C_j, j < i$) or all the side clauses of C_i were used, then we go back to the previous iteration $(i-1)$ and we choose another possible side clause for C_{i-1} to continue the resolution process.

The algorithm stops in two cases:
- the empty clause was derived and the conclusion is that S is inconsistent.
- for the top clause all the possible side clauses were used, but the empty clause was not derived, then we conclude that the set S is consistent.

The consistency of a set of clauses is proved after a complete search without the derivation of the empty clause.

Special cases of linear resolution [11]:
- ***unit resolution***: the central clauses have at least a unit clause as a parent clause.
- ***input resolution***: all side clauses are initial (input) clauses.

The equivalence of unit resolution and input resolution is expressed by the following theorem.

Theorem 5.10. [11]
Let S be a set of clauses. $S \vdash_{\text{Res}}^{input} \square$ *if and only if* $S \vdash_{\text{Res}}^{unit} \square$.

These two refinements of resolution are sound, but they are not complete:

1. **soundness**: If $S \vdash_{\text{Res}}^{input/unit} \square$ then S is inconsistent;
2. **incompleteness**: there exist inconsistent sets of clauses from which the empty clause cannot be derived using input or unit resolution.

Example 5.10.
The set $S = \{p \vee q, \neg p \vee q, p \vee \neg q, \neg p \vee \neg q\}$ is inconsistent (see Example 5.1), but there is no unit refutation from S because there is no unit clause in S. According to Theorem 5.10 there is no input refutation from S. This is an example of incompleteness of unit/input resolution.

Definition 5.3.
A clause is called a ***positive clause*** if it contains only positive literals. A clause is called a ***negative clause*** if it contains only negative literals. A clause is called ***Horn clause*** if it contains exactly one positive literals, all the other literals are negative.

Theorem 5.11.
The input resolution is complete on a set of Horn clauses with a negative top clause. (PROLOG).

Example 5.11. knowledge base
From the hypotheses:

$H_1 : U_1 \wedge U_2 \wedge ... \wedge U_n \rightarrow V$

$H_2 : X_1 \wedge X_2 \wedge ... \wedge X_l \rightarrow Y$

...

$H_j : W_1 \wedge W_2 \wedge ... \wedge W_m \rightarrow R$

is $C = Z_1 \wedge Z_2 \wedge ... \wedge Z_m$ deducible?

To prove by refutation using resolution we have to transform into clausal normal forms the hypotheses and the negation of the conclusion.

A formula of type: $U_1 \wedge U_2 \wedge ... \wedge U_n \rightarrow V$ provides a Horn clause:

$\neg U_1 \vee \neg U_2 \vee ... \vee \neg U_n \vee V$

The hypothesis H_i provides the Horn clause $C_i, i = 1,2,..., j$.

The negation of C is $\neg(Z_1 \wedge Z_2 \wedge ... \wedge Z_m)$ and provides a negative clause:

$C_{j+1} = \neg Z_1 \vee \neg Z_2 \vee ... \vee \neg Z_m$

We have to apply the input resolution to the set $S = \{C_1, C_2, ..., C_j, C_{j+1}\}$ with the top clause C_{j+1}.

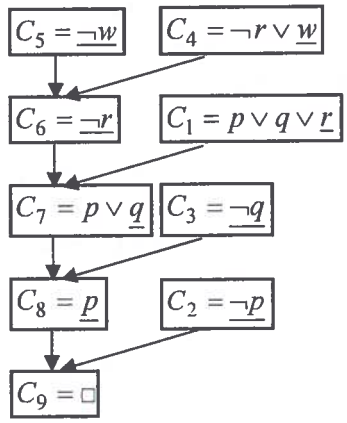
According to Theorem 5.11 we have that:

$H_1, H_2, ..., H_j \vdash C$ if and only if $S \vdash_{Res}^{input} \square$.

Example 5.12.
Using linear resolution prove the inconsistency of the set S of clauses: $S = \{C_1, C_2, C_3, C_4, C_5\}$.

$C_1 = p \vee q \vee r$, $C_2 = \neg p$, $C_3 = \neg q$, $C_4 = \neg r \vee w$, $C_5 = \neg w$

The derivation tree is as follows:



Top clause: C_5, side clauses: C_4, C_1, C_3, C_2, central clauses: C_6, C_7, C_8, C_9.

This linear resolution is also an *input* resolution and a *unit* resolution

$S \vdash^{lin}_{Res} \square$, therefore S is an inconsistent set.

Example 5.13.

Check the consistency/inconsistency of the set S of clauses using linear resolution.

$S = \{C_1 = p \vee q, C_2 = \neg p \vee q, C_3 = \neg p \vee \neg q\}$

We begin the linear search of the derivation of the empty clause using the backtracking strategy:

First iteration:

C_1 **is the top clause and it has 3 possible side clauses:** C_2, C_3, C_3.

Note that C_1 and C_3 can resolve in two ways: with p and q respectively as the literal resolved upon.

I) C_2 **is used as a side clause for** C_1

Second iteration:

$C_4 = \text{Res}_p(C_1, C_2) = q$ – central clause having one possible side clause: C_3

Third iteration:

$C_5 = \text{Res}_q(C_4, C_3) = \neg p$ – central clause having one possible side clause: C_1

Fourth iteration:

$C_6 = \text{Res}_p(C_5, C_1) = q = C_4$ (an existing clause)

The search process is blocked, otherwise it will be an infinite loop.

We go back to the previous iteration (*third iteration*): all the possible side clauses of C_5 were used.

We go back to the previous iteration (*second iteration*): all the possible side clauses of C_4 were used.

We go back to the previous iteration (*first iteration*).

II) C_3 **(with $\neg p$ as the literal resolved upon) is used as a side clause for** C_1

Second iteration:

$C_7 = \text{Res}_p(C_1, C_3) = q \vee \neg q \equiv T$ (tautology)

The search process is blocked, because a tautology cannot help to derive $\square$ (inconsistency).

We go back to the previous iteration (*first iteration*).

III) C_3 **(with $\neg q$ as the literal resolved upon) is used as a side clause for** C_1

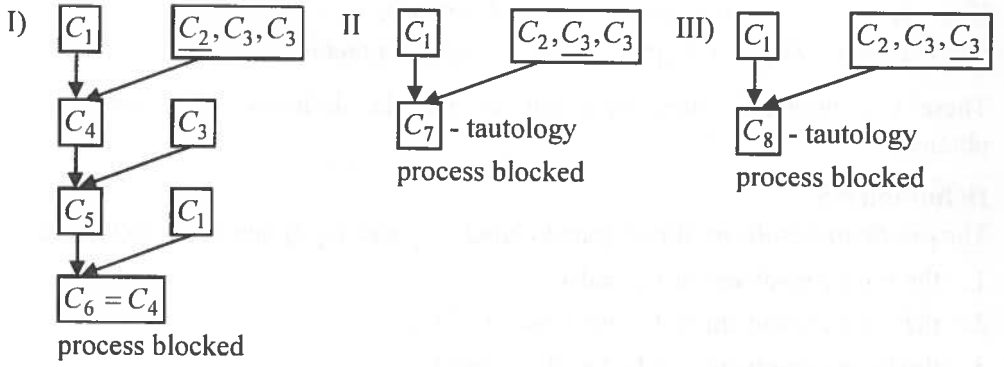
$C_8 = \text{Res}_q(C_1, C_3) = p \vee \neg p \equiv T$ (tautology)

The search process is blocked.

We go back to the previous iteration (*first iteration*).

All the possible side clauses of C_1 were used. A complete linear derivation search was performed, but the empty clause was not derived, so S is a consistent set.

The graphical representation of the resolution process, following the backtracking strategy is depicted below.



5.5. *Resolution in first-order (predicate) logic*

We present the ***predicate resolution*** as a formal (axiomatic) system according to the approach from the paper [63]:

$\text{Res}^{\text{Pr}} = (\Sigma_{\text{Res}}^{\text{Pr}}, F_{\text{Res}}^{\text{Pr}}, A_{\text{Res}}^{\text{Pr}}, R_{\text{Res}}^{\text{Pr}})$, where:

- $\Sigma_{\text{Res}}^{\text{Pr}} = \Sigma_{\text{Pr}} - \{\rightarrow, \leftrightarrow, \wedge, \exists, \forall\}$ is the alphabet;
- $F_{\text{Res}}^{\text{Pr}} \cup \{\square\}$ is the set of well formed formulas;
  - $F_{\text{Res}}^{\text{Pr}}$ is the set of all clauses built using the alphabet $\Sigma_{\text{Res}}^{\text{Pr}}$;
  - $\square$ is empty clause which does not contain any literal and symbolizes inconsistency;
- $A_{\text{Res}}^{\text{Pr}} = \varnothing$ is the set of axioms;
- $R_{\text{Res}}^{\text{Pr}} = \{res^{\text{Pr}}, fact\}$ is the set of inference rules containing the ***resolution rule*** (res^{Pr}) and the ***factoring rule*** (*fact*).

  $f \vee l_1, g \vee \neg l_2 \vdash_{res^{\text{Pr}}} \lambda(f) \vee \lambda(g)$, where $\lambda = mgu(l_1, l_2)$, $f, g \in F_{\text{Res}}^{\text{Pr}}$.

  $l_1 \vee l_2 \vee ... \vee l_k \vee l_{k+1} \vee ... \vee l_n \vdash_{fact} l_1 \vee l_{k+1} \vee ... \vee l_n$, where $\lambda = mgu(l_1, l_2, ..., l_k)$

Definition 5.4.
The predicate clauses $C_1 = f \vee l_1$ and $C_2 = g \vee \neg l_2$, without common free variables, are called ***clashing clauses*** if the literals l_1 and l_2 are unifiable: there exists $\lambda = mgu(l_1, l_2)$.

The ***binary resolvent*** of C_1 and C_2 is $C = \text{Res}_{\lambda}^{\text{Pr}}(C_1, C_2) = \lambda(f) \vee \lambda(g)$.
If $C = l_1 \vee l_2 \vee ... \vee l_k \vee l_{k+1} \vee ... \vee l_n$ and $\lambda = mgu(l_1, l_2, ..., l_k)$,
$Fact(C) = \lambda(l_1) \vee \lambda(l_{k+1}) \vee ... \vee \lambda(l_n)$ is called a ***factor*** of C.

These two inference rules are combined and the definition of a resolvent is obtained.

Definition 5.5.
The ***predicate resolvent*** of two parent clauses C_1 and C_2 is one of the following:

1. the binary resolvent of C_1 and C_2;
2. the binary resolvent of C_1 and a factor of C_2;
3. the binary resolvent of a factor of C_1 and C_2;
4. the binary resolvent of a factor of C_1 and a factor of C_2.

Example 5.14.
a) $C_1 = P(f(x), g(y)) \vee Q(x, y)$ and $C_2 = \neg P(f(f(a)), g(z)) \vee Q(f(a), g(z))$

$\text{Res}_{[x \leftarrow f(a), y \leftarrow z]}^{\text{Pr}}(C_1, C_2) = Q(f(a), g(z)) \vee Q(f(a), z)$

b) $C = P(x, g(y), z) \vee P(a, g(b), a) \vee Q(x, z)$

$l_1 = P(x, g(y), z), l_2 = P(a, g(b), a)$

$Fact(C) = \lambda(C)$, where $\lambda = [x \leftarrow a, y \leftarrow b, z \leftarrow a] = mgu(l_1, l_2)$.

$Fact(C) = P(a, g(b), a) \vee Q(a, a)$ is a factor of C

Theorem 5.12.

Let $U_1, U_2, \ldots U_n, V$ be first-order formulas.

1. V is a theorem *if and only if* $(\neg V)^c \vdash_{\text{Res}}^{\text{Pr}} \square$.
2. $U_1, U_2, \ldots, U_n \vdash V$ *if and only if* $\{U_1^c, U_2^c, \ldots, U_n^c, (\neg V)^c\} \vdash_{\text{Res}}^{\text{Pr}} \square$.

All the refinements and strategies for propositional logic can be used in predicate logic. It is recommended to rename the free variables in the initial set of clauses, such that they will be distinct in different clauses.

Algorithm predicate_ resolution:

input: $U_1, U_2, \ldots U_n, V$ - first-order formulas.

output: message: "$U_1, U_2, \ldots U_n \vdash V$" or „$U_1, U_2, \ldots U_n \nvdash V$" or

"we cannot decide if $U_1, U_2, \ldots U_n \vdash V$ or $U_1, U_2, \ldots U_n \nvdash V$"

begin

build the set of clauses: $S := \{U_1^c, U_2^c, \ldots, U_n^c, (\neg V)^c\}$;

do{

select l_1, l_2, C_1, C_2 such that:

C_1, C_2 are clauses or factors of clauses of S;

$l_1 \in C_1$, and $\neg l_2 \in C_2$;

if (l_1 and l_2 are unifiable with $\theta := mgu(l_1, l_2)$) **then**

$C := \text{Res}_\theta^{\text{Pr}}(C_1, C_2)$;

if ($C = \square$) **then** write "$U_1, U_2, \ldots U_n \vdash V$"; **exit**;

else $S := S \cup \{C\}$;

end_if

end_if

}**until** (no new resolvents can be derived **or** a predefined quantity of effort was done)

if (no new resolvents can be derived) **then** write "$U_1, U_2, \ldots U_n \nvdash V$";

else write "we cannot decide if $U_1, U_2, \ldots U_n \vdash V$ or $U_1, U_2, \ldots U_n \nvdash V$"

end_if

end

The resolution algorithm for predicate logic is a semi-decision procedure.

Theorem 5.13. (Church 1936 [13])
The problem of validity of a first-order formula is *undecidable*, but it is *semi-decidable*. If a procedure *Proc* is used to check the validity of a first-order formula U we have the following cases:

1. if the formula U is valid, then *Proc* ends with the corresponding answer.
2. if the formula U is not valid, then *Proc* ends with the corresponding answer or *Proc* may never stop.

Example 5.15.
Check if $U \vdash V$, where $U = (\forall x)(P(x) \to Q(x))$ and $V = (\forall x)P(x) \to (\forall x)Q(x)$.

We transform the formulas U and $\neg V$ into clausal normal forms:

- Prenex normal forms:

  $U^P = (\forall x)(\neg P(x) \vee Q(x))$

  $(\neg V)^P = \neg((\forall z)P(z) \to (\forall y)Q(y)) \equiv (\forall z)P(z) \wedge (\exists y)\neg Q(y) \equiv$

  $\equiv (\forall z)(\exists y)(P(x) \wedge \neg Q(y))$

- Skolem normal forms:

  $U^S = (\forall x)(\neg P(x) \vee Q(x))$,

  $(\neg V)^S = (\forall z)(P(z) \wedge \neg Q(f(z)))$, f is a unary Skolem function

- Clausal normal forms:

  $U^C = \neg P(x) \vee Q(x)$, $\qquad (\neg V)^C = P(z) \wedge \neg Q(f(z))$

We consider the following set of clauses: $S = \{C_1, C_2, C_3\}$. The free variables are renamed such that they are distinct in the set S.

$C_1 = \neg P(x) \vee Q(x)$, $C_2 = P(y)$, $C_3 = \neg Q(f(z))$

We derive the empty clause using predicate resolution as follows:

$C_1, C_2 \vdash_{res^{Pr}}^{[y \leftarrow x]} C_4 = Q(x)$

$C_4, C_5 \vdash_{res^{Pr}}^{[x \leftarrow f(z)]} \square$

According to Theorem 5.12 we proved that $U \vdash V$.

Example 5.16.
Using linear resolution prove that $S = \{C_1, C_2, C_3, C_4\}$ is an inconsistent set of clauses:

$C_1 = P(x, f(x), e)$, $\qquad C_2 = \neg R(x) \vee \neg R(y) \vee \neg P(x, f(y), z) \vee R(z)$,

$C_3 = R(a)$, $\qquad C_4 = \neg R(e)$

Constants: a, e, function symbols: f, predicate symbols: P, R

For linear resolution we choose the top clause: C_4

$C_2 \vdash_{fact}^{[y \leftarrow x]} C_5 = \neg R(x) \vee \neg P(x, f(x), z) \vee R(z)$, C_5 is a factor of C_2

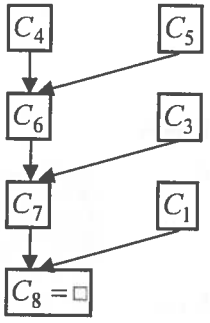
$C_4, C_5 \vdash_{res^{Pr}}^{[z \leftarrow e]} C_6 = \neg R(x) \vee \neg P(x, f(x), e)$, central clause

$C_6, C_3 \vdash_{res^{Pr}}^{[x \leftarrow a]} C_7 = \neg P(a, f(a), e)$, central clause

$C_7, C_1 \vdash_{res^{Pr}}^{[x \leftarrow a]} C_8 = \square$

From the set S we derived the empty clause, therefore S *is inconsistent.*

The refutation tree is depicted below.



top clause: C_4

side clauses: C_5, C_3, C_1

central clauses: C_6, C_7, C_8

Example 5.17.

Check the inconsistency of the set S of clauses using lock resolution.

$S = \{\neg P(x) \vee Q(x) \vee R(x), \neg Q(y) \vee R(y), P(a), \neg R(a)\}$

a) We index the literals as follows:

$C_1 = {}_{(3)}\neg P(x) \vee_{(2)} Q(x) \vee_{(1)} R(x)$

$C_2 = {}_{(5)}\neg Q(y) \vee_{(4)} R(y)$

$C_3 = {}_{(6)} P(a)$, $C_4 = {}_{(7)}\neg R(a)$

The following resolvents are obtained:

$C_5 = \mathrm{Res}^{Pr}_{[x \leftarrow a]}(C_1, C_4) = {}_{(3)}\neg P(a) \vee_{(2)} Q(a)$

$C_6 = \mathrm{Res}^{Pr}_{[y \leftarrow a]}(C_2, C_4) = {}_{(5)}\neg Q(a)$

$C_7 = \mathrm{Res}^{Pr}(C_5, C_6) = {}_{(3)}\neg P(a)$

$C_8 = \mathrm{Res}^{Pr}(C_6, C_7) = \square$

$S \vdash^{Pr}_{Res} \square$ and thus S *is inconsistent.*

b) another indexing of the literals:

$C_1 =_{(2)} \neg P(x) \vee_{(1)} Q(x) \vee_{(3)} R(x)$

$C_2 =_{(4)} \neg Q(y) \vee_{(5)} R(y)$

$C_3 =_{(6)} P(a)$ $C_4 =_{(7)} \neg R(a)$

The following resolvents are derived:

$C_5 = \mathrm{Res}^{\mathrm{Pr}}_{[y \leftarrow x]}(C_1, C_2) =_{(2)} \neg P(x) \vee_{(3)} R(x)$

$C_6 = \mathrm{Res}^{\mathrm{Pr}}_{[x \leftarrow a]}(C_3, C_5) =_{(3)} R(a)$

$C_7 = \mathrm{Res}^{\mathrm{Pr}}(C_4, C_6) = \square$

Another lock refutation from S was obtained.

Example 5.18.

Prove the semidistributivity of „$\forall$" over „$\rightarrow$":

$\vdash (\forall x)(P(x) \rightarrow Q(x)) \rightarrow ((\forall x)P(x) \rightarrow (\forall x)Q(x))$ and

$\nvdash ((\forall x)P(x) \rightarrow (\forall x)Q(x)) \rightarrow (\forall x)(P(x) \rightarrow Q(x))$

by applying predicate resolution.

We consider the predicate formulas.

$U_1 = (\forall x)(P(x) \rightarrow Q(x)) \rightarrow ((\forall x)P(x) \rightarrow (\forall x)Q(x))$

$U_2 = ((\forall x)P(x) \rightarrow (\forall x)Q(x)) \rightarrow (\forall x)(P(x) \rightarrow Q(x))$

According to Theorem 5.12:

- $\vdash U_1$ if and only if $(\neg U_1)^c \vdash^{\mathrm{Pr}}_{\mathrm{Res}} \square$ and
- $\nvdash U_2$ *if and only if* $(\neg U_2)^c \nvdash^{\mathrm{Pr}}_{\mathrm{Res}} \square$.

We apply syntactic transformations in order to obtain the prenex, Skolem and clausal normal forms of the formulas $\neg U_1$ and $\neg U_2$.

$\neg U_1 = \neg((\forall x)(P(x) \rightarrow Q(x)) \rightarrow ((\forall x)P(x) \rightarrow (\forall x)Q(x)))$

- the logical equivalence $\neg(A \rightarrow B) \equiv A \wedge \neg B$ is applied

  $\equiv (\forall x)(P(x) \rightarrow Q(x)) \wedge \neg((\forall x)P(x) \rightarrow (\forall x)Q(x)) \equiv$

  $\equiv (\forall x)(P(x) \rightarrow Q(x)) \wedge (\forall x)P(x) \wedge \neg(\forall x)Q(x)$
- infinitary DeMorgan's law: $\neg(\forall x)A(x) \equiv (\exists x)\neg A(x)$ is applied

  $\equiv (\forall x)(\neg P(x) \vee Q(x)) \wedge (\forall x)P(x) \wedge (\exists x)\neg Q(x)$
- rename the bound variables

  $\equiv (\forall x)(\neg P(x) \vee Q(x)) \wedge (\forall y)P(y) \wedge (\exists z)\neg Q(z)$
- extraction of the quantifiers in front of the formula, we extract first the existential quantifier

  $\equiv (\exists z)(\forall x)(\forall y)((\neg P(x) \vee Q(x)) \wedge P(y) \wedge \neg Q(z)) = (\neg U_1)^p$ - prenex form

$(\neg U_1)^S = (\forall x)(\forall y)((\neg P(x) \vee Q(x)) \wedge P(y) \wedge \neg Q(a))$ - Skolem normal form

$[z \leftarrow a]$, a- Skolem constant

$(\neg U_1)^c = (\neg P(x) \vee Q(x)) \wedge P(y) \wedge \neg Q(a)$ - clausal normal form

The set of clauses used in predicate resolution is:

$S_1 = \{C_1 = \neg P(x) \vee Q(x), C_2 = P(y), C_3 = \neg Q(a)\}$

The following resolvents are derived:

$C_4 = \mathrm{Res}^{\mathrm{Pr}}_{[x \leftarrow a]}(C_1, C_3) = \neg P(a)$

$C_5 = \mathrm{Res}^{\mathrm{Pr}}_{[y \leftarrow a]}(C_2, C_4) = \square$

We proved: $(\neg U_1)^c \vdash^{\mathrm{Pr}}_{\mathrm{Res}} \square$, thus $\vdash (\forall x)(P(x) \rightarrow Q(x)) \rightarrow ((\forall x)P(x) \rightarrow (\forall x)Q(x))$.

$\neg U_2 = \neg(((\forall x)P(x) \rightarrow (\forall x)Q(x)) \rightarrow (\forall x)(P(x) \rightarrow Q(x)))$

- the logical equivalence $\neg(A \rightarrow B) \equiv A \wedge \neg B$ is applied

  $\equiv ((\forall x)P(x) \rightarrow (\forall x)Q(x)) \wedge \neg(\forall x)(P(x) \rightarrow Q(x))$
- the logical equivalence $A \rightarrow B \equiv \neg A \vee B$ is applied

  $\equiv (\neg(\forall x)P(x) \vee (\forall x)Q(x)) \wedge \neg(\forall x)(P(x) \rightarrow Q(x))$
- infinitary DeMorgan's law: $\neg(\forall x)A(x) \equiv (\exists x)\neg A(x)$ and

  $\neg(A \rightarrow B) \equiv A \wedge \neg B$ are applied

  $\equiv ((\exists x)\neg P(x) \vee (\forall x)Q(x)) \wedge (\exists x)(P(x) \wedge \neg Q(x))$
- rename the bound variables

  $\equiv ((\exists x)\neg P(x) \vee (\forall y)Q(y)) \wedge (\exists z)(P(z) \wedge \neg Q(z))$
- the quantifiers $(\exists x),(\forall y),(\exists z)$ are independent: none of them is within the scope of the other one, so we can extract them in any order. To obtain the simplest Skolem form (with fewest Skolem functions) we extract first the existential quantifiers.

  $\equiv (\exists z)(\exists x)(\forall y)((\neg P(x) \vee Q(y)) \wedge P(z) \wedge \neg Q(z)) = (\neg U_2)^p$ - prenex form

$(\neg U_1)^S = (\forall x)(\forall y)((\neg P(x) \vee Q(x)) \wedge P(y) \wedge \neg Q(a))$ - Skolem normal form

$[z \leftarrow a, x \leftarrow b]$, a,b - Skolem constants

$(\neg U_2)^c = (\neg P(b) \vee Q(y)) \wedge P(a) \wedge \neg Q(a)$ - clausal normal form

The set of clauses used in predicate resolution is:

$S_2 = \{C_1' = \neg P(b) \vee Q(y), C_2' = P(a), C_3' = \neg Q(a)\}$

The only resolvent that can be derived is:

$C_4' = \text{Res}^{Pr}_{[y \leftarrow a]}(C_1', C_3') = \neg P(b)$

The literals $P(a), P(b)$ are not unifiable because a,b are distinct constants, so the clauses $C_2' = P(a), C_4' = \neg P(b)$ do not resolve and thus $\square$ cannot be derived.

We proved: $(\neg U_2)^c \not\vdash^{Pr}_{Res} \square$, so $\not\vdash (\forall x)(P(x) \to Q(x)) \to ((\forall x)P(x) \to (\forall x)Q(x))$

U_1 is a theorem, but U_2 is not a theorem, therefore we conclude that the quantifier '$\forall$' *is not distributive over the connective* '$\to$', *it is only semi-distributive.*

Example 5.19.

Using predicate resolution check the validity of the formula:

$U = (\forall y)(\exists x)\neg(P(x,y) \leftrightarrow \neg P(x,x))$

We apply Theorem 5.12 and we prove by contradiction that U is valid.

$\neg U = \neg((\forall y)(\exists x)\neg(P(x,y) \leftrightarrow \neg P(x,x))) \equiv (\exists y)(\forall x)(P(x,y) \leftrightarrow \neg P(x,x)) \equiv$

$\equiv (\exists y)(\forall x)((\neg P(x,y) \vee \neg P(x,x)) \wedge (P(x,x) \vee P(x,y)))$ - prenex normal form

$(\neg U)^S = (\forall x)(\neg P(x,a) \vee \neg P(x,x)) \wedge (P(x,x) \vee P(x,a))$ - Skolem normal form,

$[x \leftarrow a]$, a- Skolem constant

$(\neg U)^c = (\neg P(x,a) \vee \neg P(x,x)) \wedge (P(x,x) \vee P(x,a))$ - clausal normal form

From the above clausal normal form we consider the clauses:

$C_1 = \neg P(x,a) \vee \neg P(x,x)$ and

$C_2 = P(x,x) \vee P(x,a)$

Resolving these two clauses we can derive only tautologies.

It is important to apply the factoring rule for C_1 and C_2 and try to resolve their factors.

$C_1 \vdash^{[x \leftarrow a]}_{fact} C_3 = \neg P(a,a)$, C_3 is a factor of C_1.

$C_2 \vdash^{[x \leftarrow a]}_{fact} C_4 = P(a,a)$, C_4 is a factor of C_2.

$C_3, C_4 \vdash_{res} \square$

$(\neg U)^c \vdash^{Pr}_{Res} \square$, therefore $\neg U$ is inconsistent and U *is a valid formula.*

5.6. Semantic resolution

Semantic resolution, proposed by J.R.Slagle in 1965 [57], is a refinement of general resolution. In order to reduce the number of irrelevant and redundant clauses in the resolution process, this refinement of resolution:

- divides the set of clauses into two subsets using an interpretation (semantic aspect), to avoid resolving clauses within the same subset;
- uses the ordering of propositional/predicate symbols to impose a restriction on the literals resolved upon: the literal resolved upon from the parent clauses contains the largest symbol.

We suppose that the initial set of clauses, S, was simplified: tautologies, subsumed clauses and the clauses which contain pure literals were deleted. S is divided into the subsets: S_E and S_N using an interpretation I such that all the clauses from S_E are falsified by I and all the clauses from S_N are satisfied by I:

$S = S_E \cup S_N$.

There always exists at least one interpretation such that S_E and S_N are non-empty sets.

The basic idea is to avoid resolving two clauses which are both falsified or both satisfied by the same interpretation, because the resolvent is irrelevant in the process of deriving the empty clause.

The ordering of the propositional/predicate symbols and the restriction on the literal resolved upon cut down the number of useless clauses generated in the resolution process.

Another important idea is to unify all the variants of generating the same resolvent, where the order of using the parent clauses is immaterial.

Example 5.20.

Let $S = \{E_1 = p, E_2 = q, E_3 = r, N = \neg p \vee \neg q \vee \neg r\}$ be a set of propositional clauses.

There are six possible derivations of the empty clause, with the only difference being the order of using the clauses E_1, E_2, E_3 as parent clauses. We present in the following two such derivations of $\square$.

Version 1:

$R_2 = \text{Res}(N, E_1) = \neg q \vee \neg r$;

$R_3 = \text{Res}(R_2, E_2) = \neg r$;

$R_4 = \text{Res}(R_3, E_3) = \square$;

Version 2:

$R'_2 = \text{Res}(N, E_2) = \neg p \vee \neg r$;

$R'_3 = \text{Res}(R_2, E_3) = \neg p$;

$R'_4 = \text{Res}(R_3, E_1) = \square$;

To avoid all these versions being considered distinct derivations of the empty clause we shall unify them.

The combination of all the ideas presented informally above is formalized in the following definition.

Definition 5.6.
Let I be an interpretation and P an ordering of the propositional/ predicate symbols. A finite set of clauses: $\{E_1, E_2, ..., E_q, N\}, q \geq 1$, is called a ***semantic clash*** with respect to P and I (or ***PI-clash***) if and only if $E_1, E_2, ..., E_q$ (called ***electrons***) and N (called ***nucleus***) satisfy the following conditions:

1. $E_1, E_2, ..., E_q$ are falsified by I.
3. Let $R_1 = N$ and $R_{i+1} = \text{Res}_P(R_i, E_i), i = 1, ..., q$, where the literal resolved upon in E_i contains the largest symbol in that clause. The resolvents $R_1, R_2, ..., R_q$, are all satisfied by I.
2. R_{q+1} is falsified by I. $R_{q+1} = \text{Res}_{PI}(N, E_1, E_2, ..., E_q)$ is called a ***PI-resolvent*** of the *PI*-clash $\{E_1, E_2, ..., E_q, N\}$.

Remarks:

- The order of the electrons is immaterial.
- In semantic resolution we can use any ordering of propositional/predicate symbols and any interpretation.
- The resolvents $R_1, R_2, ..., R_q$, are all satisfied by I. We do not keep them because they will not be used further in the resolution process.
- All the *PI*-resolvents are falsified by the interpretation I, so they are added to the set of electrons.

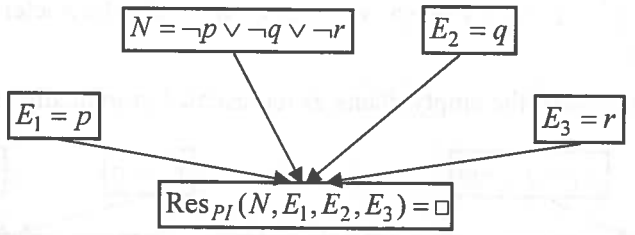
Example 5.21.
$S = \{p, q, r, \neg p \vee \neg q \vee \neg r\}$ is a set of propositional clauses, I is an interpretation, $I: \{p, q, r\} \to \{T, F\}$, $I(p) = F$, $I(q) = F$, $I(r) = F$, and P: $p > q > r$ is an ordering of the propositional variables.

The set S is a *PI*-clash:

- $S_E = \{E_1 = p, E_2 = q, E_3 = r\}$, the electrons are falsified by I
- $S_N = \{N = \neg p \vee \neg q \vee \neg r\}$, the nucleus is satisfied by I
- $R_1 = N = \neg p \vee \neg q \vee \neg r$ is satisfied by I
- $R_2 = \text{Res}_P(R_1, E_1) = \neg q \vee \neg r$ is satisfied by I

- $R_3 = \text{Res}_P(R_2, E_2) = \neg r$ is satisfied by I
- $R_4 = \text{Res}_P(R_3, E_3) = \square = \text{Res}_{PI}(N, E_1, E_2, E_3)$ is falsified by I.

We represent graphically the *PI*-clash as follows:



Example 5.22.

$S = \{E_1 = p \vee r, E_2 = q \vee r, N = \neg p \vee \neg q \vee r\}$ is a set of clauses, I is an interpretation, $I : \{p, q, r\} \to \{T, F\}$, $I(p) = F$, $I(q) = F$, $I(r) = F$, and P: $r > p > q$ is an ordering of the propositional symbols.

S is divided by the interpretation I as follows:

- $S_E = \{E_1 = p \vee \underline{r}, E_2 = q \vee \underline{r}\}$, the electrons are falsified by I
- $S_N = \{N = \neg p \vee \neg q \vee r\}$, the nucleus is satisfied by I

The underlined symbols are the largest in the electrons, according to the ordering P. No electrons can resolve with the nucleus, so S *is not a PI-clash.*

Definition 5.7.

Let S be a set of clauses, I an interpretation of S and P an ordering of the propositional/predicate symbols of S. A deduction from S is called a ***PI-deduction*** if and only if each clause in the deduction is either a clause of S or a *PI*-resolvent.

Theorem 5.14. Soundness and completeness [57]

Let S be set of clauses, I an interpretation of S and P an ordering of the propositional/predicate symbols of S. S is inconsistent if and only if there exists a *PI*-deduction of the empty clause from S.

Remark:

- The completeness property is preserved if this refinement of resolution is combined with the deletion strategy: because no tautologies are generated we only delete the subsumed clauses.

Example 5.23.

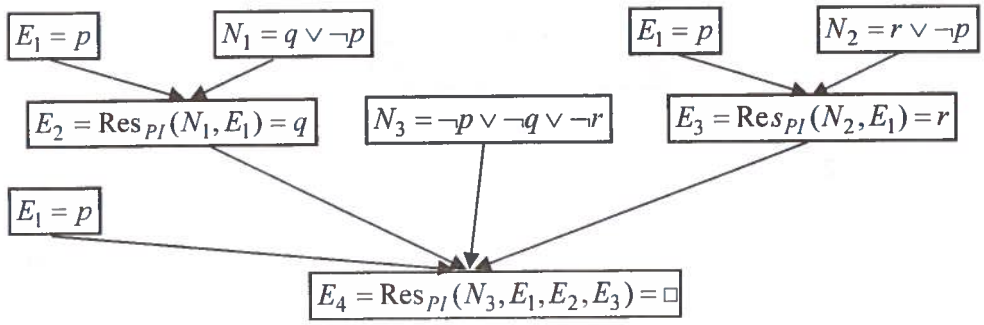
Using the semantic resolution prove that the following set of clauses is inconsistent. $S = \{p, q \vee \neg p, r \vee \neg p, \neg p \vee \neg q \vee \neg r\}$

1. We choose the interpretation $I_1 : \{p,q,r\} \to \{T,F\}$, $I_1(p) = F$, $I_1(q) = F$, $I_1(r) = F$, and the ordering $P_1 : p > q > r$

   $S_E = \{E_1 = p\}$, the electron is falsified by I_1

   $S_N = \{N_1 = q \vee \neg p, N_2 = r \vee \neg p, N_3 = \neg p \vee \neg q \vee \neg r\}$, the nuclei are satisfied by I_1

   The *PI*-deduction of the empty clause is represented graphically as follows:



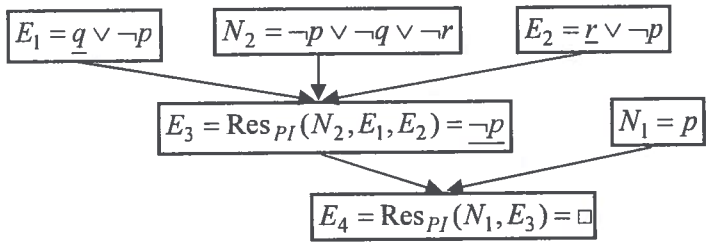
The *PI*-resolvents E_2, E_3 are falsified by the interpretation I_1 and they are used as electrons further in the semantic resolution process.

3. We choose: $I_2 : \{p,q,r\} \to \{T,F\}, I_2(p) = T, I_2(q) = F, I_2(r) = F$ and $P_2 : q > r > p$

   $S_E = \{E_1 = \underline{q} \vee \neg p, E_2 = \underline{r} \vee \neg p\}$, the electrons are falsified by I_2

   $S_N = \{N_1 = p, N_2 = \neg p \vee \neg q \vee \neg r\}$, the nuclei are satisfied by I_2

   The *PI*-deduction of the empty clause is represented as follows:



Special cases of semantic resolution are the *set-of-support strategy* [65] and *hyperresolution* (positive and negative) [51].

The set-of support strategy: avoids to resolve two clauses belonging to a consistent subset of the initial set of clauses, because the resolvents derived from a consistent set are irrelevant in the process of deriving the empty clause.

Hyperresolution is a semantic resolution with a particular interpretation that assigns the same truth value (F or T) to all the propositional variables.

Definition 5.8.
A ***clause*** is ***positive*** if it contains only non-negated atoms. A ***clause*** is ***negative*** if all its literals are negated atoms. A ***clause*** is called ***mixed*** if it is neither positive nor negative.

Definition 5.9.
A ***positive hyperresolution*** is a special case of *PI*-resolution in which the interpretation I assigns to all the propositional variables the truth value F. It is called positive hyperresolution because *all the electrons and PI-resolvents are positive clauses*.

Note: The negative and mixed clauses are nuclei.

Definition 5.10.
A ***negative hyperresolution*** is a special case of *PI*-resolution in which the interpretation I assigns to all the propositional variables the truth value *T*. It is called negative hyperresolution because *all the electrons and PI-resolvents are negative clauses*.

Note: The positive and mixed clauses are nuclei.

Often, the hypotheses of a deduction are represented by some positive and mixed clauses, and the negation of the conclusion is represented as a negative clause.
The ***positive hyperresolution*** corresponds to "**thinking forward**": from the hypotheses the conclusion is derived and together with the negation of the conclusion the empty clause is obtained.
The ***negative hyperresolution*** corresponds to "**thinking backward**": from the negation of the conclusion and the hypotheses the empty clause is derived.

Example 5.24.
Using positive and negative hyperresolution prove the following deduction:
$q \vee r, q \to r, w \vdash r \wedge w$.

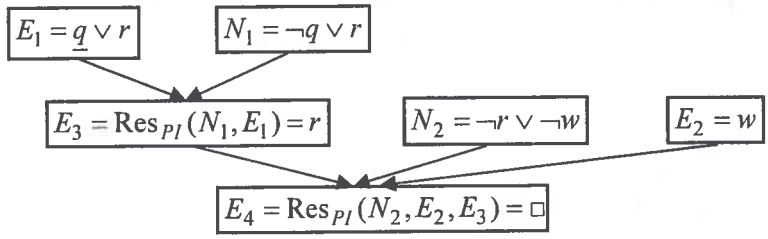
We consider the set of clauses corresponding to the hypotheses and the negation of the conclusion: $S = \{q \vee r, \neg q \vee r, w, \neg r \vee \neg w\}$

Positive hyperresolution - thinking forward
The interpretation is $I_1 : \{q, r, w\} \to \{T, F\}, I_1(q) = F, I_1(r) = F, I_1(w) = F$ and the ordering of the symbols is $P_1 : q > r > w$.

$S_E = \{E_1 = \underline{q} \vee r, E_2 = w\}$, the electrons are falsified by I_1

$S_N = \{N_1 = \neg q \vee r, N_2 = \neg r \vee \neg w\}$, the nuclei are satisfied by I_1

The *PI*-deduction of the empty clause is graphically represented as follows:



Negative hyperresolution- thinking backward

$S = \{q \vee r, \neg q \vee r, w, \neg r \vee \neg w\}$.

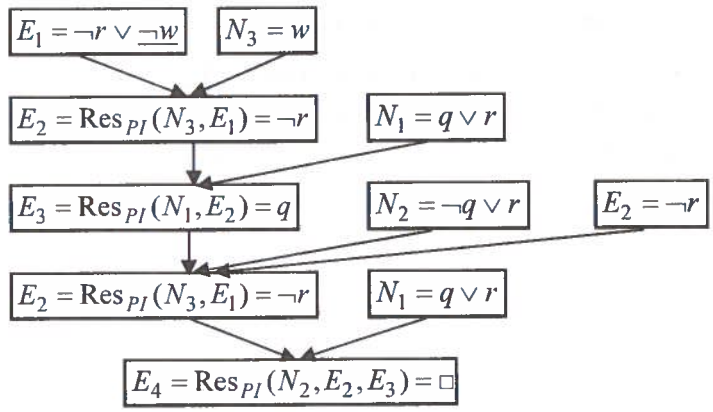
The interpretation is $I_2 : \{q, r, w\} \to \{T, F\}, I_2(q) = T, I_2(r) = T, I_2(w) = T$,

and the ordering of the propositional variables is $P_2 : w > q > r$

$S_E = \{E_1 = \neg r \vee \underline{\neg w}\}$ the electron is falsified by I_2

$S_N = \{N_1 = q \vee r, N_2 = \neg q \vee r, N_3 = w\}$ the nuclei are satisfied by I_2

The *PI*-deduction of the empty clause is



Positive hyperresolution can be simulated by lock resolution using the following rules for indexing the literals from the clauses [33]:

- **RP**$_1$: In the positive and negative clauses the indices of literals increase while the propositional symbols decrease in the ordering P of symbols.
- **RP**$_2$: In the mixed clauses the indices of literals with the negation sign are smaller than the indices of the literals without the negation sign. For the literals with the same sign, the rule for indexing is **RP**$_1$.

