Edizioni G-Tronic
9RSAA

Gottardo's Rule

Applied in TIA PORTAL V21
Vol. 1
First edition 2026

This publication is intended for the scientific community; therefore, it is not a textbook for learning Siemens PLC programming using TIA PORTAL V21.
The purpose of this monographic edition is to define an axiom, develop its theoretical implications, and prove the resulting theorems.

Written, edited, and published

By

Eng. Prof. Dr. Marco Gottardo, PhD

Series of publications for industrial automation.

Gottardo's Rule

This edition was edited and published in July 2026 by:
Eng. Prof. Dr. Marco Gottardo Ph.D.

Marco Gottardo, C.F. GTTMRC68R06G224I,
Via Colombo 14, 30030 Vigonovo (VE) Italia.
E-mail: ad.noctis@gmail.com

ISBN-13: 9798183938050
Publisher: G-Tronic by Marco Gottardo Publisher

Indice:

Preface

This publication is not conceived as a popular work or textbook, but as an academic treatise, or a monographic volume, aimed at introducing and discussing a specific theoretical proposition intended for the evaluation of the scientific community.
The main objective of the work is to present and formalize the following axiom: "When a contact is delegated to release a self-latching, it is captured in the software in a state opposite to that represented in the functional."
The purpose of the discussion is to analyze this proposition, verify its logical coherence and propose it as an axiom within the theoretical framework outlined by the author. The formalization of the axiom constitutes the first step towards a possible broader structuring of the conceptual model, within which the proposition can subsequently be developed and demonstrated in the form of a theorem.
The present work also intends to establish a clear and formally recognizable formulation of the proposed thesis[1], with the intention of attributing its authorship in an explicit and traceable way in the context of the scientific literature.
While adopting a deliberately simple and direct language, the content maintains the level of terminological precision and argumentative rigor required in scientific and academic fields, in order to ensure clarity of exposition without compromising the formality of the theoretical discourse.

[1] **Thesis:** Formulation for the definition of the logical states of the release contacts of retained cars based on the state indicated in the functional diagrams. When a physical contact in the field is delegated to release a self-latching, this is acquired in the software as opposed to how it is represented in the functional. Consequently, clear and well-defined rules for functional representation are needed.

Formal framing of the problem

In discrete industrial control, the self-holding or seal-in function realizes a bistable memory controlled by two inputs:

- S: Start control (SET)
- R: Stop Command (RESET)
- Q: Exit status (Coil)

In the PLC (ladder or FBD) formalism, the classical structure is:

- Contact NO, normalmente aperto in campo, di Start
- Contact NC, normally closed in the field, to Stop
- Contact NO in parallel with the Start
- Output Coil

A first axiom, necessary for the application of the Gottardo rule (teaching terminology widespread in the Italian technical field) establishes:
In the simultaneous presence of the excitation and de-energization commands, the de-energization must prevail.
In logical terms:

$$\boldsymbol{S = 1 \ \wedge \ R = 1 \Rightarrow Q = 0}$$

This is a reset priority rule, which is critical for functional safety reasons.

Axiomatization in Boolean Algebra
Consider axiomatic Boolean algebra:

$$\mathcal{B} = (B, +, \cdot, ', 0, 1)$$

with the following fundamental axioms:

Commutativity

$$A + B = B + A; A \cdot B = B \cdot A$$

Associativity

$$(A + B) + C = A + (B + C)$$

Distributivity

$$A \cdot (B + C) = A \cdot B + A \cdot C$$

Complementation

$$A + A' = 1; A \cdot A' = 0$$

Identity

$$A + 0 = A; A \cdot 1 = A$$

Formalization of the self-restraint

Recursive equation of state

The status of the output at the time $k + 1$ are**:**

$$Q_{k+1} = (S + Q_k) \cdot R'$$

This is the canonical formalization of the self-restraint with priority to reset.
Interpretation:

- $S + Q_k$→ initial arousal or maintenance
- R'→ Non-reset condition

First demonstration of the Gottardo rule

We want to formally demonstrate:

$$\boldsymbol{S = 1 \land R = 1 \Rightarrow Q_{k+1} = 0}$$

We substitute in the equation:

$$\boldsymbol{Q_{k+1} = (1 + Q_k) \cdot 1'}$$

Since:

$$\boldsymbol{1 + Q_k = 1}$$

e

$$\boldsymbol{1' = 0}$$

more:

$$\boldsymbol{Q_{k+1} = 1 \cdot 0 = 0}$$

Proven.
Q.E.D.

Structural Property: Reset Priority

Now let's show that the function is equivalent to an explicit form of priority:

$$\boldsymbol{Q_{k+1} = S \cdot R' + Q_k \cdot R'}$$

Applying distributivity:

$$\boldsymbol{(S + Q_k) \cdot R' = S \cdot R' + Q_k \cdot R'}$$

Collecting:

$$\boldsymbol{Q_{k+1} = R' \cdot (S + Q_k)}$$

Remark:
if $\boldsymbol{R = 1 \Rightarrow R' = 0}$, then:

$$\boldsymbol{Q_{k+1} = 0}$$

regardless of and $\boldsymbol{SQ_k}$.
This is the mathematical formalization of the Gottardo rule.

Comparison with classic SR latch

Latch SR without priority:

$$\boldsymbol{Q_{k+1} = S + Q_k \cdot R'}$$

If $\boldsymbol{S = R = 1}$:

$$\boldsymbol{Q_{k+1} = 1 + Q_k \cdot 0 = 1}$$

Here, EETS → does not comply with the Gottardo rule prevails.

Canonical Form of Shannon

We apply expansion with respect to R:

$$Q_{k+1} = R \cdot f_{R=1} + R' \cdot f_{R=0}$$

We calculate:

- Se $R = 1 \Rightarrow Q_{k+1} = 0$
- Se $R = 0 \Rightarrow Q_{k+1} = S + Q_k$

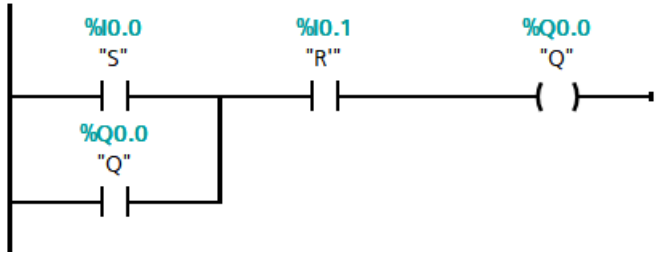
So:

$$Q_{k+1} = R \cdot 0 + R' \cdot (S + Q_k)$$
$$Q_{k+1} = R' \cdot (S + Q_k)$$

Minimal shape

Interpretation in the Siemens PLC (TIA Portal)

In the TIA Portal ladder:



The structure implements exactly:

$$Q = (S + Q) \cdot R'$$

In the scan loop:

1. Input reading
2. Logical evaluation
3. Output Image Update

The priority of the reset is guaranteed structurally, not temporally.

