Edizioni G-Tronic
9RSAA

Gotthard-Regel

Angewandt in TIA PORTAL V21
Der erste Band
Erstausgabe 2027

Diese Veröffentlichung richtet sich an die wissenschaftliche Gemeinschaft und ist daher kein nützliches Lehrbuch zum Erlernen der Siemens SPS-Programmierung über das TIA PORTAL.
Ziel ist es, ein Axiom zu definieren, seine theoretischen Implikationen zu entwickeln und die abgeleiteten Theoreme durch eine monographische Ausgabe zu beweisen.

Geschrieben, herausgegeben und veröffentlicht

Von

ing. Prof. Dr. Marco Gottardo PhD

Publikationsreihe für industrielle Automatisierung.

Gotthard-Regel

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

ISBN-13: 9798193549772
Verlag:Independently published G-Tronic von Marco Gottardo Redakteur

Indice:

Vorwort

Diese Veröffentlichung ist nicht als populäres Werk oder Lehrbuch konzipiert, sondern als wissenschaftliche Abhandlung oder als monographisches Werk, das darauf abzielt, eine spezifische theoretische These einzuführen und zu diskutieren, die der wissenschaftlichen Gemeinschaft zur Bewertung dient.
Das Hauptziel der Arbeit ist es, folgendes Axiom vorzustellen und zu formalisieren: "Wenn ein Kontakt delegiert wird, einen selbstverschließenden Zustand zu lösen, wird er in der Software in einem Zustand erfasst, der dem im Funktional dargestellten ist."
Ziel der Diskussion ist es, diese Aussage zu analysieren, ihre logische Kohärenz zu überprüfen und sie als Axiom innerhalb des vom Autor dargelegten theoretischen Rahmens vorzuschlagen. Die Formalisierung des Axioms stellt den ersten Schritt hin zu einer möglichen breiteren Strukturierung des konzeptuellen Modells dar, innerhalb derer die Proposition anschließend entwickelt und in Form eines Theorems demonstriert werden kann.
Das vorliegende Werk beabsichtigt außerdem, eine klare und formal erkennbare Formulierung der vorgeschlagenen These zu etablieren[1], mit dem Ziel, deren Urheberschaft explizit und nachverfolgbar im Kontext der wissenschaftlichen Literatur zuzuschreiben.
Obwohl eine bewusst einfache und direkte Sprache verwendet wird, bewahrt der Inhalt das Maß an terminologischer Präzision und argumentativer Strenge, das in wissenschaftlichen und akademischen Bereichen erforderlich ist, um die Klarheit der Darstellung zu gewährleisten, ohne die Formalität des theoretischen Diskurses zu beeinträchtigen.

Notizen für Ihre Abschlussarbeit oder wissenschaftliche Arbeiten
Für eine ethische und korrekte Verwendung des Inhalts des Textes muss jeder, der davon in seinen Veröffentlichungen und Doktorarbeiten profitiert hat, die übliche Zitierung in der Bibliographie im Standardformat angeben:

[1] M. Gottardo, *Regola del Gottardo*, Vigonovo (VE): G-Tronic Edizioni, 2026. ISBN: [ISBN einfügen].

Benachrichtigen Sie den Autor über die Zitation und senden Sie eine E-Mail an ad.noctis@gmail.com , in der Name, Titel, Datum und Herkunftsuniversität des Autors angegeben sind.

[1] **Thesis:** Formulierung zur Definition der logischen Zustände der Freigabekontakte der zurückgehaltenen Fahrzeuge basierend auf dem in den funktionalen Diagrammen angegebenen Zustand. Wenn ein physischer Kontakt im Feld delegiert wird, um ein Selbstverschluss freizugeben, wird dies in der Software übernommen und nicht in der Funktion dargestellt. Daher werden klare und klar definierte Regeln für die funktionale Darstellung benötigt.