These rules block the resolution of two mixed clauses and the resolution of a negative clause and a mixed one. Therefore the resolution is permitted only to a positive clause (electron) and a negative or a mixed clause (nucleus). The literal resolved upon from the positive clause (electron) is the one which contains the largest propositional/predicate symbol.

Example 5.25.

$S = \{p \vee q, \neg p \vee q, p \vee \neg q, \neg p \vee \neg q\}$ is a set of clauses and

$P : p > q$ an ordering of the propositional symbols.

We shall simulate positive hyperresolution by lock resolution.

$I : \{p, q\} \to \{T, F\}$, $I(p) = F$, $I(q) = F$

The literals from the clauses are indexed according to the above rules as follows:

$E_1 =_{(1)} \underline{p} \vee_{(2)} q$ is an electron, the literal resolved upon must be p.

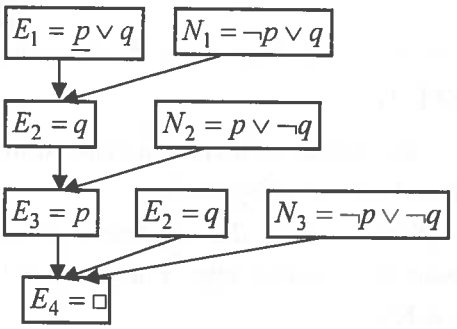
The nuclei are the clauses N_1, N_2, N_3 and we apply the rules $\mathbf{RP_1}$ and $\mathbf{RP_2}$

$N_1 =_{(3)} \neg p \vee_{(4)} q$,

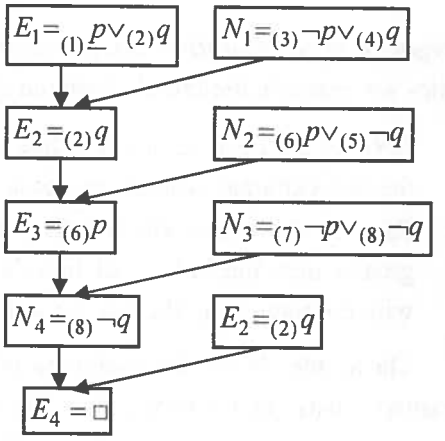
$N_2 =_{(6)} p \vee_{(5)} \neg q$,

$N_3 =_{(7)} \neg p \vee_{(8)} \neg q$

positive hyperresolution:



lock resolution:



Algorithm positive_hyperresolution_propositional_logic **[33]**
input: S - a set of clauses, P - an ordering of the propositional symbols;
output: message:"S ***is inconsistent***" or "S ***is consistent***"
begin

E^0 := all positive clauses from S indexed according to RP_1

N^0 := all non-positive (negative or mixed) clauses from S, indexed according to RP_1 and RP_2

$i := 0$;

// we will generate levels of electrons E^i and levels of nuclei N^i, $i > 0$

do

{

$i := i + 1$

$W := \{\text{Res}^{lock}(C_1, C_2) \mid C_1 \in E^{i-1}, C_2 \in N^{i-1}\}$

if ($\square \in W$) **then write** "S is inconsistent"; **exit**

end_if

$E^i := E^{i-1} \cup \{$ all positive clauses from $W\}$

$N^i := N^{i-1} \cup \{$ all non-positive clauses from $W\}$

} **until** ($E^i = E^{i-1}$ **and** $N^i = N^{i-1}$)

write "S is consistent"

end

Negative hyperresolution can be simulated by lock resolution using the following rules for indexing the literals from the clauses [33]:

- **RN_1:** In the positive and negative clauses the indices of literals increase while the propositional symbols decrease in the ordering P of symbols.
- **RN_2**: In the mixed clauses the indices of literals with the negation sign are greater than the indices of literals without the negation sign. For the literals with the same sign, the rule for indexing is **RN_1**.

These rules block the resolution of two mixed clauses and the resolution of a positive clause and a mixed one. Therefore the resolution is permitted only to a negative clause (electron) and a positive or mixed clause (nucleus). The literal resolved upon from the negative clause (electron) must contain the largest propositional/predicate symbol.

Example 5.26.

$S = \{p \vee q, \neg p \vee q, p \vee \neg q, \neg p \vee \neg q\}$ is a set of propositional clauses and

$P : q > p$ is an ordering of the propositional symbols.

We shall simulate negative hyperresolution by lock resolution.

$I : \{p, q\} \to \{T, F\}, I(p) = T, I(q) = T$

The literals from clauses are indexed according to the above rules as follows:

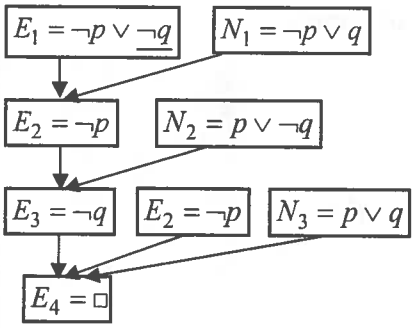
$E_1 =_{(2)} \neg p \vee_{(1)} \underline{\neg q}$ is an electron, the literal resolved upon must be $\neg q$.

The nuclei are the clauses N_1, N_2, N_3:

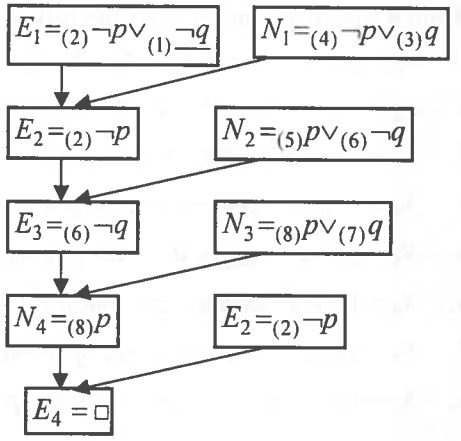
$N_1 =_{(4)} \neg p \vee_{(3)} q$, $N_2 =_{(5)} p \vee_{(6)} \neg q$, $N_3 =_{(8)} p \vee_{(7)} q$

For comparison, the negative hyperresolution process and the lock resolution process are represented graphically in the following.

negative hyperresolution:



lock resolution:



5.7. Exercises

Exercise 5.1.

Using general resolution prove that the following formulas are theorems.

1. $U_1 = (\neg B \to \neg A) \to ((\neg B \to A) \to B)$;
2. $U_2 = (B \to A) \wedge (C \to A) \to (B \wedge C \to A)$;
3. $U_3 = (B \to A) \wedge (C \to A) \to (B \vee C \to A)$;
4. $U_4 = (A \to C) \to ((\neg A \to B) \to (\neg B \to C))$;
5. $U_5 = A \vee (B \to C) \to (A \vee B) \to (A \vee C)$;
6. $U_6 = (A \to B) \to ((C \to A) \to (C \to B))$;
7. $U_7 = (A \to B) \to ((\neg A \to C) \to (\neg B \to C))$;
8. $U_8 = (A \to B \wedge C) \to (A \to B) \wedge (A \to C)$.

Exercise 5.2.
Using lock resolution check the inconsistency of the following sets of clauses. Choose two different indexings for the literals:

1. $S_1 = \{p \vee q, p \vee \neg q \vee r, p \vee \neg q \vee \neg r, \neg p \vee r, \neg p \vee \neg r\}$;
2. $S_2 = \{\neg p \vee \neg q, \neg p \vee q \vee \neg r, p \vee \neg r, \neg p \vee r, p \vee r\}$;
3. $S_3 = \{p \vee q, p \vee \neg q \vee \neg r, \neg p \vee \neg r, r, \neg p \vee r\}$;
4. $S_4 = \{p \vee q, \neg p \vee q \vee \neg r, \neg p \vee q \vee r, \neg q \vee \neg r, \neg q \vee r\}$;
5. $S_5 = \{p \vee \neg q, \neg p \vee \neg q \vee r, \neg p \vee q \vee r, p \vee q, \neg r\}$;
6. $S_6 = \{p \vee q, \neg p \vee q \vee \neg r, \neg p \vee \neg q \vee \neg r, p \vee \neg q, r\}$;
7. $S_7 = \{p \vee \neg q, \neg p \vee \neg q \vee r, \neg p \vee \neg q \vee \neg r, r \vee q, \neg r \vee q\}$;
8. $S_8 = \{p \vee r, p \vee q \vee \neg r, \neg p \vee \neg q \vee r, \neg p \vee q \vee r, \neg r\}$.

Exercise 5.3.
Build a linear refutation from the following set of clauses:

1. $S_1 = \{p \vee q \vee r, \neg q \vee r, \neg r, \neg p \vee r\}$;
2. $S_2 = \{p \vee \neg r, q \vee r, \neg q \vee r, \neg p \vee \neg r\}$;
3. $S_3 = \{q \vee r, \neg p, \neg q \vee r, p \vee \neg r\}$;
4. $S_4 = \{\neg p \vee q, p \vee \neg q \vee r, \neg r, p \vee q \vee r, \neg p \vee \neg q\}$;
5. $S_5 = \{p \vee r, \neg q, p \vee q \vee \neg r, \neg p \vee \neg r, q \vee r\}$;
6. $S_6 = \{p \vee q, \neg p \vee q, \neg p \vee \neg q, p \vee \neg q\}$;
7. $S_7 = \{p, q \vee r, \neg p \vee q \vee \neg r, \neg p \vee \neg q\}$;
8. $S_8 = \{p \vee \neg q \vee r, q, \neg p \vee \neg q \vee r, \neg p \vee \neg q \vee \neg r, p \vee \neg r\}$.

Exercise 5.4.
Prove the consistency of the following sets of clauses.

1. $S_1 = \{p \vee q \vee r, \neg q \vee r, \neg r, \neg p \vee r\}$;
2. $S_2 = \{p \vee \neg r, q \vee r, \neg q \vee r, \neg p \vee \neg r\}$;
3. $S_3 = \{q \vee r, \neg p, \neg q \vee r, p \vee \neg r\}$;
4. $S_4 = \{\neg p \vee q, p \vee \neg q \vee r, \neg r, p \vee q \vee r, \neg p \vee \neg q\}$;
5. $S_5 = \{p \vee r, \neg q, p \vee q \vee \neg r, \neg p \vee \neg r, q \vee r\}$;
6. $S_6 = \{p \vee q, \neg p \vee q, \neg p \vee \neg q, p \vee \neg q\}$;
7. $S_7 = \{p, q \vee r, \neg p \vee q \vee \neg r, \neg p \vee \neg q\}$;
8. $S_8 = \{p \vee \neg q \vee r, q, \neg p \vee \neg q \vee r, \neg p \vee \neg q \vee \neg r, p \vee \neg r\}$.

Exercise 5.5.
Using the set-of-support strategy prove the following deductions:

1. $\neg(p \vee q) \rightarrow r, \neg p \vee q \vee r, \neg r \vdash q \wedge \neg r$;
2. $p \vee \neg r, \neg q \rightarrow r, \neg q \vdash \neg(p \rightarrow q)$;
3. $q \wedge r \rightarrow p, p \vee q, q \rightarrow r \vdash p$;
4. $r \rightarrow p \vee q, \neg p \rightarrow r, \neg q \vdash p \wedge \neg q$;
5. $\neg p \rightarrow q, (q \rightarrow r) \wedge \neg r \vdash p \wedge \neg r$;
6. $q \rightarrow p, q \vee r, p \rightarrow r \vdash r$;
7. $\neg p \rightarrow q \vee r, \neg q, p \rightarrow q \vdash \neg(p \vee q) \wedge r$;
8. $r \rightarrow p, \neg p, q \rightarrow p \vee r \vdash \neg(\neg p \rightarrow q \vee r)$.

Exercise 5.6.
Prove the inconsistency of the following set of clauses using lock resolution. Try two different indexings for the literals.

1. $S_1 = \{\neg\, p(x) \vee q(x),\ p(a),\ \neg q(x) \vee \neg r(x),\ \neg w(a),\ r(y) \vee w(y)\}$;
2. $S_2 = \{\ p(x) \vee \neg q(x),\ \neg p(a) \vee r(x),\ q(x),\ w(z),\ \neg r(y) \vee \neg w(y)\}$;
3. $S_3 = \{\ p(x) \vee q(x) \vee r(x),\ \neg p(a),\ \neg q(x),\ \neg w(a),\ \neg r(y) \vee w(y)\}$;
4. $S_4 = \{\ p(x) \vee q(x),\ \neg p(x) \vee r(x),\ \neg q(y) \vee r(y),\ \neg r(x) \vee w(x),\ \neg w(f(z))\}$
5. $S_5 = \{\ p(x) \vee q(x),\ \neg p(a) \vee w(x),\ \neg q(y) \vee r(y),\ \neg r(x) \vee w(x),\ \neg w(a)\}$;
6. $S_6 = \{\neg p(x) \vee \neg q(x),\ p(z) \vee w(x),\ q(y) \vee w(y) \vee \neg r(y),\ \neg r(x) \vee \neg w(x),\ r(g(a,b))\}$
7. $S_7 = \{\ p(x) \vee q(x),\ \neg p(x),\ \neg q(f(a)) \vee r(z),\ \neg w(z),\ \neg r(y) \vee w(y)\}$
8. $S_8 = \{\neg p(x) \vee q(x) \vee \neg r(x),\ p(f(b)),\ \neg q(x),\ \neg w(y),\ r(y) \vee w(y)\}$.

Exercise 5.7.
Prove the following deductions using linear resolution

1. $(\forall x)(\forall y)(p(y,x) \wedge q(x) \rightarrow q(y)), (\forall x)(\forall y)(r(y,x) \rightarrow q(y)), r(b,a),$
$r(b,a), p(c,b) \vdash (\exists z) q(z)$;
2. $(\forall x)(p(x) \rightarrow r(x)), (\forall y)(r(y) \rightarrow q(y)), p(a), p(b) \vdash (\exists z) q(z)$;
3. $(\forall x)(\neg p(x) \wedge \neg q(x) \rightarrow r(x)), (\forall y)(r(y) \rightarrow w(y)), (\forall x)(w(x) \rightarrow p(x)),$
$\neg p(a), \neg p(b), \neg w(c) \vdash (\exists z) q(z)$;
4. $(\forall x)(p(x) \rightarrow r(x)), (\forall y)(r(y) \rightarrow q(y)), r(a), r(b), \neg r(c) \vdash (\exists z) q(z)$;
5. $(\forall x)(\neg p(x) \wedge \neg q(x) \rightarrow r(x)), (\forall y)(r(y) \rightarrow w(y)), (\forall x)(w(x) \rightarrow p(x)),$
$\neg p(a), \neg w(c) \vdash (\exists z) q(z)$;

6. $(\forall x)(\forall y)(\neg p(y,x) \to q(y)),(\forall x)(\forall y)(r(y,x) \wedge q(x) \to q(y)),$
$r(b,a),\neg p(a,b) \vdash (\exists z)q(z)$;
7. $(\forall x)(p(x) \to r(x)),(\forall y)(p(y) \to q(y)), p(a),\neg r(c) \vdash (\exists z)q(z)$;
8. $(\forall x)(p(x) \to r(x)),(\forall y)(p(y) \to q(y)), p(a), p(b),\neg p(c) \vdash (\exists z)q(z)$.

Exercise 5.8.
Using a refinement of predicate resolution prove:
1. the semidistributivity of ' $\forall$ ' over ' $\vee$ ':
$\vdash (\forall x)p(x) \vee (\forall x)q(x) \to (\forall x)(p(x) \vee q(x))$ and
$\nvdash (\forall x)(p(x) \vee q(x)) \to (\forall x)p(x) \vee (\forall x)q(x)$
2. the semidistributivity of ' $\exists$ ' over ' $\wedge$ ':
$\vdash (\exists x)(p(x) \wedge q(x)) \to (\exists x)p(x) \wedge (\exists x)q(x)$ and
$\nvdash (\exists x)p(x) \wedge (\exists x)q(x) \to (\exists x)(p(x) \wedge q(x))$
3. $\vdash (\exists x)(p(x) \to q(x)) \leftrightarrow ((\forall x)p(x) \to (\exists x)q(x))$;
4. the distributivity of ' $\exists$ ' over ' $\vee$ ':
$\vdash (\exists x)(p(x) \vee q(x)) \leftrightarrow (\exists x)p(x) \vee (\exists x)q(x)$;
5. $\vdash (\exists x)p(x) \vee (\exists x)(p(x) \wedge q(x)) \leftrightarrow (\exists x)p(x)$;
6. the semidistributivity of ' $\forall$ ' over ' $\to$ ':
$\vdash (\forall x)(p(x) \to q(x)) \to ((\forall x)p(x) \to (\forall x)q(x))$ and
$\nvdash ((\forall x)p(x) \to (\forall x)q(x)) \to (\forall x)(p(x) \to q(x))$
7. $\vdash (\forall x)p(x) \wedge ((\forall x)p(x) \vee (\forall x)q(x)) \leftrightarrow (\forall x)p(x)$;
8. the distributivity of ' $\forall$ ' over ' $\wedge$ ':
$\vdash (\forall x)p(x) \wedge (\forall x)q(x) \leftrightarrow (\forall x)(p(x) \wedge q(x))$.

Exercise 5.9.
Check if the following formulas are theorems using lock resolution.
1. $U_1 = (\forall x)(\forall y)p(x,y) \leftrightarrow (\forall y)(\forall x)p(x,y)$;
2. $U_2 = (\exists y)(\exists x)p(x,y) \leftrightarrow (\exists x)(\exists y)p(x,y)$;
3. $U_3 = (\forall x)(\forall y)p(x,y) \leftrightarrow (\exists x)(\forall y)p(x,y)$;
4. $U_4 = (\exists x)(\forall y)p(x,y) \leftrightarrow (\forall y)(\exists x)p(x,y)$;
5. $U_5 = (\exists y)(\exists x)p(x,y) \leftrightarrow (\forall x)(\exists y)p(x,y)$;
6. $U_6 = (\forall y)(\forall x)p(x,y) \leftrightarrow (\forall x)(\exists y)p(x,y)$;
7. $U_7 = (\exists y)(\exists x)p(x,y) \leftrightarrow (\exists x)(\forall y)p(x,y)$;
8. $U_8 = (\forall y)(\exists x)p(x,y) \leftrightarrow (\exists y)(\exists x)p(x,y)$.

6. REASONING MODELING AND PROGRAM VERIFICATION

Propositional logic and predicate logic can be used to model simple types of human common-sense reasoning: decide if a statement (*conjecture, conclusion*) is derivable from a set of statements (*hypotheses*). The hypotheses and the conclusion are expressed in natural language. Since predicate logic allows reasoning about the objects of some universe and the relations among these objects, this formalism is also appropriate to model mathematical reasoning. The proof methods presented in the previous chapters are applied in numerous examples.

6.1. Common-sense reasoning modeling using propositional logic

Example 6.1. Party
Hypotheses:

H_1 : Mary will go to the party if Lucy will go and George will not go.

H_2 : If John will go to the party then Lucy will go too.

H_3 : If John is in town he will go to the party.

H_4 : George is sick and can't go to the party.

H_5 : Yesterday John has returned in town from Paris.

Conclusion:

C : Will Mary go to the party?

We have to check if the following deduction holds.

$H_1, H_2, H_3, H_4, H_5 \vdash C$

The following notations for the propositional variables are used:

M – Mary will go to the party

L – Lucy will go to the party

G – George will go to the party

J – John will go to the party

Jt – John is in town

The hypotheses and the conclusion are transformed into propositional formulas as follows:

$H_1 : L \wedge \neg G \rightarrow M$ $H_2 : J \rightarrow L$ $H_3 : Jt \rightarrow J$

$H_4 : \neg G$ $H_5 : Jt$ $C : M$

The definition of the deduction (Definition 1.8) and the axiomatic system are used:

$f_1 = H_1 : L \wedge \neg G \rightarrow M$ (hypothesis)

$f_2 = H_2 : J \rightarrow L$ (hypothesis)

$f_3 = H_3 : Jt \rightarrow J$ (hypothesis)

$f_4 = H_4 : \neg G$ (hypothesis)

$f_5 = H_5 : Jt$ (hypothesis)

$f_5, f_3 \vdash_{mp} J : f_6$

$f_6, f_3 \vdash_{mp} L : f_7$

$f_4, f_7 \vdash L \wedge \neg G : f_8$ (conjunction in conclusions)

$f_8, f_1 \vdash_{mp} M : f_9 = C$

Modus ponens inference rule was applied to derive the formulas f_6, f_7, f_9.

The sequence of formulas: $(f_1, f_2, f_3, f_4, f_5, f_6, f_7, f_8, f_9)$ is the deduction of C from the hypotheses, therefore, based on the hypotheses, *Mary will go to the party.*

Example 6.2.

Hypotheses:

H_1 : If it is sunny, Diane and Alice go to the swimming pool.

H_2 : Ben goes to the swimming pool on Thursdays.

H_3 : It was sunny last Thursday.

Conclusion:

C : Did Diane meet Ben at the swimming pool last Thursday?

We have to check if the following deduction holds:

$H_1, H_2, H_3 \vdash C$

The resolution, a refutation proof method, is applied and according to the Theorem of soundness and completeness of resolution, we have to check if:

$\text{CNF}(H_1 \wedge H_2 \wedge H_3 \wedge \neg C) \vdash_{\text{Res}} \square$.

We use the following notations for the propositional variables:

D – Diane goes to the swimming pool

A – Alice goes to the swimming pool

B – Ben goes to the swimming pool

S – it is sunny

Th – it is Thursday

The following propositional formulas and their conjunctive normal forms correspond to the hypotheses and the negation of the conclusion:

$$H_1 : S \to D \wedge A \equiv \neg S \vee (D \wedge A) \equiv (\neg S \vee D) \wedge (\neg S \vee A) : C_1 \wedge C_2$$

$$H_2 : Th \to B \equiv \neg Th \vee B : C_3$$

$$H_3 : S \wedge Th : C_4 \wedge C_5$$

$$C : Th \wedge D \wedge B$$

$$\neg C : \neg(Th \wedge D \wedge B) \equiv \neg Th \vee \neg D \vee \neg B : C_6$$

The hypotheses and the negation of the conclusion were transformed into CNFs and the set of clauses $S = \{C_1, C_2, C_3, C_4, C_5, C_6\}$ was obtained:.

If we want to apply the set-of-support strategy, we have to avoid resolving two clauses belonging to the consistent subset of clauses: $\{C_1, C_2, C_3, C_4, C_5\}$ provided by the hypotheses. The support set of S is $Y = \{C_6\}$ and corresponds to the negation of the conclusion. All the resolvents are added to Y.

The refutation from S (the derivation of $\square$ from S) is provided below:

$$C_7 = \text{Res}(C_5, C_6) = \neg D \vee \neg B$$

$$C_8 = \text{Res}(C_7, C_3) = \neg D \vee \neg Th$$

$$C_9 = \text{Res}(C_8, C_5) = \neg D$$

$$C_{10} = \text{Res}(C_9, C_1) = \neg S$$

$$C_{11} = \text{Res}(C_{10}, C_4) = \square$$

We have proved that $\text{CNF}(H_1 \wedge H_2 \wedge H_3 \wedge \neg C) \vdash_{\text{Res}} \square$, so C is deducible from the hypotheses, therefore: "*Diane met Ben at the swimming pool last Thursday*".

We shall prove the deduction $H_1, H_2, H_3 \vdash_{\text{Res}} C$ using another proof method, a direct method, the sequent calculus.

The up-side down binary tree corresponding to the reduction process of the initial sequent $H_1, H_2, H_3 \Rightarrow C$ is built.

The initial sequent was reduced to five basic sequents (overlined), therefore it is a true sequent and according to the theorem of soundness and completeness of this method we conclude that C is deducible from the hypotheses H_1, H_2, H_3.

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{\underline{D}, A, B, S, Th \Rightarrow \underline{D}}(\wedge_l)}{B, \underline{S}, Th \Rightarrow \underline{S}, D \qquad D \wedge A, B, S, Th \Rightarrow D}(\rightarrow_l)}}{S \rightarrow D \wedge A, \underline{B}, S, Th \Rightarrow \underline{B} \qquad S \rightarrow D \wedge A, B, S, Th \Rightarrow D}(\wedge_r)}}{S \rightarrow D \wedge A, S, \underline{Th} \Rightarrow \underline{Th}, D \wedge B \qquad S \rightarrow D \wedge A, B, S, Th \Rightarrow D \wedge B}(\rightarrow_l) \text{ for } Th \rightarrow B}}{S \rightarrow D \wedge A, Th \rightarrow B, S, Th \Rightarrow D \wedge B \qquad S \rightarrow D \wedge A, Th \rightarrow B, S, \underline{Th} \Rightarrow \underline{Th}}(\wedge_l, \wedge_r)$$

$$S \rightarrow D \wedge A, Th \rightarrow B, S \wedge Th \Rightarrow D \wedge B \wedge Th$$

Example 6.3.
A client describes the requirements of a software application:

R_1. If condition A is satisfied then condition B must also be satisfied.

R_2. If conditions B and C are satisfied, then D must also be satisfied.

R_3. If condition D is satisfied then condition A is not satisfied.

R_4. If condition C is satisfied then A must also be satisfied.

R_5. If A is satisfied then D or C are satisfied.

R_6. C is satisfied if neither B nor A are satisfied.

R_7. B is not satisfied if C is not satisfied.

Are these requirements simultaneously satisfiable?

In order to answer the question we have to check the consistency/inconsistency of $U = R_1 \wedge R_2 \wedge R_3 \wedge R_4 \wedge R_5 \wedge R_6 \wedge R_7$.

The statements corresponding to the requirements are transformed into propositional formulas and further into their conjunctive normal forms:

$R_1. A \to B \equiv \neg A \vee B:\ C_1$

$R_2. B \wedge C \to D \equiv \neg B \vee \neg C \vee D:\ C_2$

$R_3. D \to \neg A \equiv \neg D \vee \neg A:\ C_3$

$R_4. C \to A \equiv \neg C \vee A:\ C_4$

$R_5. A \to C \vee D \equiv \neg A \vee C \vee D:\ C_5$

$R_6. \neg A \wedge \neg B \to C \equiv A \vee B \vee C:\ C_6$

$R_7. \neg C \to \neg B \equiv C \vee \neg B:\ C_7$

$S = \{C_1, C_2, C_3, C_4, C_5, C_6, C_7\}$.

We have to check the consistency/inconsistency of the set S of clauses corresponding to the conjunction of the requirements. The general resolution procedure is applied:

$C_8 = \text{Res}(C_1, C_7) = C \vee \neg A$

$C_9 = \text{Res}(C_7, C_6) = C \vee A$

$C_{10} = \text{Res}(C_8, C_9) = C$

$C_{11} = \text{Res}(C_4, C_{10}) = A$

$C_{12} = \text{Res}(C_{11}, C_1) = B$

$C_{13} = \text{Res}(C_{11}, C_3) = \neg D$

$C_{14} = \text{Res}(C_{13}, C_2) = \neg B \vee \neg C$

$C_{15} = \text{Res}(C_{14}, C_{12}) = \neg C$

$C_{16} = \text{Res}(C_{10}, C_{15}) = \square$

The empty clause was derived from S, so S *is inconsistent and the requirements are contradictory.*

Resolution as a proof method is very efficient compared to the truth table method as we shall see in the following.

To check the consistency/inconsistency of the requirements $R_1 - R_7$ we have to build the truth table of the propositional formula representing the conjunction of all the requirements: $U = R_1 \wedge R_2 \wedge R_3 \wedge R_4 \wedge R_5 \wedge R_6 \wedge R_7$.

Each requirement has a conjunctive normal form composed of one clause (see the normalization process presented before), thus:

$U \equiv C_1 \wedge C_2 \wedge C_3 \wedge C_4 \wedge C_5 \wedge C_6 \wedge C_7$.

The formula U has 4 propositional variables: A, B, C, D, so its truth table has $2^4 = 16$ rows, corresponding to all 16 interpretations which assign in all the possible ways the truth values $\{T, F\}$ to the variables A, B, C, D.

	A	B	C	D	R_1	R_2	R_3	R_4	R_5	R_6	R_7	U
i_1	T	T	T	T	T	T	F	T	T	T	T	F
i_2	T	T	T	F	T	F	T	T	T	T	T	F
i_3	T	T	F	T	T	T	F	T	T	T	F	F
i_4	T	T	F	F	T	T	T	T	F	T	F	F
i_5	T	F	T	T	F	T	F	T	T	T	T	F
i_6	T	F	T	F	F	T	T	T	T	T	T	F
i_7	T	F	F	T	F	T	F	T	T	T	T	F
i_8	T	F	F	F	F	T	T	T	F	T	T	F
i_9	F	T	T	T	T	T	T	F	T	T	T	F
i_{10}	F	T	T	F	T	F	T	F	T	T	T	F
i_{11}	F	T	F	T	T	T	T	T	T	T	F	F
i_{12}	F	T	F	F	T	T	T	T	T	T	F	F
i_{13}	F	F	T	T	T	T	T	F	T	T	T	F
i_{14}	F	F	T	F	T	T	T	F	T	T	T	F
i_{15}	F	F	F	T	T	T	T	T	T	F	T	F
i_{16}	F	F	F	F	T	T	T	T	T	F	T	F

Each requirement is evaluated in all 16 interpretations and the column (truth table) of U is obtained as the conjunction of the columns of the requirements.
For each row (interpretation) there is at least one value F in the columns of $R_1 - R_7$, meaning that in each interpretation at least one requirement is evaluated as false, so U is evaluated as false in that interpretation.
The column of U contains only the truth value F, so U is evaluated as false in all 16 interpretations, therefore U is an inconsistent formula.

The conclusion is that *the requirements are contradictory, they cannot be satisfied simultaneously by the software application.*
Note that the truth table method is time consuming and it is not efficient at the implementation level.

Example 6.4.
Consider the following hypotheses:
H_1. Mary will go to London this summer if both her friends Kate and Susan go.
H_2. If Kate passes the English exam in May then she will go to London.
H_3. Kate was in hospital from April until July and she didn't take the English exam.
H_4. This summer Susan will go to London on a business trip.

and the conclusion:
C. Will Mary go to London this summer?

Using the semantic tableaux method check if the conclusion C is a logical consequence of the set of hypotheses: $\{H_1, H_2, H_3, H_4\}$.

We transform the hypotheses and the conclusion into propositional formulas:

$H_1 : K \wedge S \rightarrow M$

$H_2 : KE \rightarrow K$

$H_3 : \neg KE$

$H_4 : S$

$C : M$

where the notations for the propositional variables are as follows:

M – Mary will go to London

S – Susan will go to London

K – Kate will go to London

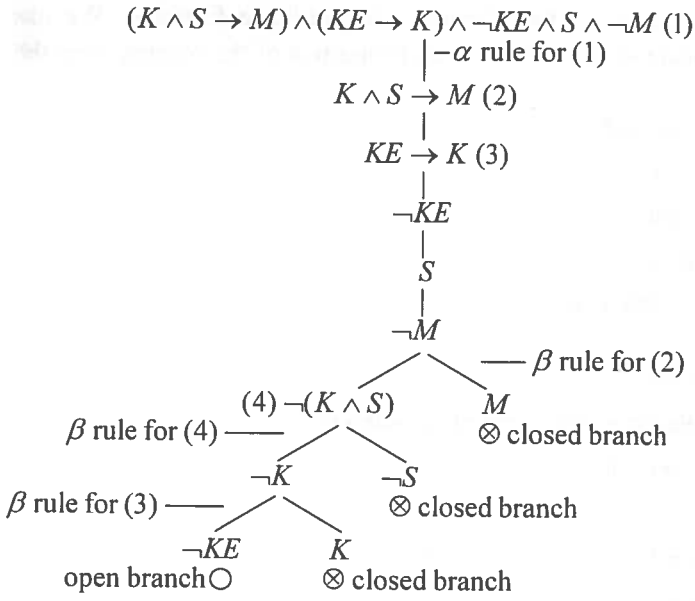
KE – Kate passed the English exam

The semantic tableau method is a refutation proof method, thus we have to negate the conclusion and use the Theorem of soundness and completeness:

$H_1, H_2, H_3, H_4 \vdash C$ *if and only if*

$H_1 \wedge H_2 \wedge H_3 \wedge H_4 \wedge \neg C$ has a closed semantic tableau.

The semantic tableau corresponding to the conjunction of the hypotheses and the negation of the conclusion is depicted below.



We have obtained a complete and open tableau with one open branch (the left-most one) and three closed branches containing the following pairs of opposite literals: $(\neg K, K), (S, \neg S), (M, \neg M)$.

Therefore $H_1, H_2, H_3, H_4 \not\models C$ and based on the hypotheses we conclude that *'Mary will not go to London this summer'*.

6.2. *Reasoning modeling using predicate logic*

Example 6.5.
Hypotheses:

H_1: All hummingbirds are richly colored.

H_2: No large birds live on honey.

H_3: Birds that do not live on honey are dull in color.

H_4: *Piky* is a hummingbird.

Conclusions:

D_1: All hummingbirds are small.

D_2: *Piky* is a small bird and lives on honey.

We have to check if the following deductions hold or not.

$H_1, H_2, H_3 \vdash D_1$ and $H_1, H_2, H_3, H_4 \vdash D_2$.

The natural language sentences are transformed into predicate formulas. We use the following unary predicate symbols to express properties of the objects from the universe of birds.

$hb(x)$ – x is a hummingbird,

$rc(x)$ – x is richly colored

$sb(x)$ – x is a small bird

$lh(x)$ – x lives on honey

Piky is a constant of the universe.

$H_1: (\forall x)(hb(x) \to rc(x))$

$H_2: \neg(\exists x)(\neg sb(x) \wedge lh(x)) \equiv (\forall x)(\neg sb(x) \to \neg lh(x))$

$H_3: (\forall x)(\neg lh(x) \to \neg rc(x))$

$H_4: hb(Piky)$

$D_1: (\forall x)(hb(x) \to sb(x))$

$D_2: sb(Piky) \wedge lh(Piky)$

We prove the deduction $H_1, H_2, H_3 \vdash D_1$ using the sequent calculus method, a direct and syntactic proof method.

The reduction process of the initial sequent $H_1, H_2, H_3 \Rightarrow D_1$ is depicted below:

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dots\dots}{(\neg r)\dfrac{rc(t), H_1, H_2, H_3, hb(t), \underline{lh(t)} \Rightarrow \underline{lh(t)}, sb(t)}{rc(t), H_1, H_2, H_3, hb(t) \Rightarrow \neg lh(t), lh(t), sb(t)}} \quad \dfrac{\dots\dots}{\underline{rc(t)}, H_1, H_2, H_3, hb(t) \Rightarrow \underline{rc(t)}, lh(t), sb(t)}(\neg l)}{rc(t), \neg rc(t), H_1, H_2, H_3, hb(t) \Rightarrow lh(t), sb(t)}(\rightarrow_l \text{ for } C)}{\underline{rc(t), C, H_1, H_2, H_3, hb(t), sb(t)} \Rightarrow \underline{sb(t)}} \quad \dfrac{rc(t), C, H_1, H_2, H_3, hb(t) \Rightarrow lh(t), sb(t)}{rc(t), \neg lh(t), C, H_1, H_2, H_3, hb(t) \Rightarrow sb(t)}(\neg l)}{(\neg r)} \quad \dfrac{\dots\dots}{\underline{B}, C, H_1, H_2, H_3, \underline{hb(t)} \Rightarrow \underline{hb(t)}, sb(t)} \quad \dfrac{rc(t), C, H_1, H_2, H_3, hb(t) \Rightarrow \neg sb(t), sb(t)}{rc(t), B, C, H_1, H_2, H_3, hb(t) \Rightarrow sb(t)}(\rightarrow_l \text{ for } B)}{(\rightarrow_l \text{ for } A)} \quad \dfrac{A, B, C, H_1, H_2, H_3, hb(t) \Rightarrow sb(t)}{(\rightarrow_r)}$$

$$\dfrac{hb(t) \rightarrow rc(t), \neg sb(t) \rightarrow \neg lh(t), \neg lh(t) \rightarrow \neg rc(t), H_1, H_2, H_3 \Rightarrow hb(t) \rightarrow sb(t)}{(\forall x)(hb(x) \rightarrow rc(x)), (\forall y)(\neg sb(y) \rightarrow \neg lh(y)), (\forall z)(\neg lh(z) \rightarrow \neg rc(z)) \Rightarrow hb(t) \rightarrow sb(t)}(\forall_l)[x \leftarrow t, y \leftarrow t, z \leftarrow t]$$

$$\dfrac{(\forall x)(hb(x) \rightarrow rc(x)), (\forall y)(\neg sb(y) \rightarrow \neg lh(y)), (\forall z)(\neg lh(z) \rightarrow \neg rc(z)) \Rightarrow hb(t) \rightarrow sb(t)}{(\forall x)(hb(x) \rightarrow rc(x)), (\forall y)(\neg sb(y) \rightarrow \neg lh(y)), (\forall z)(\neg lh(z) \rightarrow \neg rc(z)) \Rightarrow (\forall t)(hb(t) \rightarrow sb(t))}(\forall_r)$$

- the bound variables from the formulas H_1, H_2, H_3, D_1 were renamed such that they are distinct in the initial sequent:

  $H_1 : (\forall x)(hb(x) \rightarrow rc(x))$,

  $H_2 : (\forall y)(\neg sb(y) \rightarrow \neg lh(y))$,

  $H_3 : (\forall z)(\neg lh(z) \rightarrow \neg rc(z))$,

  $D_1 : (\forall t)(hb(t) \rightarrow sb(t))$

- in the first step the universal quantified variable t from the consequent becomes a free variable: the rule $(\forall_r)$ is applied;
- in the second step the rule $(\forall_l)$ is applied three times. All three universal quantified formulas H_1, H_2, H_3, from the antecedent, are instantiated using t and A, B, C open formulas are obtained. H_1, H_2, H_3 are duplicated (for further instantiations);

  $A = hb(t) \to rc(t)$,

  $B = \neg sb(t) \to \neg lh(t)$,

  $C = \neg lh(t) \to \neg rc(t)$.
- in the third step the rule $(\to_r)$ is applied;
- in the following steps the formulas A, B, C are decomposed using the rule $(\to_l)$;
- the complete reduction tree has four leaf nodes containing basic sequents.

The initial sequent $H_1, H_2, H_3 \Rightarrow D_1$ was reduced to four basic sequents, so it is a true sequent and the deduction $H_1, H_2, H_3 \vdash D_1$ holds.

Based on the hypotheses we conclude that:

'*All hummingbirds are small*'.

The deduction $H_1, H_2, H_3, H_4 \vdash D_2$ is proved by contradiction by applying the semantic tableaux method.

The semantic tableau of $H_1 \wedge H_2 \wedge H_3 \wedge H_4 \wedge \neg D_2$ is built.