Demonstration of system stability

Let's look for steady states:

$$\boldsymbol{Q_{k+1} = Q_k = Q}$$
$$\boldsymbol{Q = (S + Q) \cdot R'}$$

Case 1: $R = 1$

$$\boldsymbol{Q = 0}$$

Unique balance.
Case 2: $R = 0$

$$\boldsymbol{Q = S + Q}$$

If $\boldsymbol{S = 0 \Rightarrow Q = Q}$(Memory)
If $\boldsymbol{S = 1 \Rightarrow Q = 1}$
Consistent system.

Safety Theorem (formulation)
Theorem:
The Boolean function

$$\boldsymbol{Q_{k+1} = (S + Q_k) \cdot R'}$$

is the only minimal three-variable function that satisfies:

1. Self-restraint
2. Key reset
3. Absence of prohibited states
4. Idempotency with respect to $R = 1$

Proof (synthetic):
Imposing the constraints in the truth table and minimizing them using Karnaugh maps yields a single minimum implying coverage equivalent to the above form.

Extension: matrix form in state space

We can write:

$$\boldsymbol{Q_{k+1} = f(Q_k, S, R)}$$

Nonlinear discrete dynamical system on Boolean field:

$$\boldsymbol{Q_{k+1} = R'S + R'Q_k}$$

Structurally equivalent to:

$$\boldsymbol{Q_{k+1} = R'(S + Q_k)}$$

which represents a first-order logic system with dominant cancellation input.

Didactic application synthesis

The Gottardo rule is not a ladder graphic artifice, but:

- a structural property of the Boolean function
- a priority constraint in the discrete domain
- a functional safety condition
- A design choice consistent with machine safety regulations

Its axiomatic formalization allows:

- formal analysis
- Proof of fairness
- symbolic verification
- automatic synthesis

Field application starting from input terminal blocks.

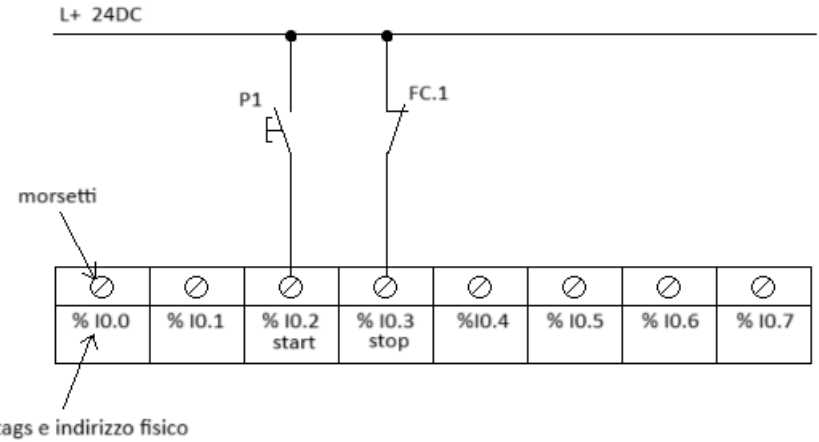
The programmer's work begins with an interview phase, in which the technical specifications of the system to be built are collected, documented and formalized through the customer's signatures.

These should not change during the course of the work, which, however, punctually happens, upsetting the work already done in progress.

When you start programming the system or machine, you must provide the programmer with two documents, the wiring diagram and the P&ID.

The first shows the connections between the sensory equipment and the inputs of the PLC as well as between the outputs and the actuators or their control devices (relays, contactors, solenoid valves, etc.).

The image shows an excerpt of a typical wiring diagram[2] provided to the programmer in the initial phase of the development of the work.

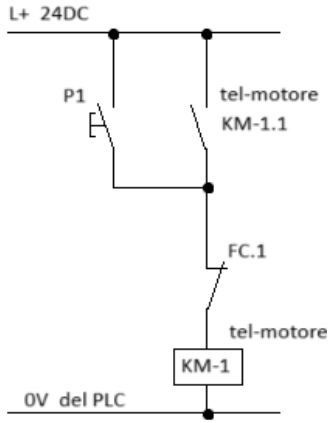


The P1 button brings the travel control signal to the %I0.2 input, i.e. the third bit of the first input terminal block, while the electromechanical limit switch is connected to the next digital input %I0.3.

The programmer probably realizes a "self-retention command", which maintains the movement up to the assigned limit.

The following should not appear on the wiring diagram:

[2] In the wiring diagram, the inputs and outputs are shown as rectangles, organized in rows of eight, or one byte. They have an address field and a field for the Tags (symbolic label of the variable). At a glance, if the rectangle is at the bottom of the page, it's inputs, if it's at the top, it's outputs.
Internal variables such as Merker and those defined in the DBs are not shown in the schematics.

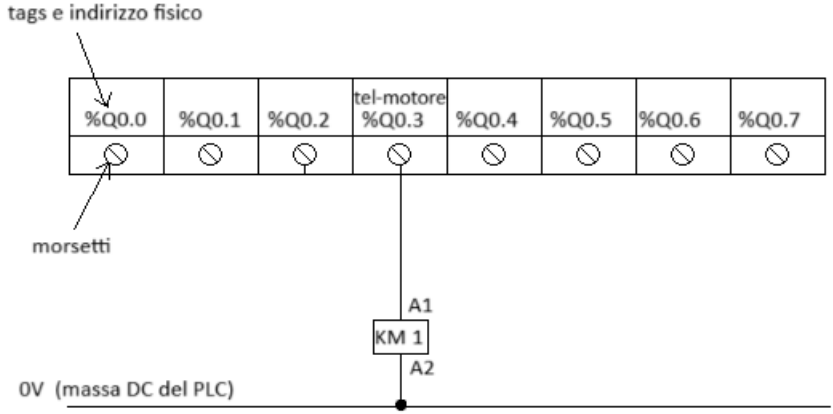


This does not appear in the schema because it represents the internally implemented logic[3].
It is an electromechanical Flip Flop capable of storing a bit until the unhooking event occurs.
In this case, the monostable[4] drive command is stored in the internal bit KM-1, preferably declared inside a Data Block, rather than as a Merker in the Tags Table.