Formale Darstellung des Problems

Bei diskreter industrieller Steuerung realisiert die selbsthaltende oder Dichtungsfunktion einen bistabilen Speicher, der von zwei Eingängen gesteuert wird:

- S: Startkontrolle (SET)
- R: Stopp-Befehl (RESET)
- Q: Austrittsstatus (Spule)

Im PLC- (Leiter- oder FBD-) Formalismus lautet die klassische Struktur:

- KEIN Kontakt, normalerweise offen auf dem Feld, von Start
- NC-Kontakt, normalerweise geschlossen im Feld, von Stop
- Selbstverriegelnder KEIN Kontakt parallel zum Start
- Ausgangsspule

Ein erstes Axiom, das für die Anwendung der Gotthard-Regel (Lehrterminologie, die im italienischen technischen Bereich weit verbreitet ist) notwendig ist, belegt:
Bei gleichzeitiger Anwesenheit der Anregungs- und Entspannungsbefehle muss die Entspannung vorherrschen.
In logischen Worten:

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

Dies ist eine Reset-Priority-Regel, die aus Gründen der funktionalen Sicherheit entscheidend ist.

Axiomatisierung in der Boolesche Algebra
Betrachten wir axiomatische Boolesche Algebra:

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

mit den folgenden fundamentalen Axiomen:

1. **Kommutativität**
$$\boldsymbol{A} + \boldsymbol{B} = \boldsymbol{B} + \boldsymbol{A}; \boldsymbol{A} \cdot \boldsymbol{B} = \boldsymbol{B} \cdot \boldsymbol{A}$$

2. **Assoziativität**
$$(\boldsymbol{A} + \boldsymbol{B}) + \boldsymbol{C} = \boldsymbol{A} + (\boldsymbol{B} + \boldsymbol{C})$$

3. **Distributivität**
$$\boldsymbol{A} \cdot (\boldsymbol{B} + \boldsymbol{C}) = \boldsymbol{A} \cdot \boldsymbol{B} + \boldsymbol{A} \cdot \boldsymbol{C}$$

4. **Komplementierung**
$$\boldsymbol{A} + \boldsymbol{A}' = \mathbf{1}; \boldsymbol{A} \cdot \boldsymbol{A}' = \mathbf{0}$$

5. **Identität**
$$\boldsymbol{A} + \mathbf{0} = \boldsymbol{A}; \boldsymbol{A} \cdot \mathbf{1} = \boldsymbol{A}$$

Formalisierung der Selbstbeschränkung
Rekursive Zustandsgleichung

Der Status der Ausgabe zu diesem Zeitpunkt $\boldsymbol{k} + \mathbf{1}$**è:**

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

Dies ist die kanonische Formalisierung der Selbstbeherrschung mit Priorität zum Zurücksetzen.
Interpretation:

- $\boldsymbol{S} + \boldsymbol{Q}_{\boldsymbol{k}}$→ Anfängliche Erregung oder Erhaltung
- $\boldsymbol{R}'$→ Nicht-Reset-Bedingung

Erste Demonstration der Gotthard-Regel

Wir möchten es formell demonstrieren:

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

Wir setzen in die Gleichung ein:

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

Da:

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

e

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

Mehr:

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

Bewiesen.

Strukturelle Eigenschaft: Reset-Priorität

Zeigen wir nun, dass die Funktion äquivalent zu einer expliziten Form der Priorität ist:

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

Anwendung der Distributivität:

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

Sammeln:

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

Anmerkung:
Das ist es $\boldsymbol{R = 1 \Rightarrow R' = 0}$, dann:

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

unabhängig davon, ob $\boldsymbol{Q_k}$.
Dies ist die mathematische Formalisierung der Gotthard-Regel.

Vergleich mit dem klassischen SR-Riegel

Unpriorisierte SR-Verschlüsse:

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

Das ist es $\boldsymbol{S = R = 1}$:

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

Hier gilt das EETS-→ nicht mit der Gotthard-Regel.

Kanonische Form von Shannon

Wir wenden Expansion bezüglich auf R:

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

Wir berechnen:

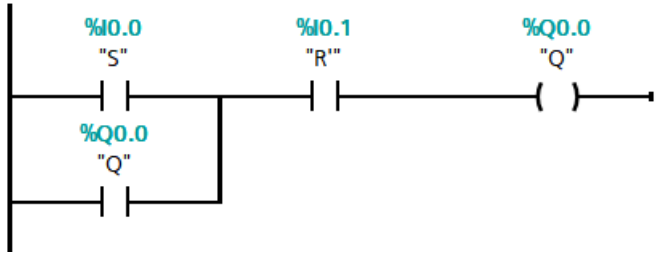
- Se $R = 1 \Rightarrow Q_{k+1} = 0$
- Se $R = 0 \Rightarrow Q_{k+1} = S + Q_k$

Also:

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

Minimale Form

Interpretation in der Siemens PLC (TIA Portal)
In der TIA-Ladder:



Die Struktur implementiert genau:

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

In der Scan-Schleife:

1. Eingangsmessung
2. Logische Bewertung
3. Ausgabebild-Update

Die Priorität des Zurücksetzens ist strukturell garantiert, nicht zeitlich.