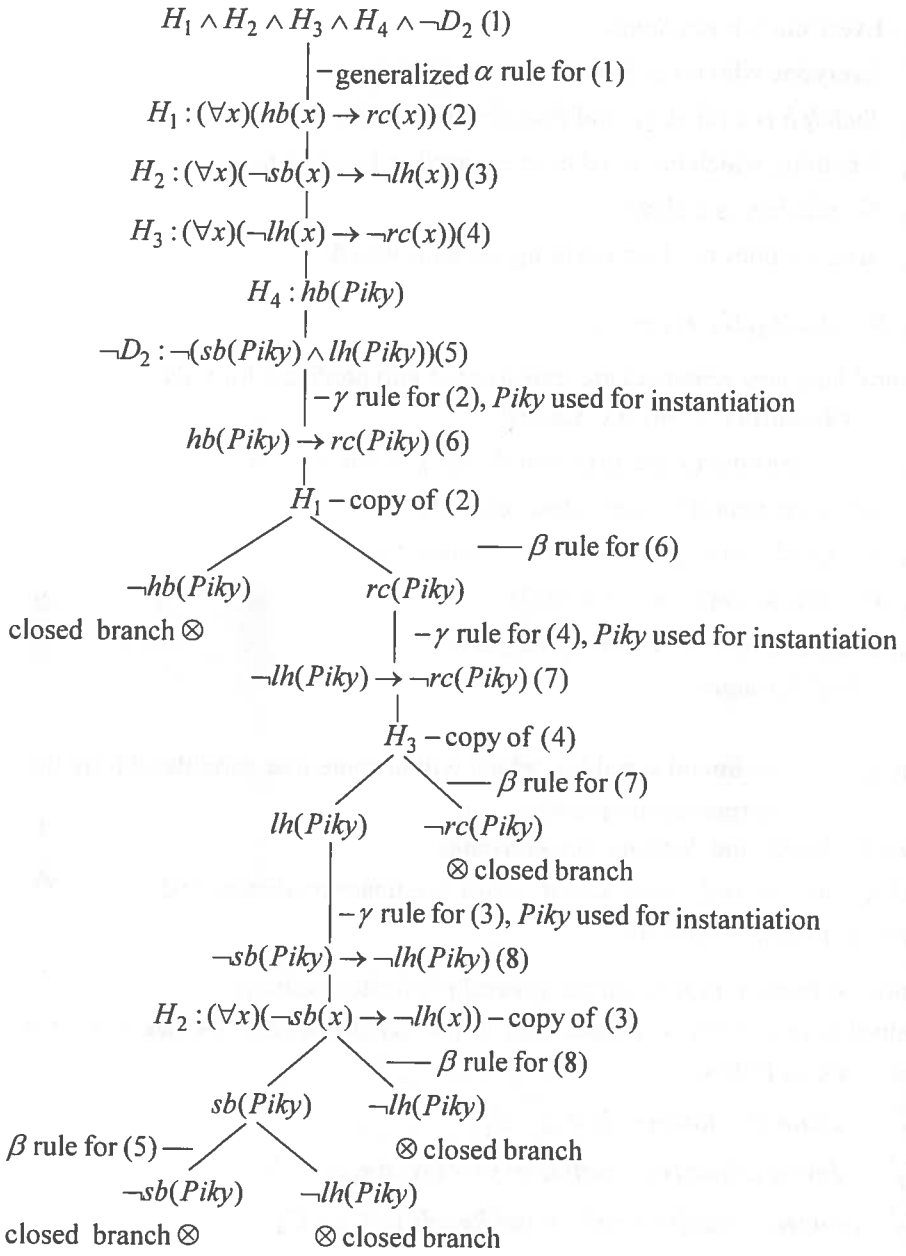
The copies of the universal quantified formulas H_1, H_2, H_3 are not used further because no new constants were introduced after their instantiations.

The semantic tableau is closed, having five closed branches. The set of the hypotheses is consistent, therefore there is a contradiction among the hypotheses and the negation of the conclusion.

According to the 'reduction ad absurdum' principle, the conclusion D_2 is derivable from the hypotheses H_1, H_2, H_3, H_4.

Based on the hypotheses we conclude that:

'*Piky is a small bird and lives on honey.*'



Example 6.6.

Consider the following set of hypotheses $\{H_1, H_2, H_3, H_4, H_5, H_6\}$ and check the validity of the conclusion (C):

C . *Scrooge* is not a child.

H_1. Every child loves *Santa*.

H_2. Everyone who loves *Santa* loves any reindeer.

H_3. *Rudolph* is a reindeer, and *Rudolph* has a red nose.

H_4. Anything which has a red nose is weird or is a clown.

H_5. No reindeer is a clown.

H_6. *Scrooge* does not love anything which is weird.

$H_1, H_2, H_3, H_4, H_5, H_6 \overset{?}{\vdash} C$.

The natural language sentences are transformed into predicate formulas:

$H_1 : (\forall x)(child(x) \rightarrow loves(x, Santa))$

$H_2 : (\forall x)(\forall y)(loves(x, Santa) \wedge reindeer(y) \rightarrow loves(x, y))$

$H_3 : reindeer(Rudolf) \wedge red\_nose(Rudolf)$

$H_4 : (\forall z)(red\_nose(z) \rightarrow weird(z) \vee clown(z))$

$H_5 : (\forall s)(reindeer(s) \rightarrow \neg clown(s))$

$H_6 : (\forall t)(weird(t) \rightarrow \neg loves(Scrooge, t))$

$C : \neg child(Scrooge)$

where:

x, u, y, z, s, t are bound variables, which will become free variables during the normalization process,

Rudolf, *Santa* and *Scrooge* are constants,

child, *reindeer*, *red_nose*, *weird*, *clown* are unary predicates and

loves is a binary predicate

We apply a refutation proof method: general predicate resolution.

The clausal normal forms corresponding to the hypotheses and the negation of the conclusion are as follows:

$H_1{}^C : \neg child(x) \vee loves(x, Santa) = C_1$

$H_2{}^C : \neg loves(x, Santa) \vee \neg reindeer(y) \vee loves(x, y) = C_2$

$H_3{}^C : reindeer(Rudolf) \wedge red\_nose(Rudolf) = C_3 \wedge C_4$

$H_4{}^C : \neg red\_nose(z) \vee weird(z) \vee clown(z) = C_5$

$H_5{}^C : \neg reindeer(s) \vee \neg clown(s) = C_6$

$H_6{}^C : \neg weird(t) \vee \neg loves(Scrooge, t) = C_7$

$(\neg C)^C : child(Scrooge) = C_8$

To the set of clauses $S=\{C_1,C_2,C_3,C_4,C_5,C_6,C_7,C_8\}$, general predicate resolution is applied.
In the resolution process, the following resolvents are obtained:

$C_9=\text{Res}^{Pr}_{[x\leftarrow Scrooge]}(C_8,C_1)=loves(Scrooge,Santa)$

$C_{10}=\text{Res}^{Pr}_{[s\leftarrow Rudolf]}(C_3,C_6)=\neg clown(Rudolf)$

$C_{11}=\text{Res}^{Pr}_{[z\leftarrow Rudolf]}(C_4,C_5)=weird(Rudolf)\vee clown(Rudolf)$

$C_{12}=\text{Res}^{Pr}(C_{10},C_{11})=weird(Rudolf)$

$C_{13}=\text{Res}^{Pr}_{[t\leftarrow Rudolf]}(C_{12},C_7)=\neg loves(Scrooge,Rudolf)$

$C_{14}=\text{Res}^{Pr}_{[y\leftarrow Rudolf]}(C_2,C_3)=\neg loves(x,Santa)\vee loves(x,Rudolf)$

$C_{15}=\text{Res}^{Pr}_{[x\leftarrow Scrooge]}(C_{13},C_{14})=\neg loves(Scrooge,Santa)$

$C_{16}=\text{Res}^{Pr}(C_9,C_{15})=\square$

The most general unifier generated during the resolution process is the substitution:
$[x\leftarrow Scrooge, y\leftarrow Rudolf, z\leftarrow Rudolf, s\leftarrow Rudolf, t\leftarrow Rudolf]$

$S\vdash^{Pr}_{Res}\square$, therefore S is an inconsistent set and the deduction
$H_1,H_2,H_3,H_4,H_5,H_6\vdash C$ holds.

From the hypotheses we conclude that '*Scrooge is not a child*'.

Example 6.7. Succession to the British throne
Hypotheses:

H_1: If x is the king and y is his oldest son, then y can become the king.

H_2: If x is the king and y defeats x, then y will become the king.

H_3: *RichardIII* is the king.

H_4: *HenryVII* defeated *RichardIII*.

H_5: *HenryVIII* is *HenryVII*'s oldest son.

Conclusion:

C: Can *HenryVIII* become the king?

Check if the conclusion C is derivable from the set of hypotheses $\{H_1,H_2,H_3,H_4,H_5\}$ using a syntactic proof method.

We transform the hypotheses and the conclusion into predicate formulas using:
variables: x,y,z,t
constants: *RichardIII*, *HenryVII*, *HenryVIII*
predicate symbols: **unary**: *king*, **binary**: *oldest_son*, *defeat*

$H_1 : (\forall x)(\forall y)(king(x) \wedge oldest\_son(y,x) \rightarrow king(y))$

$H_2 : (\forall z)(\forall t)(king(z) \wedge defeat(t,z) \rightarrow king(t))$

$H_3 : king(RichardIII)$

$H_4 : defeat(HenryVII, RichardIII)$

$H_5 : oldest\_son(HenryVIII, HenryVII)$

$C : king(HenryVIII)$

We prove the deduction $H_1, H_2, H_3, H_4, H_5 \vdash C$ using the axiomatic system of predicate logic and Definition 2.2.

The following sequence of predicate formulas: $(f_1, f_2, \ldots, f_{13})$ is generated.

The inference rules used in the deduction process are **universal instantiation**: *univ_inst* and **modus ponens**: *mp*.

$f_1 = H_1 : (\forall x)(\forall y)(king(x) \wedge oldest\_son(y,x) \rightarrow king(y))$

$f_2 = H_2 : (\forall z)(\forall t)(king(z) \wedge defeat(t,z) \rightarrow king(t))$

$f_3 = H_3 : king(RichardIII)$

$f_4 = H_4 : defeat(HenryVII, RichardIII)$

$f_5 = H_5 : oldest\_son(HenryVIII, HenryVII)$

$f_2 \vdash_{univ\_inst} (\forall t)(king(RichardIII) \wedge defeat(t, RichardIII) \rightarrow king(t)) : f_6$,

the universal variable z was instantiated using the constant *RichardIII*

$f_6 \vdash_{univ\_inst} king(RichardIII) \wedge defeat(HenryVII, RichardIII) \rightarrow king(HenryVII) : f_7$

the universal variable t was instantiated using the constant *HenryVII*

$f_3 \wedge f_4 = king(RichardIII) \wedge defeat(HenryVII, RichardIII) : f_8$ (conjunction in conclusions)

$f_8, f_7 \vdash_{mp} king(HenryVII) : f_9$

$f_1 \vdash_{univ\_inst} (\forall y)\ (king(HenryVII) \wedge oldest\_son(y, HenryVII) \rightarrow king(y)) : f_{10}$,

the universal variable x was instantiated using the constant *HenryVII*

$f_{10} \vdash_{univ\_inst} king(HenryVII) \wedge oldest\_son(HenryVIII, HenryVII) \rightarrow king(HenryVIII) : f_{11}$

the universal variable y was instantiated using the constant *HenryVIII*

$f_5 \wedge f_9 = king(HenryVII) \wedge oldest\_son(HenryVIII, HenryVII) : f_{12}$ (conjunction in conclusions)

$f_{10}, f_{12} \vdash_{mp} king(HenryVIII) : f_{13}$=**C**

The sequence of formulas $(f_1, f_2, \ldots, f_{13})$ is the deduction of C from the hypotheses H_1, H_2, H_3, H_4, H_5, therefore based on the hypotheses we conclude that '*HenryVIII can become the king*'.

Example 6.8.
Hypotheses:

H_1 : Any Computer Science student likes *logic* and likes any programming language.

H_2 : Someone who likes *logic* is a Computer Science student or a Philosophy student.

H_3 : *Java* is a programming language.

H_4 : *John* doesn't like *Java* but he likes *logic*.

Conclusion:

C: John is a Philosophy student but he is not a Computer Science student.

The hypotheses and the conclusion are translated into first-order language. Symbols used:

- x, y, z are variables;
- *logic*, *Java*, *John* are constants,
- CS and P are unary predicate symbols with the meanings:
  $CS(x)$:' x is a Computer Science student'
  $P(x)$:' x is a Philosophy student'
- pl is a unary predicate, $pl(x)$: ' x is a programming language'
- *likes* is a binary predicate, $likes(x, y)$: ' x likes y '

$H_1 : (\forall x)(\forall y)(CS(x) \wedge pl(y) \rightarrow likes(x, logic) \wedge likes(x, y))$

$H_2 : (\forall z)(likes(z, logic) \rightarrow CS(z) \vee P(z))$

$H_3 : pl(Java)$

$H_4 : likes(John, \text{logic}) \wedge \neg likes(John, Java)$

$C : \neg CS(John) \wedge P(John)$

We transform the hypotheses and the negation of the conclusion into clausal normal forms:

$(H_1)^c = \neg CS(x) \vee \neg pl(y) \vee likes(x, logic) \wedge likes(x, y) \equiv$

$\equiv (\neg CS(x) \vee \neg pl(y) \vee likes(x, logic)) \wedge (\neg CS(x) \vee \neg pl(y) \vee likes(x, y)) = C_1 \wedge C_2$

$C_1 = \neg CS(x) \vee \neg pl(y) \vee likes(x, logic), \quad C_2 = \neg CS(x) \vee \neg pl(y) \vee likes(x, y)$

$(H_2)^c = \neg likes(z, logic) \vee CS(z) \vee P(z) = C_3$

$(H_3)^c = pl(Java) = C_4$

$(H_4)^c = likes(John, logic) \wedge \neg likes(John, Java) = C_5 \wedge C_6$

$C_5 = likes(John, logic), \quad C_6 = \neg likes(John, Java)$

$(\neg C)^c = CS(John) \vee \neg P(John) = C_7$

We apply linear resolution to the set $S=\{C_1, C_2, C_3, C_4, C_5, C_6, C_7\}$ of clauses, with C_3 as the top clause.

$C_8 = \text{Res}^{Pr}_{[z \leftarrow John]}(C_3, C_5) = CS(John) \vee P(John)$

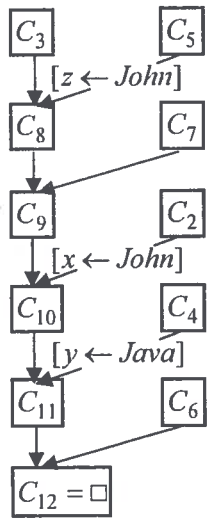
$C_9 = \text{Res}(C_8, C_7) = CS(John)$

$C_{10} = \text{Res}^{Pr}_{[x \leftarrow John]}(C_9, C_2) = \neg pl(y) \vee likes(John, y)$

$C_{11} = \text{Res}^{Pr}_{[y \leftarrow Java]}(C_{10}, C_4) = likes(John, Java)$

$C_{12} = \text{Res}(C_{11}, C_6) = \square$

The linear refutation process is represented graphically as follows:



$S \vdash^{lin}_{\text{Res}} \square$, therefore S is an inconsistent set and based on the hypotheses we conclude that:

'*John is a Philosophy student but he is not a Computer Science student*'.

Example 6.9. Reasoning modeling in geometry using predicate logic

The domain is the set of all the lines in a plane. We use variables: *x*, *y*, *z* to denote arbitrary objects (lines) and constants: d, d_1, d_2 to denote constant objects (lines).

Hypotheses:

H_1: If x is perpendicular to y then x intersects y.

H_2: If x is parallel to y then x doesn't intersect y.

H_3: If x is perpendicular to y and z is perpendicular to y then x is parallel to z.

H_4: d_1 is perpendicular to d.

H_5: d is perpendicular to d_2.

Conclusion.

C: d_2 does not intersect d_1.

Check if the conclusion C is derivable from the set of hypotheses $\{H_1, H_2, H_3, H_4, H_5\}$ using a syntactic proof method.

In order to translate the hypotheses and the conclusion into first-order formulas we use the binary predicate symbols with the same names as the corresponding geometric relations. Distinct names for the bound variables in the first-order formulas are used.

$H_1 : (\forall x_1)(\forall x_2)(perpendicular(x_1, x_2) \rightarrow intersects(x_1, x_2))$

$H_2 : (\forall x_3)(\forall x_4)(parallel(x_3, x_4) \rightarrow \neg intersects(x_3, x_4))$

$H_3 : (\forall x_5)(\forall x_6)(\forall x_7)(perpendicular(x_5, x_6) \wedge perpendicular(x_7, x_6) \rightarrow parallel(x_5, x_7))$

$H_4 : perpendicular(d_1, d)$

$H_5 : perpendicular(d, d_2)$

$C : \neg intersects(d_2, d_1)$

We have to add predicate formulas which express the properties of ***symmetry*** for the geometric relations: *parallel*, *perpendicular*, *intersects*.

$P_1 : (\forall x_8)(\forall x_9)(parallel(x_8, x_9) \rightarrow parallel(x_9, x_8))$

$P_2 : (\forall x_{10})(\forall x_{11})(perpendicular(x_{10}, x_{11}) \rightarrow perpendicular(x_{11}, x_{10}))$

$P_3 : (\forall x_{12})(\forall x_{13})(intersects(x_{12}, x_{13}) \rightarrow intersects(x_{13}, x_{12}))$

The properties of ***reflexivity*** for *parallel* and *intersects*, and ***transitivity*** for *parallel* must be also added:

$P_4 : (\forall x_{14}) parallel(x_{14}, x_{14})$

$P_5 : (\forall x_{15}) intersects(x_{15}, x_{15})$

$P_6 : (\forall x_{16})(\forall x_{17})(\forall x_{17})(parallel(x_{16}, x_{17}) \wedge parallel(x_{17}, x_{18}) \rightarrow parallel(x_{16}, x_{18}))$

The clausal normal forms of the hypotheses: H_1, H_2, H_3, H_4, H_5, the negation of the conclusion C and the properties: $P_1, P_2, P_3, P_4, P_5, P_6$ are as follows:

$H_1^C : \neg perpendicular(x_1, x_2) \vee intersects(x_1, x_2) : C_1$

$H_2^C : \neg parallel(x_3, x_4) \vee \neg intersects(x_3, x_4) : C_2$

$H_3^C : \neg perpendicular(x_5, x_6) \vee \neg perpendicular(x_7, x_6) \vee parallel(x_5, x_7) : C_3$

$H_4^C : perpendicular(d_1, d) : C_4$

$H_5{}^C : perpendicular(d, d_2) : C_5$

$(\neg C)^C : intersects(d_2, d_1) : C_6$

$P_1{}^C : \neg parallel(x_8, x_9) \vee parallel(x_9, x_8) : C_7$

$P_2{}^C : \neg perpendicular(x_{10}, x_{11}) \vee perpendicular(x_{11}, x_{10}) : C_8$

$P_3{}^C : \neg intersects(x_{12}, x_{13}) \vee intersects(x_{13}, x_{12}) : C_9$

$P_4{}^C : parallel(x_{14}, x_{14}) : C_{10}$

$P_5{}^C : intersects(x_{15}, x_{15}) : C_{11}$

$P_6{}^C : \neg parallel(x_{16}, x_{17}) \vee \neg parallel(x_{17}, x_{18}) \vee parallel(x_{16}, x_{18}) : C_{12}$

We apply general resolution to the set of clauses: $S=\{C_1, C_2, \ldots, C_{12}\}$.
The following resolvents are obtained during the resolution process:

$C_{13} = \text{Res}^{Pr}_{[x_{10} \leftarrow d, x_{11} \leftarrow d_2]}(C_5, C_8) = perpendicular(d_2, d)$

$C_{14} = \text{Res}^{Pr}_{[x_5 \leftarrow d_1, x_6 \leftarrow d]}(C_3, C_4) = \neg perpendicular(x_7, d) \vee parallel(d_1, x_7)$

$C_{15} = \text{Res}^{Pr}_{[x_7 \leftarrow d_2]}(C_{13}, C_{14}) = parallel(d_1, d_2)$

$C_{16} = \text{Res}^{Pr}_{[x_3 \leftarrow d_1, x_4 \leftarrow d_2]}(C_{15}, C_2) = \neg intersects(d_1, d_2)$

$C_{17} = \text{Res}^{Pr}_{[x_{13} \leftarrow d_1, x_{12} \leftarrow d_2]}(C_{16}, C_9) = \neg intersects(d_2, d_1)$

$C_{18} = \text{Res}(C_{17}, C_6) = \square$

The empty clause was derived from the set S of clauses, so S is inconsistent and the deduction $H_1, H_2, H_3, H_4, H_5 \vdash C$ holds.
The conclusion 'd_2 *does not intersect* d_1' is valid, based on the validity of the hypotheses.

Example 6.10. Mathematical reasoning modeling in algebra.
Prove: "**If every element of a group G is its own inverse, then G is an Abelian group**".

We will introduce the axioms, H_1, H_2 which define the group, the hypothesis, H_3: "every element of the group is its own inverse", and the conclusion C: "the group is an Abelian group".

Mathematical language:

$H_1 : (\forall x)(\forall y)(\forall z)[(x * y) * z = x * (y * z)]$ - associativity

$H_2 : (\forall x)[x * e = e * x = x]$ - e - neutral element

$H_3 : (\forall x)[x * x = e]$ - every element is its own inverse

$C : (\forall x)(\forall y)[x * y = y * x]$ - conclusion: G is an Abelian group

First-order logic language:

We use the ternary predicate symbol P, with the meaning: $P(x,y,z)$:"$x * y = z$". 'e' is a constant.

The formulas U_1 and U_2 correspond to H_1:

$U_1 : (\forall x)(\forall y)(\forall z)(\forall u)(\forall v)(\forall w)[P(x,y,u) \wedge P(u,z,w) \wedge P(y,z,v) \rightarrow P(x,v,w)]$

$U_2 : (\forall x)(\forall y)(\forall z)(\forall u)(\forall v)(\forall w)[P(y,z,v) \wedge P(x,v,w) \wedge P(x,y,u) \rightarrow P(u,z,w)]$

$U_1{}^C : \neg P(x,y,u) \vee \neg P(u,z,w) \vee \neg P(y,z,v) \vee P(x,v,w) = C_1$

$U_2{}^C : \neg P(y,z,v) \vee \neg P(x,v,w) \vee \neg P(x,y,u) \vee P(u,z,w) = C_2$

The formulas U_3 and U_4 correspond to H_2:

$U_3 : (\forall x)P(x,e,x)$, $\qquad U_3{}^C : P(s,e,s) = C_3$

$U_4 : (\forall x)P(e,x,x)$, $\qquad U_4{}^C : P(e,r,r) = C_4$

The formula U_5 corresponds to H_3:

$U_5 : (\forall x)P(x,x,e)$, $\qquad U_5{}^C : P(l,l,e) = C_5$

The formula U_6 corresponds to the conclusion:

$U_6 : (\forall x)(\forall y)(\exists t)(P(x,y,t) \rightarrow P(y,x,t))$

$\neg U_6 : \neg((\forall x)(\forall y)(\exists t)(P(x,y,t) \rightarrow P(y,x,t))) \equiv$

$\equiv (\exists x)(\exists y)(\forall t)(P(x,y,t) \wedge \neg P(y,x,t))$

$(\neg U_6)^C : P(a,b,t) \wedge \neg P(b,a,t) = C_6 \wedge C_7$ a,b – Skolem constants

In the clauses we ranamed some of the free variables.

Checking whether $H_1, H_2, H_3 \vdash C$ was reduced to checking the inconsistency of the set of clauses: $S = \{C_1, C_2, C_3, C_4, C_5, C_6, C_7\}$.

The refutation from S is presented below. At each application of the resolution rule, the literals resolved upon from the parent clauses are underlined.

$C_1 = \underline{\neg P(x,y,u)} \vee \neg P(u,z,w) \vee \neg P(y,z,v) \vee P(x,v,w)$, $\qquad C_5 = \underline{P(l,l,e)}$

$\theta_1 = [x \leftarrow l, y \leftarrow l, u \leftarrow e] = mgu(P(x,y,u), P(l,l,e))$

- $C_8 = \mathrm{Res}^{\mathrm{Pr}}_{\theta_1}(C_1, C_5) = \neg P(e,z,w) \vee \neg P(l,z,v) \vee P(l,v,w)$

$C_2 = \neg P(y,z,v) \vee \neg P(x,v,w) \vee \underline{\neg P(x,y,u)} \vee P(u,z,w)$, $\qquad C_6 = \underline{P(a,b,t)}$

$\theta_2 = [x \leftarrow a, y \leftarrow b, u \leftarrow t] = mgu(P(a,b,t), P(x,y,u))$

- $C_9 = \mathrm{Res}^{\mathrm{Pr}}_{\theta_2}(C_2, C_6) = \neg P(b,z,v) \vee \neg P(a,v,w) \vee P(t,z,w)$

$C_4 = \underline{P(e,r,r)}$, $\quad C_8 = \underline{\neg P(e,z,w)} \vee \neg P(l,z,v) \vee P(l,v,w)$

$\theta_3 = [z \leftarrow r, w \leftarrow r] = mgu(P(e,z,w), P(e,r,r))$

- $C_{10} = \text{Res}^{\text{Pr}}_{\theta_3}(C_4, C_8) = \neg P(l,r,v) \vee P(l,v,r)$

  $C_5 = \underline{P(l,l,e)}$, $\quad C_9 = \underline{\neg P(b,z,v)} \vee \neg P(a,v,w) \vee P(t,z,w)$

  $\theta_4 = [l \leftarrow b, z \leftarrow b, v \leftarrow e] = mgu(P(l,l,e), P(b,z,v))$

- $C_{11} = \text{Res}^{\text{Pr}}_{\theta_4}(C_5, C_9) = \neg P(a,e,w) \vee P(t,b,w)$

  $C_3 = \underline{P(s,e,s)}$, $\quad C_{11} = \underline{\neg P(a,e,w)} \vee P(t,b,w)$

  $\theta_5 = [s \leftarrow a, w \leftarrow a] = mgu(P(s,e,s), P(a,e,w))$

- $C_{12} = \text{Res}^{\text{Pr}}_{\theta_5}(C_3, C_{11}) = P(t,b,a)$,

  $C_{10} = \underline{\neg P(l,r,v)} \vee P(l,v,r)$, $\quad C_{12} = \underline{P(t,b,a)}$

  $\theta_6 = [l \leftarrow t, r \leftarrow b, v \leftarrow a] = mgu(P(t,b,a), P(l,r,v))$

- $C_{13} = \text{Res}^{\text{Pr}}_{\theta_6}(C_{10}, C_{12}) = P(t,a,b)$

  $C_2 = \underline{\neg P(y,z,v)} \vee \neg P(x,v,w) \vee \neg P(x,y,u) \vee P(u,z,w)$, $\quad C_5 = \underline{P(l,l,e)}$

  $\theta_7 = [y \leftarrow l, z \leftarrow l, v \leftarrow e] = mgu(P(l,l,e), P(y,z,v))$

- $C_{14} = \text{Res}^{\text{Pr}}_{\theta_7}(C_2, C_5) = \neg P(x,e,w) \vee \neg P(x,l,u) \vee P(u,l,w)$

  $C_3 = \underline{P(s,e,s)}$, $\quad C_{14} = \underline{\neg P(x,e,w)} \vee \neg P(x,l,u) \vee P(u,l,w)$

  $\theta_8 = [x \leftarrow s, w \leftarrow s] = mgu(P(s,e,s), P(x,e,w))$

- $C_{15} = \text{Res}^{\text{Pr}}_{\theta_8}(C_3, C_{14}) = \neg P(s,l,u) \vee P(u,l,s)$

  $C_{13} = \underline{P(t,a,b)}$, $\quad C_{15} = \underline{\neg P(s,l,u)} \vee P(u,l,s)$

  $\theta_9 = [s \leftarrow t, l \leftarrow a, u \leftarrow b] = mgu(P(t,a,b), P(s,l,u))$

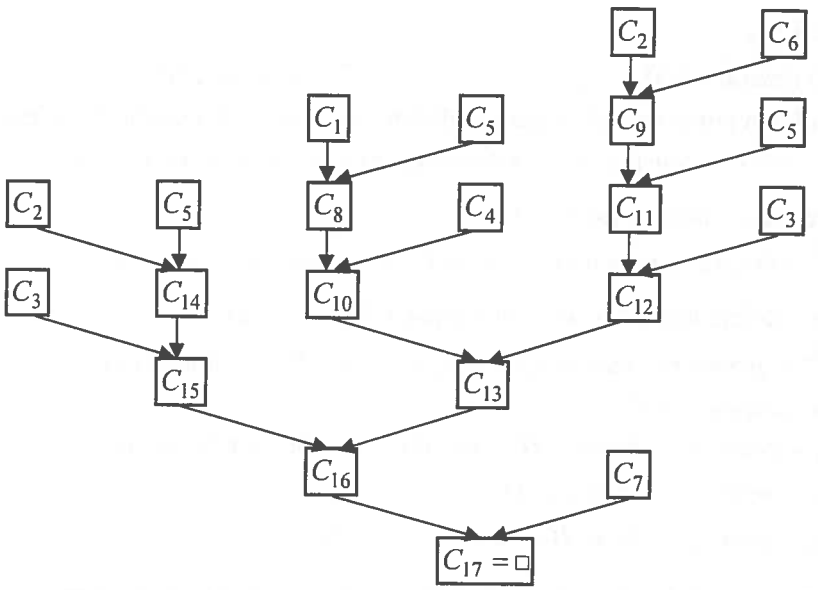
- $C_{16} = \text{Res}^{\text{Pr}}_{\theta_9}(C_{13}, C_{15}) = P(b,a,t)$

  $C_7 = \underline{\neg P(b,a,t)}$, $\quad C_{16} = \underline{P(b,a,t)}$

- $C_{17} = \text{Res}^{\text{Pr}}(C_{16}, C_7) = \square$, so the set S is inconsistent and $U_1, U_2, U_3, U_4, U_5 \vdash U_6$.

We conclude that the statement *"If every element of a group G is its own inverse, then G is an Abelian group"* is valid.

The resolution process is represented graphically by the following binary tree.



Example 6.11. Formalization of mathematical reasoning
Prove the statement: "*There exists an infinite number of primes*".

We do not have hypotheses, we need mathematical axioms in order to prove the statement. The axioms in mathematical language, predicate logic language and their clausal forms are provided below.

$a_1 : x \not< x$;

$(\forall x)\neg less(x,x)$, $\quad C_1 = \neg less(x,x)$

a_2 : if $x < y$ then $y \not< x$

$(\forall x)(\forall y)(less(x,y) \rightarrow \neg less(y,x))$, $\quad C_2 = \neg less(x,y) \vee \neg less(y,x)$

$a_3 : x$ divides x

$(\forall x) divides(x,x)$, $\quad C_3 = divides(x,x)$

a_4 : if x divides y and y divides z, then x divides z

$(\forall x)(\forall y)(\forall z)(divides(x,y) \wedge divides(y,z) \rightarrow divides(x,z))$,

$C_4 = \neg divides(x,y) \vee \neg divides(y,z) \vee divides(x,z))$

a_4 : if $x < y$ then y does not divide x

$(\forall x)(\forall y)(less(x,y) \rightarrow \neg divides(y,x))$, $\quad C_5 = \neg less(x,y) \vee \neg divides(y,x)$

a_6 : if y divides $F(x)$=x!+1, then $x < y$

$(\forall x)(\forall y)(divides(y,F(x)) \rightarrow less(x,y))$ $\quad C_6 = \neg divides(y,F(x)) \vee less(x,y)$

$a_7 : x < F(x)$

$(\forall x) less(x, F(x))$, $\qquad C_7 = less(x, F(x))$

a_8 : if x is not prime, then there is a y such that y divides x, y is prime and y is less than x.

$U = (\forall x)(\neg prime(x) \rightarrow (\exists y)(divides(y, x) \wedge prime(y) \wedge less(y, x)))$

The prenex normal form of U is:

$U^P = (\forall x)(\exists y)(prime(x) \vee (divides(y, x) \wedge prime(y) \wedge less(y, x)))$

The Skolem normal form without quantifiers of U is:

$U^{Sq} = prime(x) \vee (divides(H(x), x) \wedge prime(H(x)) \wedge less(H(x), x)))$

The clauses provided:

$C_8 = prime(x) \vee divides(H(x), x)$, $H(x)$ is a Skolem function

$C_9 = prime(x) \vee prime(H(x))$

$C_{10} = prime(x) \vee less(H(x), x))$

The conclusion: "If x is a prime then there is a y such that y is a prime, x is less than y and $F(x)$ is not less than y."

$C = (\forall x)(prime(x) \rightarrow (\exists y)(prime(y) \wedge less(x, y) \wedge \neg less(F(x), y)))$

$\neg C = \neg(\forall x)(prime(x) \rightarrow (\exists y)(prime(y) \wedge less(x, y) \wedge \neg less(F(x), y))) \equiv$

$\equiv (\exists x)(prime(x) \wedge (\forall y)(\neg prime(y) \vee \neg less(x, y) \vee less(F(x), y))) \equiv$

$\equiv (\exists x)(\forall y)(prime(x) \wedge (\neg prime(y) \vee \neg less(x, y) \vee less(F(x), y)))$

– prenex normal form

The clausal normal form of $\neg C$ is:

$prime(a) \wedge (\neg prime(y) \vee \neg less(a, y) \vee less(F(a), y))$ providing the clauses:

$C_{11} = prime(a)$,

$C_{12} = \neg prime(y) \vee \neg less(a, y) \vee less(F(a), y)$

We want to prove that the conclusion is derivable from the set of axioms: $a_1, a_2, a_3, a_4, a_5, a_6, a_7, a_8$.

Using resolution we prove that $S=\{C_1, C_2, \ldots, C_{12}\}$ is an inconsistent set.

$C_{12} = \neg less(a, y) \vee less(F(a), y) \vee \underline{\neg prime(y)}$,

$C_9 = \underline{prime(x)} \vee prime(H(x))$,

- $C_{13} = \mathrm{Res}^{\mathrm{Pr}}_{[y \leftarrow x]}(C_9, C_{12}) = \neg less(a, x) \vee less(F(a), x) \vee [\neg prime(x)] \vee prime(H(x))$

$C_8 = \underline{prime(x)} \vee divides(H(x), x)$,

$C_{12} = \underline{\neg prime(y)} \vee \neg less(a, y) \vee less(F(a), y))$

- $C_{14} = \mathrm{Res}^{\mathrm{Pr}}_{[y \leftarrow x]}(C_8, C_{12}) = divides(H(x), x) \vee \neg less(a, x) \vee less(F(a), x)$

$C_{13} = \underline{prime(H(x))} \vee \neg less(a,x) \vee less(F(a),x)$

$C_{12} = \underline{\neg prime(y)} \vee \neg less(a,y) \vee less(F(a),y)$

- $C_{15} = \mathrm{Res}^{Pr}_{[y \leftarrow H(x)]}(C_{13}, C_{12}) = \neg less(a,x) \vee \underline{\neg less(a,H(x))} \vee less(F(a),x) \vee less(F(a),H(x))$

$C_6 = \neg divides(u,F(v)) \vee \underline{less(v,u)}$

$C_{16} = \mathrm{Res}^{Pr}_{[v \leftarrow a, u \leftarrow H(x)]}(C_6, C_{15}) = \neg less(a,x) \vee less(F(a),x) \vee \underline{less(F(a),H(x))} \vee \neg divides(H(x),F(a))$

$C_5 = \underline{\neg less(u,v)} \vee \neg divides(v,u)$

- $C_{17} = \mathrm{Res}^{Pr}_{[u \leftarrow F(a), v \leftarrow H(x)]}(C_5, C_{16}) = \neg less(a,x) \vee less(F(a),x) \vee \underline{\neg divides(H(x),F(a))}$

$C_{14} = \underline{divides(H(x),x)} \vee \neg less(a,x) \vee less(F(a),x)$

- $C_{18} = \mathrm{Res}^{Pr}_{[x \leftarrow a]}(C_{17}, C_{14}) = \underline{\neg less(a,F(a))} \vee less(F(a),F(a))$

$C_7 = \underline{less(x,F(x))}$

- $C_{19} = \mathrm{Res}(C_1, C_{18}) = less(F(a),F(a))$

$C_1 = \neg less(x,x)$

- $C_{20} = \mathrm{Res}^{Pr}_{[x \leftarrow F(a)]}(C_1, C_{19}) = \square$

The empty clause was derived from the set of clauses $S=\{C_1, C_5, C_6, C_7, C_8, C_9, C_{12}\}$, so there is a contradiction among the formulas $\{U_1, U_2, \ldots, U_8, \neg C\}$.

By applying the 'reductio ad absurdum' principle we conclude that C is derivable from the set of axioms and the statement:

"There exists an infinite number of primes" is true.

6.3. *Program verification using predicate logic*

We present an approach of the program verification task introduced in paper [62]. A program is described by a set of predicate formulas and then using the resolution method, the execution of the program is modeled.

Let P be a ***program***: an algorithm described using a chart flow, pseudocode or a programming language.

For a program we have to solve these problems:

- the ***stop problem***: having an input data $\bar{x}$, will the program stop?
- the ***answer problem***: given the input data $\bar{x}$ which is the output data $\bar{z}$ of a program?
- the ***correctness problem***: if a program stops, will the relations input data – output data be satisfied?
- the ***equivalence problem***: having two programs, will they provide the same output data for the same input data?

A program uses three data vectors denoted by:

- $\bar{x} = (x_1,...,x_n)$ - the input vector (the variables are used only on the right side of an assignment)
- $\bar{y} = (y_1,...,x_k)$ - the work vector
- $\bar{z} = (z_1,...,z_m)$ - the output vector (the variables are used only on the left side of an assignment)

Definition 6.1.
To a program P we associate a graph $G = (X,U)$ as follows:

1. there is a unique initial vertex (without incoming arcs) called ***Start***, denoted by S;
2. there is a unique final vertex (without outgoing arcs) called ***Halt***, denoted by H;
3. each arc a that is not incident with H has associated a formula $P_a(\bar{x},\bar{y})$ and an assignment $\bar{y} = f_a(\bar{x},\bar{y})$
4. each arc a that is incident with H has associated a formula $P_a(\bar{x},\bar{y})$ and an assignment $\bar{z} = f_a(\bar{x},\bar{y})$
5. if a_1 and a_2 are two outgoing arcs from the same vertex $v \neq H$, only one of the formulas $P_{a_1}(\bar{x},\bar{y})$ and $P_{a_2}(\bar{x},\bar{y})$ is true.

Note that for a given input data $\bar{x}$, the vector $\bar{y}$ is well defined and the formulas $P_a(\bar{x},\bar{y})$ can be evaluated as true or false for each arc a.

Definition 6.2.
The ***condition for accessing the node (vertex)*** v_i from S is expressed by the predicate $q_i(\bar{x},\bar{y})$ which must be true.

Note: $q_S(\bar{x},\bar{y}) = T, \forall \bar{x}, \bar{y}$.

Definition 6.3.
The ***stop condition*** is the condition for accessing H from S and it is expressed by the predicate $q_H(\bar{x},\bar{z})$, which must be true.

Definition 6.4.
The formula W_a is called the ***description formula for the arc*** $a=(v_i,v_j)$ and has the form: $W_a : q_i(\bar{x},\bar{y}) \wedge P_a(\bar{x},\bar{y}) \rightarrow q_j(\bar{x}, f_a(\bar{x},\bar{y}))$.

Note that W_a must be true if the arc a is traversed. This formula is in fact a clause: $W_a \equiv \neg q_i(\bar{x},\bar{y}) \vee \neg P_a(\bar{x},\bar{y}) \vee q_j(\bar{x}, f_a(\bar{x},\bar{y}))$.

Definition 6.5.
Let $a_1,...,a_r$ be all the arcs of the graph associated to the program P. The formula $U_P = W_{a_1} \wedge ... \vee W_{a_r}$ is called the ***description formula*** of P. We will also identify U_P with the set of all description clauses of P.

Theorem 6.1.
Let P be a program and U_P the set of all description clauses of P. U_P is a consistent set of clauses.

A program is characterized not only by U_P but also by U_I (a formula that defines the properties of input data) and U_r (the set of formulas representing the axioms for the predicates $P_a(\bar{x},\bar{y})$ and the functions $f_a(\bar{x},\bar{y})$).

Definition 6.6.
A program P ***is convergent*** **for an input data** $\bar{x}$ if there is a path from S to H using $\bar{x}$. ***A program*** P ***is convergent*** if for all $\bar{x}$ there is a path from S to H.

Theorem 6.2.
Let P be a program and U_T a set of formulas obtained from U_P by eliminating all the literals containing the predicate $q_H(\bar{x},\bar{z})$. P is convergent if and only if the set $U_T \cup U_I \cup U_r$ is inconsistent.

Definition 6.7.
A clause containing only the literal $q_H(\bar{x},\bar{z})$ is called a ***halt clause***.

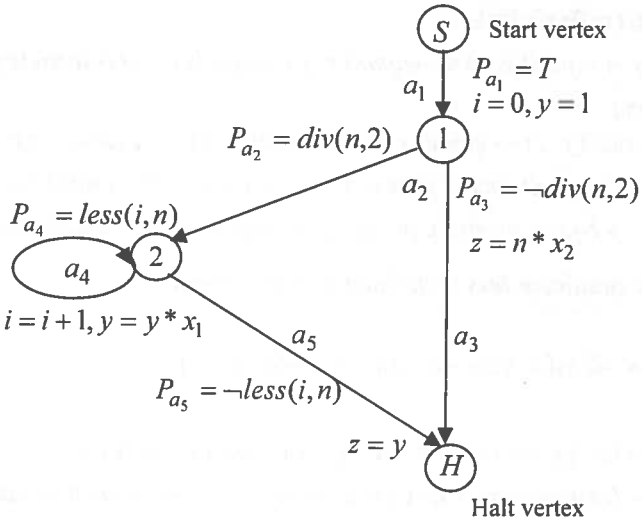
Theorem 6.3.
Let P be a program and X the set of clauses corresponding to $U_P \cup U_I \cup U_r$. The program P is convergent if and only if there is a deduction of a halt clause from X using resolution.