When the P1 button is pressed, the positive signal, at 24V DC, is carried to the node below.
At the node It encounters a normally closed wired electromechanical device, then the signal passes through it energizing the coil.
The energized coil closes its contact placed parallel to the control button.
The operator can release the button but the reel is held by its own contact placed in parallel.
Some moving part of the car will intercept FC.1 which, cutting the line, releases the retained car.
As we will see in several parts of the text, this symbology, although having strong similarities, is not an electrical diagram and takes the name of functional diagram.
Many things, which would be mandatory for wiring diagrams are negligible in functional diagrams, for example, many numbering.
It makes no sense to number a cable that is actually non-existent since it is represented by software.
The same connections have a direct correspondence with the software, for

[3] In computer science, implementing is synonymous with realizing, therefore programming.
[4] Input signals can be monostable or bistable in nature. Monostable signals are impulsive, such as those generated by buttons, which generate the TRUE state only for as long as the operator remains with the finger. Bistable signals, on the other hand, are typical of switches. Each action corresponds to a transition from a stable state to the complementary one where it remains until a subsequent action of the operator. In automation, monostable action is preferred and any self-restraint created in the software so as to make the automata self-operating when conditions occur. In addition, self-retained systems are more suitable for automatic release during safety interventions.

example the parallel represents the Boolean operation[5] OR and the series the operation AND.
The wiring diagram, of the outputs, gives meaning to the connection of the contactor coil.



It is in this diagram that the length of wire between terminal %Q0.3 and terminal A1 of the contactor must be numbered, even if it is not present here for the sake of simplicity.

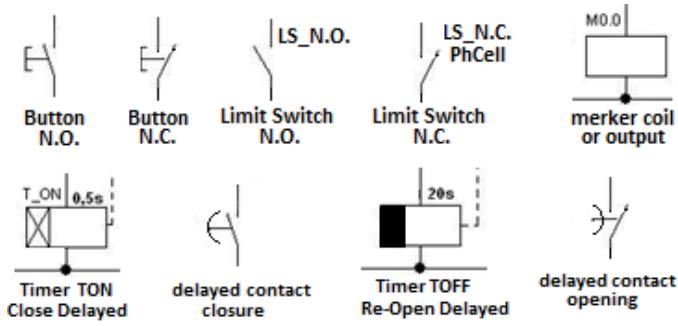
The same terminals A1 and A2 will be represented in the real diagram in standard mode as the "crossed out dot" to which the wording consistent with the rest of the diagram is combined, for example X1-1 etc.

[5] Boolean operations are those contemplated in Boolean algebra. George Boole was an English mathematician and logician born in 1815, long before the advent of computers. He applied normal algebra to base-two numbering systems, postulating some missing and essential elements, such as tautology, which allows the simplification of redundant signals. To give an example, the expression "today it rains or it doesn't rain" is a tautology, which can be achieved through the parallel "OR" of the two possibilities that can be expressed with the same contact in the normally open or normally closed situation. Since this parallel is always true (TRUE), it can be eliminated or better replaced with a wire that from the +24V D.C. power line. It connects directly to the coil of the expression closing relay. The most important of Boolean minimizations is thus implemented, as can be found in the Karnaugh map technique. A compendium of the basic rules of Boolean logic is the contribution of another scientist, De Morgan, who introduces the equalities and therefore the substitutability between OR and AND if subject to appropriate constraints of signal negation.

The fundamental elements and symbols of functional logic.
Functional logic expresses the actions that the controller must take and is a useful aid for software development. The basic elements are:

1. Power line, horizontal at the top, indicated by L+ or L if powered by direct or alternating current. Since it is not an electrical circuit but an algorithm to be implemented in the internal logic of the PLC, through the software, it makes more sense to always be continuous.
2. Return line, indicated by M (ground as the closure of an L+ line) or N (neutral as the closure of an alternating line).
3. Drops, which derive downwards the logical flow. In the functional real, each segment starts from a descent and does not derive horizontally from the previous segment.
4. Closing contact, (i.e. wired normally open), with a dry contact appearance drawn to the left of the descent.
5. Opening contact (i.e. normally closed wired), with a dry contact appearance drawn to the right of the descent.
6. Timed coil delayed on excitation and its contact in opening and closing.
7. Timed coil delayed at de-energization and its contact when opening and closing.
8. Counter, represented as a rectangular load coil but with up, down, set and reset inputs. The preset value must also be represented.

The symbols useful for writing a functional are summarized in the image below. As we can see, triangles on the end stops such as the closing bars are not essential since the various contacts represent only the Boolean state of a bit inside or present in the terminal block, not the electromechanical object in the field.

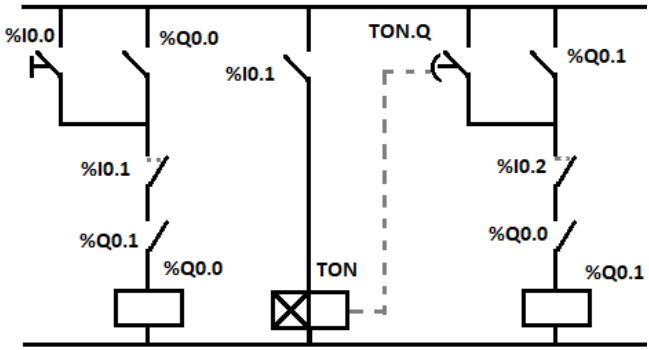


Since, in this context, the functional diagram is not a schematic but an algorithm to be followed to implement the Ladder program, the quantity of symbols and the rules of the drawing are greatly simplified.

First, the symbols present in the functional represent the signals that electromechanical objects provide.

The functional diagram is a schematic and intuitive representation of the program that will have to be implemented with the software. It performs the same function as the flow chart [6]in high-level languages.

Only in a first approximation does the functional rotated 90 degrees represent the implementation of the software. In general, this is not true. In this regard, the **Gottardo Rule should be considered.**

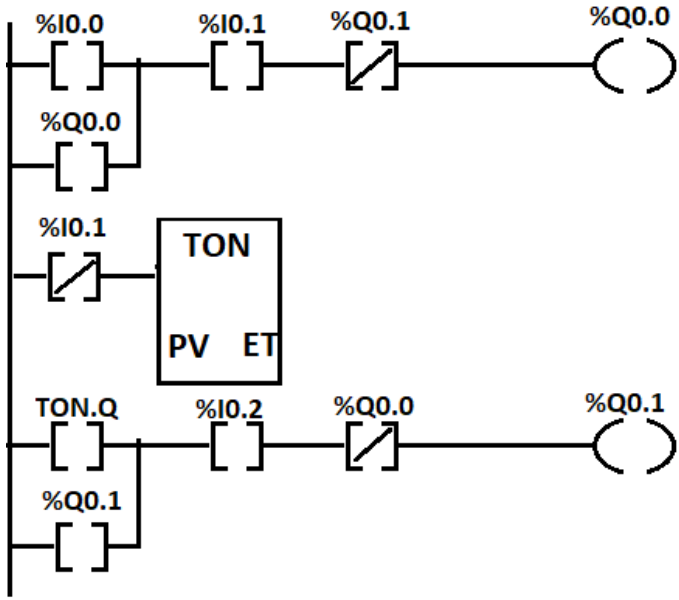


Regola del Gottardo: Quando un contatto* è delegato allo sgancio di un'auto ritenuta, viene acquisito nel software al contrario di come è rappresentato nel funzionale.

***si intende contatto fisico di tipo fine corsa elettromeccanico. Quando il medesimo segnale perviene da una virtualizzazione o è in qualche modo simulato la regola ne viene confermata in quanto l'eventuale contatto negato presente allo sgancio non è un contatto reale o non è un fine corsa elettromeccanico.**

[6] Flow charts should not be applied in the development phase of PLC programs because they are inadequate. These are replaced, where possible, with functional diagrams. The symbol sets for flow charts are oval for departures and arrivals, rectangles of various types for actions, diamonds for IF-type decision-makers, arrows to indicate the directions of flows. By analogy, functional diagrams have their own limited set of symbols, the open contact to the left of the descent, the closed contact to the right of the descent, the coil, the timed coils, while the logics are made with parallels for the ORs and series for the ANDs.

The proposed functional diagram leads to the implementation of the standardized code IEC 61131 below.
Attention should be drawn from the contacts %I0.1 and %I0.2 which in the functional are shown on the right side of the abseil, so normal 1 -> pressed 0.
In these Ladder segments, on the other hand, they are normally open, and this is explained in the aforementioned "Gottardo rule".[7]
The need for the rule arises after thirty years of experience in the educational field and is no less trivial than stating that two parallel lines never meet.



Nota sulla sicurezza intrinseca:
Un sistema in autoritenuta si trova in sicurezza intrinseca quando la rottura o la mancanza del sensore di fine corsa invia al controllore il medesimo stato dell'intervento.

Demonstration with counternominal technique
: For the Gottardo Rule we apply the technique of counternominal proof which consists in starting from the negation of the thesis to arrive at the negation of the hypothesis, and not at a contradiction with the hypothesis. Both the **thesis** is the need to inverted the release contact of the restraint

[7] Until proven otherwise, the author uses the status of Ph.D. researcher to validate its existence and the demonstration formulated here.

car with respect to how it is represented in the functional.
The negation of the thesis is that the state of physical contact on the pitch is not denied.

The **hypothesis**, however, is that the physical contacts on the field delegated to the release of the restraint car are normally closed wired.

Let's assume, absurdly, to inverted the state in the field of physical contact delegated to unhooking.

It can be observed that no signal reaches the terminal delegated to read it.

Since this is connected, according to the internal logic of the software, by means of a Boolean combination AND, the combinatorial result, whatever the state of the start button, will be zero, preventing the coil from hooking in self-retained.

The **negation of the hypothesis** is that the state of the release contact delegated to the restraint car should not be denied with respect to how it is shown in the functional.

By not inverteding the state of this contact, it implements the Boolean double negation theorem[8] realized by the statement "not(not(unhooked) = unhooked", and therefore the motor does not start because it is unhooked.

the double negation allows you to eliminate a negation from the expression, i.e. to place the state of the contact in the ladder step 7 segment open instead of closed, definitively validating the Gottardo rule, until proven otherwise.

Quod Erat Demonstrandum Q.E.D.

Please note: The hypothesis mentions "closed contacts in the field" so it should not apply when the aforementioned contacts do not exist, i.e. when they are virtualized in an HMI panel.

Proof of the opposite: The interlocking contact should not be reversed as it does not satisfy the initial assumptions by proving the thesis.

[8] Demonstrable both through Hilbert's deductive systems and in classical propositional logic even with the sole aid of tautologies. According to the rules of double negation, theoremized in Boolean and combinatorial algebras, then the double negation allows one to eliminate a negation from the expression, definitively validating the Gotthard rule, until proven otherwise.

Correct technical statement

In the case of **physical contact of a normally closed wired (NC) electromechanical limit switch** and delegated to the **release of a self-retaining device**, the following rule applies:

In the PLC software, the contact must be programmed as negated with respect to its functional representation.

In practice:

- In the field → NC contact (closed at rest).
- In Ladder → must be inserted as a normally open contact (—| |—) associated with the input bit.

This is because the PLC reads **electrical logic levels**, not functional symbology.

Analysis of why inversion is needed

Real physical state (NC limit switch)

Physical state	**PLC terminal**	**Input Bits**
Non premuto	1	TRUE
Premuto	0	FALSE

But functionally:

1. Pressed = trip condition
2. Not pressed = machine enabled

Therefore the functional meaning is inverted with respect to the logical level read.

Correct form of self-restraint

Correct equation:
Q_{n+1}= (START ∨ Q_n) ∧ INPUT

Dove:

- INPUT is the bit read by the PLC (wired NC).
- When the limit switch is pressed → INPUT = 0 → la AND Cancel all → Guaranteed release.

If the programmer denies it in the software:
Q_{n+1}= (START ∨ Q_n) ∧ ¬INPUT

then:

- At rest → INPUT = 1 → ¬1 = 0 → The machine never starts.
- Pressed → INPUT = 0 → ¬0 = 1 → The machine may start with the limit switch active (serious error).

And this is exactly the heart of the counternominal demonstration reported in the text.

Corollary: Q_{n+1} represents the state assumed by the output in the scan cycle following the evaluation of the logical state of the memory location, as provided for by the PLC runtime. In PLC systems that conform to the IEC 61131-3 language execution model, inputs are sampled at the beginning of the scan cycle and copied to the process image of the inputs, while output assignments update the process image of the outputs during program execution. Only at the end of the cycle is the image of the outputs transferred to the physical terminals. This implies that the program can observe the updated state of a coil in the same cycle, while the field will see it in the next cycle.

Logical translation of the proof

Hypothesis:
Physical contact is NC in the field.

Thesis:
It should not be negated in software.

Counternominal demonstration:

- If you inverted it in the software,
- then you would prevent the latch from engaging or you would enable the machine in the release condition,
- therefore the hypothesis of correct functioning would be false,
- therefore it must not be denied.