In the following we will study the convergence of the following program according to the theory presented above. Predicate resolution is used to model the execution of the program.

Program P:
input: $\bar{x} = (x_1, x_2, n)$
output: $z := x_1^n$ (if $n \vdots 2$) or $z := n * x_2$ (if $n \not\vdots 2$)
begin
  $y := 1;$
  $i := 0;$
  if $(n \vdots 2)$
    then
      while $(i < n)$
        $y := y * x_1;$
        $i := i + 1;$
      end_while
    else
      $y := n * x_2;$
  end_if
  $z := y;$
end

- the input vector: $\bar{x} = (x_1, x_2, n)$;
- the output vector: $\bar{z} = (z)$;
- the work vector: $\bar{y} = (i, y)$

We associate the following graph to the program P:



The description formulas for the arcs are as follows:

$W_{a_1} : q_S(x_1,x_2,n,i,y) \wedge P_{a_1} \rightarrow q_1(x_1,x_2,n,0,1) \equiv T \wedge T \rightarrow q_1(x_1,x_2,n,0,1) \equiv$

$\equiv q_1(x_1,x_2,n,0,1)$

$W_{a_2} : q_1(x_1,x_2,n,i,y) \wedge div(n,2) \rightarrow q_2(x_1,x_2,n,i,y)$

$W_{a_3} : q_1(x_1,x_2,n,i,y) \wedge \neg div(n,2) \rightarrow q_H(x_1,x_2,n,n * x_2)$

$W_{a_4} : q_2(x_1,x_2,n,i,y) \wedge less(i,n) \rightarrow q_2(x_1,x_2,n,i+1,y * x_1)$

$W_{a_5} : q_2(x_1,x_2,n,i,y) \wedge \neg less(i,n) \rightarrow q_H(x_1,x_2,n,y)$

The clausal normal forms of these formulas are:

$C_1 : q_1(x_1,x_2,n,0,1)$

$C_2 : \neg q_1(x_1,x_2,n,i,y) \vee \neg div(n,2) \vee q_2(x_1,x_2,n,i,y)$

$C_3 : \neg q_1(x_1,x_2,n,i,y) \vee div(n,2) \vee q_H(x_1,x_2,n,n * x_2)$

$C_4 : \neg q_2(x_1,x_2,n,i,y) \vee \neg less(i,n) \vee q_2(x_1,x_2,n,i+1,y * x_1)$

$C_5 : \neg q_2(x_1,x_2,n,i,y) \vee less(i,n) \vee q_H(x_1,x_2,n,y)$

$U_P = \{C_1,C_2,C_3,C_4,C_5\}$

The axioms which define the predicates: ***equal***, ***less*** and ***div*** for integers are expressed by predicate formulas having universal quantified variables.
We omit the quantifiers obtaining the free formulas presented below.

The functions: $pred(y) = y-1$, $succ(y) = y+1$ are used.

The binary predicate ***equal*** is defined by the axioms:

e_1: $equal(x,x)$ **(reflexivity)**

e_2: $equal(x,y) \to equal(y,x) \equiv \neg equal(x,y) \vee equal(y,x)$ **(symmetry)**

e_3: **(transitivity)**

$equal(x,y) \wedge equal(y,z) \to equal(x,z) \equiv \neg equal(x,y) \vee \neg equal(y,z) \vee equal(x,z)$

e_4: $equal(x,y) \to equal(succ(x),succ(y)) \equiv \neg equal(x,y) \vee equal(succ(x),succ(y))$

e_5: $equal(x,y) \to equal(pred(x),pred(y)) \equiv \neg equal(x,y) \vee equal(pred(x),pred(y))$

The binary predicate ***less*** is defined by the axioms:

l_0: $\neg less(x,x)$

l_1: $less(x,y) \to \neg less(y,x) \equiv \neg less(x,y) \vee \neg less(y,x)$

l_2: $less(0,1)$

l_3: $less(x,y) \to less(x,succ(y)) \equiv \neg less(x,y) \vee less(x,succ(y))$

l_4: $less(x,y) \to less(succ(x),succ(y)) \equiv \neg less(x,y) \vee less(succ(x),succ(y))$

l_5: $less(x,y) \to less(pred(x),y) \equiv \neg less(x,y) \vee less(pred(x),y)$

l_6: $less(x,y) \to less(pred(x),pred(y)) \equiv \neg less(x,y) \vee less(pred(x),pred(y))$

The predicate ***div (divisibility by 2)*** is defined by the axioms:

d_1: $div(0,2)$

d_2: $div(k,2) \to div(k+2,2) \equiv \neg div(k,2) \vee div(k+2,2)$

d_3: $div(k,2) \to div(k-2,2) \equiv \neg div(k,2) \vee div(k-2,2)$

d_4: $div(k,2) \to \neg div(k+1,2) \equiv \neg div(k,2) \vee \neg div(k+1,2)$

d_5: $div(k,2) \to \neg div(k-1,2) \equiv \neg div(k,2) \vee \neg div(k-1,2)$

The set of formulas representing the axioms for the predicates and the functions used in the description of the program is

$U_r = \{e_1, e_2, e_3, e_4, e_5, l_0, l_1, l_2, l_3, l_4, l_5, l_6, d_1, d_2, d_3, d_4, d_5\}$,

$U_I = \varnothing$

For $n = 3$ $(n \geq 2)$ we prove the convergence of P. We follow the resolution process in order to derive a halt clause and obtain the output data. At each resolution step the corresponding unifier is written.

$C_6 = \mathrm{Res}^{\mathrm{Pr}}_{[i \leftarrow 0, y \leftarrow 1]}(C_1, C_3) = div(3,2) \vee q_H(x_1, x_2, 3, 3 * x_2)$

$d_6' = \mathrm{Res}^{\mathrm{Pr}}_{[k \leftarrow 0]}(d_1, d_2) = div(2,2)$

$d_7' = \mathrm{Res}^{\mathrm{Pr}}_{[k \leftarrow 2]}(d_6', d_4) = \neg div(3,2)$

$C_7 = \mathrm{Res}(C_6, d_7') = q_H(x_1, x_2, 3, 3 * x_2)$ halt clause, the output data is $z = 3 * x_2$

$U_P \cup U_r \vdash_{\mathrm{Res}} C_7$, therefore *the program P is convergent.*

For $n=4$ $(n{:}2)$ we prove the convergence of P. In the while loop four iterations are executed and the resolution process models this execution as follows:

$C_8 = \mathrm{Res}^{\mathrm{Pr}}_{[i\leftarrow 0, y\leftarrow 1]}(C_1, C_2) = \neg div(4,2) \vee q_2(x_1, x_2, 4, 0, 1)$

$d_6' = \mathrm{Res}^{\mathrm{Pr}}_{[k\leftarrow 0]}(d_1, d_2) = div(2,2)$

$d_8' = \mathrm{Res}^{\mathrm{Pr}}_{[k\leftarrow 2]}(d_6', d_2) = div(4,2)$

$C_9^0 = \mathrm{Res}(C_8, d_8') = q_2(x_1, x_2, 4, 0, 1)$

$C_{10}^0 = \mathrm{Res}^{\mathrm{Pr}}_{[i\leftarrow 0, y\leftarrow 1]}(C_9^0, C_4) = \neg less(0,4) \vee q_2(x_1, x_2, 4, 1, x_1)$

Using the axioms $l_0, ..., l_6$ we derive $l_7' = less(0,4)$ as follows:

$c_1 = \mathrm{Res}(l_2, l_3) = less(0,2)$, $c_3 = \mathrm{Res}(c_1, l_3) = less(0,3)$

$l_7' = \mathrm{Res}(c_3, l_3) = less(0,4)$

Iteration 1:

$C_9^1 = \mathrm{Res}(C_{10}^0, l_7') = q_2(x_1, x_2, 4, 1, x_1)$

$C_{10}^1 = \mathrm{Res}^{\mathrm{Pr}}_{[i\leftarrow 1, y\leftarrow x_1]}(C_9^1, C_4) = \neg less(1,4) \vee q_2(x_1, x_2, 4, 2, x_1 * x_1)$

Using the axioms $l_0, \ldots, l_6$ we derive $l_8' = less(1,4)$

Iteration 2:

$C_9^2 = \mathrm{Res}(C_{10}^1, l_8') = q_2(x_1, x_2, 4, 2, x_1 * x_1)$

$C_{10}^2 = \mathrm{Res}^{\mathrm{Pr}}_{[i\leftarrow 2, y\leftarrow x_1 * x_1]}(C_9^2, C_4) = \neg less(2,4) \vee q_2(x_1, x_2, 4, 3, x_1 * x_1 * x_1)$

Using the axioms $l_0, \ldots, l_6$ we derive $l_9' = less(2,4)$

Iteration 3:

$C_9^3 = \mathrm{Res}(C_{10}^2, l_9') = q_2(x_1, x_2, 4, 3, x_1 * x_1 * x_1)$

$C_{10}^3 = \mathrm{Res}^{\mathrm{Pr}}_{[i\leftarrow 3, y\leftarrow x_1 * x_1 * x_1]}(C_9^3, C_4) = \neg less(3,4) \vee q_2(x_1, x_2, 4, 4, x_1 * x_1 * x_1 * x_1)$

Using the axioms $l_0, \ldots, l_6$ we derive $l_1' = less(3,4)$

Iteration 4:

$C_9^4 = \mathrm{Res}(C_{10}^3, l_{10}') = q_2(x_1, x_2, 4, 4, x_1 * x_1 * x_1 * x_1)$

The exit of the loop:

$C_{11} = \mathrm{Res}(C_9^4, C_5) = less(4,4) \vee q_H(x_1, x_2, 4, x_1 * x_1 * x_1 * x_1)$

$l_0 = \neg less(4,4), [x \leftarrow 4]$

$C_{12} = \mathrm{Res}(C_{11}, l_0) = q_H(x_1, x_2, 4, x_1 * x_1 * x_1 * x_1)$

C_{12} is a halt clause and provides the output data: $z = x_1 * x_1 * x_1 * x_1$

$U_P \cup U_r \vdash_{\mathrm{Res}} C_{12}$, therefore P *is a convergent program.*

7. BOOLEAN ALGEBRAS AND BOOLEAN FUNCTIONS

George Boole introduced Boolean algebras in 1854 in the paper [7]. In 1938 Claude Shannon proved that a two-valued binary Boolean algebra can describe the operations of two-valued electrical switching circuits. In modern times Boolean algebras and Boolean functions are indispensable in the design of computer chips and digital circuits.

In this chapter the theoretical concepts of Boolean algebras and Boolean functions are introduced. These concepts are used further in the simplification of Boolean functions. Three simplification methods: Veitch-Karnaugh diagrams method, Quine's method and Moisil's method are described and applied in numerous examples. The following papers [8, 14, 25, 26, 37, 53, 62] were used as bibliographic references.

7.1. Boolean algebras

Definition 7.1.

A **Boolean algebra** is a structure $(A, \wedge, \vee, \bar{}, 0, 1)$ described by the following axioms:

1. $|A| \geq 2$, A contains at least two elements: 0 and 1, $0 \neq 1$
2. $\wedge, \vee$ are binary operations
3. $\bar{}$ is a unary operation
4. 0 is the zero element:
   $x \wedge 0 = 0 \wedge x = 0$ and $x \vee 0 = 0 \vee x = x, \forall x \in A$
5. 1 is the unity element:
   $x \wedge 1 = 1 \wedge x = x$ and $x \vee 1 = 1 \vee x = 1, \forall x \in A$
6. 0 is the first element, 1 is the last element
   $x \wedge \bar{x} = 0$ and $x \vee \bar{x} = 1, \forall x \in A$
7. double negation: $\bar{\bar{x}} = x, \forall x \in A$
8. commutativity:
   $x \wedge y = y \wedge x$ and $x \vee y = y \vee x, \forall x, y \in A$
9. associativity:
   $x \wedge (y \wedge z) = (x \wedge y) \wedge z \; (= x \wedge y \wedge z)$ and
   $x \vee (y \vee z) = (x \vee y) \vee z \; (= x \vee y \vee z), \forall x, y, z \in A$
10. distributivity:
    $x \wedge (y \vee z) = (x \wedge y) \vee (x \wedge z)$ and $x \vee (y \wedge z) = (x \vee y) \wedge (x \vee z), \forall x, y, z \in A$

11. idempotency:
    $x \wedge x = x$ and $x \vee x = x, \forall x \in A$
12. De Morgan's laws:
    $\overline{x \wedge y} = \bar{x} \vee \bar{y}$ and $\overline{x \vee y} = \bar{x} \wedge \bar{y}, \forall x, y \in A$
13. absorption laws:
    $x \wedge (x \vee y) = x$ and $x \vee (x \wedge y) = x, \forall x, y \in A$

Note: A **Boolean algebra** is a complemented distributive lattice.

In the literature there are alternative symbols used for the binary operations of a Boolean algebra: $(\wedge,\vee)$ or $(*,+)$. In this paper we use the pair of symbols: $(\wedge,\vee)$.

The **duality principle** in a Boolean algebra: „For any equality of two Boolean expressions, $U = V$, there is another equality, $U' = V'$, obtained by interchanging the Boolean operations: $\wedge$, $\vee$ and the Boolean values: 0, 1".

Example 7.1.
The **binary Boolean algebra** is $B = (B_2 = \{0,1\}, \wedge, \vee, \bar{}, 0, 1)$, where the truth tables for the Boolean operations are as follows:

$\vee$	0	1
0	0	1
1	1	1

$\wedge$	0	1
0	0	0
1	0	1

x	$\bar{x}$
0	1
1	0

Example 7.2.
The structure $(F_P, \wedge, \vee, \neg, F, T)$ is a Boolean algebra.

- F_P is the set of all well-formed propositional formulas;
- binary operations: $\wedge$ (conjunction), $\vee$ (disjunction);
- unary operation: $\neg$ (negation);
- zero element: F, unity element: T, where *F(false)*, *T(true)* are the truth values.

Propositional logic is a logical system that is intimately connected to Boolean algebra. Many syntactic concepts of Boolean algebra carry over to propositional logic with only minor changes in notation and terminology, while the semantics of propositional logic are defined via Boolean algebras in a way that the tautologies (theorems) of propositional logic correspond to equational theorems of Boolean algebra.

The semantics of propositional logic is based on truth assignments. The basic idea of a truth assignment is that the propositional variables are mapped to elements of a fixed Boolean algebra, and then the truth value of a propositional formula is the element of the Boolean algebra that is obtained by computing the value of the Boolean expression corresponding to the formula.

Example 7.3.
The structure $(\mathcal{P}(X), \cap, \cup, C_X, \varnothing, X)$ is a Boolean algebra.

- X is a set and $\mathcal{P}(X)$ is the set of all subsets of X;
- binary operations: $\cap$ (intersection), $\cup$ (reunion);
- unary operation: $C_X(A) = X \setminus A$ - complementary set of A with respect to X;
- zero element: $\varnothing$ and unity element: X.

The properties from Definition 7.1 can be easily verified.

7.2. Boolean functions

Definition 7.2.
Let $\boldsymbol{B} = (B_2, \wedge, \vee, \bar{}, 0, 1)$ be the binary Boolean algebra, $B_2 = \{0,1\}$ and $n \in \boldsymbol{N}^*$. A ***Boolean function of n variables*** is a function $f : (B_2)^n \to B_2$ defined as follows:

1. the projection function: $P_i : B_2^n \to B_2$, $P_i(x_1,..., x_i,..., x_n) = x_i$, is a Boolean function.
2. if $f, g : B_2^n \to B_2$ are Boolean functions then $f \wedge g, f \vee g, \overline{f}$ are Boolean functions of n variables, where:
   $(f \wedge g)(x_1,..., x_n) = f(x_1,..., x_n) \wedge g(x_1,..., x_n)$;
   $(f \vee g)(x_1,..., x_n) = f(x_1,..., x_n) \vee g(x_1,..., x_n)$;
   $\overline{f}(x_1,...,x_n) = \overline{f(x_1,...,x_n)}$.
3. any Boolean function can be obtained by the application of the rules 1 and 2.

Theorem 7.1.

$\forall\, n \in \mathbf{N}^*$, there exist 2^{2^n} Boolean functions of n variables.

Theorem 7.2.
The structure $(\mathbf{FB}(\boldsymbol{n}), \wedge, \vee, \bar{}, f_0, f_{2^{2^n}-1})$ is a Boolean algebra, where $\mathbf{FB}(\boldsymbol{n})$ is the set of all Boolean functions of $n \in \mathbf{N}^*$ variables, where $\wedge$, $\vee$ and $\bar{}$ are defined in the above definition and $f_0(x_1,..., x_n) = 0$ and $f_{2^{2^n}-1}(x_1,..., x_n) = 1$ are the constant functions corresponding to *contradiction* and *tautology* respectively.

Example 7.4.

For $n = 1$, there exist $2^{2^1} = 4$ Boolean functions of one variable represented by their expressions and tables of (truth) values as follows:

x	$f_0(x) = 0$ *contradiction*	$f_1(x) = x$	$f_2(x) = \bar{x}$	$f_0(x) = 1$ *tautology*
0	0	0	1	1
1	0	1	0	1

Example 7.5.

The $2^{2^2} = 2^4 = 16$ Boolean functions of 2 variables are represented in the table below.

x	y	$f_0(x,y)$	$f_1(x,y)$	$f_2(x,y)$	$f_3(x,y)$	$f_4(x,y)$	$f_5(x,y)$	$f_6(x,y)$	$f_7(x,y)$
		contra-diction	$x \wedge y$	$x \wedge \bar{y}$	x	$\bar{x} \wedge y$	y	$x \oplus y$	$x \vee y$
0	0	0	0	0	0	0	0	0	0
0	1	0	0	0	0	1	1	1	1
1	0	0	0	1	1	0	0	1	1
1	1	0	1	0	1	0	1	0	1

x	y	$f_8(x,y)$	$f_9(x,y)$	$f_{10}(x,y)$	$f_{11}(x,y)$	$f_{12}(x,y)$	$f_{13}(x,y)$	$f_{14}(x,y)$	$f_{15}(x,y)$
		$x \downarrow y$	$x \leftrightarrow y$	$\bar{y}$	$y \rightarrow x$	$\bar{x}$	$x \rightarrow y$	$x \uparrow y$	*tauto-logy*
0	0	1	1	1	1	1	1	1	1
0	1	0	0	0	0	1	1	1	1
1	0	0	0	1	1	0	0	1	1
1	1	0	1	0	1	0	1	0	1

Each function has an inverse function obtained as its negation. The pairs of inverse functions are: (f_0, f_{15}), (f_1, f_{14}), (f_2, f_{13}), (f_3, f_{12}), (f_4, f_{11}), (f_5, f_{10}), (f_6, f_9), (f_7, f_8).

We use the following symbols for operations:

- $\downarrow$ (nor) – Pierce's function, $x \downarrow y = \overline{(x \vee y)} = \bar{x} \wedge \bar{y}$
- $\uparrow$ (nand) – Schaffer's function, $x \uparrow y = \overline{(x \wedge y)} = \bar{x} \vee \bar{y}$
- $\rightarrow$ (logical implication), $x \rightarrow y = \bar{x} \vee y$
- $\leftrightarrow$ (logical equivalence), $x \leftrightarrow y = (x \rightarrow y) \wedge (y \rightarrow x) = (x \wedge y) \vee (\bar{x} \wedge \bar{y})$
- $\oplus$ („xor"- exclusive 'or'), $x \oplus y = \overline{(x \leftrightarrow y)} = (\bar{x} \wedge y) \vee (x \wedge \bar{y})$

For a uniform representation of the variables and their negations we introduce the following notation: $x^\alpha = \begin{cases} x, & \text{if } \alpha = 1 \\ \bar{x}, & \text{if } \alpha = 0 \end{cases}$, $\quad x \in \{0, 1\}$.

For $x, \alpha \in \{0, 1\}$ we have: $x^0 = \bar{x}$, $x^1 = x$ and

$$0^0 = \bar{0} = 1; \quad 0^1 = 0; \quad 1^0 = \bar{1} = 0; \quad 1^1 = 1$$

$$x^\alpha = \begin{cases} 1, & \text{if } x = \alpha \\ 0, & \text{if } x \neq \alpha \end{cases}, \quad x, \alpha \in \{0, 1\}.$$

7.3. *Canonical forms of Boolean functions*

All Boolean expressions, regardless of their form, can be transformed into two standard (canonical) forms: the *disjunction-of-conjunctions* form or the *conjunction-of-disjunctions* form.

In the (*, +) notation of the Boolean operations, these forms are called *sum-of-products* (*SOP*) and *product-of-sums* (*POS*) respectively. Standardization makes the evaluation, simplification, and implementation of Boolean expressions much more systematic and easier.

Theorem 7.3.

A Boolean function $f:(B_2)^n \to B_2, n \in \mathbf{N}^*$ can be transformed into two equivalent forms:

1. ***disjunctive canonical form (DCF):*** *disjunction* of conjunctions

   (1) $f(x_1,...,x_n) = \bigvee_{(\alpha_1,...,\alpha_n)\in B_2^n} (f(\alpha_1,...,\alpha_n) \wedge x_1^{\alpha_1} \wedge ... \wedge x_n^{\alpha_n})$

2. ***conjunctive canonical form (CCF):*** *conjunction* of disjunctions

   (2) $f(x_1,...,x_n) = \bigwedge_{(\alpha_1,...,\alpha_n)\in B_2^n} (f(\alpha_1,...,\alpha_n) \vee x_1^{\overline{\alpha_1}} \vee ... \vee x_n^{\overline{\alpha_n}})$

Theorem 7.4.

A Boolean function $f:(B_2)^n \to B_2$ is unique determined by its values $f(\alpha_1,...,\alpha_n)$, where $(\alpha_1,...,\alpha_n) \in B_2^n$:

1. *disjunctive canonical form* (*DCF*):

   (1) $\Leftrightarrow$ (1') $f(x_1,...,x_n) = \bigvee_{(\alpha_1,...,\alpha_n)\in B_2^n \text{ and } f(\alpha_1,...,\alpha_n)=1} (x_1^{\alpha_1} \wedge ... \wedge x_n^{\alpha_n})$

2. *conjunctive canonical form* (*CCF*):

   (2) $\Leftrightarrow$ (2') $f(x_1,...,x_n) = \bigwedge_{(\alpha_1,...,\alpha_n)\in B_2^n \text{ and } f(\alpha_1,...,\alpha_n)=0} (x_1^{\overline{\alpha_1}} \vee ... \vee x_n^{\overline{\alpha_n}})$

Example 7.6.

For n=2, the disjunctive canonical forms of the functions f_8, f_{13}, f_6, f_{12} from Example 7.5 are built using formula (1):

$f_8(x,y) = (1 \wedge x^0 \wedge y^0) \vee (0 \wedge x^0 \wedge y^1) \vee (0 \wedge x^1 \wedge y^0) \vee (0 \wedge x^1 \wedge y^1) = \overline{x} \wedge \overline{y}$

$f_{13}(x,y) = (1 \wedge x^0 \wedge y^0) \vee (1 \wedge x^0 \wedge y^1) \vee (0 \wedge x^1 \wedge y^0) \vee (1 \wedge x^1 \wedge y^1) =$

$= (\overline{x} \wedge \overline{y}) \vee (\overline{x} \wedge y) \vee (x \wedge y)$

$f_6(x,y) = (0 \wedge x^0 \wedge y^0) \vee (1 \wedge x^0 \wedge y^1) \vee (1 \wedge x^1 \wedge y^0) \vee (0 \wedge x^1 \wedge y^1) = (\overline{x} \wedge y) \vee (x \wedge \overline{y})$

$f_{12}(x,y) = (1 \wedge x^0 \wedge y^0) \vee (1 \wedge x^0 \wedge y^1) \vee (0 \wedge x^1 \wedge y^0) \vee (0 \wedge x^1 \wedge y^1) =$

$= (\overline{x} \wedge \overline{y}) \vee (\overline{x} \wedge y) = \overline{x} \wedge (\overline{y} \vee y) = \overline{x}$

Note that the functions' indices, represented in binary using 4 binary digits, are the binary numbers obtained reading the corresponding column.

For $f_{12}(x,y)$, $12=1100_{(2)}$ and the column of the function's values is: $\begin{pmatrix}1\\1\\0\\0\end{pmatrix}$.

Remarks:

1. The disjunctive canonical form is recommended if the function has more values 0 than values 1.
2. The conjunctive canonical form is recommended if the function has more values 1 than values 0.

Example 7.7.

Write the canonical forms of $f_9(x,y)$.

$$f_9(x,y)=(1\wedge x^0\wedge y^0)\vee(0\wedge x^0\wedge y^1)\vee(0\wedge x^1\wedge y^0)\vee(1\wedge x^1\wedge y^1)=(\bar{x}\wedge\bar{y})\vee(x\wedge y)$$

--- *DCF*

and by applying distributivity we obtain:

$$=((\bar{x}\wedge\bar{y})\vee x)\wedge((\bar{x}\wedge\bar{y})\vee y)=(\bar{x}\vee x)\wedge(\bar{y}\vee x)\wedge(\bar{x}\vee y)\wedge(\bar{y}\vee y)=$$
$$=1\wedge(x\vee\bar{y})\wedge(\bar{x}\wedge y)\wedge 1=(x\vee\bar{y})\wedge(\bar{x}\vee y)\ \text{--- } CCF$$

Example 7.8.

Write the corresponding canonical forms of f_2, f_5, f_3, f_0, according to the previous remark:

Formula (1') will be used for f_2, f_5 and formula (2') will be used for f_3, f_0:

$$f_2(x,y)=(x^1\wedge y^0)=x\wedge\bar{y}\ \ (DCF)$$

$$f_5(x,y)=(x^0\wedge y^1)\vee(x^1\wedge y^1)=(\bar{x}\wedge y)\vee(x\wedge y)=(\bar{x}\vee x)\wedge y=1\wedge y=y\ \ (DCF)$$

$$f_3(x,y)=(x^{\bar{0}}\vee y^{\bar{0}})\wedge(x^{\bar{0}}\vee y^{\bar{1}})=(x\vee y)\wedge(x\vee\bar{y})=x\wedge(y\vee\bar{y})=x\wedge 1=x\ \ (CCF)$$

$$f_0(x,y)=(x^{\bar{0}}\vee y^{\bar{0}})\wedge(x^{\bar{0}}\vee y^{\bar{1}})\wedge(x^{\bar{1}}\vee y^{\bar{0}})\wedge(x^{\bar{1}}\vee y^{\bar{1}})$$
$$=(x\vee y)\wedge(x\vee\bar{y})\wedge(\bar{x}\vee y)\wedge(\bar{x}\vee\bar{y})=((x\vee y)\wedge(x\vee\bar{y}))\wedge((\bar{x}\vee y)\wedge(\bar{x}\vee\bar{y}))$$
$$=(x\vee(y\wedge\bar{y}))\vee(\bar{x}\vee(y\wedge\bar{y}))=(x\vee 0)\wedge(\bar{x}\vee 0)=x\wedge\bar{x}=0\ \ (CCF)$$

Remarks:

- The Boolean function f_0 does not have a disjunctive canonical form.
- The Boolean function $f_{2^{2^n}-1}$ does not have a conjunctive canonical form.

Definition 7.3.

Let $f:(B_2)^n \to B_2$, $n \in N^*$ be a Boolean function of n variables.

1. A conjunction of variables is called a ***monom***.
2. A monom which contains all n variables is called ***canonical monom (conjunction)*** or ***minterm*** of n variables and has the form: $x_1^{\alpha_1} \wedge ... \wedge x_n^{\alpha_n}, \alpha_i \in B_2$.
3. A disjunction containing all n variables: $x_1^{\alpha_1} \vee ... \vee x_n^{\alpha_n}, \alpha_i \in B_2$ is called ***canonical disjunction*** or ***maxterm*** of n variables.

Example 7.9.

For n=3:

- $x_1 \wedge x_2$ is a monom, but not a minterm (canonical monom);
- $x_1 \wedge \bar{x}_2 \wedge x_3$ is a minterm (canonical conjunction) of 3 variables;
- $\bar{x}_1 \vee \bar{x}_2 \vee x_3$ is a maxterm (canonical disjunction) of 3 variables.

Properties:

- $\forall n \in \mathbf{N}^*$, there are 2^n maxterms, denoted by $M_0, M_1, ..., M_{2^n-1}$ and 2^n minterms, denoted by $m_0, m_1, ..., m_{2^n-1}$.
- A maxterm is a Boolean function which is 0 for only one argument.
- A minterm is a Boolean function which is 1 for only one argument.

Remarks:

- The index of the standard notation of a minterm of n variables is obtained by converting into decimal of the binary number composed of the digits representing the powers of all n variables from the minterm expression.
- The index of the standard notation of a maxterm of n variables is obtained by the conversion into decimal of the binary number composed of the duals of the digits, digits representing the powers of all n variables from the maxterm expression.

Example 7.10.

For n =1, there are $2^n = 2^1 = 2$ minterms and 2 maxterms:

x	$m_0 = M_1 = \bar{x}$	$m_1 = M_0 = x$
0	1	0
1	0	1

Example 7.11.

For n =2, there are 4 minterms and 4 maxterms having the following expressions and tables of values:

x	y	$m_0 = \bar{x} \wedge \bar{y}$	$m_1 = \bar{x} \wedge y$	$m_2 = x \wedge \bar{y}$	$m_3 = x \wedge y$	$M_0 = x \vee y$	$M_1 = \bar{x} \vee y$	$M_2 = x \vee \bar{y}$	$M_3 = \bar{x} \vee \bar{y}$
0	0	1	0	0	0	0	1	1	1
0	1	0	1	0	0	1	0	1	1
1	0	0	0	1	0	1	1	0	1
1	1	0	0	0	1	1	1	1	0

To be noted that each minterm/maxterm has a unique value 1/0 on its column.

$m_0 = m_{00_{(2)}} = x^0 \wedge y^0 = \bar{x} \wedge \bar{y}$, $\quad m_3 = m_{11_{(2)}} = x^1 \wedge y^1 = x \wedge y$,

$M_1 = M_{01_{(2)}} = x^{\bar{0}} \vee y^{\bar{1}} = x \vee \bar{y}$, $\quad M_2 = M_{10_{(2)}} = x^{\bar{1}} \vee y^{\bar{0}} = \bar{x} \vee y$.

Example 7.12.

For n =3, there are 8 minterms ($m_0, m_1, ..., m_7$) and 8 maxterms ($M_0, M_1, ..., M_7$).

$m_0 = m_{000_{(2)}} = x^0 \wedge y^0 \wedge z^0 = \bar{x} \wedge \bar{y} \wedge \bar{z}$, $\quad M_0 = M_{000_{(2)}} = x^{\bar{0}} \vee y^{\bar{0}} \vee z^{\bar{0}} = x \vee y \vee z$

$m_1 = m_{001_{(2)}} = x^0 \wedge y^0 \wedge z^1 = \bar{x} \wedge \bar{y} \wedge z$, $\quad M_1 = M_{001_{(2)}} = x^{\bar{0}} \vee y^{\bar{0}} \vee z^{\bar{1}} = x \vee y \vee \bar{z}$

$m_2 = m_{010_{(2)}} = x^0 \wedge y^1 \wedge z^0 = \bar{x} \wedge y \wedge \bar{z}$, $\quad M_2 = M_{010_{(2)}} = x^{\bar{0}} \vee y^{\bar{1}} \vee z^{\bar{0}} = x \vee \bar{y} \vee z$

$m_3 = m_{011_{(2)}} = x^0 \wedge y^1 \wedge z^1 = \bar{x} \wedge y \wedge z$, $\quad M_3 = M_{011_{(2)}} = x^{\bar{0}} \vee y^{\bar{1}} \vee z^{\bar{1}} = x \vee \bar{y} \vee \bar{z}$

$m_4 = m_{100_{(2)}} = x^1 \wedge y^0 \wedge z^0 = x \wedge \bar{y} \wedge \bar{z}$, $\quad M_4 = M_{100_{(2)}} = x^{\bar{1}} \vee y^{\bar{0}} \vee z^{\bar{0}} = \bar{x} \vee y \vee z$

$m_5 = m_{101_{(2)}} = x^1 \wedge y^0 \wedge z^1 = x \wedge \bar{y} \wedge z$, $\quad M_5 = M_{101_{(2)}} = x^{\bar{1}} \vee y^{\bar{0}} \vee z^{\bar{1}} = \bar{x} \vee y \vee \bar{z}$

$m_6 = m_{110_{(2)}} = x^1 \wedge y^0 \wedge z^0 = x \wedge \bar{y} \wedge \bar{z}$, $\quad M_6 = M_{110_{(2)}} = x^{\bar{1}} \vee y^{\bar{1}} \vee z^{\bar{0}} = \bar{x} \vee \bar{y} \vee z$

$m_7 = m_{111_{(2)}} = x^1 \wedge y^1 \wedge z^1 = x \wedge y \wedge z$, $\quad M_7 = M_{111_{(2)}} = x^{\bar{1}} \vee y^{\bar{1}} \vee z^{\bar{1}} = \bar{x} \vee \bar{y} \vee \bar{z}$

Theorem 7.5.

1. The conjunction of two distinct minterms is 0.
   $m_i \wedge m_j = 0, \quad \forall i \neq j,\ i, j \in \{0, ..., 2^{n-1}\}$.
2. The disjunction of two distinct maxterms is 1:
   $M_i \vee M_j = 1, \forall i \neq j,\ i, j \in \{0, ..., 2^{n-1}\}$.
3. A minterm and a maxterm with the same index are inverse functions.
   $M_i = \bar{m}_i, \bar{M}_i = m_i, \quad \forall i \in \{0, ..., 2^{n-1}\}$.

Example 7.13.

- $m_2 \wedge m_6 = (x^0 \wedge y^1 \wedge z^0) \wedge (x^1 \wedge y^1 \wedge z^0) = (\bar{x} \wedge y \wedge \bar{z}) \wedge (x \wedge y \wedge \bar{z}) =$
  $= (\bar{x} \wedge x) \wedge y \wedge \bar{z} = 0 \wedge y \wedge \bar{z} = 0$
- $M_6 \vee M_2 = (x^{\bar{1}} \vee y^{\bar{1}} \vee z^{\bar{0}}) \vee (x^{\bar{0}} \vee y^{\bar{1}} \vee z^{\bar{0}}) = (\bar{x} \vee \bar{y} \vee z) \vee (x \vee \bar{y} \vee z) =$
  $= (\bar{x} \vee x) \vee \bar{y} \vee z = 1 \vee \bar{y} \vee z = 1$
- $\bar{m}_3 = \overline{x^0 \wedge y^1 \wedge z^1} = x^{\bar{0}} \vee y^{\bar{1}} \vee z^{\bar{1}} = x \vee \bar{y} \vee \bar{z} = M_3$

Remarks:

- The conjunctive canonical form, *CCF*, is the conjunction of the maxterms corresponding to the values 0 of the function.
- The disjunctive canonical form, *DCF*, is the disjunction of the minterms corresponding to the values 1 of the function.

Example 7.14.

A Boolean function of 3 variables: $f(x_1, x_2, x_3)$ is given below by its table of values. Write the canonical forms of f.

x_1	x_2	x_3	$f(x_1, x_2, x_3)$	M_2	M_4	M_5	m_0	m_1	m_3	m_6	m_7
0	0	0	**1**	1	1	1	**1**	0	0	0	0
0	0	1	**1**	1	1	1	0	**1**	0	0	0
0	1	0	***0***	***0***	1	1	0	0	0	0	0
0	1	1	**1**	1	1	1	0	0	**1**	0	0
1	0	0	***0***	1	***0***	1	0	0	0	0	0
1	0	1	***0***	1	1	***0***	0	0	0	0	0
1	1	0	**1**	1	1	1	0	0	0	**1**	0
1	1	1	**1**	1	1	1	0	0	0	0	**1**

The last eight columns of the table represent the tables of values of the maxterms and the minterms used to write the canonical forms of f.

In the conjunctive canonical form, the maxterms: $\boldsymbol{M_2}$, $\boldsymbol{M_4}$ and $\boldsymbol{M_5}$ are used, and they correspond to the values ***0*** (on rows 3,5,6) of the function f:

$$CCF(f) = M_2 \wedge M_4 \wedge M_5 = (x_1 \vee \bar{x}_2 \vee x_3) \wedge (\bar{x}_1 \vee x_2 \vee x_3) \wedge (\bar{x}_1 \vee x_2 \vee \bar{x}_3)$$

There are five minterms: $\boldsymbol{m_0}$, $\boldsymbol{m_1}$, $\boldsymbol{m_3}$, $\boldsymbol{m_6}$ and $\boldsymbol{m_7}$ corresponding to the values **1** (on rows 1, 2, 4, 7, 8) of the function and they are used to write *DCF*:

$$DCF(f) = m_0 \vee m_1 \vee m_3 \vee m_6 \vee m_7 =$$
$$= (\bar{x}_1 \wedge \bar{x}_2 \wedge \bar{x}_3) \vee (\bar{x}_1 \wedge \bar{x}_2 \wedge x_3) \vee (\bar{x}_1 \wedge x_2 \wedge x_3) \vee (x_1 \wedge x_2 \wedge \bar{x}_3) \vee (x_1 \wedge x_2 \wedge x_3)$$

7.4. Simplification of Boolean functions

Although the truth table representation of a Boolean function is unique, its algebraic expression may be of many different forms. Boolean functions may be simplified by algebraic means using the axioms from Definition 7.1, or by applying simplification methods. Simplification of Boolean functions is essential for the development of simple logic circuits, as we shall see in the next chapter.

In this section the theoretical concepts and a general simplification algorithm are introduced, then three simplification methods are presented and applied in examples.

To simplify a Boolean function means to obtain an equivalent expression of the function with a minimum number of occurrences of variables and connectives. As input data for the simplification process canonical forms are used. We present the simplification procedure based on the disjunctive canonical form.

Definition 7.4.

Let $f:(B_2)^n \to B_2$ be a Boolean function of n variables.

The set $S_f = \{(x_1,x_2,...,x_n) \mid f(x_1,x_2,...,x_n)=1\}$, containing all the groups $(x_1,x_2,...,x_n) \in B_2^n$ for which f takes the value 1, is called **the support of** f.

Example 7.15.

For $n = 4$ variables, the support of the monom $m = x_1 \wedge \bar{x}_3$, is: $S_m = \{(1,1,0,1),(1,0,0,1),(1,1,0,0),(1,0,0,0)\}$

Definition 7.5.

The monom m is **smaller or equal** than the monom m' if the support of m is included or equal to the support of m':

$m \le m'$ if $S_m \subseteq S_{m'}$

Example 7.16.

Let $m = x_1 \wedge \bar{x}_2 \wedge x_4$ and $m' = x_1 \wedge \bar{x}_2$, be two monoms of $n = 4$ variables.

$S_m = \{(1,0,0,1),(1,0,1,1)\}$

$S_{m'} = \{(1,0,0,0),(1,0,0,1),(1,0,1,0),(1,0,1,1)\}$

$S_m \subseteq S_{m'}$ and thus $m \le m'$, $x_1 \wedge \bar{x}_2 \wedge x_4 \le x_1 \wedge \bar{x}_2$.

Note that all the variables of the bigger monom m' occur in the smaller monom m.

Definition 7.6.

For a Boolean function of n variables let us consider the monoms:

$m = x_{k_1}^{\alpha_{k_1}} \wedge .... \wedge x_{k_j}^{\alpha_{k_j}} \wedge x_{k_i} \wedge x_{k_l}^{\alpha_{k_l}} \wedge ... \wedge x_{k_s}^{\alpha_{k_s}}$ and

$m' = x_{k_1}^{\alpha_{k_1}} \wedge .... \wedge x_{k_j}^{\alpha_{k_j}} \wedge \bar{x}_{k_i} \wedge x_{k_l}^{\alpha_{k_l}} \wedge ... \wedge x_{k_s}^{\alpha_{k_s}}$, where :

$k_1,...,k_j,k_i,k_l,...,k_s \in \{1,2,...,n\}$ and $k_1 < ... < k_j < k_i < k_l < ... < k_s$,

1. m and m' are called ***adjacent (neighbor) monoms*** because they differ only by the power of the variable with the index "k_i".
2. ***the factorization*** *of the monoms* m and m' is:

$m \vee m' = x_{k_1}^{\alpha_{k_1}} \wedge .... \wedge x_{k_j}^{\alpha_{k_j}} \wedge x_{k_l}^{\alpha_{k_l}} \wedge ... \wedge x_{k_s}^{\alpha_{k_s}}$ obtained by eliminating the variable with the index "k_i".