The reference to double negation:

$$\neg(\neg S) = S$$

it only serves to formalize that:
Physical NC + NO contact in ladder = implicit double negation → consistent behavior.

When it does NOT apply

The book footnote is correct:
The rule applies **only if the contact is physical electromechanical NC wired in the field**.

It does not apply when:

- The signal is virtual (HMI).
- The signal is simulated.
- Contact is physical NO.
- These are internal logic interlocks.

Operational summary (from plant technician)

If the safety device is:

- NC Physical Cablato → DO NOT reverse in the PLC.
- NO physical → evaluate according to function.
- Virtual → no Gottardo rule.

Possiamo ora sviluppare i seguenti punti:

- riscrivere la dimostrazione in forma rigorosa di logica proposizionale pura (senza narrativa),
- Mostrare il caso con timer TON come nello schema presentato in precedenza
- Analizzare l'errore tipico in TIA Portal quando si confonde rappresentazione funzionale e stato elettrico.

without prejudice to the Gottardo rule, already present in the text with ISBN: 9798345038970, let's proceed by developing the analysis of all three points listed.

Let's keep as an axiom the formulation of the Gottardo Rule, already present in many texts, and let's proceed in a structured way on the three points required.

Proof in strict propositional logic

Definizioni formali

Let it be:

- F = physical state of the NC contact in the field
  F=1 → closed loop (normal condition)
  F=0 → open circuit (limit switch depressed)
- I = bit read by PLC
  Being NC wired:

$$I=F$$

1. Q = self-latching exit
2. S = start

Correct self-restraint:

$$Q_{n+1}= (S \vee Q_n) \wedge I$$

Thesis (Gottardo Rule)

Don't invert that I in software.

Counternominal demonstration

Suppose the thesis is false:
You inverted contact in the software.

Then:

$$Q_{n+1} = (S \vee Q_n) \wedge \neg I$$

Case 1 — normal condition (F = 1 → I = 1):

$$Q_{n+1} = (S \vee Q_n) \wedge 0 = 0$$

The restraint cannot be linked → non-compliant behavior.

Case 2 — Active limit switch (F = 0 → I = 0):

$$Q_{n+1} = (S \vee Q_n) \wedge 1$$

The machine may start in a tripping condition → serious functional violation.

Therefore, the negation of the thesis implies the negation of the hypothesis of correct functioning.

Q.E.D.

Case with TON (as in the attached diagram)

A TON appears in the diagram that delays undocking or enabling.

Remind:

TON.Q=1 solo dopo PT con IN = 1

If the NC limit switch is denied in the software:

- Under normal conditions → IN = 0 → TON never starts.
- In trip conditions → IN = 1 → TON starts when it shouldn't.

Conseguenze:

- Inverted timings.
- Self-latching that engages with the limit switch pressed.
- Trip delays instead of start delays.

With timers, the violation becomes even more insidious because it introduces inconsistent time dynamics.

Typical error in TIA Portal

Common students mistake:

Misunderstanding:

1. Functional Graphic Symbol (NC Drawn)
2. Logic status read by PLC

In TIA:

1. Contact —|/|— does not mean "physical NC contact".
2. It means "Negation (inverting) logical condition".

Many programmers:

1. NC Wiring in the Field.
2. Ladder —|/|—.
3. They get double wrong reversal.

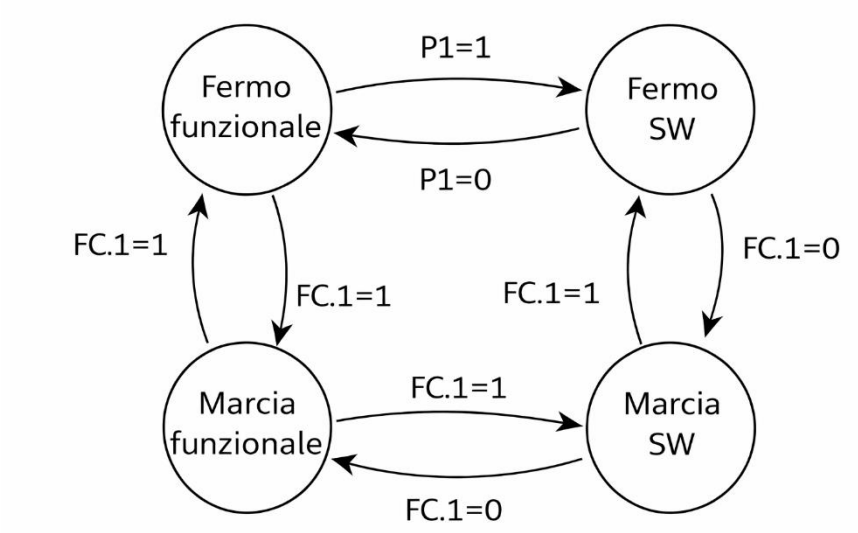
Effect:

- Machine that won't start.
- Or it starts in open safety condition.
- Misleading diagnostics.

The Gottardo Rule serves precisely to avoid this ambiguity between the physical and Boolean levels.

Finite state automaton representation

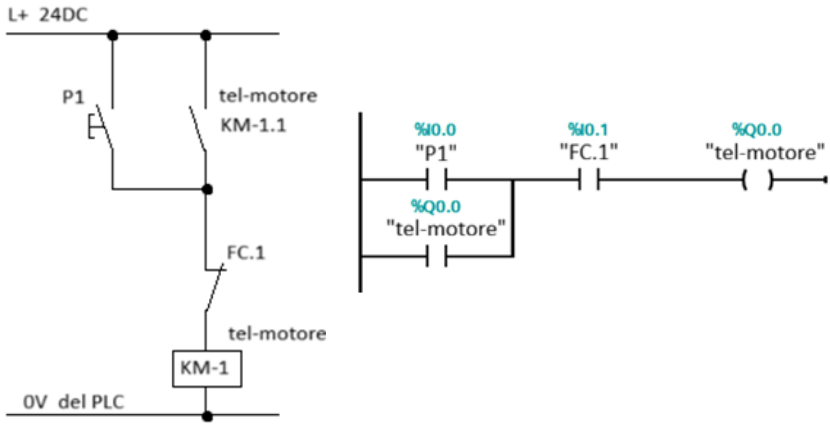
The finite state automaton represented in the graph formalizes the behavior of a circuit with **electromechanical self-restraint** when it is acquired within a programmable control system according to the principle called *the Gottardo rule*. This principle states that, if a contact is delegated to the release of a self-latching, its logical acquisition in the software takes place with reverse polarity with respect to the functional representation present in the pseudo-electromechanical scheme.