Adjacency (neighborhood) relation is defined by a single variable change.
The factorization is based on the application of the distributivity of „$\wedge$" over „$\vee$". Applied in the particular case of the previous definition, where y is the common part of the monoms, $m = y \wedge x_{k_i}$ and $m' = y \wedge \bar{x}_{k_i}$, factorization provides:

$$(y \wedge x_{k_i}) \vee (y \wedge \bar{x}_{k_i}) = y \wedge (x_{k_i} \vee \bar{x}_{k_i}) = y \wedge 1 = y .$$

As a conclusion, the result of factorization is a monom containing the common part of m and m', this monom is bigger than both m and m' and it covers them.

Note: In this chapter and the next one we shall use the concatenation of the variables to symbolize their conjunction. For example, the monom $m = x_1 \wedge \bar{x}_2 \wedge x_4$ is represented as $m = x_1\bar{x}_2x_4$.

Definition 7.7. [62]
The set $M(f)$ is called the ***set of the maximal monoms*** for the function $f:(B_2)^n \to B_2$ if:

1. $\forall m \in M(f), m \in FB(n), m \leq f$ and
2. $\forall m \in M(f), \exists m' \in FB(n)$ such that $m < m' \leq f$.

Example 7.17.
$f(x,y) = (x \wedge y) \vee (\bar{x} \wedge y) = m_3 \vee m_1$. We have that:

- $m_3 < f$ and $m_1 < f$;
- $m_3 < y \leq f$ and $m_1 < y \leq f$;

So, $m_3, m_1 \notin M(f)$ and $M(f)=\{y\}$

Note: The maximal monoms are minterms or monoms obtained by applying factorizations.

Definition 7.8. [62]
The set $C(f)$ is called the ***set of the central monoms*** for the function $f:(B_2)^n \to B_2$ if:

1. $\forall m \in C(f), m \in M(f)$ and
2. $\forall m \in C(f), m \not\leq \vee_{m' \in M(f) \setminus \{m\}} m'$

Remarks:

- A maximal monom is a central monom for a function *f*, if the monom is not smaller than the disjunction of all the other maximal monoms of *f*.
- All the central monoms belong to all the disjunctive simplified forms of the function *f*.

Properties:

Let $f:(B_2)^n \to B_2$ be a Boolean function.

1. $M(f) \neq \varnothing$, but $C(f)$ can be empty.
2. $f(x_1,...,x_n) = \bigvee_{m \in M(f)} m$ but this is not always the simplest form of *f*.

Informally, to simplify a Boolean function given in disjunctive canonical form, means to cover all its minterms with a minimum number of maximal monoms and with a minimum number of overlaps.

The simplification process is formalized by the steps below. The steps 2, 3, 4 are specific to the applied simplification method.

1. The initial function *f* is transformed into its equivalent disjunctive canonical form $DCF(f)$.
2. Factorization process => the set of maximal monoms $M(f)$.
3. From the set of maximal monoms the central monoms are selected => $C(f)$
4. The case of the simplification algorithm (presented below) is identified and all the simplified forms are obtained.

The following algorithm ([62]) is a general one and it is used in all three simplification methods presented in the next sections:

- Veitch-Karnaugh diagrams method – a graphical method;
- Quine-Mc'Clusky's method – an analytical method;
- Moisil's method – an algebraic method which uses propositional logic.

Algorithm Simplification:
input: f – *a Boolean function in disjunctive canonical form*
output: $f^{S}_1, f^{S}_2, ..., f^{S}_k$ – *all the disjunctive simplified forms of* f
begin
calculate $M(f)$ and $C(f)$
if $M(f) = C(f)$ **then**

$f^S = \bigvee_{m \in M(f)} m$; STOP 1 // **case 1**--- one solution

else
if $C(f) \neq \varnothing$ **then** // f has central monoms

$g = \bigvee_{m \in C(f)} m$

$f^S_i = g \vee h_i$, $i = \overline{1,k}$, h_i is a disjunction of a minimum number of maximal monoms such that $S_{h_i} = S_f \setminus S_g$

STOP 2 // **case 2** --- k solutions
else // f does not have central monoms
$f^S_i = h_i$, $i = \overline{1,k}$, h_i is a disjunction of a minimum number of maximal monoms such that $S_{h_i} = S_f$

STOP 3 // **case 3** --- k solutions
end_if
end_if
end

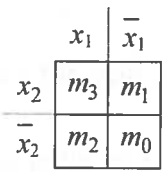
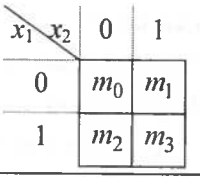
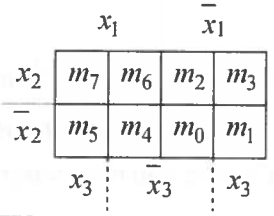
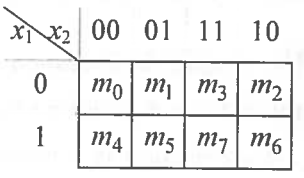
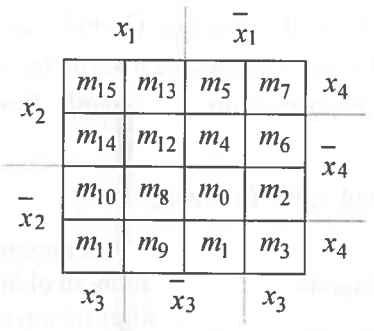
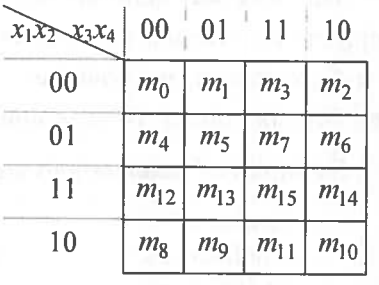
7.5. Veitch-Karnaugh diagrams method

This method is based on a graphical representation of the disjunctive canonical form of the initial function. It is recommended to be used for Boolean functions of 2, 3 and 4 variables. For more than 4 variables the graphical representation using diagrams is not so intuitive.

For a function of n variables, the diagram (Veitch or Karnaugh) has 2^n cells, and each cell (intersection of a line and a column) corresponds to a minterm. The headers of the lines and columns represent:

- the variables simple or negated in Veitch diagram
- all the possible combinations of the powers (0,1) of the variables in Karnaugh diagram.

The minterms from the *DCF* are represented in the diagrams as follows:

No. var.	Veitch diagram	Karnaugh diagram
2		
3		
4		

The cells in a Veitch/Karnaugh diagram are arranged so that there is only a single variable change between neighbor (adjacent cells). Adjacency (graphical neighborhood) relation is defined by a single variable change.

The diagrams are circular, meaning that the first row/column and the last row/column are neighbors. Two minterms belonging to two neighbor cells (graphical neighborhood relation) are neighbors according to Definition 7.6 and they factorize.

For $n = 2$ variables:

- $m_0 = \bar{x}_1\bar{x}_2$ and $m_2 = x_1\bar{x}_2$ are neighbors, they differ only by variable x_1 and they factorize: $m_0 \vee m_2 = \bar{x}_2$
- $m_2 = x_1\bar{x}_2$ and $m_3 = x_1x_2$ are neighbors, they differ only by variable x_2 and they factorize: $m_2 \vee m_3 = x_1$

For $n = 3$ variables:

- $m_5 = x_1\bar{x}_2x_3$ and $m_7 = x_1x_2x_3$ are neighbors, they differ only by variable x_2 and they factorize: $m_5 \vee m_7 = x_1x_3$

- $m_4 = x_1\bar{x}_2\bar{x}_3$ and $m_6 = x_1x_2\bar{x}_3$ are neighbors, they differ only by variable x_2 and they factorize: $m_4 \vee m_6 = x_1\bar{x}_3$

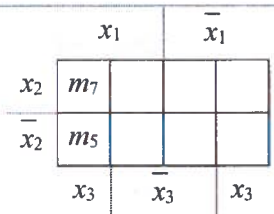
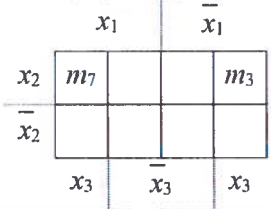
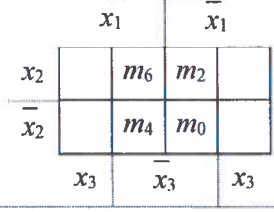
For $n = 4$ variables:

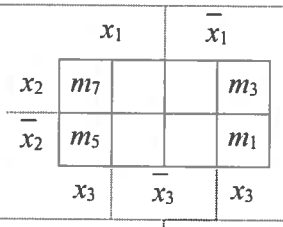
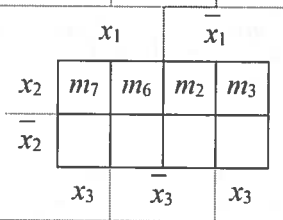
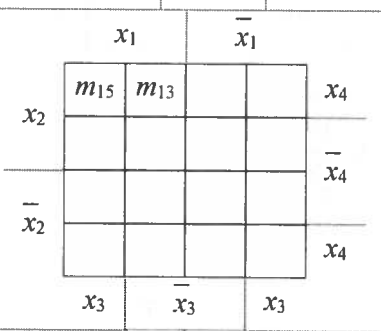
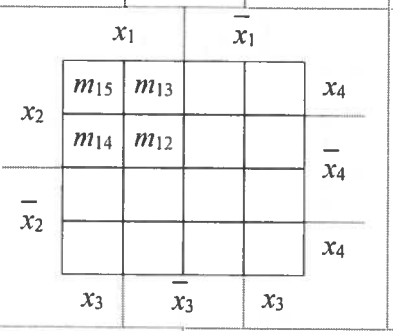
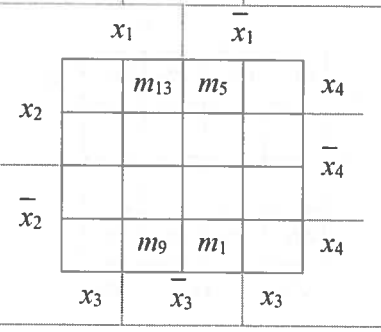
- $m_{12} = x_1\bar{x}_2\bar{x}_3\bar{x}_4$ and $m_8 = x_1\bar{x}_2\bar{x}_3\bar{x}_4$ are neighbors, they differ only by variable x_2 and they factorize: $m_{12} \vee m_8 = x_1\bar{x}_3\bar{x}_4$
- $m_1 = \bar{x}_1\bar{x}_2\bar{x}_3x_4$ and $m_3 = \bar{x}_1\bar{x}_2x_3x_4$ are neighbors, they differ only by variable x_3 and they factorize: $m_1 \vee m_3 = \bar{x}_1\bar{x}_2x_4$

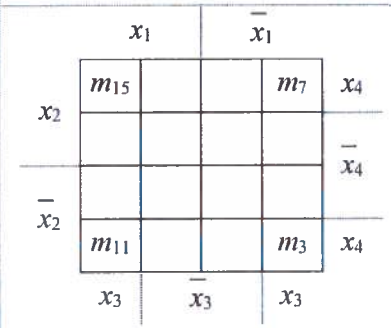
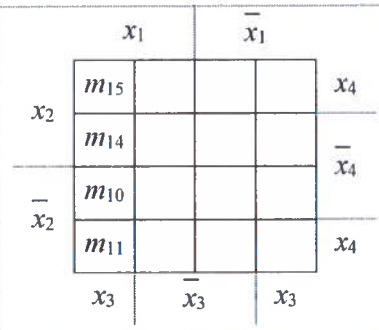
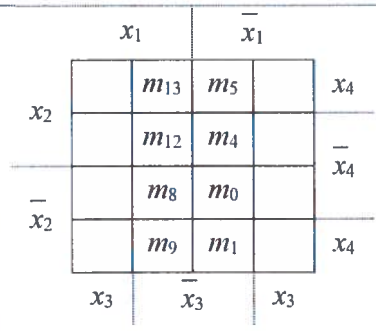
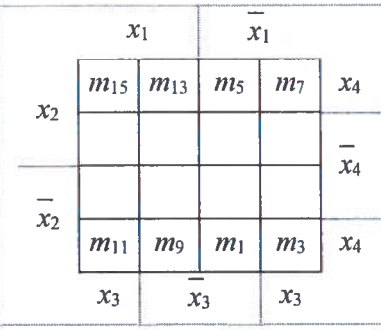
The *simple factorization* operation between two neighbor minterms can be generalized to a *k-factorization:* 2^k neighbor minterms (cells) are used. The result of a *k-factorization* is a monom which contains the common variables of all 2^k minterms.

The **maximal monoms** are obtained from the diagram (Veitch/Karnaugh) by applying factorizations as follows: for a Boolean function of ***n*** variables we try first a ***n-factorization***, we continue with ***n-1*** *factorizations*, ... , *simple factorizations* and *0-factorizations* (isolated minterms).

Examples of factorizations are presented in the following table.

Number of simplified variables	Veitch diagram	The maximal monom obtained after factorization
1 from 3 simple factorization		$m_7 \vee m_5 = x_1x_3$
1 from 3 simple factorization		$m_7 \vee m_3 = x_2x_3$
2 from 3 double factorization		$m_0 \vee m_2 \vee m_4 \vee m_6 = \bar{x}_3$

Number of simplified variables	Veitch diagram	The maximal monom obtained after factorization
2 from 3 double factorization		$m_1 \vee m_3 \vee m_5 \vee m_7 =$ $= x_3$
2 from 3 double factorization		$m_2 \vee m_3 \vee m_6 \vee m_7 =$ $= x_2$
1 from 4 simple factorization		$m_{13} \vee m_{15} = x_1 x_2 x_4$
2 from 4 double factorization		$m_{12} \vee m_{13} \vee m_{14} \vee m_{15}$ $= x_1 x_2$
2 from 4 double factorization		$m_1 \vee m_5 \vee m_9 \vee m_{13} =$ $= \bar{x}_3 x_4$

Number of simplified variables	Veitch diagram	The maximal monom obtained after factorization
2 from 4 double factorization		$m_3 \vee m_7 \vee m_{11} \vee m_{15} =$ $= x_3 x_4$
2 from 4 double factorization		$m_{10} \vee m_{11} \vee m_{14} \vee m_{15}$ $= x_1 x_3$
3 from 4 triple factorization		$m_0 \vee m_1 \vee m_4 \vee m_5 \vee$ $m_8 \vee m_9 \vee m_{12} \vee m_{13} =$ $= \bar{x}_3$
3 from 4 triple factorization		$m_1 \vee m_3 \vee m_5 \vee m_7 \vee$ $m_9 \vee m_{11} \vee m_{13} \vee m_{15} =$ $= x_4$

The **central monoms** are the maximal monoms which cannot be covered by the disjunction of all the other maximal monoms. In the diagrams, the identification of the central monoms is very simple: if the group of the minterms covered by a maximal monom contains at least one cell (minterm) circled once, then the maximal monom is a central one and it belongs to all the simplified forms of the function.

We follow the general simplification algorithm and we have to identify the simplification case and obtain all the simplified forms.

The groups of the minterms corresponding to the central monoms are shaded in the diagram. The minterms from the function's expression which are unshaded, are not covered by the central monoms and they will be covered in all the possible ways using a minimum number of unused maximal monoms and with a minimum number of overlaps, resulting all the simplified forms of the initial function.

Example 7.18.

Simplify the following Boolean function given by its Boolean *DCF* expression.

$f(x_1, x_2, x_3) = x_1x_2x_3 \vee x_1\bar{x}_2x_3 \vee \bar{x}_1x_2\bar{x}_3 \vee x_1\bar{x}_2x_3 \vee x_1\bar{x}_2\bar{x}_3 \vee \bar{x}_1\bar{x}_2x_3$

Using the standard notations for the minterms, the function can be written as:

$f(x_1, x_2, x_3) = m_7 \vee m_6 \vee m_2 \vee m_5 \vee m_4 \vee m_1$

The representation of the minterms in the diagrams:

Veitch diagram

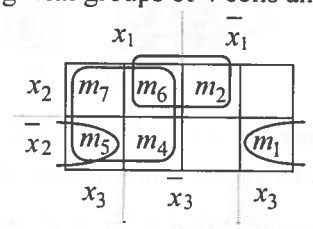
	x_1		$\bar{x}_1$
x_2	m_7	m_6	m_2	
$\bar{x}_2$	m_5	m_4		m_1
	x_3	$\bar{x}_3$	x_3

Karnaugh diagram

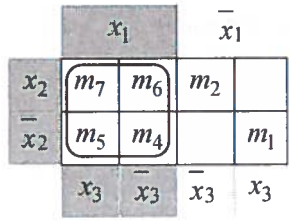
x_1 \ x_2x_3	00	01	11	10
0		m_1		m_2
1	m_4	m_5	m_7	m_6

In the next steps we shll use the Veitch diagram.

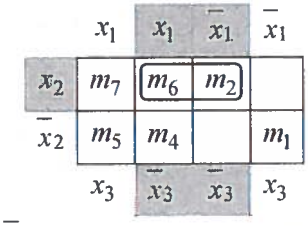
The factorization process consists in factorizing all the possible groups of neighbor minterms, beginning with groups of 4 cells and then groups of 2 cells.



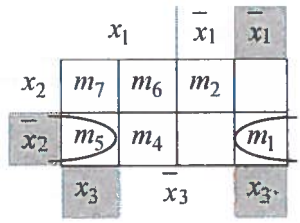
There is a group of 4 neighbor minterms and two groups of 2 neighbor minterms, circled in the diagram. The maximal monoms are obtained as follows:



Maximal monom: $max_1 = x_1 = m_7 \vee m_6 \vee m_5 \vee m_4$ (double factorization)

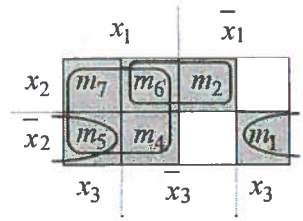


Maximal monom: $max_2 = x_2\bar{x}_3 = m_6 \vee m_2$ (simple factorization).



Maximal monom: $max_3 = \bar{x}_2x_3 = m_5 \vee m_1$ (simple factorization).

The set of the maximal monoms is $M(f) = \{max_1, max_2, max_3\} = \{x_1, x_2\bar{x}_3, \bar{x}_2x_3\}$.
The **central monoms** are identified. A maximal monom is a central monom if its corresponding group of minterms contains at least one minterm circled once. The minterms covered by the central monoms are shaded in the diagram.



The set of the central monoms is $C(f) = \{max_1, max_2, max_3\} = \{x_1, x_2\bar{x}_3, \bar{x}_2x_3\}$
$M(f) = C(f)$ --- the first case of the simplification algorithm.

There is a unique disjunctive simplified form of f, obtained as the disjunction of all the central monoms: $f^S(x_1, x_2, x_3) = x_1 \vee x_2\bar{x}_3 \vee \bar{x}_2x_3$

Example 7.19.

Simplify the following Boolean function:

$f(x_1, x_2, x_3) = x_2\bar{x}_3 \vee \bar{x}_1x_2x_3 \vee x_1\bar{x}_2 \vee \bar{x}_1\bar{x}_2x_3.$

The function's expression is a disjunctive form but not a canonical one because the first and the third monoms are not minterms. In order to obtain the *DCF* we replace these monoms by the minterms covered by them:

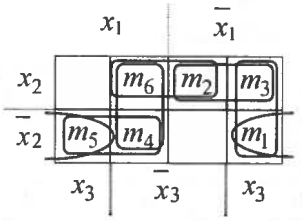
- the monom $x_2\bar{x}_3$ covers the minterms $x_1x_2\bar{x}_3$ and $\bar{x}_1x_2\bar{x}_3$ obtained from the initial monom and adding the missing variable, simple: x_1 and negated: $\bar{x}_1$.
- the monom $x_1\bar{x}_2$ covers the minterms $x_1\bar{x}_2x_3$ and $x_1\bar{x}_2\bar{x}_3$ obtained from the initial monom and adding the missing variable, simple: x_3 and negated: $\bar{x}_3$.

The disjunctive canonical form of f is provided below using the Boolean expression and the standard notations of the minterms.

$DCF(f) = x_1x_2\bar{x}_3 \vee \bar{x}_1x_2\bar{x}_3 \vee \bar{x}_1x_2x_3 \vee x_1\bar{x}_2x_3 \vee x_1\bar{x}_2\bar{x}_3 \vee \bar{x}_1\bar{x}_2x_3 =$

$= m_6 \vee m_2 \vee m_3 \vee m_5 \vee m_4 \vee m_1$

The corresponding Veitch diagram is depicted below.



The set of the **maximal monoms** is obtained by applying six simple factorizations on groups of two neighbor cells. All these six groups are circled in the diagram.

$max_1 = x_1\bar{x}_2 = m_5 \vee m_4,$ $\quad max_2 = x_1\bar{x}_3 = m_4 \vee m_6$

$max_3 = x_2\bar{x}_3 = m_6 \vee m_2,$ $\quad max_4 = \bar{x}_2x_3 = m_5 \vee m_1$

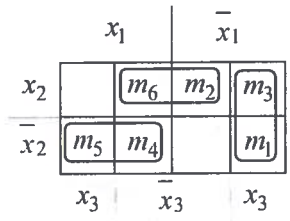
$max_5 = \bar{x}_1x_2 = m_2 \vee m_3,$ $\quad max_6 = \bar{x}_1x_3 = m_3 \vee m_1$

$M(f) = \{max_1, max_2, max_3, max_4, max_5, max_6\} = \{x_1\bar{x}_2, x_1\bar{x}_3, x_2\bar{x}_3, \bar{x}_2x_3, \bar{x}_1x_2, \bar{x}_1x_3\}$

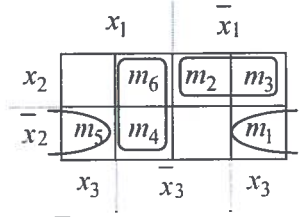
The set of the **central monoms**: $C(f) = \varnothing$ because all the minterms are circled twice, meaning that each minterm can be covered by two maximal monoms.

$M(f) \neq C(f)$ and $C(f) = \varnothing$, thus the third case of the simplification algorithm is applied.

All the minterms must be covered by a minimum number of maximal monoms, with a minimum number of overlaps. There are two such possibilities: all the minterms are covered by three maximal monoms and without overlaps, obtaining two equivalent disjunctive simplified forms of f.



$$f_1^S(x_1, x_2, x_3) = x_1\bar{x}_2 \vee x_2\bar{x}_3 \vee \bar{x}_1x_3 = max_1 \vee max_3 \vee max_6$$



$$f_2^S(x_1, x_2, x_3) = x_1\bar{x}_3 \vee \bar{x}_1x_2 \vee \bar{x}_2x_3 = max_2 \vee max_5 \vee max_4$$

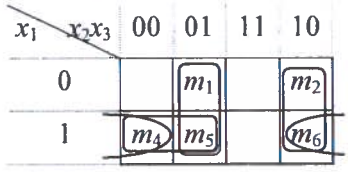
Example 7.20.
Using Karnaugh diagram simplify the Boolean function f given by its values 0: $f(0,0,0) = f(0,1,1) = f(1,1,1) = 0$.
The table of its values is provided below.

x_1	x_2	x_3	$f(x_1, x_2, x_3)$
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	0

The disjunctive canonical form of f is obtained from the table:

$$f(x_1, x_2, x_3) = \bar{x}_1\bar{x}_2x_3 \vee \bar{x}_1x_2\bar{x}_3 \vee x_1\bar{x}_2\bar{x}_3 \vee x_1\bar{x}_2x_3 \vee x_1x_2\bar{x}_3 = m_1 \vee m_2 \vee m_4 \vee m_5 \vee m_6$$

We fill the Karnaugh diagram as follows:



$M(f) = \{ max_1, max_2, max_3, max_4\} = \{x_1\bar{x}_2, x_1\bar{x}_3, x_2\bar{x}_3, \bar{x}_2x_3\}$ is obtained by applying four simple factorizations:

$max_1 = x_1\bar{x}_2 = m_5 \vee m_4$
$max_2 = x_1\bar{x}_3 = m_4 \vee m_6$
$max_3 = x_2\bar{x}_3 = m_6 \vee m_2$
$max_4 = \bar{x}_2x_3 = m_5 \vee m_1$

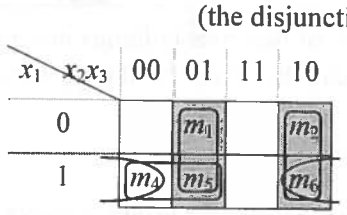
The set of the central monoms is $C(f) = \{max_3, max_4\} = \{x_2\bar{x}_3, \bar{x}_2x_3\}$

- ***max*$_1$** is not a central monom because it is covered by the disjunction of max_2 and max_4:
  $max_1 \le max_2 \vee max_4$;
- ***max*$_2$** is not a central monom because it is covered by the disjunction of max_1 and max_3:
  $max_2 \le max_1 \vee max_3$;
- ***max*$_3$** is a central monom because it is not covered by the disjunction of all the other maximal monoms:
  $max_3 \not\le max_1 \vee max_2 \vee max_4$, the minterm m_2 is circled once;
- ***max*$_4$** is a central monom because it is not covered by the disjunction of all the other maximal monoms:
  $max_4 \not\le max_1 \vee max_2 \vee max_3$, the minterm m_1 is circled once.

$M(f) \neq C(f)$ and $C(f) \neq \varnothing$, this is the second case of the simplification algorithm.

We denote by $g(x_1, x_2, x_3) \overset{not.}{=} x_2\bar{x}_3 \vee \bar{x}_2x_3 = max_3 \vee max_4$

(the disjunction of the central monoms)



The groups of cells corresponding to the central monoms are shaded in the diagram. The only unshaded minterm m_4 must be covered in all the possible ways by a minimum number of unused maximal monoms and with a minimum number of overlaps.

We can use two such functions h (see the simplification algorithm):

$h_1(x_1, x_2, x_3) \overset{not.}{=} x_1\bar{x}_2 = max_1$ or $h_2(x_1, x_2, x_3) \overset{not.}{=} x_1\bar{x}_3 = max_2$

Correspondingly, there are two disjunctive simplified forms of the initial function:

$f_1^S(x_1, x_2, x_3) = g \vee h_1 = max_3 \vee max_4 \vee max_1 = x_2\bar{x}_3 \vee \bar{x}_2x_3 \vee x_1\bar{x}_2$

$f_2^S(x_1, x_2, x_3) = g \vee h_2 = max_3 \vee max_4 \vee max_2 = x_2\bar{x}_3 \vee \bar{x}_2x_3 \vee x_1\bar{x}_3$.

Based on the simplification algorithm, having as input a *DCF*, a dual simplification algorithm that works on *CCF* (conjunction of maxterms) can be obtained.

While in the simplification of *DCF* the values 1 of the function (expressed by the minterms) are covered, in the simplification of *CCF* the values 0 of the function (expressed by the maxterms) are covered.

Similar steps are applied and conjunctive simplified forms are obtained. We build minimal and central disjunctions of a function, the duals of maximal and central monoms (conjunctions). In dual factorizations we work with groups of 2^k neighbor maxterms and the minimal disjunctions of a function are calculated. The maxterms (canonical disjunctions) from the function's expression are covered by a minimum number of minimal disjunctions and with a minimum number of overlaps. Thus, the conjunctive simplified forms are built.

We apply the dual simplification algorithm for the conjunctive canonical form of f $CCF(f) = M_0 \wedge M_3 \wedge M_7 = (x_1 \vee x_2 \vee x_3) \wedge (x_1 \vee \bar{x}_2 \vee \bar{x}_3) \wedge (\bar{x}_1 \vee \bar{x}_2 \vee \bar{x}_3)$ is obtained from its table of values.

The Karnaugh diagram for $CCF(f)$ is depicted below.

x_1 \ x_2x_3	00	01	11	10
0	M_0		M_3	
1			M_7	

In the diagram, the headers of the lines-columns are used to express the indices of the maxterms and represent the duals of the powers of the variables from the maxterms' expressions.

$$M_3 = M_{011_{(2)}} = x_1^{\bar{0}} \vee x_2^{\bar{1}} \vee x_3^{\bar{1}} = x_1 \vee \bar{x}_2 \vee \bar{x}_3$$

There is an isolated maxterm (M_0) and a group of two neighbor maxterms (M_3, M_7). By applying a simple dual factorization we obtain a minimal disjunction of f:

- $min_1 = M_3 \wedge M_7 = (x_1 \vee \bar{x}_2 \vee \bar{x}_3) \wedge (\bar{x}_1 \vee \bar{x}_2 \vee \bar{x}_3) = (x_1 \wedge \bar{x}_1) \vee \bar{x}_2 \vee \bar{x}_3 =$
  $= 0 \vee \bar{x}_2 \vee \bar{x}_3 = \bar{x}_2 \vee \bar{x}_3$ (the common part of the maxterms).
- min_1 is smaller than both M_3 and M_7 and it covers the values 0 of both maxterms.

The minimal disjunctions of f are min_1 and $min_2 = M_0 = x_1 \vee x_2 \vee x_3$.
These are also central disjunctions of f, because each group of cells has at least one maxterm circled once. This is the first case of the dual simplification algorithm.

The function f has a unique conjunctive simplified form obtained as the conjunction of min_1 and min_2.

$$f^{CS}(x_1, x_2, x_3) = (x_1 \vee x_2 \vee x_3) \wedge (\bar{x}_2 \vee \bar{x}_3).$$

Let us compare the disjunctive simplified forms with the conjunctive form:

- f_1^S and f_2^S have 6 occurrences of variables, 5 binary operations and 3 unary operations.
- f^{CS} has 5 occurrences of variables, 4 binary operations and 2 unary operations.

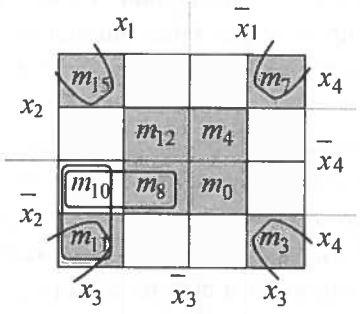
We conclude that $f^{CS}(x_1, x_2, x_3) = (x_1 \vee x_2 \vee x_3) \wedge (\bar{x}_2 \vee \bar{x}_3)$ is the simplest equivalent form of f.

Example 7.21.

Using Veitch diagram simplify the Boolean function:

$$f(x_1, x_2, x_3, x_4) = x_1x_2x_3x_4 \vee x_1\bar{x}_2\bar{x}_3x_4 \vee \bar{x}_1x_2\bar{x}_3\bar{x}_4 \vee x_1\bar{x}_2x_3\bar{x}_4 \vee x_1x_2x_3\bar{x}_4 \vee \bar{x}_1\bar{x}_2\bar{x}_3\bar{x}_4 \vee$$
$$\vee x_1\bar{x}_2x_3x_4 \vee \bar{x}_1x_2x_3x_4 \vee \bar{x}_1\bar{x}_2\bar{x}_3x_4 =$$
$$= m_{15} \vee m_{12} \vee m_4 \vee m_{10} \vee m_7 \vee m_8 \vee m_{11} \vee m_3 \vee m_0$$

The function's expression is the disjunctive canonical form and we fill with the minterms the Veitch diagram of four variables:



$m_{15} \vee m_7 \vee m_{11} \vee m_3 = x_3x_4 = max_1$ (double factorization)

$m_{12} \vee m_4 \vee m_0 \vee m_8 = \bar{x}_3\bar{x}_4 = max_2$ (double factorization)

$m_{10} \vee m_{11} = x_1\bar{x}_2x_3 = max_3$ (simple factorization)

$m_{10} \vee m_8 = x_1\bar{x}_2\bar{x}_4 = max_4$ (simple factorization)

$M(f) = \{x_3x_4, \bar{x}_3\bar{x}_4, x_1\bar{x}_2x_3, x_1\bar{x}_2\bar{x}_4\}$ - the set of *maximal monoms*

$C(f) = \{max_1, max_2\}$ – the set of central monoms

$M(f) \neq C(f)$ and $C(f) \neq \varnothing$ --- the second case of the simplification algorithm.

We denote by $g = max_1 \vee max_2 = x_3x_4 \vee \bar{x}_3\bar{x}_4$ the disjunction of all the central monoms, belonging to all the simplified forms of f.

The minterms covered by the central monoms are shaded in the diagram.
Only the minterm m_{10} is not covered by the central monoms and we can cover it in two ways: using max_3 or using max_4.

Thus, there are two disjunctive simplified forms of the initial function:

$$f_1^S(x_1, x_2, x_3, x_4) = g \vee max_3 = x_3x_4 \vee \bar{x}_3\bar{x}_4 \vee x_1\bar{x}_2x_3$$
$$f_2^S(x_1, x_2, x_3, x_4) = g \vee max_4 = x_3x_4 \vee \bar{x}_3\bar{x}_4 \vee x_1\bar{x}_2\bar{x}_4$$

7.6. Quine-Mc'Clusky's method

This method, used to simplify Boolean functions, is an analytical one which can be easily implemented and its advantage over the previous graphical method is that it can be applied for functions of any number of variables.

The Boolean function, f, is given in *DCF*.
The first step is to order the support set of f,
$S_f = \{(x_1, x_2, ..., x_n) \mid f(x_1, x_2, ..., x_n) = 1\}$, in ascending/descending order, with respect to the number of values "1" in each n-uple.

The minterms from the function's expression are represented using the powers of the variables, in a tableau, each minterm on a row, in ascending/descending order, with respect to the number of the values "1" in the n-uples of the support of the function. The header of the tableau contains the variables names. We will group the minterms (delimited by horizontal lines), such that all the minterms belonging to the same group have the same number of values 1 as powers of the variables. A double horizontal line marks the end of the representation of the initial function.

We continue with simple factorizations obtaining new groups of monoms which will participate in double factorizations and so on.

Only two neighbor groups may contain two adjacent (neighbor) monoms (according to Definition 7.6), which factorize. The result of the **factorization** of two neighbor monoms (m and m') is a new monom represented in a new row at the end of the tableau. The row contains the same values (0,1, or –) in the columns corresponding to the common variables (of m and m') and the symbol "–" for the variable which is eliminated. The rows corresponding to m and m' are marked (on the left side), with the meaning that they are not maximal monoms.
All the monoms obtained as results of the factorization of two neighbor groups will form a new group (delimited by a horizontal line), used further in higher order factorizations (double, triple,...).

The symbol "–" cannot be combined with anything else. Thus, beginning with double factorization only rows from two neighbor groups having "–" on the same position (column) will be combined.

A double horizontal line symbolizes the end of a simple, double, triple,... factorization. The end of the factorization process is represented by a triple horizontal line and the tableau is closed.

The set of the **maximal monoms** contains the monoms corresponding to all the unmarked rows from the tableau.

For obtaining the **central monoms** a new tableau representing the correspondence between the maximal monoms (on columns) and the minterms from the function's expression (on rows) is used. A cell in the tableau is marked

with an asterisk (*) if the minterm corresponding to the row was used in factorization to obtain the maximal monom from the column. A maximal monom is a central monom if there is a * (on its column), which is unique on its row. The disjunction of all central monoms belongs to all the simplified forms of the initial function.

According to the previous tableau, the minterms from the function's expression which are uncovered by the central monoms will be covered in all the possible ways using a minimum number of unused maximal monoms, with a minimum number of overlaps, resulting all the simplified forms of the initial function.

Example 7.22.

Let f be a Boolean function of 3 variables given by the following expression: $f(x_1, x_2, x_3) = \bar{x}_1\bar{x}_2x_3 \vee \bar{x}_1x_2\bar{x}_3 \vee \bar{x}_1x_2x_3 \vee x_1\bar{x}_2\bar{x}_3$.

Using the standard notations for the minterms: $f(x_1, x_2, x_3) = m_1 \vee m_2 \vee m_3 \vee m_4$.
The ascending order, with respect to the number of "1" values in each triplet, of the support of f is $S_f = \{ (0,0,1), (0,1,0), (1,0,0), (0,1,1)\}$

Representation: Group		x_1	x_2	x_3	
	√	0	0	1	m_1
I	√	0	1	0	m_2
		1	0	0	$m_4 = x_1\bar{x}_2\bar{x}_3 = max_1$
II	√	0	1	1	m_3
Simple factorizations: III=I+II		0	–	1	$m_1 \vee m_3 = \bar{x}_1x_3 = max_2$
		0	1	–	$m_2 \vee m_3 = \bar{x}_1x_2 = max_3$

1. Only simple factorizations can be applied: $m_1 \vee m_3$ and $m_2 \vee m_3$. The minterm m_4 does not have neighbors, it is an isolated minterm which is a maximal monom. The unmarked rows correspond to the maximal monoms:
   $M(f) = \{ \bar{x}_1\bar{x}_2x_3, \bar{x}_1x_3, \bar{x}_1x_2\} = \{max_1, max_2, max_3\}$.
2. We build the following correspondence tableau (minterms - maximal monoms) and we circled the '*' which are unique on their rows. We can read the tableau on columns or on rows as follows:

- third row: the minterm m_3 is covered by max_2 and by max_3
- second column: the maximal monom max_2 was obtained as a result of a simple factorization between the minterms m_1 and m_3

All the columns have a ⊛, thus all the maximal monoms are central monoms:
$M(f) = C(f)$.

minterms \ max.monoms	max_1	max_2	max_3
m_1		⊛	
m_2			⊛
m_3		*	*
m_4	⊛		

3. This is the first case of the simplification algorithm.
4. f has a unique disjunctive simplified form obtained as the disjunction of all the maximal monoms:

$$f^S(x_1, x_2, x_3) = max_1 \vee max_2 \vee max_3 = x_1\bar{x}_2\bar{x}_3 \vee \bar{x}_1x_3 \vee \bar{x}_1x_2$$

Example 7.23.

Let f be a Boolean function of 3 variables given by the following table.

x_1	x_2	x_3	$f(x_1, x_2, x_3)$
0	0	0	1
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	1

The disjunctive canonical form is:

$$f(x_1, x_2, x_3) = \bar{x}_1\bar{x}_2\bar{x}_3 \vee \bar{x}_1\bar{x}_2x_3 \vee \bar{x}_1x_2x_3 \vee x_1\bar{x}_2\bar{x}_3 \vee x_1x_2\bar{x}_3 \vee x_1x_2x_3 =$$
$$= m_0 \vee m_1 \vee m_3 \vee m_4 \vee m_6 \vee m_7$$

$S_f = \{(0,0,0), (0,0,1), (0,1,1), (1,0,0), (1,1,0), (1,1,1)\}$

The ascending order, with respect to the number of "1" values in each triplet, of the support of f is $S_f = \{(0,0,0), (1,0,0), (0,0,1), (1,1,0), (0,1,1), (1,1,1)\}$

Representation

Group		x_1	x_2	x_3	
I	√	0	0	0	m_0
II	√	1	0	0	m_4
	√	0	0	1	m_1
III	√	1	1	0	m_6
	√	0	1	1	m_3
IV	√	1	1	1	m_7

Simple factorizations

V=I+II	–	0	0	$m_0 \vee m_4 = \bar{x}_2\bar{x}_3 = max_1$
	0	0	–	$m_0 \vee m_1 = \bar{x}_1\bar{x}_2 = max_2$
VI=II+III	1	–	0	$m_4 \vee m_6 = x_1\bar{x}_3 = max_3$
	0	–	1	$m_1 \vee m_3 = \bar{x}_1x_3 = max_4$
VII=III+IV	1	1	–	$m_6 \vee m_7 = x_1x_2 = max_5$
	–	1	1	$m_3 \vee m_7 = x_2x_3 = max_6$

1. Only simple factorizations can be applied. The neighbor groups (V, VI) and (VI,VII) do not have neighbor monoms to be used in a double factorization. The unmarked rows correspond to the maximal monoms:
   $M(f) = \{\bar{x}_2\bar{x}_3, \bar{x}_1\bar{x}_2, x_1\bar{x}_3, \bar{x}_1x_3, x_1x_2, x_2x_3\} = \{max_1, max_2, max_3, max_4, max_5, max_6\}$

We build the following correspondence tableau:

minterms \ max.monoms	max_1	max_2	max_3	max_4	max_5	max_6
m_0	*	❋				
m_4	*		❋			
m_1		❋		*		
m_6			❋		*	
m_3				*		❋
m_7					*	❋

2. There is no '*' which is unique on a certain row, meaning that all the minterms are covered by two maximal monoms, so there are no central monoms: $C(f) = \varnothing$.
3. This is the third case of the simplification algorithm.
4. There are two simplified forms of f, obtained by covering all the minterms with a minimum number of maximal monoms, without overlaps:
   $f_1^S(x_1, x_2, x_3) = max_1 \vee max_4 \vee max_5 = \bar{x}_2\bar{x}_3 \vee \bar{x}_1x_3 \vee x_1x_2$ (*)
   $f_2^S(x_1, x_2, x_3) = max_2 \vee max_3 \vee max_6 = \bar{x}_1\bar{x}_2 \vee x_1\bar{x}_3 \vee x_2x_3$ (❋)

Example 7.24.
Simplify the following Boolean function of 4 variables:

$f(x_1, x_2, x_3, x_4) = x_1\bar{x}_2x_3\bar{x}_4 \vee \bar{x}_1x_2x_3 \vee \bar{x}_1\bar{x}_2x_3x_4 \vee x_1\bar{x}_2\bar{x}_3x_4 \vee \bar{x}_1\bar{x}_3x_4 \vee \bar{x}_1x_3 \vee \bar{x}_2\bar{x}_3x_4$

The support set of f is obtained as the reunion of the supports of the monoms from the function's expression, which is a disjunctive form but not a canonical one:

S_f ={(1,0,1,0)} ∪ {(0,1,1,0), (0,1,1,1)} ∪ {(0,1,0,1)} ∪ {(1,1,0,0)} ∪
∪ {(0,0,0,1), (0,1,0,1)} ∪ {(0,0,1,0), (0,0,1,1), (0,1,1,0), (0,1,1,1)} ∪
∪ {(0,0,0,1), (1,0,0,1)} =
= {(0,1,1,1), (1,0,1,0), (0,1,1,0), (1,1,0,0), (0,1,0,1), (1,0,0,1), (0,0,1,1),
(0,0,0,1), (0,0,1,0)}

From S_f the disjunctive canonical form of f is written as follows:
$f(x_1, x_2, x_3, x_4) = m_7 \vee m_{10} \vee m_6 \vee m_{12} \vee m_5 \vee m_9 \vee m_3 \vee m_1 \vee m_2$

1. The support of f is represented in descending order, with respect to the number of "1" values in each 4-uple and then factorizations are applied as can be seen in the following tableau.

	Group		x_1	x_2	x_3	x_4	
Representation	I	√	0	1	1	1	m_7
		√	1	0	1	0	m_{10}
		√	0	1	1	0	m_6
	II		1	1	0	0	$m_{12} = x_1x_2\bar{x}_3\bar{x}_4 = max_1$
		√	0	1	0	1	m_5
		√	1	0	0	1	m_9
		√	0	0	1	1	m_3
	III	√	0	0	0	1	m_1
		√	0	0	1	0	m_2
Simple factorization	IV=I+II	√	0	1	1	–	$m_7 \vee m_6$
		√	0	1	–	1	$m_7 \vee m_5$
		√	0	–	1	1	$m_7 \vee m_3$
	V=II+III		–	0	1	0	$m_{10} \vee m_2 = \bar{x}_2x_3\bar{x}_4 = max_2$
		√	0	–	1	0	$m_6 \vee m_2$
		√	0	–	0	1	$m_5 \vee m_1$
			–	0	0	1	$m_9 \vee m_1 = \bar{x}_2\bar{x}_3x_4 = max_3$
		√	0	0	–	1	$m_3 \vee m_1$
		√	0	0	1	–	$m_3 \vee m_2$
Double factorization	VI=IV+V		0	–	1	–	$m_7 \vee m_6 \vee m_3 \vee m_2 = \bar{x}_1x_3 = max_4$
			0	–	–	1	$m_7 \vee m_5 \vee m_3 \vee m_1 = \bar{x}_1x_4 = max_5$

The monoms from the groups IV and V are obtained by applying simple factorizations and further the neighbor monoms from these groups are used in double factorizations obtaining group VI.
There is an isolated minterm (without neighbors): m_{12}, which is a maximal monom.

$M(f) = \{max_1, max_2, max_3, max_4, max_5\} = \{ x_1\bar{x}_2\bar{x}_3\bar{x}_4, \bar{x}_2\bar{x}_3\bar{x}_4, \bar{x}_2\bar{x}_3x_4, \bar{x}_1\bar{x}_3, \bar{x}_1x_4\}$

Note that a result of a double factorization is obtained twice as follows:

$max_4 = (m_7 \vee m_6) \vee (m_3 \vee m_2) = (m_7 \vee m_3) \vee (m_6 \vee m_2)$ and

$max_5 = (m_7 \vee m_5) \vee (m_3 \vee m_1) = (m_7 \vee m_3) \vee (m_5 \vee m_1)$

2. We identify the central monoms from the correspondence tableau built below.

max.monoms / minterms	max_1	max_2	max_3	max_4	max_5
m_7				*	*
m_{10}		⊛			
m_6				⊛	
m_{12}	⊛				
m_5					⊛
m_9			⊛		
m_3				*	*
m_1			*		*
m_2		*		*	

We circled the '*' which are unique on their rows. All the columns have a ⊛, thus all the maximal monoms are central monoms: $M(f) = C(f)$.

3. According to the first case of the simplification algorithm there is a unique disjunctive simplified form of the initial function, obtained as the disjunction of all the central monoms:

4. $f^S(x_1, x_2, x_3, x_4) = max_1 \vee max_2 \vee max_3 \vee max_4 \vee max_5 =$

$= x_1\bar{x}_2\bar{x}_3\bar{x}_4 \vee \bar{x}_2\bar{x}_3\bar{x}_4 \vee \bar{x}_2\bar{x}_3x_4 \vee \bar{x}_1\bar{x}_3 \vee \bar{x}_1x_4$

Example 7.25.

Simplify the following Boolean function of 4 variables:

$f(x_1, x_2, x_3, x_4) = m_4 \vee m_6 \vee m_7 \vee m_8 \vee m_9 \vee m_{10} \vee m_{11} \vee m_{12}$.

The support of f is represented in a descending order, with respect to the number of "1" values in each 4-uple.

$S_f = \{(0,1,1,1),(1,0,1,1),(0,1,1,0),(1,0,0,1),(1,0,1,0),(1,1,0,0),(0,1,0,0),(1,0,0,0)\}$

1. **Factorization** process: there are 4 simple factorizations and one double factorization according to the following table.
The set of the maximal monoms is:

$M(f) = \{max_1, max_2, max_3, max_4, max_5\} = \{\bar{x}_1x_2x_3, \bar{x}_1x_2\bar{x}_4, x_2\bar{x}_3\bar{x}_4, x_1\bar{x}_3\bar{x}_4, x_1\bar{x}_2\}$

	Group		x_1	x_2	x_3	x_4	
Representation	I	√	0	1	1	1	m_7
		√	1	0	1	1	m_{11}
	II	√	0	1	1	0	m_6
		√	1	0	0	1	m_9
		√	1	0	1	0	m_{10}
		√	1	1	0	0	m_{12}
	III	√	0	1	0	0	m_4
		√	1	0	0	0	m_8
Simple factorization			0	1	1	–	$m_7 \vee m_6 = \bar{x}_1 x_2 x_3 = max_1$
	IV=I+II	√	1	0	–	1	$m_{11} \vee m_9$
		√	1	0	1	–	$m_{11} \vee m_{10}$
			0	1	–	0	$m_6 \vee m_4 = \bar{x}_1 x_2 \bar{x}_4 = max_2$
		√	1	0	0	–	$m_9 \vee m_8$
	V=II+III	√	1	0	–	0	$m_{10} \vee m_8$
			–	1	0	0	$m_{12} \vee m_4 = x_2 \bar{x}_3 \bar{x}_4 = max_3$
			1	–	0	0	$m_{12} \vee m_8 = x_1 \bar{x}_3 \bar{x}_4 = max_4$
Double factorization	VI=IV+V		1	0	–	–	$m_{11} \vee m_9 \vee m_{10} \vee m_8 = x_1 \bar{x}_2 = max_5$

2. Identification of the central monoms:

minterms \ max.monoms	max_1	max_2	max_3	max_4	max_5
m_7	⊛				
m_{11}					⊛
m_6	*	*			
m_9					⊛
m_{10}					⊛
m_{12}			*	*	
m_4		*	*		
m_8				*	*

The shaded columns (max_1 and max_5) correspond to the central monoms because they have at least one '*', which is unique on its row.

max_1 has on the first row a '*', which is unique on its row, meaning that m_7 is covered only by max_1, so max_1 must belong to all simplified forms of f.

The minterms m_{11}, m_9, m_{10} are covered only by max_5 (see the second, fourth and fifth rows), therefore max_5 is a central monom.

$C(f) = \{ max_1, max_5 \}$

3. $M(f) \neq C(f)$ and $C(f) \neq \emptyset$, the second case of the simplification algorithm.

$g(x_1, x_2, x_3, x_4) \overset{not.}{=} max_1 \vee max_5 = \bar{x}_1 x_2 x_3 \vee x_1 \bar{x}_2$.

4. We also shade the rows (minterms) covered by the central monoms. To identify the simplified forms of f we have to choose functions $h(x_1, x_2, x_3, x_4)$, as simple as possible, such that they cover the minterms m_{12} and m_4 uncovered by $g(x_1, x_2, x_3, x_4)$.

We choose the maximal monom corresponding to the column with the maximum number of unshaded '*' : $max_3 = x_2 \bar{x}_3 \bar{x}_4$.

Note that $h(x_1, x_2, x_3, x_4) = max_3$, covers $S_f \setminus S_g$.

There is a unique simplified form of f:

$f^s(x_1, x_2, x_3, x_4) = g(x_1, x_2, x_3, x_4) \vee h(x_1, x_2, x_3, x_4) = \bar{x}_1 x_2 x_3 \vee x_1 \bar{x}_2 \vee x_2 \bar{x}_3 \bar{x}_4$

7.7. Moisil's simplification method

Grigore Constantin Moisil was a Romanian mathematician, computer pioneer. His research was mainly in the fields of mathematical logic, algebraic logic and differential equations. Moisil is considered to be the father of Romanian informatics with his invention of three-stable circuits.

Moisil's method uses propositional logic to obtain the simplified forms of a Boolean function from the set of maximal monoms. The set of the maximal monoms are calculated using one of the previous methods (Veitch-Karnaugh diagrams, Quine's method). In the next example we follow the approach proposed in paper [62].

Example 7.26.

We apply Moisil's method to the function from Example 7.22:

$f(x_1, x_2, x_3) = \bar{x}_1 \bar{x}_2 \bar{x}_3 \vee \bar{x}_1 \bar{x}_2 x_3 \vee \bar{x}_1 x_2 x_3 \vee x_1 \bar{x}_2 x_3 \vee x_1 x_2 \bar{x}_3 \vee x_1 x_2 x_3 =$

$= m_0 \vee m_1 \vee m_3 \vee m_4 \vee m_6 \vee m_7$ - *DCF* (disjunctive canonical form)

$S_f = \{(0,0,0), (0,0,1), (0,1,1), (1,0,0), (1,1,0), (1,1,1)\}$

The maximal monoms were obtained using Quine's method by applying six double factorizations:

$max_1 = m_0 \vee m_4 = \bar{x}_2 \bar{x}_3$ $\qquad$ $max_2 = m_0 \vee m_1 = \bar{x}_1 \bar{x}_2$

$max_3 = m_4 \vee m_6 = x_1 \bar{x}_3$ $\qquad$ $max_4 = m_1 \vee m_3 = \bar{x}_1 x_3$

$max_5 = m_6 \vee m_7 = x_1 x_2$ $\qquad$ $max_6 = m_3 \vee m_7 = x_2 x_3$

$M(f) = \{\bar{x}_2 \bar{x}_3, \bar{x}_1 \bar{x}_2, x_1 \bar{x}_3, \bar{x}_1 x_3, x_1 x_2, x_2 x_3\} = \{max_1, max_2, max_3, max_4, max_5, max_6\}$.

Because the simplified forms of a function contain only maximal monoms we consider the following propositional sentences:

p_i: " max_i belongs to a simplified form of f", $i=1,2,..,6$.

Each minterm from the function's expression must be covered by a maximal monom in a simplified form, therefore according to the result of the factorization process we have the following true sentences:

„m_0 is covered by max_1 or by max_2", translated as: $p_1 \vee p_2 \equiv T$

„m_4 is covered by max_1 or by max_3", translated as: $p_1 \vee p_3 \equiv T$

„m_1 is covered by max_2 or by max_4", translated as: $p_2 \vee p_4 \equiv T$

„m_6 is covered by max_3 or by max_5", translated as: $p_3 \vee p_5 \equiv T$

„m_3 is covered by max_4 or by max_6", translated as: $p_4 \vee p_6 \equiv T$

„m_7 is covered by max_5 or by max_6", translated as: $p_5 \vee p_6 \equiv T$

In order to obtain a simplified form of f, all the minterms from the function's expression must be covered by a minimum number of maximal monoms, with a minimum number of overlaps. This statement is modeled by the following propositional formula obtained as a conjunction of all the previous true sentences:

$$(p_1 \vee p_2) \wedge (p_1 \vee p_3) \wedge (p_2 \vee p_4) \wedge (p_3 \vee p_5) \wedge (p_4 \vee p_6) \wedge (p_5 \vee p_6) \equiv T$$

The formula is in conjunctive normal form (*CNF* – a conjunction of clauses) and we have to transform it into its equivalent disjunctive normal form (*DNF*).

We transform syntactically the previous formula:

$$T \equiv (p_1 \vee (p_2 \wedge p_3)) \wedge (p_4 \vee (p_2 \wedge p_6)) \wedge (p_5 \vee (p_3 \wedge p_6))$$

Apply distributive laws:

$$\begin{aligned} T \equiv\ & (p_1 \wedge p_4 \wedge p_5) \vee (p_1 \wedge p_4 \wedge p_3 \wedge p_6) \vee (p_1 \wedge p_2 \wedge p_6 \wedge p_5) \vee \\ & \vee (p_1 \wedge p_2 \wedge p_6 \wedge p_3 \wedge p_6) \vee (p_2 \wedge p_3 \wedge p_4 \wedge p_5) \vee \\ & \vee (p_2 \wedge p_3 \wedge p_4 \wedge p_3 \wedge p_6) \vee (p_2 \wedge p_3 \wedge p_2 \wedge p_6 \wedge p_5) \vee \\ & \vee (p_2 \wedge p_3 \wedge p_2 \wedge p_6 \wedge p_3 \wedge p_6) \end{aligned}$$

Apply idempotency:

$$\begin{aligned} T \equiv\ & (p_1 \wedge p_4 \wedge p_5) \vee (p_1 \wedge p_4 \wedge p_3 \wedge p_6) \vee (p_1 \wedge p_2 \wedge p_6 \wedge p_5) \vee \\ & \vee (p_1 \wedge p_2 \wedge p_3 \wedge p_6) \vee (p_2 \wedge p_3 \wedge p_4 \wedge p_5) \vee (p_2 \wedge p_3 \wedge p_4 \wedge p_3 \wedge p_6) \\ & \vee (p_2 \wedge p_3 \wedge p_6 \wedge p_5) \vee (p_2 \wedge p_3 \wedge p_6) \end{aligned}$$

Apply absorption:

$$\begin{aligned} T \equiv\ & \mathbf{(p_1 \wedge p_4 \wedge p_5)} \vee (p_1 \wedge p_4 \wedge p_3 \wedge p_6) \vee (p_1 \wedge p_2 \wedge p_6 \wedge p_5) \vee \\ & \vee (p_2 \wedge p_3 \wedge p_4 \wedge p_5) \vee \mathbf{(p_2 \wedge p_3 \wedge p_6)} - \textit{DNF} \text{ with 5 cubes} \end{aligned}$$

A *DNF* (disjunction of cubes) is true if at least one of its cubes is true.

From *DNF* we consider the cubes with the minimum number of propositional variables (marked in bold) and correspondingly we obtain the simplified forms of f.

For the cube $p_1 \wedge p_4 \wedge p_5 \equiv T$ the corresponding simplified form is:

$$f_1^S(x_1, x_2, x_3) = max_1 \vee max_4 \vee max_5 = \bar{x}_2\bar{x}_3 \vee \bar{x}_1x_3 \vee x_1x_2.$$

For the cube $p_2 \wedge p_3 \wedge p_6 \equiv T$, the corresponding simplified form is:

$$f_2^S(x_1, x_2, x_3) = max_2 \vee max_3 \vee max_6 = \bar{x}_1\bar{x}_2 \vee x_1\bar{x}_3 \vee x_2x_3.$$

If we take the other cubes from *DNF*, there will be overlaps: a minterm will be covered by two maximal monoms and these forms do not correspond to the simplest forms of *f*.

7.8. Application

A direct application of Boolean functions simplification is automatic simplification of conditional expressions. In the beginning, the simple conditional expressions must be identified. These simple conditional expressions are connected by logical operations. They are denoted by variables and thus a logical expression that can be seen as a Boolean function is obtained. The simplification process is easily implemented according to the steps described in the following.

- **Step1:** The conditional expression is transformed from current form into disjunctive normal form, using the appropriate algorithm from the first chapter.
- **Step2:** To obtain the disjunctive canonical form of the Boolean function, attach to each cube all missing variables, both simple and negated.
- **Step3:** The Quine's method is applied to perform factorizations and to obtain the maximal monoms.
- **Step4:** The simplified form is obtained using Moisil's method.

Example 7.27.

We consider the following Pascal conditional expression:

$$((x < y)\,and\,(y < z)\,and\,(z < 5))\,or\,((x < y)\,and\,(y >= z)\,and\,(z < 5))$$

Note that the expression $y >= z$ can be written as: $\neg(y < z)$.

So, the initial expression becomes:

$$((x < y)\,and\,(y < z)\,and\,(z < 5))\,or\,((x < y)\,and\,(not(y < z))\,and\,(z < 5))$$

In order to track more easily the simplification process, we denote the simple conditional expressions with variables, as follows:

- $x < y$ by x_1, $y < z$ by x_2, $z < 5$ by x_3.

The following Boolean function is obtained:

$$f(x_1, x_2, x_3) = (x_1 \wedge x_2 \wedge x_3) \vee (x_1 \wedge \bar{x}_2 \wedge x_3)$$

The simplified form of f function is: $f^S(x_1, x_2, x_3) = x_1 \wedge x_3$

Thus, the initial expression was reduced to $((x < y)\,and\,(z < 5))$.

An application which implements the above algorithm was presented at the ICMI2006 conference, see [39].

7.9. DCF versus CCF in simplification

All three simplification methods presented in the previous sections use the disjunctive canonical form (*DCF*) of the initial function. The simplification purpose is to obtain the simplest form of the function. This form can be a conjunctive or a disjunctive form. But by negation, the *CCF* of the function is transformed into *DCF* of the negated function, which can be simplified using the previous presented methods and then the negated simplified form can be changed back into a conjunctive simplified form of the initial function.

Example 7.28.
Let us consider the following function given in its conjunctive canonical form:

$$f(x_1, x_2, x_3, x_4) = M_0 \wedge M_4 ,$$

$$\bar{f}(x_1, x_2, x_3, x_4) = m_0 \vee m_4 = \bar{x}_1\bar{x}_2\bar{x}_3\bar{x}_4 \vee \bar{x}_1x_2\bar{x}_3\bar{x}_4 = \bar{x}_1\bar{x}_3\bar{x}_4 (x_2 \vee \bar{x}_2) = \bar{x}_1\bar{x}_3\bar{x}_4$$

– the simplified form

By negating the simplified form obtained previously, the simplified form of the initial function f is obtained: $f^S(x_1, x_2, x_3, x_4) = x_1 \vee x_3 \vee x_4$

Sometimes, the simplified form of a Boolean function is a conjunctive one, not a disjunctive form, as can be seen from the following example. Thus, it is recommended to simplify both: the function and its negation, and in the end to choose the simplest form.

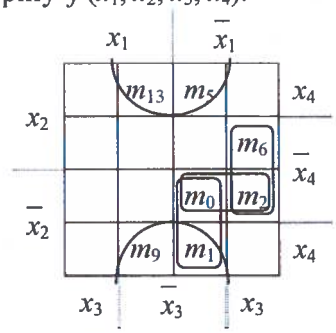
Example 7.29.
Determine the simplest equivalent form of the following Boolean function:

$$f(x_1, x_2, x_3, x_4) = x_1\bar{x}_2\bar{x}_3x_4 \vee \bar{x}_1\bar{x}_2\bar{x}_3x_4 \vee \bar{x}_1x_2\bar{x}_3\bar{x}_4 \vee \bar{x}_1\bar{x}_2\bar{x}_3\bar{x}_4 \vee \bar{x}_1\bar{x}_2\bar{x}_3\bar{x}_4 \vee$$
$$\vee x_1\bar{x}_2\bar{x}_3\bar{x}_4 \vee \bar{x}_1\bar{x}_2\bar{x}_3x_4 =$$
$$= m_{13} \vee m_5 \vee m_6 \vee m_0 \vee m_2 \vee m_9 \vee m_1 \ (DCF)$$
$$= M_{15} \wedge M_7 \wedge M_{14} \wedge M_{12} \wedge M_4 \wedge M_{10} \wedge M_8 \wedge M_{11} \wedge M_3 \ (CCF)$$

The negation function of f is:

$$\bar{f}(x_1, x_2, x_3, x_4) = m_{15} \vee m_7 \vee m_{14} \vee m_{12} \vee m_4 \vee m_{10} \vee m_8 \vee m_{11} \vee m_3$$

We use the Veitch simplification method in order to obtain the simplified form for both f and $\bar{f}$. First we simplify $f(x_1, x_2, x_3, x_4)$:



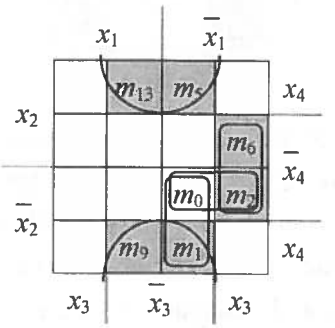
1. The set of the maximal monoms is:

$M(f) = \{\bar{x}_3x_4, \bar{x}_1\bar{x}_2\bar{x}_3, \bar{x}_1\bar{x}_2\bar{x}_4, \bar{x}_1x_3\bar{x}_4\}$

$max_1 = \bar{x}_3x_4 = m_{13} \vee m_5 \vee m_9 \vee m_1$, $max_2 = \bar{x}_1\bar{x}_2\bar{x}_3 = m_1 \vee m_0$

$max_3 = \bar{x}_1\bar{x}_2\bar{x}_4 = m_0 \vee m_2$ $max_4 = \bar{x}_1x_3\bar{x}_4 = m_6 \vee m_2$

2. $C(f) = \{ max_1=\bar{x}_3x_4, max_4=\bar{x}_1x_3\bar{x}_4\}$



3. This is the **second case** of the simplification algorithm.
4. There is only one uncovered minterm, m_0, which can be covered in two ways, with the same complexity, using max_2 or max_3.

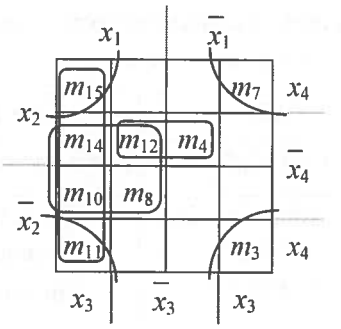
$f_1^S(x_1, x_2, x_3, x_4) = \bar{x}_3x_4 \vee \bar{x}_1x_3\bar{x}_4 \vee \bar{x}_1\bar{x}_2\bar{x}_3$

$f_2^S(x_1, x_2, x_3, x_4) = \bar{x}_3x_4 \vee \bar{x}_1x_3\bar{x}_4 \vee \bar{x}_1\bar{x}_2\bar{x}_4$

Both *disjunctive simplified forms* of the function contain 8 occurrences of the variables, 7 binary operations and 6 unary operations.

In the following we simplify the function:

$\bar{f}(x_1, x_2, x_3, x_4) = m_{15} \vee m_7 \vee m_{14} \vee m_{12} \vee m_4 \vee m_{10} \vee m_8 \vee m_{11} \vee m_3$

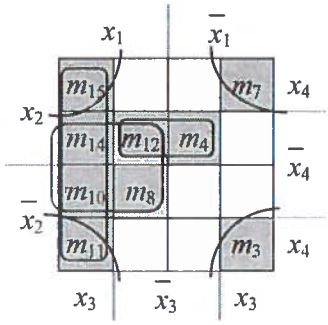


1. The set of maximal monoms is:

$M(\bar{f}) = \{x_1x_3, x_1\bar{x}_4, x_2\bar{x}_3\bar{x}_4, x_3x_4\}$

$max'_1 = x_1x_3 = m_{15} \vee m_{14} \vee m_{10} \vee m_{11}$ $max'_2 = x_1\bar{x}_4 = m_{14} \vee m_{12} \vee m_{10} \vee m_8$

$max'_3 = x_2\bar{x}_3\bar{x}_4 = m_{12} \vee m_4$ $max'_4 = x_3x_4 = m_{15} \vee m_{11} \vee m_7 \vee m_3$



2. $C(\bar{f}) = \{max'_2 = x_1\bar{x}_4,\ max'_3 = x_2\bar{x}_3\bar{x}_4,\ max'_4 = x_3x_4\}$
3. This is the **second case** of the simplification algorithm, but as can be seen in the above diagram, the central monoms cover all the minters of $\bar{f}$.
4. Thus, there is a unique simplified form of the function $\bar{f}$, obtained as a disjunction of all the central monoms: $\bar{f}^S(x_1, x_2, x_3, x_4) = x_1\bar{x}_4 \vee x_2\bar{x}_3\bar{x}_4 \vee x_3x_4$

Negating again we obtain the *conjunctive simplified form* of f:

$f^{CS}(x_1, x_2, x_3, x_4) = (\bar{x}_1 \vee x_4) \wedge (\bar{x}_2 \vee x_3 \vee x_4) \wedge (\bar{x}_3 \vee \bar{x}_4)$.

Note that this new simplified form of the initial function contains only 7 variable occurrences, 6 binary operatotions and 4 unary operatotions, so is simpler than the disjunctive simplified forms obtained earlier.

We conclude that the simplest form of f is the conjunctive form, obtained by negating the simplified form of $\bar{f}$:

$f^{CS}(x_1, x_2, x_3, x_4) = (\bar{x}_1 \vee x_4) \wedge (\bar{x}_2 \vee x_3 \vee x_4) \wedge (\bar{x}_3 \vee \bar{x}_4)$.

Conclusion:

In order to obtain the simplest equivalent form (conjunctive or disjunctive) of a Boolean function f we have two possibilities:

see Example 7.10.	see Example 7.19.
- simplify $DCF(f)$ => disjunctive simplified form: f^S	
- simplify $CCF(f)$ using a dual simplification algorithm => conjunctive simplified form: f^{CS}	- $CCF(f)$ is negated obtaining a disjunctive canonical form - simplify $\overline{CCF(f)}$=> disjunctive simplified form: $\bar{f}^S$ - $\bar{f}^S$ is negated and we obtain the conjunctive simplified form f^{CS}
- from f^S and f^{CS} choose the form with the fewest occurrences of variables and connectives	

7.10. Exercises

Exercise 7.1.

For the following Boolean functions of 3 variables, given by their tables of values, write the corresponding *disjunctive canonical form* (*DCF*) and *conjunctive canonical form* (*CCF*). Using Veitch diagrams simplify the functions.

x	y	z	f_1	f_2	f_3	f_4	f_5	f_6	f_7	f_8
0	0	0	0	1	1	1	0	1	0	1
0	0	1	1	1	1	0	1	0	0	0
0	1	0	0	0	1	0	1	1	1	1
0	1	1	1	0	0	1	0	1	0	0
1	0	0	1	0	1	0	0	0	1	0
1	0	1	0	1	0	0	1	0	1	1
1	1	0	0	0	0	1	1	1	0	1
1	1	1	1	1	0	1	0	0	1	0

Exercise 7.2.

Simplify the following Boolean functions of 4 variables using Veitch diagrams.

1. $f_1(x_1,x_2,x_3,x_4)= x_1\bar{x}_2\bar{x}_3\bar{x}_4 \vee x_1x_2\bar{x}_3\bar{x}_4 \vee x_1x_2x_3\bar{x}_4 \vee x_1\bar{x}_2x_3x_4 \vee \bar{x}_1x_2x_3x_4 \vee \bar{x}_1x_2\bar{x}_3\bar{x}_4$
$\vee \bar{x}_1\bar{x}_2x_3x_4 \vee \bar{x}_1x_2x_3x_4 \vee \bar{x}_1\bar{x}_2x_3\bar{x}_4;$
2. $f_2(x_1,x_2,x_3,x_4)= x_1\bar{x}_2\bar{x}_3\bar{x}_4 \vee x_1x_2\bar{x}_3\bar{x}_4 \vee x_1\bar{x}_2x_3\bar{x}_4 \vee x_1x_2x_3\bar{x}_4 \vee \bar{x}_1x_2x_3\bar{x}_4 \vee \bar{x}_1\bar{x}_2\bar{x}_3\bar{x}_4$
$\vee \bar{x}_1\bar{x}_2x_3x_4 \vee \bar{x}_1x_2x_3x_4 \vee \bar{x}_1\bar{x}_2x_3\bar{x}_4;$
3. $f_3(x_1,x_2,x_3,x_4)= \bar{x}_1x_2\bar{x}_3\bar{x}_4 \vee \bar{x}_1x_2x_3\bar{x}_4 \vee \bar{x}_1x_2\bar{x}_3x_4 \vee x_1x_2\bar{x}_3x_4 \vee x_1\bar{x}_2x_3x_4 \vee \bar{x}_1x_2x_3x_4$
$\vee x_1\bar{x}_2\bar{x}_3x_4 \vee x_1x_2x_3\bar{x}_4 \vee x_1\bar{x}_2x_3\bar{x}_4;$
4. $f_4(x_1,x_2,x_3,x_4)= \bar{x}_1\bar{x}_2\bar{x}_3\bar{x}_4 \vee x_1\bar{x}_2\bar{x}_3\bar{x}_4 \vee x_1x_2\bar{x}_3\bar{x}_4 \vee \bar{x}_1x_2x_3\bar{x}_4 \vee x_1\bar{x}_2x_3x_4 \vee \bar{x}_1x_2\bar{x}_3\bar{x}_4$
$\vee \bar{x}_1\bar{x}_2x_3x_4 \vee x_1x_2x_3x_4 \vee \bar{x}_1\bar{x}_2x_3\bar{x}_4;$
5. $f_5(x_1,x_2,x_3,x_4)= x_1\bar{x}_2\bar{x}_3\bar{x}_4 \vee x_1x_2\bar{x}_3\bar{x}_4 \vee x_1x_2x_3\bar{x}_4 \vee \bar{x}_1x_2\bar{x}_3\bar{x}_4 \vee \bar{x}_1x_2x_3\bar{x}_4 \vee x_1\bar{x}_2x_3x_4$
$\vee \bar{x}_1\bar{x}_2\bar{x}_3x_4 \vee \bar{x}_1x_2x_3x_4 \vee x_1\bar{x}_2x_3\bar{x}_4;$
6. $f_6(x_1,x_2,x_3,x_4)= x_1\bar{x}_2\bar{x}_3\bar{x}_4 \vee \bar{x}_1x_2\bar{x}_3\bar{x}_4 \vee \bar{x}_1\bar{x}_2x_3\bar{x}_4 \vee x_1x_2x_3\bar{x}_4 \vee \bar{x}_1x_2\bar{x}_3x_4 \vee x_1\bar{x}_2x_3x_4$
$\vee \bar{x}_1\bar{x}_2\bar{x}_3x_4 \vee \bar{x}_1x_2x_3x_4 \vee x_1\bar{x}_2x_3\bar{x}_4;$
7. $f_7(x_1,x_2,x_3,x_4)= x_1\bar{x}_2x_3\bar{x}_4 \vee \bar{x}_1x_2\bar{x}_3\bar{x}_4 \vee \bar{x}_1x_2x_3\bar{x}_4 \vee x_1x_2\bar{x}_3x_4 \vee x_1\bar{x}_2x_3\bar{x}_4 \vee x_1x_2x_3x_4$
$\vee x_1\bar{x}_2\bar{x}_3x_4 \vee \bar{x}_1x_2x_3x_4 \vee x_1\bar{x}_2x_3\bar{x}_4;$
8. $f_8(x_1,x_2,x_3,x_4)= x_1\bar{x}_2\bar{x}_3\bar{x}_4 \vee \bar{x}_1x_2\bar{x}_3\bar{x}_4 \vee x_1x_2\bar{x}_3\bar{x}_4 \vee \bar{x}_1\bar{x}_2x_3\bar{x}_4 \vee x_1x_2x_3\bar{x}_4 \vee x_1\bar{x}_2x_3x_4$
$\vee \bar{x}_1\bar{x}_2\bar{x}_3x_4 \vee \bar{x}_1x_2x_3x_4 \vee x_1\bar{x}_2x_3\bar{x}_4.$

Exercise 7.3.
Using Karnaugh diagrams, simplify the following Boolean functions:

1. $f_1(x_1,x_2,x_3)=\bar{x}_1(x_2 \downarrow x_3)\vee \bar{x}_1\bar{x}_2x_3 \vee \overline{(x_1 \vee (x_2 \uparrow x_3))}\vee x_1x_2\bar{x}_3$
2. $f_2(x_1,x_2,x_3)=\bar{x}_1\bar{x}_2\bar{x}_3 \vee x_1(\bar{x}_2 \downarrow x_3)\vee \overline{(x_1 \vee (x_2 \uparrow \bar{x}_3))}\vee x_1\bar{x}_2x_3$
3. $f_3(x_1,x_2,x_3)=\bar{x}_1(x_2 \downarrow x_3)\vee x_1\bar{x}_2x_3 \vee \overline{(x_1 \vee (x_2 \uparrow x_3))}\vee x_1x_2x_3$
4. $f_4(x_1,x_2,x_3)=\bar{x}_1\bar{x}_2\bar{x}_3 \vee x_1(\bar{x}_2 \downarrow x_3)\vee \overline{(x_1 \vee (\bar{x}_2 \uparrow x_3))}\vee x_1\bar{x}_2x_3$
5. $f_5(x_1,x_2,x_3)=x_1(x_2 \downarrow x_3)\vee \bar{x}_1\bar{x}_2x_3 \vee \overline{(\bar{x}_1 \vee (\bar{x}_2 \uparrow x_3))}\vee \bar{x}_1x_2\bar{x}_3$
6. $f_6(x_1,x_2,x_3)=x_1\bar{x}_2x_3 \vee \bar{x}_1(\bar{x}_2 \downarrow x_3)\vee \overline{(\bar{x}_1 \vee (x_2 \uparrow x_3))}\vee x_1\bar{x}_2\bar{x}_3$
7. $f_7(x_1,x_2,x_3)=\bar{x}_1(x_2 \downarrow x_3)\vee x_1\bar{x}_2x_3 \vee \overline{(x_1 \vee (x_2 \uparrow x_3))}\vee x_1\bar{x}_2\bar{x}_3$
8. $f_8(x_1,x_2,x_3)=\bar{x}_1\bar{x}_2x_3 \vee x_1(x_2 \downarrow x_3)\vee \overline{(x_1 \vee (x_2 \uparrow x_3))}\vee \bar{x}_1x_2\bar{x}_3$

Exercise 7.4.
Simplify the following Boolean functions of 4 variables using Karnaugh diagrams.

1. $f_1(x_1,x_2,x_3,x_4)= x_1\bar{x}_4 \vee x_1\bar{x}_2\bar{x}_3x_4 \vee \bar{x}_1x_2x_4 \vee \bar{x}_1x_3 \vee x_3\bar{x}_4$;
2. $f_2(x_1,x_2,x_3,x_4)= x_1x_2 \vee x_1\bar{x}_2\bar{x}_3x_4 \vee \bar{x}_1\bar{x}_2x_4 \vee \bar{x}_1x_3 \vee x_2x_3$;
3. $f_3(x_1,x_2,x_3,x_4)= x_1x_4 \vee x_1\bar{x}_2\bar{x}_3x_4 \vee \bar{x}_1\bar{x}_2x_4 \vee \bar{x}_1x_3 \vee x_3x_4$;
4. $f_4(x_1,x_2,x_3,x_4)= x_1\bar{x}_2 \vee x_1\bar{x}_2\bar{x}_3x_4 \vee \bar{x}_1x_2\bar{x}_4 \vee \bar{x}_1x_3 \vee \bar{x}_2x_3$;
5. $f_5(x_1,x_2,x_3,x_4)= \bar{x}_3x_4 \vee x_1\bar{x}_2\bar{x}_3x_4 \vee \bar{x}_3x_2x_4 \vee \bar{x}_1\bar{x}_3 \vee \bar{x}_1x_4$;
6. $f_6(x_1,x_2,x_3,x_4)= \bar{x}_1x_4 \vee \bar{x}_1x_2\bar{x}_3x_4 \vee x_1x_2x_4 \vee x_1\bar{x}_3 \vee \bar{x}_3\bar{x}_4$;
7. $f_7(x_1,x_2,x_3,x_4)= \bar{x}_3\bar{x}_4 \vee \bar{x}_1x_2\bar{x}_3x_4 \vee x_2x_3x_4 \vee x_1x_3 \vee x_1\bar{x}_4$;
8. $f_8(x_1,x_2,x_3,x_4)= x_3x_4 \vee x_1\bar{x}_2\bar{x}_3x_4 \vee \bar{x}_2\bar{x}_3x_4 \vee \bar{x}_1\bar{x}_3 \vee \bar{x}_1x_4$.

Exercise 7.5.
Using Quine's method, simplify the following Boolean functions given by their values 0.

1. $f_1(0,1,0)=f_1(0,1,1)=f_1(1,0,1)=0$;
2. $f_2(0,0,0)=f_2(0,0,1)=f_2(1,1,1)=0$;
3. $f_3(0,0,1)=f_3(0,1,0)=f_3(1,1,0)=0$;
4. $f_4(0,0,0)=f_4(0,1,1)=f_4(1,0,0)=0$;
5. $f_5(0,0,0)=f_5(1,1,0)=f_5(1,1,1)=0$;
6. $f_6(0,1,0)=f_6(1,0,0)=f_6(1,0,1)=0$;
7. $f_7(0,1,1)=f_7(1,0,0)=f_7(1,1,1)=0$;
8. $f_8(0,0,1)=f_8(1,0,1)=f_8(1,1,0)=0$.

Exercise 7.6.
Using Quine's method, simplify the following Boolean functions given in *DCF* (disjunction of minterms):

1. $f_1(x_1,x_2,x_3)= m_0 \vee m_3 \vee m_4 \vee m_5 \vee m_6 \vee m_7$;
2. $f_2(x_1,x_2,x_3)= m_1 \vee m_2 \vee m_4 \vee m_5 \vee m_6 \vee m_7$;
3. $f_3(x_1,x_2,x_3)= m_1 \vee m_2 \vee m_3 \vee m_4 \vee m_5 \vee m_7$;
4. $f_4(x_1,x_2,x_3)= m_0 \vee m_1 \vee m_2 \vee m_3 \vee m_5 \vee m_6$;

5. $f_5(x_1,x_2,x_3)= m_0 \vee m_1 \vee m_2 \vee m_4 \vee m_6 \vee m_7$;
6. $f_6(x_1,x_2,x_3)= m_0 \vee m_1 \vee m_3 \vee m_5 \vee m_6 \vee m_7$;
7. $f_7(x_1,x_2,x_3)= m_0 \vee m_1 \vee m_2 \vee m_3 \vee m_4 \vee m_7$;
8. $f_8(x_1,x_2,x_3)= m_0 \vee m_2 \vee m_3 \vee m_4 \vee m_5 \vee m_6$.

Exercise 7.7.

Simplify the following Boolean functions of 4 variables given by their values 1, using Quine's method.

1. $f_1(1,1,1,1) = f_1(1,1,0,1) = f_1(0,1,1,1) = f_1(1,1,0,0) = f_1(0,1,0,0) = f_1(0,0,0,0) = = f_1(0,0,0,1) = f_1(0,0,1,1) = 1$;
2. $f_2(1,1,0,1) = f_2(0,1,0,1) = f_2(0,1,0,0) = f_2(0,0,0,0) = f_2(0,0,1,0) = f_2(1,0,1,1) = = f_2(1,0,0,1) = f_2(0,0,1,1) = 1$;
3. $f_3(0,1,0,1) = f_3(0,1,0,0) = f_3(0,1,1,0) = f_3(1,0,1,0) = f_3(1,0,0,0) = f_3(0,0,1,0) = = f_3(1,0,0,1) = f_3(0,0,0,1) = 1$;
4. $f_4(0,1,0,1) = f_4(0,1,1,1) = f_4(1,1,1,0) = f_4(1,1,0,0) = f_4(0,1,1,0) = f_4(1,0,0,0) = = f_4(0,0,0,0) = f_4(0,0,0,1) = 1$;
5. $f_5(1,1,1,1) = f_5(0,1,0,1) = f_5(0,1,1,1) = f_5(1,1,1,0) = f_5(1,1,0,0) = f_5(1,0,0,0) = = f_5(1,0,0,1) = f_5(0,0,0,1) = 1$;
6. $f_6(1,1,0,1) = f_6(0,1,0,1) = f_6(0,1,1,1) = f_6(1,1,1,0) = f_6(0,1,1,0) = f_6(1,0,1,0) = = f_6(1,0,1,1) = f_6(1,0,0,1) = 1$;
7. $f_7(1,1,1,1) = f_7(1,1,0,1) = f_7(0,1,0,1) = f_7(0,1,0,0) = f_7(0,1,1,0) = f_7(0,0,1,0) = = f_7(1,0,1,1) = f_7(0,0,1,1) = 1$;
8. $f_8(1,1,1,1) = f_8(1,1,1,0) = f_8(1,1,0,0) = f_8(1,0,0,0) = f_8(0,0,0,0) = f_8(0,0,1,0) = = f_8(1,0,1,1) = f_8(0,0,1,1) = 1$.