Regola del Gottardo

The system in question consists of a control circuit of a contactor, indicated by KM-1, equipped with self-latching made by means of an auxiliary contact of the same contactor in parallel with the start button P1.
The stop chain is made by means of the FC.1 contact placed in series with the reel.
In the electromechanical functional domain, the behavior of the system is bistable: the momentary activation of the start command energizes the contactor coil, which then maintains its state through

the self-latching contact; Opening the stop contact interrupts the current in the coil and returns to its idle state.
When such behavior is implemented within a PLC, the logic must be interpreted in light of the cyclic scanning pattern and the use of process images of inputs and outputs.
In this context, the automaton introduces a conceptual distinction between the **functional state of the physical circuit** and the **software state internal to the controller**, thus producing four distinct states that describe the combination of the two representations.
The states called *Functional Stop* and *Functional Run* describe the expected behaviour of the physical system, while the *SW Stop* and *SW Run states* represent the state of the internal logic variable that performs the self-latching in the PLC program.
In the operation of the automaton, pressing the P1 button determines a transition from the standstill functional domain to the corresponding software state, in which the self-latching logic is acquired by the controller.



This step reflects the fact that, during the scan cycle, activation of the input causes the assignment of the logical coil that implements self-latching.
Once this condition is established, the automaton evolves towards the running state, in which both the functional model and the

software model are consistent in representing the excitation of the contactor.
The presence of the FC.1 contact, on the other hand, introduces the self-locking release condition. In the functional domain, opening the contact immediately results in the interruption of the coil supply chain. However, in the software domain, this event is captured as a change in the input that causes the transition to the standstill state only through the update of the self-latching logic in the scan cycle. In this phase, the principle expressed by the Gottardo rule emerges: the contact that functionally represents the stop condition is treated in the software with a logical polarity opposite to the electrical representation, since the internal variable that maintains the self-latching must be disabled when the contact is open.
The automaton therefore represents a formalization of the semantic misalignment between the electromechanical representation of the circuit and its logical implementation in the PLC.
Through the distinction between functional states and software states, it is possible to rigorously describe the process by which the latching is established and subsequently released, highlighting how the acquisition of the inputs and the updating of the outputs in the scan cycle introduce a logical transformation that justifies the conceptual reversal of the stop contact. In this sense, the automaton provides a formal basis for interpreting the Gottardo rule as an emergent property of the execution model of programmable controllers and of the translation of relay circuits into software logic.

Critical analysis of the book (ISBN 9798345038970)

Technical evaluation, not editorial.

Strengths points

- Introduces logical formalization (counternominal, double negation).
- Connect physical wiring and Boolean algebra.
- Stimulates structured reasoning, not just "programming by imitation".
- Good for college students who need to understand why.

Overall Rating

From a technical point of view:

- Conceptually solid.
- Correct logic setting.
- Useful for technical vocational training.

From an educational point of view:

- Good for motivated students.
- It can be complex for those who do not have a basis in formal logic.
- It would have benefited from greater symbolic rigor and less rhetoric.

It is not an advanced industrial manual, but for the ITI/IPSIA/ITS target it is above the average of introductory texts.

Formalization according to IEC 61131-3

IEC 61131-3 defines the languages (LD, FBD, ST, SFC) and the cyclic execution model.

Logical model of self-restraint

Variables:

```
VAR
    Start     : BOOL;   // NO button
    StopNC    : BOOL;   // Input from physical NC contact
    Motor     : BOOL;   // Self-latching exit
END_VAR
```

Physical intake (Gottardo rule – book hypothesis):

- NC wired STOP physical contact.
- At rest → input = TRUE.
- Input → Enabled = FALSE.

Correct implementation in Structured Text

```
Motor := (Start OR Motor) AND StopNC;
```

This form is:

- deterministic,
- reset-dominant,
- Consistent with fail-safe logic.

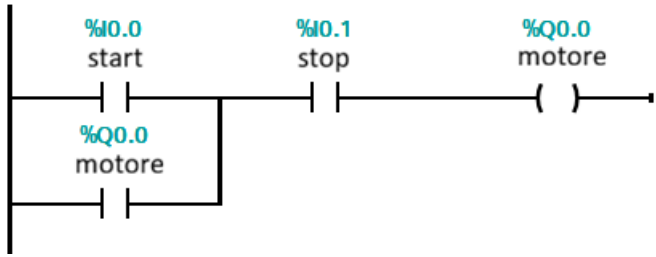
Incorrect implementation (violation)

```
Motor := (Start OR Motor) AND NOT StopNC;
```

Consequences:

- At rest → NOT StopNC = FALSE → Motor never starts.
- In stop condition → NOT StopNC = TRUE → possible dangerous start.

Correct equivalent ladder shape



Note: The StopNC contact is **NO in the ladder**, because it represents the logical level read, not the physical nature of the device.

Full Truth Table

We consider:

$$Q_{n+1} = (S \vee Q_n) \wedge I$$

Where:

- S = Start
- I = StopNC
- Qn = Previous status

Correct case S	Q_n	I (NC)	Q_{n+1}	Meaning
0	0	1	0	Stop
1	0	1	1	Startup
0	1	1	1	Retention
1	1	1	1	Retention
X	X	0	0	STOP always dominant

Fundamental property:

$$I = 0 \Rightarrow Q_{n+1} = 0$$

Reset key guarantee.

Incorrect case (software denial)

$$Q_{n+1} = (S \vee Qn) \wedge \neg I$$

S	Q_n	I	Q_{n+1}
0	0	1	0
1	0	1	0 □
0	1	1	0 □
1	0	0	1 □
0	1	0	1 □

Violations:

- Unable to start in normal state.
- Possible start with active STOP.

Analysis from a functional safety perspective

Regulatory references (Italy):

- EN ISO 13849-1
- IEC 62061

Relevant safety principle

Fail-safe principle:

Energy loss or circuit opening must lead to the safe state.

With NC contact:

1. Input → wire break = 0
2. PLC reads FALSE
3. AND cancels exit
4. Engine stops

If you violate the Gottardo Rule:

1. Input → wire break = 0
2. NOT 0 = 1
3. Possible consent to the march

This compromises:

- Circuit category
- Performance Level (PL)
- Functional integrity

Reset Dominance

In safety systems, the reset must be:

- priority,
- unmaskable,
- Non-self-resetting.

The correct form:

$$Q=Fset \wedge Fsafe$$

where:

$$Fsafe = \bigwedge_i Ii$$

Every safety input is in global AND.

Connection with safety PLC

In PLC fail-safe (es. CPU F):

- Safety contacts are treated as "high = safe" signals".
- Arbitrary negation in software breaks SIL/PL consistency.

The Gottardo Rule, in this sense, is not only didactic but consistent with certified architecture.

In-depth technical evaluation of the book

More detailed analysis.

Conceptual correctness

Correct interpretation of the report:

- Electrical level
- Logic level
- ladder representation

Correct reset-dominant setting

Formal rigorousness

Partially strict:

- Use counternominal correctly.
- But there is a lack of systematic formalization with tables of truth.
- The references to Hilbert's systems are theoretically correct but not formally developed.

Educational level

For college targets:

- Above average.
- Stimulates logical reasoning.
- Introduces concepts of applied Boolean algebra.

Limit:

- Sometimes redundant language.
- Formalism not always linear.

Overall Rating

From a technical-professional point of view:

It is a good operational didactic text with logical-formal ambition.

It is not an advanced functional safety manual, but it is well structured to:

- Understanding Self-Restraint,
- avoid typical errors in PLC,
- develop applied Boolean reasoning.

Formalization of the Gottardo Rule in Axiomatic Boolean Algebra

Definitions and Assumptions

Let it be defined as a self-retained controlled by:

- $S \in \{0,1\}$ — START command (NO)
- $F \in \{0,1\}$ — physical state of the NC trip contact
- $I \in \{0,1\}$ $I \in \{0,1\}$ — logical value read by PLC
- $Q_n \in \{0,1\}$ — Coil status at cycle n
- Q_{n+1} — state to the next cycle

Physical Hypothesis (Wired NC Contact):

$$I - F$$

with:

- $F=1 \rightarrow$ Contact closed (normal condition)
- $F=0 \rightarrow$ Open contact (release)

Correct self-restraint

Definizione:

$$Q_{n+1} = (S \vee Q_n) \wedge I$$

Boolean Algebra axioms used

Classical axioms (Huntington's) are considered:

1. Commutativity
   $A \vee B = B \vee A$
   $A \wedge B = B \wedge A$
2. Distributivity
   $A \wedge (B \vee C) = (A \wedge B) \vee (A \wedge C)$

3. Neutral elements
   $A \wedge 1 = A$
   $A \vee 0 = A$
4. Null elements
   $A \wedge 0 = 0$
   $A \vee 1 = 1$
5. Complementation
   $A \wedge \neg A = 0$
   $A \vee \neg A = 1$

Dominant Reset Property

Proposition 1
La funzione è reset-dominant rispetto a I.

Demonstration

Let it be $I = 0$.

$Q_{n+1} = (S \vee Q_n) \wedge 0$

By axiom 4:

$A \wedge 0 = 0$

then:

$Q_{n+1} = 0$

Regardless of S and Qn

Quod Erat Demonstrandum
Q.E.D.

Rule Violation

Suppose software denial:

$Q_{n+1} = (S \vee Q_n) \wedge \neg I$

Sia I = 1 (Normal condition).

$Q_{n+1} = (S \vee Q_n) \wedge 0 = 0$

The function loses the ability to set.

Let it be I = 0 (Release).

$Q_{n+1} = (S \vee Q_n) \wedge 1 = S \vee Q_n$

Reset is no longer dominant.

Formal conclusion:
Software negation destroys the dominance property of the reset.

Finite State System (FSM) Modeling

System definition

We define the state machine:

$M = (X,U,\delta)$

where:

- $X = \{OFF,ON\}$
- $U = \{S,I\}$
- $\delta : X \times U \rightarrow X$

2.2 Correct transition function

Current status	**S**	**I**	**Next status**
OFF	0	1	OFF
OFF	1	1	ON
ON	0	1	ON
ON	1	1	ON
*	*	0	OFF

Properties:

$I = 0 \Rightarrow \delta(x, S, I) = OFF$

The OFF state is absorbent compared to I = 0.

Incorrect case (software denial) **Status**	**S**	**I**	**Stato successivo**
OFF	1	1	OFF
OFF	1	0	ON

Dangerous transition is generated:

(OFF, S=1, I=0) → ON

The system is not monotonous with respect to the safety variable.

Formal property violated

We define safety properties:

I = 0 ⇒ X = OFF

In the wrong system this property is not invariant.

Dual-Channel Analysis – Category 3/4

Reference: EN ISO 13849-1

Architecture Category 3

Features:

- Two independent channels.
- Consistency monitoring.
- A single failure does not lead to the loss of the safety function.

We define:

- I_1 = channel A
- I_2 = channel B

Safety Function:

$F_{safe} = I_1 \wedge I_2$

Self-restraint:

$Q_{n+1} = (S \vee Qn) \wedge I_1 \wedge I_2$

Fault tolerance properties

Case: Wire break on channel A

$I_1=0$

Then:

$Q_{n+1} = 0$

Secure system.

Caso con inversione errata software
$Q_{n+1} = (S \vee Qn) \wedge \neg I_1 \wedge \neg I_2$

Wire break:

$I_1=0 \Rightarrow \neg I_1=1$

Possible consent to the march.

Category Loss.

Category 4

Requires:

- Redundancy
- Continuous diagnostics
- Fault accumulation detection

Logical model:

$F_{safe} = (I_1 \wedge I_2) \wedge D$

where D = diagnostics.

The Gottardo Rule is a necessary condition for maintaining the semantics "1 = safe".

Inverting in software alters the semantic mapping between:

- Electrical level,
- Logic level,
- PL certification level.

Engineering Conclusions

1. The Gottardo Rule formally guarantees:
    - dominant reset,
    - Invariance of the safe state,
    - Monotony with respect to the safety variable.
2. In Boolean algebra it is provable by classical axioms.
3. In the theory of automata is a property of the absorbing state[9].

1. In functional safety, it is a structural requirement to maintain PL and SIL category[10].

---

[9] In the context of automata theory, an **absorbing state** is a state of a finite-state automaton characterized by the property that, once reached, the system can no longer evolve to different states. Formally, given a deterministic automaton , a state is said to be absorbent if for every input symbol . This property implies that all transitions coming out of the state lead back to the state itself, making it a point of stability of the dynamical system described by the automaton. In modeling terms, absorbing states are frequently used to represent terminal conditions, conditions of non-recoverable error or equilibrium configurations from which no further evolution of the system behavior is expected.$A = (Q, \Sigma, \delta, q_0) q_a \in Q x \in \Sigma \delta(q_a, x) = q_a$

[10] **Performance Level (PL)** and **Safety Integrity Level (SIL)** are regulatory metrics used to quantify the level of reliability required of safety functions in control systems. The Performance Level, defined in ISO 13849-1, expresses the ability of a safety function to reduce risk through five discrete levels, indicated from PL to (lowest) to PL and (highest), determined on the basis of the average probability of dangerous failure per hour and the architectural structure of the system. The Safety Integrity Level, defined in IEC 61508 and also adopted in industry by IEC 62061, similarly represents the probability that a safety function will perform its task correctly when required; it is divided into four increasing levels, SIL 1–SIL 4, associated with quantitative intervals of probability of dangerous failure. Both classifications are tools for risk assessment and safety system design, allowing you to specify and verify the degree of integrity required of security functions.