Exercise 7.8.

Using Moisil's method simplify the following Boolean functions of 3 variables:

1. $f_1(x_1,x_2,x_3)= m_0 \vee m_1 \vee m_2 \vee m_5 \vee m_6$;
2. $f_2(x_1,x_2,x_3)= m_0 \vee m_1 \vee m_3 \vee m_4 \vee m_7$;
3. $f_3(x_1,x_2,x_3)= m_1 \vee m_2 \vee m_3 \vee m_5 \vee m_6$;
4. $f_4(x_1,x_2,x_3)= m_0 \vee m_3 \vee m_4 \vee m_5 \vee m_7$;
5. $f_5(x_1,x_2,x_3)= m_1 \vee m_2 \vee m_5 \vee m_6 \vee m_7$;
6. $f_6(x_1,x_2,x_3)= m_0 \vee m_2 \vee m_3 \vee m_4 \vee m_7$;
7. $f_7(x_1,x_2,x_3)= m_1 \vee m_2 \vee m_4 \vee m_5 \vee m_6$;
8. $f_8(x_1,x_2,x_3)= m_0 \vee m_3 \vee m_4 \vee m_6 \vee m_7$.

8. LOGIC CIRCUITS

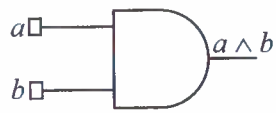
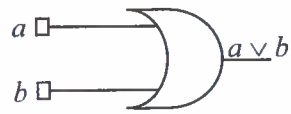
Computers and other electronic devices are composed of simple electronic circuits which are based on the two states of electricity. Digital logic circuits handle data encoded in binary form, i.e. signal that have only two values, **0** and **1**. The Boolean functions and the logic circuits describe algebraically and graphically the functionality of electronic circuits. The papers [4, 14, 25, 28, 37, 41, 62] were used as bibliographic references in this chapter.

8.1. Basic concepts

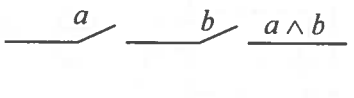
Boolean algebra provides a concise way to express the operation of a logic circuit formed by a combination of logic gates so that the output can be determined for various combinations of input values. A Boolean function models the functionality of a logic circuit. The variables used in the function's expression are represented by the inputs of the circuit. Each wire in the circuit represents some part of the expression. A **gate** takes the values from its input wires and combines them with the appropriate Boolean operation, **AND, OR, NOT**, to produce the label on its output wire. The final output of the whole circuit represents the function's expression.

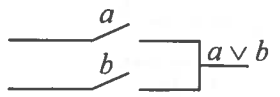
The basic elements used to build electronic circuits are called **basic gates** and they implement one of the Boolean operations: $\wedge$, $\vee$, $\bar{}$.

According to the IEEE standards, the basic gates are represented as follows:

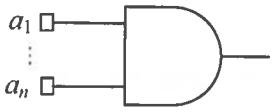
gate	**symbol for the gate**	**Boolean operation, symbols used**
AND		**conjunction** $a \wedge b$, $a \cap b$, $a * b$, ab
OR		**disjunction** $a \vee b$ $a \cup b$, $a + b$
NOT		**negation** $\neg a$, $\bar{a}$

At the physical level electronic circuits are composed of a power supply connection and contacts. A contact is described by a Boolean variable having one of the values: 0 for an **open contact** or 1 for a **closed contact**. The **AND**, **OR** gates correspond to a **series connection** and to **a parallel connection** respectively.

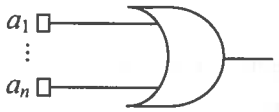




The **AND** and **OR** gates can be generalized to have more input variables:

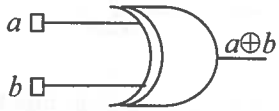
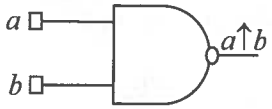
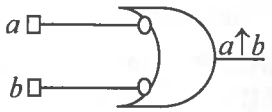
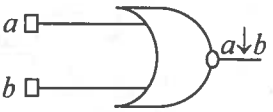
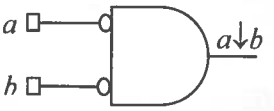


$a_1 \wedge a_2 \wedge \ldots \wedge a_n$ or $a_1\, a_2 \ldots a_n$



$a_1 \vee a_2 \vee \ldots \vee a_n$

Derived gates XOR, NAND, NOR:

gate	symbol	Boolean function	alternative symbol
XOR		$a \oplus b = \bar{a}b \vee a\bar{b}$	
NAND		$a \uparrow b = \overline{ab}$ $= \bar{a} \vee \bar{b}$	
NOR		$a \downarrow b = \overline{a \vee b}$ $= \bar{a}\bar{b}$	

The **AND, OR, NAND, NOR** gates may have any number or inputs, but practical commercial gates are mostly limited to 2, 3 and 4 inputs to fit the standard IC (integrated circuits) packages. A logic circuit can be implemented directly from the function's expression.

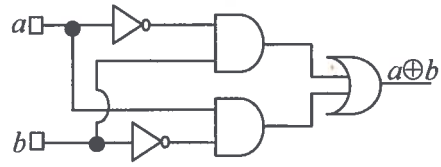
Example 8.1.

We want to build the logic circuit corresponding to "**exclusive OR**" (XOR) operation using only basic gates. The Boolean function which models its functionality is $f : \{0,1\}^2 \to \{0,1\}, f(a,b) = a \oplus b$.

Its truth table is provided below:

a	b	$a \oplus b$
0	0	0
0	1	1
1	0	1
1	1	0

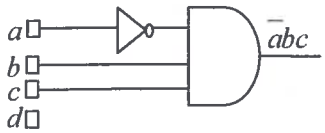
The disjunctive canonical form of $f(a,b)$ is $\bar{a}b \vee a\bar{b}$. Using only basic gates we draw the circuit diagram.



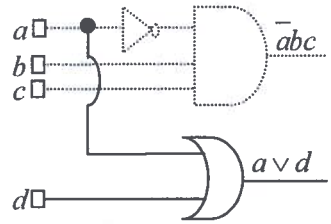
Example 8.2.

Given the Boolean function $f(a,b,c,d) = \bar{a}bc \vee \overline{(a \vee d)}$ of four variables, we draw the logic circuit having f as output.

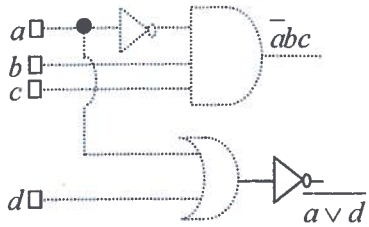
1. a NOT gate for $\bar{a}$ and the AND gate for $\bar{a}bc$ are placed:



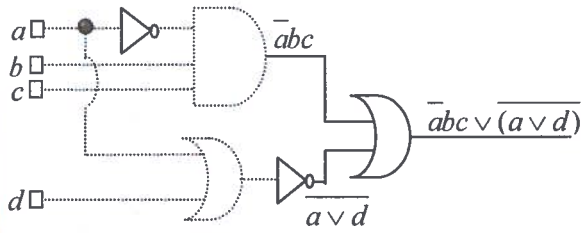
2. we draw the gate for $a \vee d$



3. the branch $a \vee d$ is negated:



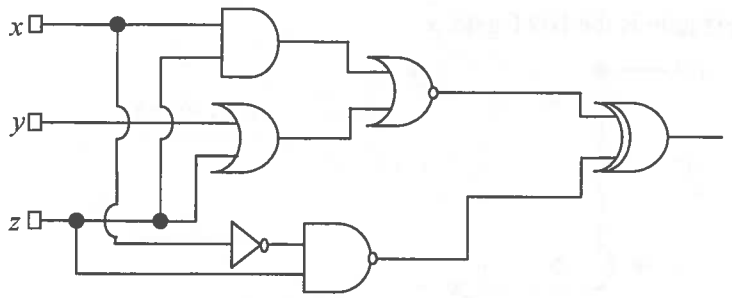
4. finally, an OR gate has as inputs the wires corresponding to $\bar{a}bc$ and $\overline{a \vee d}$:



The *logic circuit analysis* is the task of determining the functionality of a given logic circuit from its diagram. To derive the Boolean expression for a given logic circuit, begin in the diagram at the left-most inputs and work toward the final output, writing the expression for each gate.

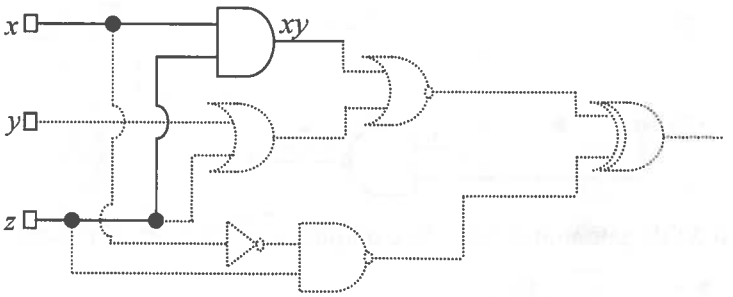
Example 8.3.

Let us consider the logic circuit depicted below, containing all types of gates, basic and derived ones.

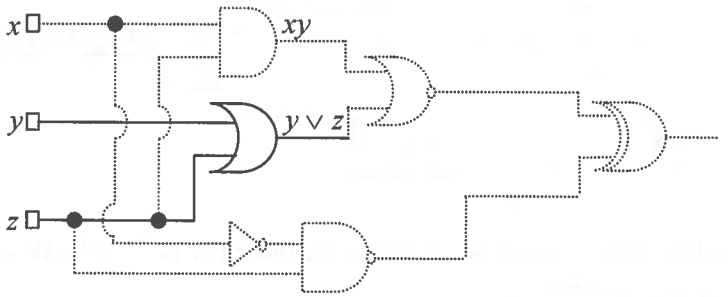


The Boolean expresion which models its functionality is deduced step by step in the following:

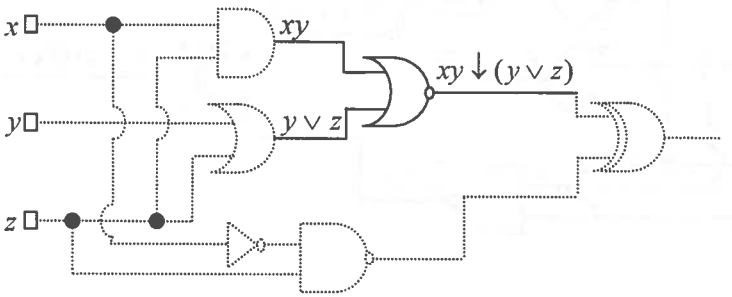
- the output of each gate is identified one at a time, starting from left to right, up to bottom.
- the first gate to be identified is the AND gate for xy:



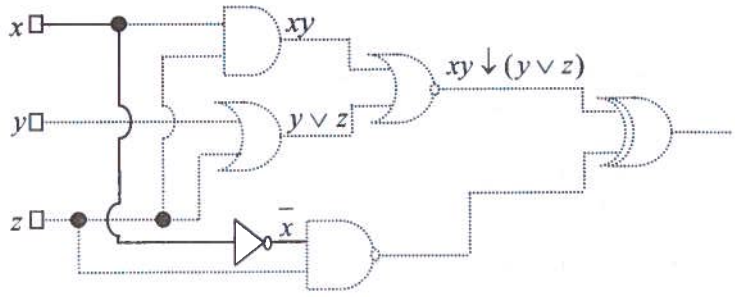
- the next gate is the OR gate for the expression $y \vee z$:



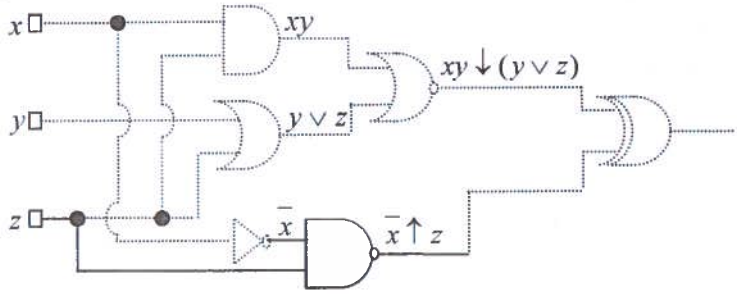
- the outputs of the previous two gates are the inputs of a NOR gate:



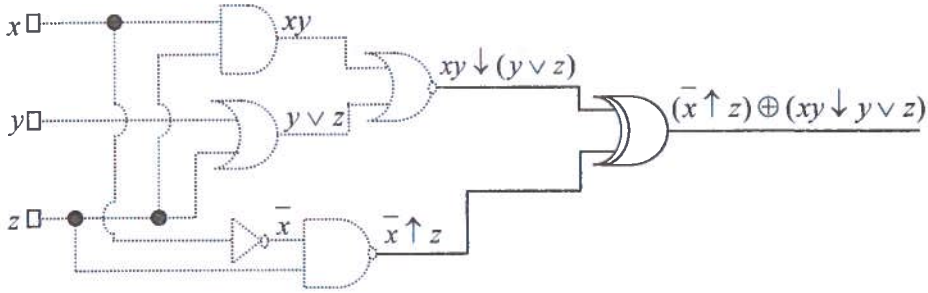
- the next gate is the NOT gate, $\bar{x}$:



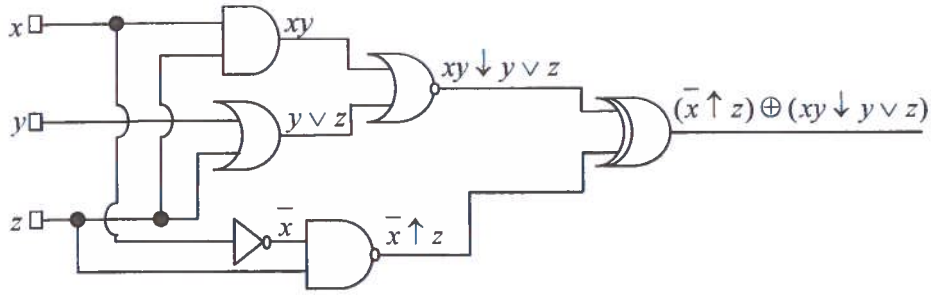
- the output of $\bar{x}$ enters into an NAND gate, $\bar{x} \uparrow z$:



- the last XOR gate unites those two outputs $(\bar{x} \uparrow z) \oplus (xy \downarrow y \vee z)$:



So, the Boolean function corresponding to the circuit is $(\bar{x} \uparrow z) \oplus (xy \downarrow y \vee z)$:



Example 8.4.

The NAND gate is a universal gate because it can be used to produce the NOT, AND, OR, and NOR functions. We express the Boolean operations: negation, conjunction, disjunction using only the NAND operation, symbolized by '$\uparrow$' and having the definition: $a \uparrow b = \overline{a \wedge b}$.

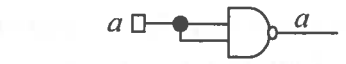
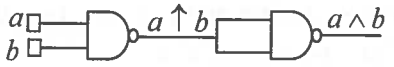
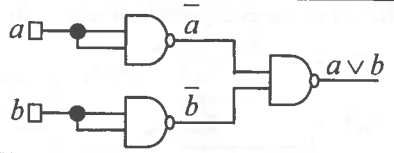
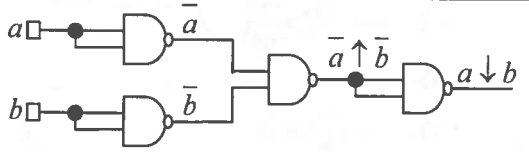
$\bar{a} = \overline{a \wedge a} = a \uparrow a$

$a \wedge b = \overline{\overline{a \wedge b}} = \overline{a \uparrow b} = (a \uparrow b) \uparrow (a \uparrow b)$

$a \vee b = \overline{\bar{a} \wedge \bar{b}} = \bar{a} \uparrow \bar{b} = (a \uparrow a) \uparrow (b \uparrow b)$

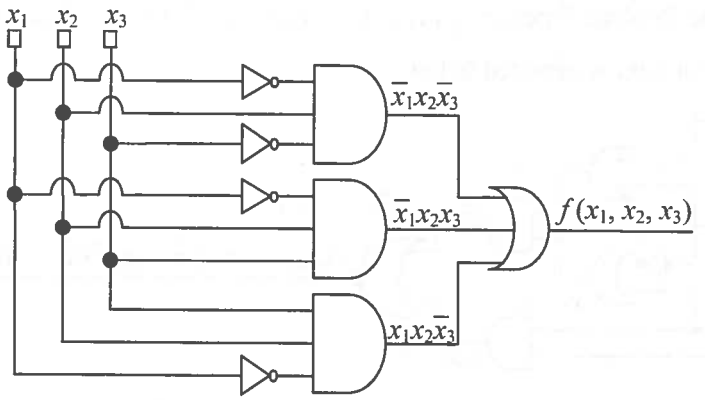
$a \downarrow b = \overline{a \vee b} = \overline{(\bar{a} \uparrow \bar{b})} = ((a \uparrow a) \uparrow (b \uparrow b)) \uparrow ((a \uparrow a) \uparrow (b \uparrow b))$

The corresponding logic circuits are as follows:

$\bar{a} = \overline{a \wedge a} = a \uparrow a$	
$a \wedge b = (a \uparrow b) \uparrow (a \uparrow b)$	
$a \vee b = \bar{a} \uparrow \bar{b}$	
$a \downarrow b = \overline{\bar{a} \uparrow \bar{b}}$	

Example 8.5.

Given the initial circuit drawn below, we want to obtain a simplified circuit (with less gates) equivalent to the initial one.

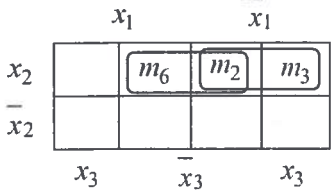


In order to obtain the simplified circuit we have to write the Boolean function corresponding to the initial circuit, then simplify the function and finally draw the circuit modeled by the simplified function.

The Boolean function which models the functionality of the logic circuit is

$$f(x_1, x_2, x_3) = \bar{x}_1 x_2 \bar{x}_3 \vee \bar{x}_1 x_2 x_3 \vee x_1 x_2 \bar{x}_3 = m_2 \vee m_3 \vee m_6$$

Using Veitch digram we simplify f:



$max_1 = m_6 \vee m_2 = x_2 \bar{x}_3$

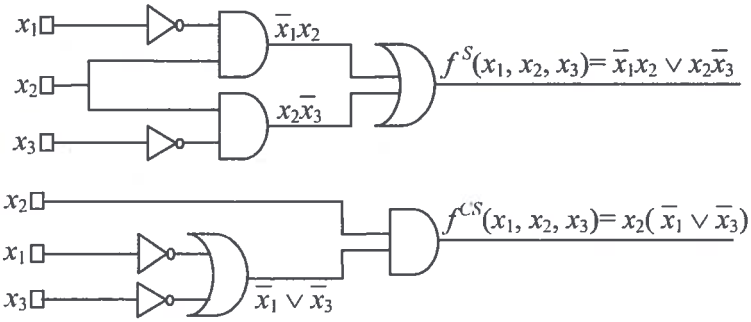
$max_2 = m_2 \vee m_3 = \bar{x}_1 x_2$

$M(f) = \{ max_1, max_2 \} = C(f)$

It is the first case of the simplification algorithm, thus there is a unique *disjunctive simplified form*: $f^S(x_1, x_2, x_3) = \bar{x}_1 x_2 \vee x_2 \bar{x}_3$.

By applying distributive and commutative laws, a *conjunctive simplified form* is obtained: $f^{CS}(x_1, x_2, x_3) = x_2(\bar{x}_1 \vee \bar{x}_3)$.

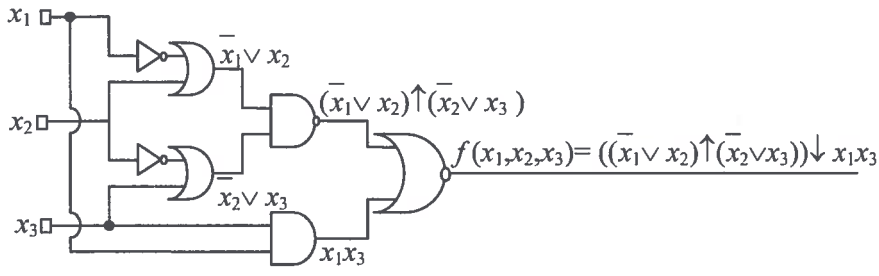
The simplified circuits equivalent to the initial one are as follows:



Example 8.6.

Draw the corresponding circuit (with basic and derived gates) and a simplified circuit for the Boolean function: $f(x_1,x_2,x_3) = ((\bar{x}_1 \vee x_2) \uparrow (\bar{x}_2 \vee x_3)) \downarrow x_1 x_3$.

The circuit diagram is depicted below.



We transform the function into its disjunctive canonical form. We replace the NAND and NOR operations with equivalent expressions using only: $\wedge$, $\vee$, $\bar{}$ operations and we apply DeMorgan, idempotency, commutativity, simplification and distributive laws.

$$f(x_1,x_2,x_3) = ((\bar{x}_1 \vee x_2)\uparrow(\bar{x}_2 \vee x_3)) \downarrow x_1x_3 =$$

$$= \overline{\overline{(\bar{x}_1 \vee x_2)(\bar{x}_2 \vee x_3)} \vee x_1x_3} =$$

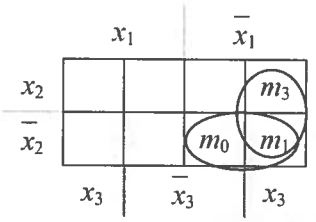
$$= (\bar{x}_1 \vee x_2)(\bar{x}_2 \vee x_3)(\bar{x}_1 \vee \bar{x}_3) = (\bar{x}_1\bar{x}_2 \vee \bar{x}_1x_3 \vee x_2\bar{x}_2 \vee x_2x_3)(\bar{x}_1 \vee \bar{x}_3) =$$

$$= (\bar{x}_1\bar{x}_2 \vee \bar{x}_1x_3 \vee 0 \vee x_2x_3)(\bar{x}_1 \vee \bar{x}_3) = (\bar{x}_1\bar{x}_2 \vee \bar{x}_1x_3 \vee x_2x_3)(\bar{x}_1 \vee \bar{x}_3) =$$

$$= \bar{x}_1\bar{x}_2\bar{x}_1 \vee \bar{x}_1\bar{x}_2x_3 \vee \bar{x}_1x_3\bar{x}_1 \vee \bar{x}_1x_3\bar{x}_3 \vee x_2x_3\bar{x}_1 \vee x_2x_3\bar{x}_3 =$$

$$= \bar{x}_1\bar{x}_2 \vee \bar{x}_1\bar{x}_2x_3 \vee \bar{x}_1x_3 \vee 0 \vee \bar{x}_1x_2x_3 \vee 0 =$$

$$= \bar{x}_1\bar{x}_2x_3 \vee \bar{x}_1\bar{x}_2\bar{x}_3 \vee \bar{x}_1x_2x_3 = m_1 \vee m_0 \vee m_3$$



By applying factorizations on Veitch diagram we obtain the maximal and central monoms:

$$max_1 = m_0 \vee m_1 = \bar{x}_1\bar{x}_2;$$

$$max_2 = m_1 \vee m_3 = \bar{x}_1x_3$$

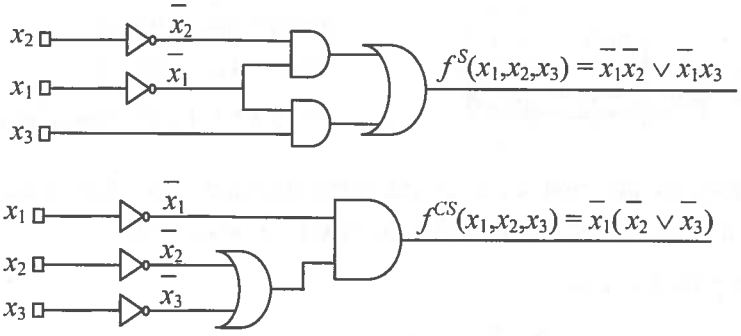
$M(f) = C(f) = \{max_1, max_2\}$

There is a unique disjunctive simplified form: $f^S(x_1,x_2,x_3) = \bar{x}_1\bar{x}_2 \vee \bar{x}_1x_3$

A simpler conjunctive form, with less operations and less occurrences of variables, is obtained by applying distributivity:

$f^{CS}(x_1,x_2,x_3) = \bar{x}_1(\bar{x}_2 \vee \bar{x}_3)$

The simplified circuits for f are depicted below:



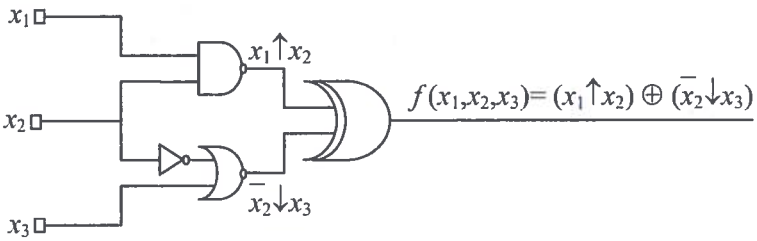
Example 8.7.

The following Boolean function contains the operations: $\oplus$, $\uparrow$, $\downarrow$:

$f(x_1,x_2,x_3) = (x_1\uparrow x_2) \oplus (\bar{x}_2\downarrow x_3)$.

Draw a simplified circuit containing only basic gates.

The circuit diagram with derived gates:



Using the definitions of the Boolean operations: $\oplus$, $\uparrow$, $\downarrow$ and by applying DeMorgan, absorption, idempotency and distributive laws we obtain a disjunctive form (not canonical):

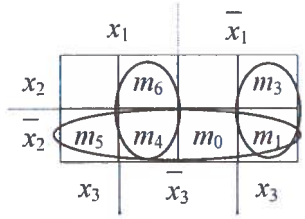
$$f(x_1,x_2,x_3) = (x_1 \uparrow x_2) \oplus (\bar{x}_2 \downarrow x_3) =$$
$$= \overline{(x_1 \uparrow x_2)}(\bar{x}_2 \downarrow x_3) \vee (x_1 \uparrow x_2)\overline{(\bar{x}_2 \downarrow x_3)} =$$
$$= \overline{\overline{(x_1 x_2)}}\,\overline{(\bar{x}_2 \vee x_3)} \vee \overline{(x_1 x_2)}\,\overline{\overline{(\bar{x}_2 \vee x_3)}} =$$
$$= x_1 x_2 \bar{x}_2 \bar{x}_3 \vee (\bar{x}_1 \vee \bar{x}_2)(\bar{x}_2 \vee x_3) = x_1 x_2 \bar{x}_3 \vee \bar{x}_2 \vee \bar{x}_1 x_3$$

This form is simple but we try to simplify it more.

We build the Veitch diagram from this disjunctive form as follows:

- for the minterm $x_1 x_2 \bar{x}_3 = m_6$ we mark the corresponding cell;
- for the monom $\bar{x}_2$ we mark in the diagram the 4 cells (m_1, m_0, m_5, m_4) which have in common the variable: $\bar{x}_2$
- for the monom $\bar{x}_1 x_3$ which covers the minterms: $\bar{x}_1 \bar{x}_2 x_3 = m_1$ and $\bar{x}_1 x_2 x_3 = m_3$ we mark the corresponding 2 cells.

The Veitch diagram is:



Factorizations:

$max_1 = \bar{x}_2 = m_1 \vee m_0 \vee m_5 \vee m_4$

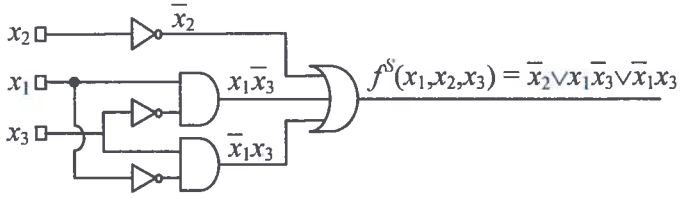
$max_2 = x_1 \bar{x}_3 = m_6 \vee m_4$

$max_3 = \bar{x}_1 x_3 = m_1 \vee m_3$

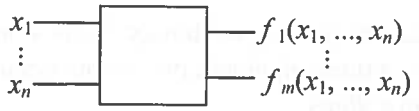
$M(f) = C(f) = \{max_1, max_2, max_3\}$

According to the first case of the simplification algorithm there is a unique disjunctive simplified form of f: $f^S(x_1,x_2,x_3) = \bar{x}_2 \vee x_1 \bar{x}_3 \vee \bar{x}_1 x_3$.

The simplified circuit:



A logic circuit which does not use any memory and whose outputs respond immediately to the inputs is called a ***combinational circuit***.
It can have m number of outputs. Each output corresponds to a Boolean function.



An elementary logic circuit is a particular case of a combinational circuit ($m = 1$).

In the *combinational logic circuit synthesis task* we have as input the problem specification describing the functionality of the desired circuit and as output the implementation of the logic circuit. To solve the task, first we build the truth tables defining the functions, then we minimize them and finally we build the circuit using the logic gates.

8.2. Examples of useful combinational circuits

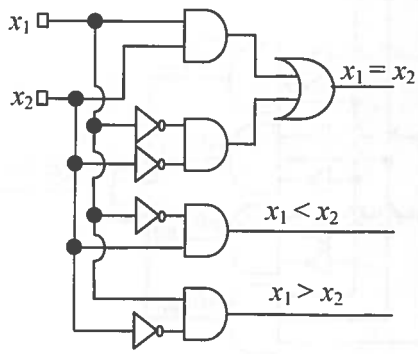
Combinational logic circuits can be used to perform arithmetic and logical operations (adders, subtractors, comparators), for data transmission (encoders, decoders, multiplexors, de-multiplexors). Some of these combinational circuits used in the hardware of computers will be described in this section.

Example 8.8.
The **comparator** circuit checks the relation between two binary digits. It is a combinational circuit with two inputs and three outputs. The truth tables of the output functions are as follows:

x_1	x_2	*equal* ($x_1 = x_2$)	*smaller* ($x_1 < x_2$)	*bigger* ($x_1 > x_2$)
0	0	1	0	0
0	1	0	1	0
1	0	0	0	1
1	1	1	0	0

We obtain the Boolean expressions for the output functions and we draw the circuit diagram.

$equal(x_1, x_2) = \overline{x}_1\overline{x}_2 \vee x_1x_2 = \overline{x}_1 \oplus x_2$
$smaller(x_1, x_2) = \overline{x}_1x_2$
$bigger(x_1, x_2) = x_1\overline{x}_2$



Boolean functions corresponding to the following binary relations can be written:

- smaller or equal: $x_1 <= x_2$, $seq(x_1, x_2) = \bar{x}_1 \vee x_2$
- bigger or equal: $x_1 >= x_2$, $beq(x_1, x_2) = x_1 \vee \bar{x}_2$

Example 8.9.

The **full adder** computes the sum of two binary digits: a and b of the same rank in two binary numbers. It is a three input and two output combinational circuit.

input variables: a, b - ***two digits***

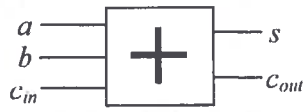
c_{in} - the ***input carry*** digit from the previous rank (less significant)

output functions: s - the ***one-digit sum*** of a and b and c_{in},

c_{out} - the generated carry digit, called ***output carry***

The table of truth values is presented below.

inputs			outputs	
a	b	c_{in}	$s = a + b + c_{in}$	c_{out}
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

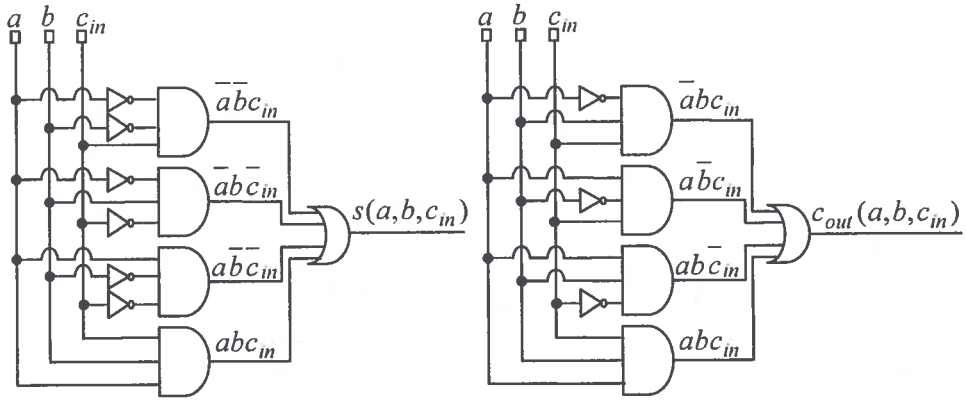


The output functions have the following disjunctive canonical forms:

$$s(a,b,c_{in}) = \bar{a}\bar{b}c_{in} \vee \bar{a}b\bar{c}_{in} \vee a\bar{b}\bar{c}_{in} \vee abc_{in}$$

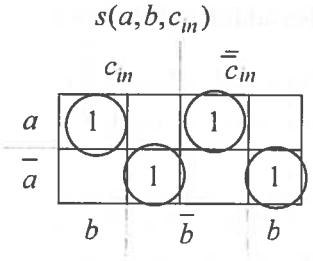
$$c_{out}(a,b,c_{in}) = \bar{a}bc_{in} \vee a\bar{b}c_{in} \vee ab\bar{c}_{in} \vee abc_{in}$$

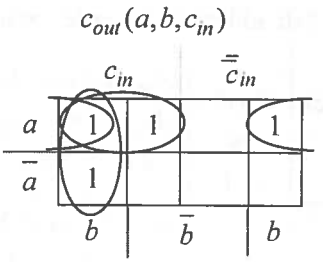
The *DCF* circuits for the functions are as follows:



Before drawing the corresponding combinational logic circuit we have to simplify the Boolean functions s and c_{out}.

Veitch diagrams are used in simplification.

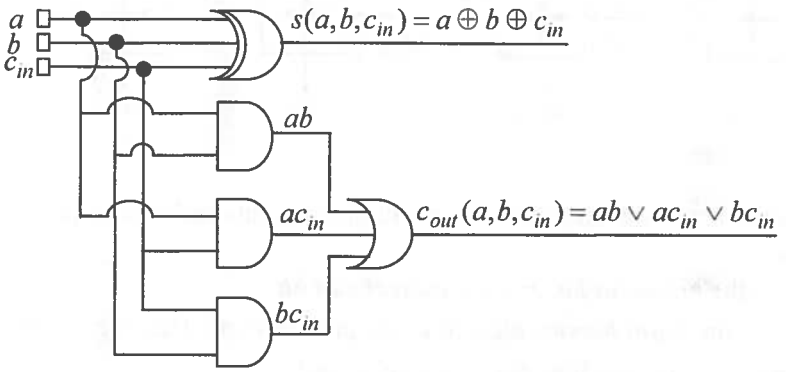




Note that the function s is in a simplified form and the simplified form of c_{out} using only basic gates is $c_{out}(a,b,c_{in}) = ab \vee ac_{in} \vee bc_{in}$.

For the function s there is a simplified form with fewest gates (derived): $s(a,b,c_{in}) = a \oplus b \oplus c_{in}$, as can be seen in the truth table.

The combinational circuit diagram is depicted below.

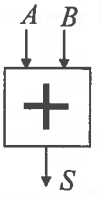


Example 8.10.

To add two n-bit binary numbers we need to use the ***n* - bit parallel adder**. It is composed of n full adders in cascade. The output carry of the previous full adder is connected to the input carry of the next full adder.

Let $A = (a_{n-1}a_{n-2}...a_0)_{(2)}$ and $B = (b_{n-1}b_{n-2}...b_0)_{(2)}$ be two binary numbers and $S = A + B$, $S = (s_n s_{n-1}...s_0)_{(2)}$. For $A = 10011011_{(2)}$ and $B = 11001111_{(2)}$, $S = A + B = 101101010_{(2)}$.

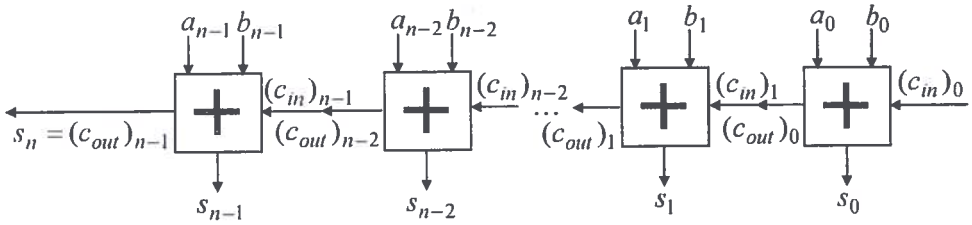
The logic circuit is symbolized as:



Eight full adders in cascade are used to perform this addition as follows:

carries	$(c_{out})_7 = s_8 = 1$	$(c_{out})_6 = (c_{in})_7 = 0$	$(c_{out})_5 = (c_{in})_6 = 0$	$(c_{out})_4 = (c_{in})_5 = 1$	$(c_{out})_3 = (c_{in})_4 = 1$	$(c_{out})_2 = (c_{in})_3 = 1$	$(c_{out})_1 = (c_{in})_2 = 1$	$(c_{out})_0 = (c_{in})_1 = 1$	$(c_{in})_0 = 0$
A		$a_7 = 1$	$a_6 = 0$	$a_5 = 0$	$a_4 = 1$	$a_3 = 1$	$a_2 = 0$	$a_1 = 1$	$a_0 = 1$
B		$b_7 = 1$	$b_6 = 1$	$b_5 = 0$	$b_4 = 0$	$b_3 = 1$	$b_2 = 1$	$b_1 = 1$	$b_0 = 1$
$S = A + B$	$s_8 = 1$	$s_7 = 0$	$s_6 = 1$	$s_5 = 1$	$s_4 = 0$	$s_3 = 1$	$s_2 = 0$	$s_1 = 1$	$s_0 = 0$

The combinational circuit is composed of n full adders:



Example 8.11.

The **full subtractor** is a combinational circuit with three inputs and two outputs.

input variables:

a - the ***minuend bit***, b - the ***subtrahend bit***

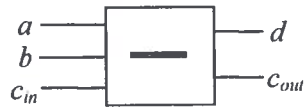
c_{in} - the ***input borrow*** digit from the previous rank (less significant)

output functions: d - the one-digit ***difference*** of a and b,

c_{out} - the ***output borrow***

The table of truth values is presented below.

inputs			outputs	
a	b	c_{in}	d	c_{out}
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1



If $a - b - c_{in} \geq 0$ then $d = a - b - c_{in}$ and $c_{out} = 0$

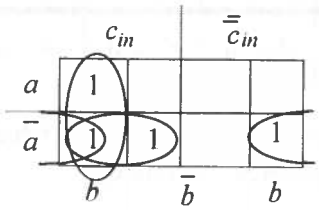
else $d = 2 + a - b - c_{in}$ and $c_{out} = 1$

The output functions have the following disjunctive canonical forms:

$$d(a,b,c_{in}) = \bar{a}\bar{b}c_{in} \vee \bar{a}b\bar{c}_{in} \vee a\bar{b}\bar{c}_{in} \vee abc_{in}$$

$$c_{out}(a,b,c_{in}) = \bar{a}\bar{b}c_{in} \vee \bar{a}b\bar{c}_{in} \vee \bar{a}bc_{in} \vee abc_{in}$$

Note that the function d is the same as the output function s from the full adder circuit. The function c_{out} is simplified using Veitch diagram:

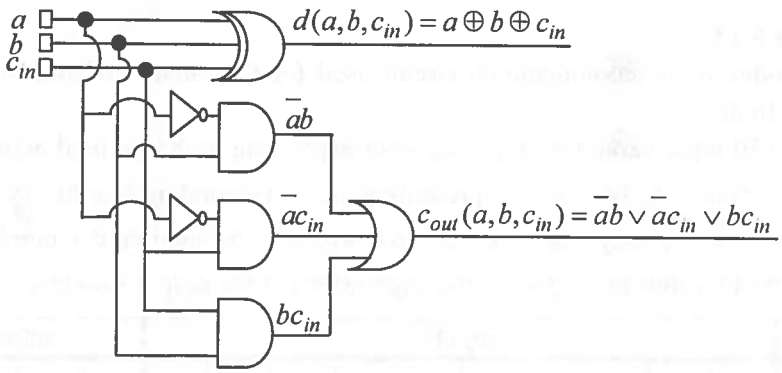


The simplified forms are:

$$d(a,b,c_{in}) = a \oplus b \oplus c_{in}$$

$$c_{out}(a,b,c_{in}) = \bar{a}b \vee \bar{a}c_{in} \vee bc_{in}$$

The combinational circuit diagram is depicted below.



Example 8.12.

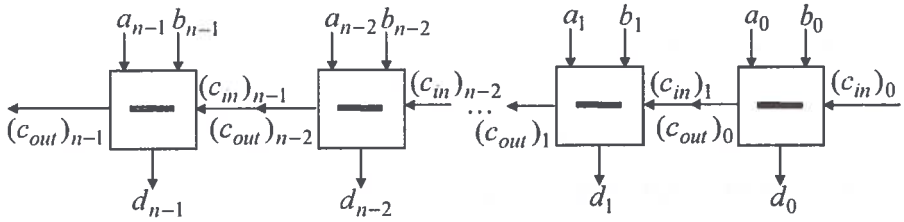
The ***n* - bit parallel subtractor** is used to subtract two n - bit binary numbers.

Let $A = (a_{n-1}a_{n-2}...a_0)_{(2)}$ be the minuend and $B = (b_{n-1}b_{n-2}...b_0)_{(2)}$ the subtrahend. The difference $D = A - B$, $D = (d_{n-1}d_{n-2}...d_0)_{(2)}$ is calculated using n full subtractors in cascade. The output borrow of the previous full subtractor is connected to the input borrow of the next full subtractor.

The subtraction in binary on 8 bits for $A = 11101010_{(2)}$ and $B = 10011111_{(2)}$, $D = A - B = 1001011_{(2)}$ is modeled as follows:

carries $(c_{out})_7 = 0$	$(c_{out})_6 = (c_{in})_7 = 0$	$(c_{out})_5 = (c_{in})_6 = 0$	$(c_{out})_4 = (c_{in})_5 = 1$	$(c_{out})_3 = (c_{in})_4 = 1$	$(c_{out})_2 = (c_{in})_3 = 1$	$(c_{out})_1 = (c_{in})_2 = 1$	$(c_{out})_0 = (c_{in})_1 = 1$	$(c_{in})_0 = 0$
A	$a_7 = 1$	$a_6 = 1$	$a_5 = 1$	$a_4 = 0$	$a_3 = 1$	$a_2 = 0$	$a_1 = 1$	$a_0 = 0$
B	$b_7 = 1$	$b_6 = 0$	$b_5 = 0$	$b_4 = 1$	$b_3 = 1$	$b_2 = 1$	$b_1 = 1$	$b_0 = 1$
$D = A - B$	$d_7 = 0$	$d_6 = 1$	$d_5 = 0$	$d_4 = 0$	$d_3 = 1$	$d_2 = 0$	$d_1 = 1$	$d_0 = 1$

The circuit diagram of the *n-bit subtractor*, which is composed of n full subtractors in cascade is represented as follows:



Example 8.13.

The **encoder** is the combinational circuit used for the binary codification of the decimal digits.

There are 10 input variables: $x_0,\ldots,x_9$ - corresponding to the decimal digits, and 4 output functions: f_1, f_2, f_3, f_4 - representing the corresponding four binary digits.

The variable $x_i = 1$ only when we want to convert the decimal digit i into binary.

First we build a tableau to identify the expressions of the output Boolean functions:

	inputs										outputs			
decimal digit	x_0	x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	x_9	f_1	f_2	f_3	f_4
0	**1**	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	**1**	0	0	0	0	0	0	0	0	0	0	0	1
2	0	0	**1**	0	0	0	0	0	0	0	0	0	1	0
3	0	0	0	**1**	0	0	0	0	0	0	0	0	1	1
4	0	0	0	0	**1**	0	0	0	0	0	0	1	0	0
5	0	0	0	0	0	**1**	0	0	0	0	0	1	0	1
6	0	0	0	0	0	0	**1**	0	0	0	0	1	1	0
7	0	0	0	0	0	0	0	**1**	0	0	0	1	1	1
8	0	0	0	0	0	0	0	0	**1**	0	1	0	0	0
9	0	0	0	0	0	0	0	0	0	**1**	1	0	0	1

The disjunctive forms of the Boolean functions are:

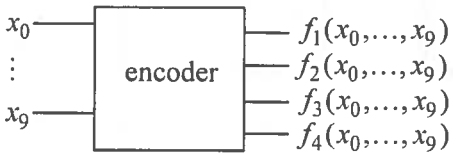
$f_1(x_0,\ldots,x_9) = x_8 \vee x_9$

$f_2(x_0,\ldots,x_9) = x_4 \vee x_5 \vee x_6 \vee x_7$

$f_3(x_0,\ldots,x_9) = x_2 \vee x_3 \vee x_6 \vee x_7$

$f_4(x_0,\ldots,x_9) = x_1 \vee x_3 \vee x_5 \vee x_7 \vee x_9$

The corresponding diagram of the combinational circuit is:



Example 8.14.

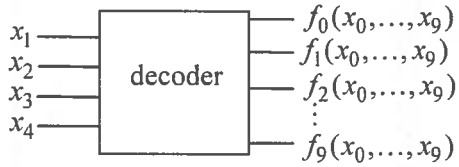
The **decoder** converts a group of 4 binary digits into a decimal digit.

$f_i(x_1,x_2,x_3,x_4) = 1$ only for $x_1x_2x_3x_4{}_{(2)} = i_{(10)}, i = \overline{0,9}$.

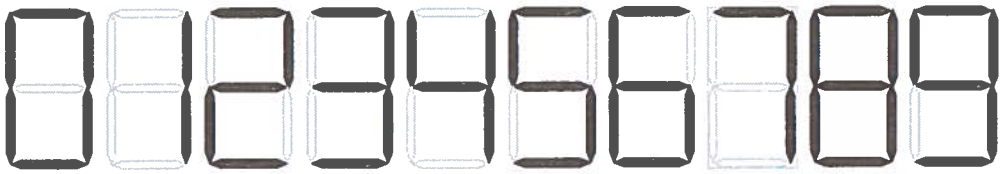
- $f_0(0,0,0,0) = 1$, $0000_{(2)} = 0_{(10)}$, for all the other arguments the function takes the value 0;
- $f_7(0,1,1,1) = 1$, $0111_{(2)} = 7_{(10)}$, for all the other arguments the function takes the value 0.

inputs				outputs										
x_1	x_2	x_3	x_4	f_0	f_1	f_2	f_3	f_4	f_5	f_6	f_7	f_8	f_9	disjunctive canonical form
0	0	0	0	**1**	0	0	0	0	0	0	0	0	0	$f_0(x_1,x_2,x_3,x_4) = \overline{x}_1\overline{x}_2\overline{x}_3\overline{x}_4$
0	0	0	1	0	**1**	0	0	0	0	0	0	0	0	$f_1(x_1,x_2,x_3,x_4) = \overline{x}_1\overline{x}_2\overline{x}_3x_4$
0	0	1	0	0	0	**1**	0	0	0	0	0	0	0	$f_2(x_1,x_2,x_3,x_4) = \overline{x}_1\overline{x}_2x_3\overline{x}_4$
0	0	1	1	0	0	0	**1**	0	0	0	0	0	0	$f_3(x_1,x_2,x_3,x_4) = \overline{x}_1\overline{x}_2x_3x_4$
0	1	0	0	0	0	0	0	**1**	0	0	0	0	0	$f_4(x_1,x_2,x_3,x_4) = \overline{x}_1x_2\overline{x}_3\overline{x}_4$
0	1	0	1	0	0	0	0	0	**1**	0	0	0	0	$f_5(x_1,x_2,x_3,x_4) = \overline{x}_1x_2\overline{x}_3x_4$
0	1	1	0	0	0	0	0	0	0	**1**	0	0	0	$f_6(x_1,x_2,x_3,x_4) = \overline{x}_1x_2x_3\overline{x}_4$
0	1	1	1	0	0	0	0	0	0	0	**1**	0	0	$f_7(x_1,x_2,x_3,x_4) = \overline{x}_1x_2x_3x_4$
1	0	0	0	0	0	0	0	0	0	0	0	**1**	0	$f_8(x_1,x_2,x_3,x_4) = x_1\overline{x}_2\overline{x}_3\overline{x}_4$
1	0	0	1	0	0	0	0	0	0	0	0	0	**1**	$f_9(x_1,x_2,x_3,x_4) = x_1\overline{x}_2\overline{x}_3x_4$

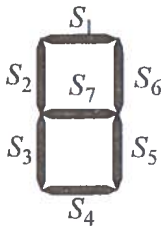
The corresponding combinational circuit is:

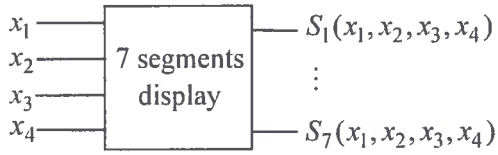


Example 8.15.
Electronic display of the decimal digits using 7 segments (LEDs):



The combinational circuit having the functionality described above has as inputs 4 variables and as outputs 7 functions corresponding to the segments.





The combinations of 4 binary digits which do not correspond to a decimal digit will generate an emission of light only for the segment S_7.

	inputs				outputs						
Decimal digit	x_1	x_2	x_3	x_4	S_1	S_2	S_3	S_4	S_5	S_6	S_7
0	0	0	0	0	1	1	1	1	1	1	0
1	0	0	0	1	0	0	0	0	1	1	0
2	0	0	1	0	1	0	1	1	0	1	1
3	0	0	1	1	1	0	0	1	1	1	1
4	0	1	0	0	0	1	0	0	1	1	1
5	0	1	0	1	1	1	0	1	1	0	1
6	0	1	1	0	1	1	1	1	1	0	1
7	0	1	1	1	1	0	0	0	1	1	0
8	1	0	0	0	1	1	1	1	1	1	1
9	1	0	0	1	1	1	0	1	1	1	1

–	1	0	1	0	0	0	0	0	0	0	1
–	1	0	1	1	0	0	0	0	0	0	1
–	1	1	0	0	0	0	0	0	0	0	1
–	1	1	0	1	0	0	0	0	0	0	1
–	1	1	1	0	0	0	0	0	0	0	1
–	1	1	1	1	0	0	0	0	0	0	1

The canonical forms of the output functions are presented.

$S_1 = m_0 \vee m_2 \vee m_3 \vee m_5 \vee m_6 \vee m_8 \vee m_9$ (DCF)

$= M_1 \wedge M_4 \wedge M_{10} \wedge M_{11} \wedge M_{12} \wedge M_{13} \wedge M_{14} \wedge M_{15}$ (CCF)

$S_2 = m_0 \vee m_4 \vee m_5 \vee m_6 \vee m_8 \vee m_9$ (DCF)

$= M_1 \wedge M_2 \wedge M_3 \wedge M_7 \wedge M_{10} \wedge M_{11} \wedge M_{12} \wedge M_{13} \wedge M_{14} \wedge M_{15}$ (CCF)

$S_3 = m_0 \vee m_2 \vee m_6 \vee m_8$ (DCF)

$= M_1 \wedge M_3 \wedge M_4 \wedge M_5 \wedge M_7 \wedge M_9 \wedge M_{10} \wedge M_{11} \wedge M_{12} \wedge M_{13} \wedge M_{14} \wedge M_{15}$ (CCF)

$S_4 = m_0 \vee m_2 \vee m_3 \vee m_5 \vee m_6 \vee m_8 \vee m_9$ (DCF)

$= M_1 \wedge M_4 \wedge M_7 \wedge M_{10} \wedge M_{11} \wedge M_{12} \wedge M_{13} \wedge M_{14} \wedge M_{15}$ (CCF)

$S_5 = m_0 \vee m_1 \vee m_3 \vee m_4 \vee m_5 \vee m_6 \vee m_7 \vee m_8 \vee m_9$ (DCF)

$= M_2 \wedge M_{10} \wedge M_{11} \wedge M_{12} \wedge M_{13} \wedge M_{14} \wedge M_{15}$ (CCF)

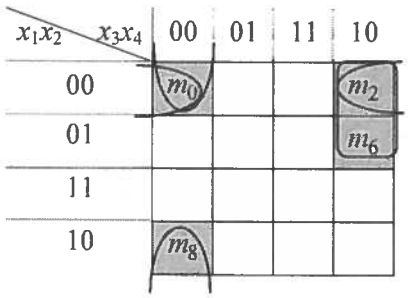
$S_6 = m_0 \vee m_1 \vee m_2 \vee m_3 \vee m_7 \vee m_8 \vee m_9$ (DCF)

$= M_5 \wedge M_6 \wedge M_{10} \wedge M_{11} \wedge M_{12} \wedge M_{13} \wedge M_{14} \wedge M_{15}$ (CCF)

$S_7 = m_2 \vee m_3 \vee m_4 \vee m_5 \vee m_8 \vee m_9 \vee m_{10} \vee m_{11} \vee m_{12} \vee m_{13} \vee m_{14} \vee m_{15}$ (DCF)

$= M_0 \wedge M_1 \wedge M_7$ (CCF)

In the following we simplify S_3, first its *DCF*, using Karnaugh diagram:



The set of the maximal monoms is $M(S_3) = \{ max_1, max_2, max_3 \}$. These maximal monoms are obtained by applying three simple factorizations:

$max_1 = \bar{x}_1 \bar{x}_2 \bar{x}_4 = m_0 \vee m_2$

$max_2 = \bar{x}_1 x_3 \bar{x}_4 = m_2 \vee m_6$

$max_3 = \bar{x}_2 x_3 \bar{x}_4 = m_0 \vee m_8$

The set of the central monoms, $C(S_3) = \{max_2, max_3\} \neq M(S_3)$.

There is a unique disjunctive simplified form of S_3:
$S_3^S = max_2 \vee max_3 = \bar{x}_1 x_3 \bar{x}_4 \vee \bar{x}_2 \bar{x}_3 \bar{x}_4$. This expression has 6 occurrences of variables, 5 binary operations and 5 unary operations.

We double negate the expression of S_3^S and we transform it using the results from Example 8.5 in order to implement the corresponding circuit using only NAND gates.

$$S_3^s = (\bar{x}_1 \wedge x_3 \wedge \bar{x}_4) \vee (\bar{x}_2 \wedge \bar{x}_3 \wedge \bar{x}_4) = \overline{\overline{(\bar{x}_1 \wedge x_3 \wedge \bar{x}_4) \vee (\bar{x}_2 \wedge \bar{x}_3 \wedge \bar{x}_4)}} =$$

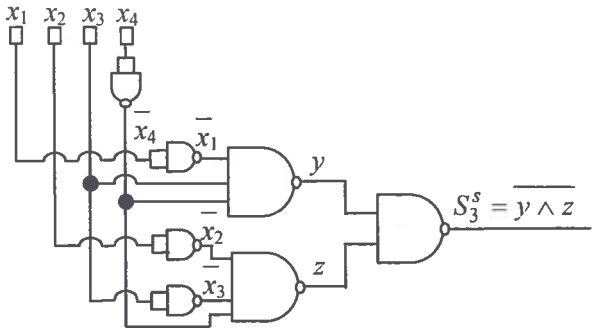
$$= \overline{\overline{x_1 \wedge x_3 \wedge \bar{x}_4} \wedge \overline{\bar{x}_2 \wedge \bar{x}_3 \wedge \bar{x}_4}} =$$

$$= \overline{\overline{(x_1 \wedge x_1) \wedge x_3 \wedge (x_4 \wedge x_4)} \wedge \overline{(x_2 \wedge x_2) \wedge (x_3 \wedge x_3) \wedge (x_4 \wedge x_4)}}$$

The logic circuit has:

- four NAND gates which implement the negations of the variables (ex: $\bar{x}_1 = \overline{x_1 \wedge x_1}$);
- two NAND gates, with 3 inputs each, for the expressions $y = \overline{\overline{(x_1 \wedge x_1) \wedge x_3 \wedge (x_4 \wedge x_4)}}$ and $z = \overline{\overline{(x_2 \wedge x_2) \wedge (x_3 \wedge x_3) \wedge (x_4 \wedge x_4)}}$
- a two-input NAND gate corresponding to $S_3^s = \overline{y \wedge z}$

The circuit diagram is depicted below:



The simplification of *CCF* (S_3) follows a dual simplifgication algorithm presented in Example 7.10. In the Karnaugh diagram for *CCF*, the headers of the lines-columns are used to express the indices of the maxterms and represent the duals of the powers of the variables from the maxterms' expressions.

x_1x_2 \ x_3x_4	00	01	11	10
00		M_1	M_3	
01	M_4	M_5	M_7	
11	M_{12}	M_{13}	M_{15}	M_{14}
10		M_9	M_{11}	M_{10}

By applying a triple dual factorization and two double dual factorizations we obtain the minimal disjunctions of S_3 :

$$min_1 = M_1 \wedge M_3 \wedge M_5 \wedge M_7 \wedge M_{13} \wedge M_{15} \wedge M_9 \wedge M_{11} = \bar{x}_4$$

$$min_2 = M_4 \wedge M_5 \wedge M_{12} \wedge M_{13} = \bar{x}_2 \vee x_3$$

$$min_3 = M_{15} \wedge M_{14} \wedge M_{11} \wedge M_{10} = \bar{x}_1 \vee \bar{x}_3$$

These minimal disjunctions are also central disjunctions of S_3, because in the diagram each group of cells has at least one maxterm circled once.

According to the first case of the dual simplification algorithm there is a unique conjunctive simplified form:

$$S_3^{CS} = \bar{x}_4 \wedge (\bar{x}_2 \vee x_3) \wedge (\bar{x}_1 \vee \bar{x}_3)$$

This expression has 5 occurrences of variables, 4 binary operations and 4 unary operations and is simpler then the disjunctive simplified form S_3^S.

From the conjunctive simplified form we can implement very easy the corresponding logic circuit using only NOR gates. The transformations applied on S_3^{CS} are presented in the following:

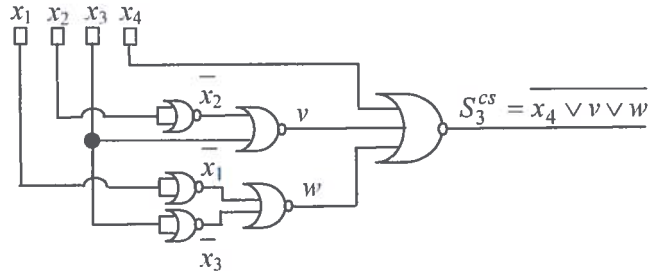
$$S_3^{cs} = \bar{x}_4 \wedge (\bar{x}_2 \vee x_3) \wedge (\bar{x}_1 \vee \bar{x}_3) = \overline{\overline{\bar{x}_4 \wedge (\bar{x}_2 \vee x_3) \wedge (\bar{x}_1 \vee \bar{x}_3)}} =$$

$$= \overline{\overline{x_4 \vee (\bar{x}_2 \vee x_3) \vee (\bar{x}_1 \vee \bar{x}_3)}} =$$

$$= \overline{\overline{x_4 \vee (\bar{x}_2 \vee x_3) \vee (\bar{x}_1 \vee \bar{x}_3)}} = x_4 \vee \overline{\overline{(x_2 \vee x_2) \vee x_3 \vee (x_1 \vee x_1) \vee (x_3 \vee x_3)}}$$

In the implementation of the logic circuit six NOR gates are used as follows:

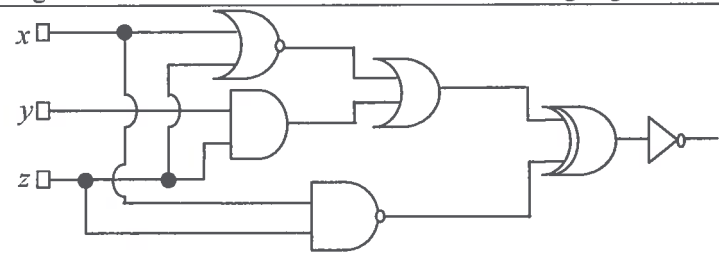
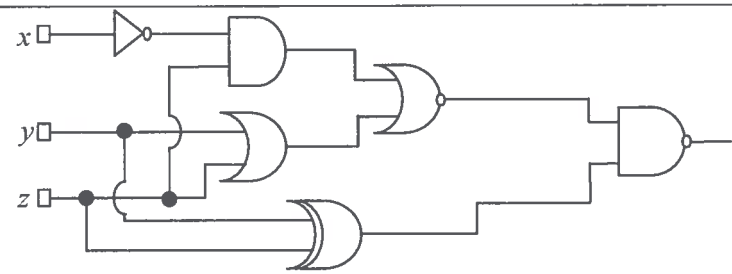
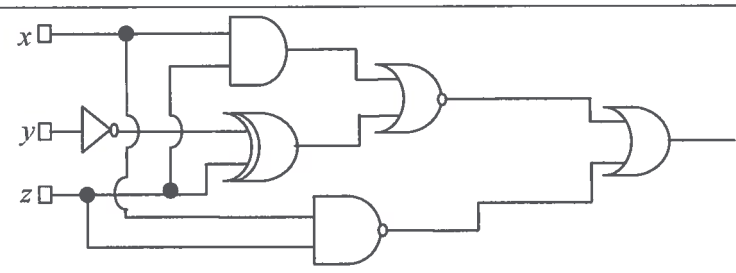
- three gates for $\overline{x_1}, \overline{x_2}, \overline{x_3}$ (ex: $\overline{x_1} = \overline{x_1 \vee x_1}$);
- two gates, with two inputs each, for the expressions:
  $v = \overline{\overline{(x_2 \vee x_2)} \vee x_3}$ and $w = \overline{\overline{(x_1 \vee x_1)} \vee \overline{(x_3 \vee x_3)}}$;
- a final gate with three inputs corresponding to $S_3^{cs} = \overline{x_4 \vee v \vee w}$

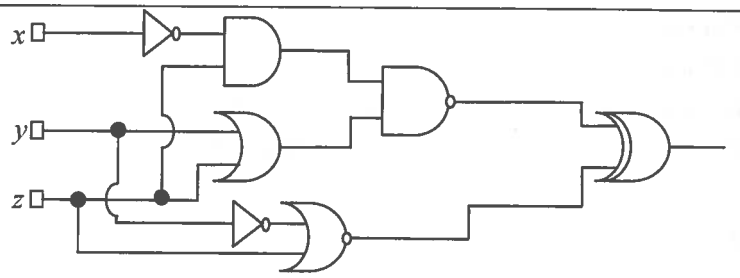
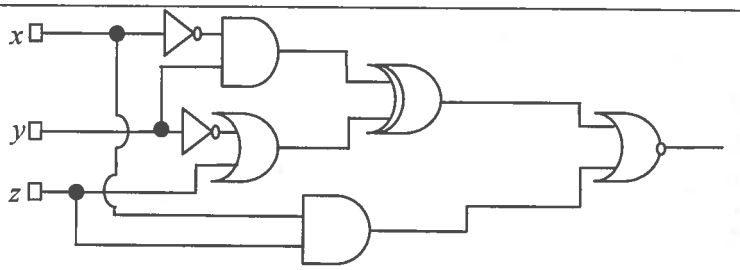
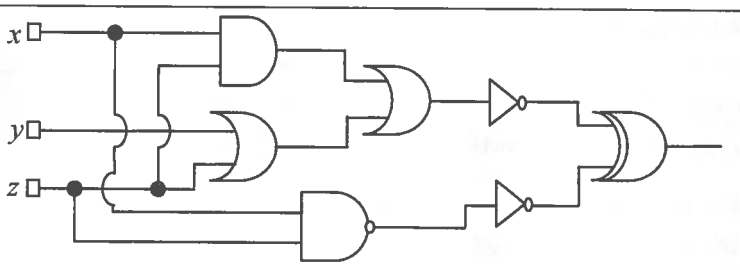
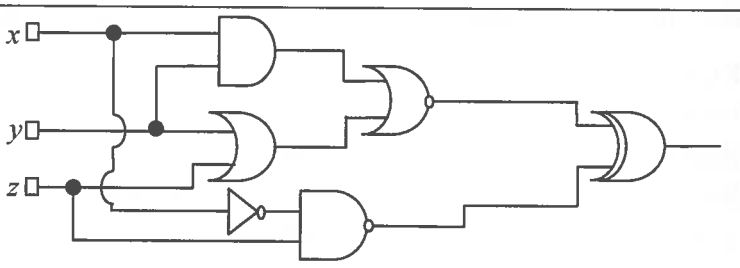
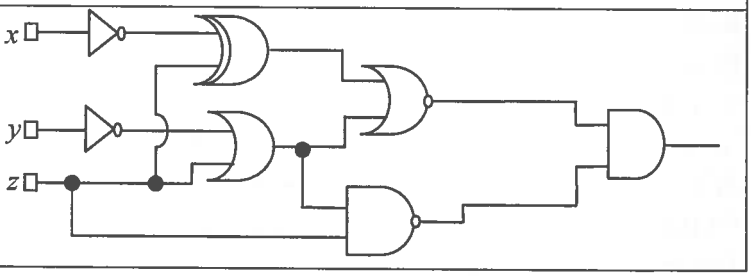


8.3. Exercises

Exercise 8.1.

Write the corresponding Boolean function associated to the following logic circuit:

1.	
2.	
3.	

4.	
5.	
6.	
7.	
8.	

Exercise 8.2.
Prove that NOR is a universal gate.

Exercise 8.3.
For each of the following Boolean functions draw the corresponding logic circuit using derived gates, simplify the function and draw the logic circuits associated to all simplified forms of the initial function using only basic gates.

1. $f_1(x, y, z) = x(y \oplus z) \vee y(x \oplus z) \vee x(\overline{y} \downarrow \overline{z}) \vee (x \downarrow y)\overline{z}$;
2. $f_2(x, y, z) = x(y \uparrow z) \vee \overline{x}(\overline{y} \oplus z) \vee y(\overline{x} \oplus \overline{z})$;
3. $f_3(x, y, z) = x(\overline{y} \oplus z) \vee y(\overline{x} \oplus z) \vee \overline{x}(\overline{y} \downarrow z) \vee (\overline{x} \downarrow y)z$;
4. $f_4(x, y, z) = \overline{x}(y \uparrow \overline{z}) \vee x(\overline{y} \oplus z) \vee \overline{y}(\overline{x} \oplus z)$;
5. $f_5(x, y, z) = \overline{x}(y \oplus \overline{z}) \vee \overline{y}(x \oplus z) \vee \overline{x}(y \downarrow z) \vee (\overline{x} \downarrow y)\overline{z}$;
6. $f_6(x, y, z) = x(\overline{y} \uparrow \overline{z}) \vee \overline{x}(y \oplus z) \vee \overline{y}(\overline{x} \oplus z)$;
7. $f_7(x, y, z) = x(y \oplus \overline{z}) \vee y(\overline{x} \oplus z) \vee x(y \downarrow z) \vee (x \downarrow y)\overline{z}$;
8. $f_8(x, y, z) = x(\overline{y} \uparrow z) \vee \overline{x}(\overline{y} \oplus z) \vee y(x \oplus \overline{z})$.

Exercise 8.4.
Draw a logic circuit having 3 input wires and containing all basic and derived gates. Write the corresponding Boolean function, simplify it and then draw a simplified circuit equivalent to the initial one.

Exercise 8.5.
Write a Boolean function of 4 variables given by its table of values, simplify it and draw the logic circuits corresponding to all its simplified forms.

Exercise 8.6.
Simplify the Boolean functions $(S_1,S_2,S_3,S_4,S_5,S_6,S_7)$ corresponding to the outputs(segments) of the electronic display of the decimal digits using 7 segments (see Example 8.15). Draw the logic circuits for the disjunctive/conjunctive simplified forms using only NAND/NOR gates.

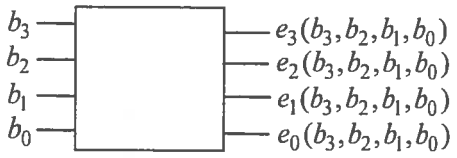
Exercise 8.7.
Binary codes are used to represent the ten decimal digits on 4 bits. There exist *weighted codes* and *unweighted codes*. In a weighted code each binary digit has a 'weight' and the decimal value is obtained as the weighted sum of all 4 bits.
BCD - Binary Coded Decimal is an example of a weighted code, with the weights for the digits: 8,4,2,1 (from the most significant digit to the least significant one).
The most used unweighted codes are *Excess 3* (obtained by adding '0011' to each *BCD* word) and the *Gray code* (a cyclic code with the property that all the successive words differ by one single digit).

The table below provides the *BCD*, *Excess 3* and *Gray* code on 4 bits.

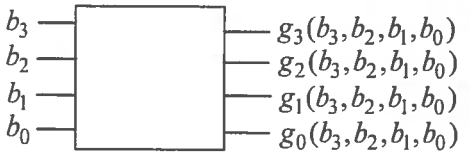
Decimal value	*BCD*				*Excess 3*				*Gray* code			
	b_3 8	b_2 4	b_1 2	b_0 1	e_3	e_2	e_1	e_0	g_3	g_2	g_1	g_0
0	0	0	0	0	0	0	1	1	0	0	0	0
1	0	0	0	1	0	1	0	0	0	0	0	1
2	0	0	1	0	0	1	0	1	0	0	1	1
3	0	0	1	1	0	1	1	0	0	0	1	0
4	0	1	0	0	0	1	1	1	0	1	1	0
5	0	1	0	1	1	0	0	0	0	1	1	1
6	0	1	1	0	1	0	0	1	0	1	0	1
7	0	1	1	1	1	0	1	0	0	1	0	0
8	1	0	0	0	1	0	1	1	1	1	0	0
9	1	0	0	1	1	1	0	0	1	1	0	1
10	1	0	1	0	-	-	-	-	1	1	1	1
11	1	0	1	1	-	-	-	-	1	1	1	0
12	1	1	0	0	-	-	-	-	1	0	1	0
13	1	1	0	1	-	-	-	-	1	0	1	1
14	1	1	1	0	-	-	-	-	1	0	0	1
15	1	1	1	1	-	-	-	-	1	0	0	0

Implement the combinational circuits for converting:

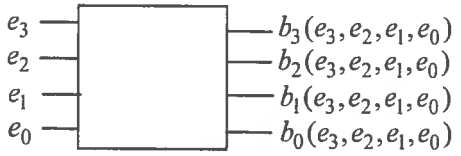
1. from *BCD* to *Excess 3* code;



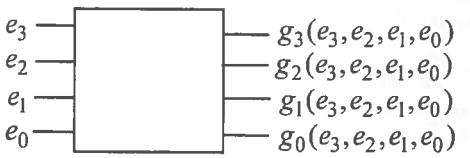
2. from *BCD* to *Gray* code;



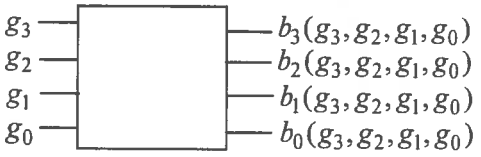
3. from *Excess 3* code to *BCD*;



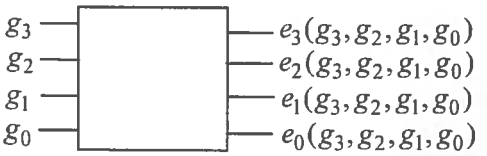
4. from *Excess 3* code to *Gray* code;



5. from *Gray* code to *BCD*;



6. from *Gray* code to *Excess 3* code;



INDEX

BIBLIOGRAPHY

1. **Ben-Ari, M.,** *Mathematical Logic for Computer Science*, Springer, 2001.
2. **Benzaken, C.,** *Systemes Formels. Introduction a la Logique*, Masson, 1991.
3. **Bibel, W.,** *Automated Theorem Proving*, View Verlag, Braunschweig, second edition, 1987.
4. **Boian, F. M.,** *De la aritmetică la calculatoare*, Presa Universitară Clujeană, 1996.
5. **Bonatti, P., Olivetti, N.,** *Sequent Calculi for Propositional Nonmonotonic Logics*, ACM Trans. Comput. Log., 2002, pag. 226-278.
6. **Boole, G.,** 1847, *The Mathematical Analysis of Logic. Being an Essay Towards a Calculus of Deductive Reasoning*, Cambridge: Macmillan, Barclay, 1847.
7. **Boole, G.,** 1854, *An Investigation of the Laws of Thought, on which are Founded the Mathematical Theories of Logic and Probabilities*, London: Walton & Maberly, 1854.
8. **Both, N.,** *Algebra logicii cu aplicaţii*, Editura Dacia, Cluj-Napoca, 1984.
9. **Both, N.,** *Capitole Speciale de Logică Matematică*, Universitatea Babeş-Bolyai, Cluj-Napoca, 1994.
10. **Boyer, R.S.,** *Locking: A Restriction of Resolution*, Ph.D. Thesis,University of Texas at Austin, Texas, 1971.
11. **Chang, C.L.,** *The unit proof and the input proof in theorem proving*, Journal Assoc. Comput. Mach. 17, 1970, pp.698-707.
12. **Chang, C.L, Lee, R.C.,** *Symbolic Logic and Mechanical Theorem Proving*, Academic Press, 1973.
13. **Church, A.,** *An unsolvable problem of number theory*, American Journal of Mathematics, 58, pag.345-363, 1936.
14. **Cocan, M., Pop, B.,** *Bazele Matematice ale Sistemelor de Calcul*, Editura Albastră, Cluj-Napoca, 2001.
15. **Davis, M., Putnam, H.,** *A computing procedure for quantification theory*, Journal Assoc. Comput. Mach., 7, 1960, pp. 201-215.
16. **Delahaye, J.P.,** *Outils Logiques pour l'Intelligence Artificielle*, Eyrolls, 1986.
17. **Dreben, B., Goldfarb, W.,** *The Decision Problem: Solvable Classes of Quantificational Formulas*, Addison-Wesley, 1979.
18. **Duffy, D.A.,** *Principles of Automated Theorem Proving*, John Wiley & Sons, 1991.
19. **Dumitrescu, D.,** *Principiile Inteligenţei Artificiale*, Editura Albastră, Cluj-Napoca, 2002.
20. **Fitting, M.,** *First-order logic and Automated Theorem Proving*, Springer Verlag, 1990.
21. **Floarea, A., Boangiu, A.,** *Inteligenţa Artificială*, Universitatea Tehnică Bucureşti, 1994
22. **Gentzen, G.K.E,** *Untersuchungen über das logische Schließen. I*, Mathematische Zeitschrift 39 (2),1934, pp. 176–210.

23. **Genesereth, M.R., Nilsson, N.J.,** *Logical Foundations of Artificial Intelligence*, Morgan Kaufman, 1992.
24. **Hilbert, D., Ackermann, W.,** *Principles of Mathematical Logic*, Chelsea, New York, 1950.
25. **Hsu, J. Z.,** *Computer logic. Design Principles and Applications*, Springer-Verlag, New York, 2002.
26. **Kleene, S.C.,** Mathematical Logic, Wiley, New York, 1967.
27. **Lenzen, W.,** *Leibniz's Logic, in Handbook of the History of Logic*, D. M. Gabbay/J. Woods (eds.), volume 3: The Rise of Modern Logic: From Leibniz to Frege, Amsterdam et al.: Elsevier-North-Holland, pp. 1–83, 2004.
28. **Livovschi, L.,** *Circuite cu contacte de relee*, Editura Academiei Republicii Socialiste România, 1968.
29. **Littlewood, J.E.,** *Varietăţi matematice*, Bucureşti, 1969.
30. **Loveland, D.W.,** *A linear format for resolution*, Proceedings IRIA Symp. Automatic Demonstration, Versailles, France, 1968, Springer-Verlag, New York, 1970, pp. 147-162.
31. **Loveland, D.W.,** *Automated Theorem Proving: A Logical Basis*, North Holland, Amsterdam, 1978.
32. **Luckham, D.,** *Refinements in resolution theory*, Proceedings IRIA Symp. Automatic Demonstration, Versailles, France, 1968, Springer-Verlag, New York, 1970, pp. 147-162.
33. **Lupea, M.,** *Lock resolution - a refinement of resolution*, Seminar on Computer Science, Editura Universitatii Babes-Bolyai, Cluj-Napoca, 1994, pp. 57-64.
34. **Lupea, M.,** *Semantic tableaux to compute extensions for different versions of default logic*, Research Seminar on Computer Science, Editura Universitatii Babes-Bolyai, Cluj-Napoca, 2000, pp. 31-48.
35. **Lupea, M.,** *Raţionament nemonoton prin logici implicite*, Ph.D Thesis, Babeş-Bolyai University, Cluj-Napoca, 2002.
36. **Lupea, M.,** *Axiomatization of credulous reasoning in rational default logic*, Studia Universitas Babeş-Bolyai, Informatica, LII(1), pag:101-111, 2007.
37. **Lupea, M., Mihiş, A.,** *Logici clasice şi circuite logice. Teorie şi exemple*, Editura Albastra, Cluj-Napoca, first edition (2008), second edition (2009), third edition (2011).
38. **Maliţa, M., Mircea M.,** *Bazele Inteligenţei Artificiale*, Vol.I, Logici propozitionale, Editura Tehnică, Bucureşti, 1987.
39. **Mihiş, A.D., Chisăliţă-Creţu C., Mihăilă C., Şerban C.,** *BOOFS - a tool that supports simplifying conditional expressions using boolean functions simplification methods*, Proceedings of International Conference of Mathematics & Informatics, ICMI45, Bacău, Septembrie 18-20, 2006, Studii şi cercetări ştiinţifice, nr.16 – 2006 Supplement, Universitatea din Bacău, Facultatea de Stiinţe, Seria Matematica, ISN 1224-2519.
40. **Moşcenski, V.A.,** *Lecţii po matematiceskoi logike*, Minsk, 1973.
41. **Nasin, P.,** *Circuite logice şi automatizări secvenţiale*, Editura Tehnică, Bucureşti 1967.

42. **Nilsson, N.J.,** *Principles of Artificial Intelligence*, Tioga, Palo Alto, Morgan Kaufmann, 1980.
43. **Paulson, L.C.,** *Logic and Proof*, Cambridge University, 2000, www.cl.cam.ac.uk/teaching/2000/LogicProof/notes.pdf.
44. **Popa, C.,** *Logica predicatelor*, Editura Hyperion, Bucureşti, 1992.
45. **Possega, M.,** *Deduction Systems*, Institute of Informatics, 2002, on-line course.
46. **Rădulescu, V.,** *Revanşa minţii*, Editura Militară, Bucureşti, 1974.
47. **Reeves S., Clarke, M.,** *Logic for Computer Science*, Eddison-Wesley, 1990.
48. **Rich, E.,** *Artificial Inteligence*, Mac Graw-Hill, NewYork, 1983.
49. **Risch, V., Schwind, C.B.,** *Tableau-Based Characterization and Theorem Proving for Default Logic*, Journal of Automated Reasoning, Vol 13, Nr. 2, 1994, pag. 223-242.
50. **Robinson, J.A.,** *A machine oriented logic based on resolution principle*, Journal Assoc. Comput. Mach, 12, 1965, pp. 23-41.
51. **Robinson, J.A.,** *Automatic deduction with hyper-resolution*, International Journal Computation Math, 1, 1965, pp. 227-234.
52. **Robinson, J.A.,** *Logic, Form and Function. The Mechanization of Deductive Reasoning*, University Press, Edinburg, 1979.
53. **Roul de Palma,** *Algebra binară a lui Boole şi aplicaţiile ei în informatică*, Editura Tehnică, Bucureşti, 1976.
54. **Russel, S., Norwig, P.,** *Artificial Intelligence*, Prentice Hall, 1995.
55. **Rusu, E.,** *Cum gândim şi rezolvăm 200 de probleme*, Editura Albastros, Bucureşti, 1972.
56. **Schwind, C.B.,** *A tableaux-based theorem prover for a decidable subset of default logic*. 10-th International Conference on Automated Deduction. Lecture Notes in A.I. 449, 1990.
57. **Slagle, J.R.,** *Automatic theorem proving with renamable and semantic resolution*, Journal Assoc. Comput. Mach, 12, 1965, pp. 536-541.
58. **Smullyan, R.M.,** *First-Order Logic*, Springer Verlag, Berlin, 1968. Revised Edition, Dover Press, New York, 1996.
59. **State, L.,** *Elemente de Logică Matematică şi Demonstrarea Automată a Teoremelor*, Universitatea Bucureşti, 1989.
60. **Tarski, A.,** *Introduction to Logic*, Oxford University Press, 1976.
61. **Thayse, A.,** editor: *From Standard Logic to Logic Programming*, John Wiley & Sons, vol1 (1989), vol2 (1989), vol3 (1990).
62. **Tătar, D.,** *Bazele Matematice ale Calculatoarelor*, Lito, Babeş-Bolyai University, Cluj-Napoca, 1999.
63. **Tătar, D.,** Inteligenţa Artificială: Demonstrare Automată de Teoreme şi NLP, Editura Microinformatica, Cluj-Napoca, 2001.
64. **Tîrnoveanu, M.,** *Elemente de logică matematică, vol.1, Logica propoziţiilor bivalente*, Editura didactică şi pedagogică, Bucureşti, 1964.
65. **Wos, L., Robinson J.A., Carson, D.F.,** *Efficiency and completeness of the set of support strategy in theorem proving*, Journal Assoc. Comput. Mach, 12, 1965, pp. 536-541.