Gottardo's thesis
Logical-Formal Foundations of the Gottardo Rule in PLC and Safety Systems

A discussion of Boolean algebra, automata theory, temporal logic, and functional safety

Abstract

The so-called ***Gottardo Rule***, formulated in the context of PLC programming with normally closed physical contacts delegated to the release of a self-retaining system, can be interpreted not only as a practical workshop rule, but as a structural property demonstrable in Boolean algebra, which can be modeled by finite state automata and verified by formal temporal logic.

In this paper it is demonstrated that this rule guarantees the dominance of the reset, the preservation of the safety invariant and the semantic consistency between the electrical level and the logic level, being a necessary condition — although not sufficient — for the maintenance of the safety requirements provided for by the EN ISO 13849-1 and IEC 62061 standards.

Logical structure of the self-restraint function

Consider a motor control system with a normally open start button and normally closed physical stop contact wired in series, read by a PLC compliant with IEC 61131-3.

We indicate with:

- S the start command,
- F the physical state of the NC contact,
- I the logical value read by the PLC,
- Qn the state of the coil at cycle n.

Since the contact is NC wired in the field, the variable read coincides with the real electrical state:

$I = F$

with the convention:

$F=1 \Rightarrow$ closed circuit (normal condition)
$F=0 \Rightarrow$ open circuit (trip)

The correct self-restraint function is defined by:

$$Q_{n+1} = (S \vee Q_n) \wedge I$$

This expression incorporates the classic latch structure with dominant reset implicit in the final conjunction.

Proof in axiomatic Boolean algebra

Classical Boolean algebra, based on Huntington's axioms, provides the tools for rigorous proof.

We want to prove that the previous function has the dominance property of the release.

Let it be $I = 0$. Then:

$Q_{n+1} = (S \vee Q_n) \wedge 0$

For the axiom of the null element:

$A \wedge 0 = 0$

follows immediately:

$Q_{n+1} = 0$

The output variable is independent of S and Qn. Reset is dominant.

Now let's analyze the function obtained by inverteding the contact in the software:

$Q_{n+1} = (S \vee Q_n) \wedge \neg I$

Putting I=0 we obtain:

$Q_{n+1} = (S \vee Q_{n)} \wedge 1 = S \vee Q_n$

The reset loses dominance. The security property is no longer an axiomatic consequence of the structure.

It follows that the Gottardo Rule is not a graphical convention, but a structural property of the Boolean function.

Modeling as a finite state system

The deterministic automaton is defined:

$M = (X, U, \delta)$

where the state space is X= {OFF, ON}, the set of inputs is U= {S, I} and the transition function is:

$\delta(x, S, I) =$

- OFF if I=0
- ON if $I=1 \wedge (S=1 \vee x=ON)$
- OFF otherwise

The fundamental property of the system is that the OFF state is absorbent with respect to the I=0 condition. Formally:

$$I = 0 \Rightarrow \delta(x, S, I) = OFF \ \forall x, S$$

If the improper denial of contact is introduced, the automaton admits the transition:

$$(OFF, S=1, I=0) \rightarrow ON$$

The safety condition is no longer invariant. The system loses monotony with respect to the safety variable.

Verification by time logic

The desired property can be expressed in Linear Temporal Logic (LTL).

Let the atomic proposition be defined:

$\text{Safe}: = (Q = \text{OFF})$

The fundamental property becomes:

$G(I = 0 \rightarrow \text{Safe})$

where G is the operator “globally”.

In the correct model, the formula is valid in all time paths.

In the wrong model, there is a path such that:

$$I = 0 \wedge S = 1 \wedge Qn = \text{OFF}$$

But:

$$Q_{n+1} = \text{ON}$$

The LTL formula is falsified. The rule can therefore be seen as a necessary condition for the validity of the safety invariant in temporal logic.

In terms of CTL:

$$AG(I = 0 \rightarrow AX(Q = \text{OFF}))$$

is true only in the correct formulation.

Extension to Category 3/4 dual-channel systems

In the regulatory framework of EN ISO 13849-1, Category 3 requires redundancy with fault detection.

Two independent channels I1 and I2 are introduced, both physical NCs.

The function becomes:

$$Q_{n+1}= (S \vee Q_n) \wedge I_1 \wedge I_2$$

Suppose a single fault with wire break on the first channel:

$$I_1=0$$

By axiomatic property:

$$Q_{n+1}=0$$

The safety function is preserved.

If you inverted the signals in the software:

$$Q_{n+1}= (S \vee Q_n) \wedge \neg I_1 \wedge \neg I_2$$

Wire break produces:

$$I_1= 0 \Rightarrow \neg I_1 = 1$$

Failure is no longer inherently safe. The structure loses fail-safe property and can compromise the Performance Level.

In Category 4, where fault accumulation detection is also required, semantic consistency "1 = safe condition" is critical for probabilistic analysis and correct implementation of diagnostic mechanisms.

Probabilistic analysis and relationship with SIL

In the context of IEC 62061, the probability of a dangerous failure depends on the ability of the system to automatically move into a safe state in the event of a fault.

With NC contacts and final AND function, the probability of dangerous failure is a function of the probability of simultaneous failure of the channels.

Negating in software, single break does not create certain shutdown. The probability of dangerous failure increases, modifying the PFHd and therefore the achievable SIL.

The Gottardo Rule thus becomes a structural condition for keeping the dangerous failure rate low.

General conclusions

The Gottardo Rule, far from being a graphic convention or a simple teaching habit, can be interpreted as:

an algebraic property of the dominance of the reset, a structural property of a deterministic automaton, an invariant that can be expressed in temporal logic, a necessary condition for maintaining the Performance Level in redundant systems.

Its validity is strict within the hypothesis of wired NC physical contact in the field. Apart from this hypothesis, the rule remains valid, on the basis of the counternominal proof, and must be reconsidered in the light of the semantics of the signal.