Demonstration der Systemstabilität

Schauen wir nach stationären Zuständen:

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

Fall 1: $R = 1$

$$\boldsymbol{Q = 0}$$

Einzigartiges Gleichgewicht.
Fall 2: $R = 0$

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

Das ist es $\boldsymbol{S = 0 \Rightarrow Q = Q}$(Erinnerung)
Das ist es $\boldsymbol{S = 1 \Rightarrow Q = 1}$
Konsistentes System.

Sicherheitssatz (Formulierung)
Satz:
Die boolesche Funktion

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

ist die einzige minimale dreidimensionale Funktion, die erfüllt:

1. Selbstbeherrschung
2. Tasten-Reset
3. Abwesenheit verbotener Staaten
4. Idempotenz bezüglich $R = 1$

Beweis (synthetisch):
Das Auferlegen der Nebenbedingungen in der Wahrheitstabelle und deren Minimierung mit Karnaugh-Abbildungen ergibt ein einziges Minimum, das eine Abdeckung entspricht der obigen Form impliziert.

Erweiterung: Matrixform im Zustandsraum
Wir können schreiben:

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

Diskretes nichtlineares dynamisches System auf Booleschem Feld:

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

Strukturell äquivalent zu:

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

was ein erstordnungslogisches System mit dominanter Auslöschungs-Eingabe darstellt.

Didaktische Anwendungssynthese

Die Gotthard-Regel ist kein grafisches Trick auf der Leiter, aber:

- eine strukturelle Eigenschaft der Boolesche Funktion
- eine Prioritätsbedingung im diskreten Bereich
- Eine funktionale Sicherheitsbedingung
- Eine Designentscheidung, die mit den Maschinensicherheitsvorschriften übereinstimmt

Seine axiomatische Formalisierung erlaubt:

- Formale Analyse
- Beweis der Fairness
- Symbolische Verifikation
- Automatische Synthese

Feldanwendung beginnend mit Eingabe-Terminalblöcken.

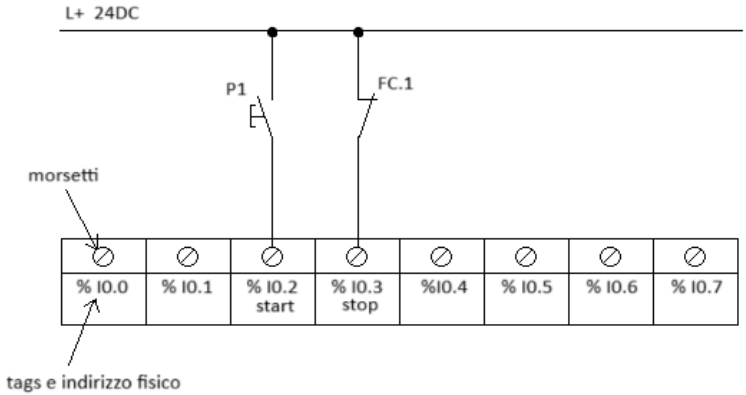
Die Arbeit des Programmierers beginnt mit einer Interviewphase, in der die technischen Spezifikationen des zu bauenden Systems gesammelt, dokumentiert und durch die Unterschriften des Kunden formalisiert werden.

Diese sollten sich im Verlauf der Arbeit nicht ändern, was jedoch pünktlich geschieht und die bereits geleistete Arbeit stört.

Wenn Sie mit der Programmierung des Systems oder der Maschine beginnen, müssen Sie dem Programmierer zwei Dokumente vorlegen: den Schaltplan und die P&ID.

Die erste zeigt die Verbindungen zwischen der Sensortechnik und den Eingängen der SPS sowie zwischen den Ausgängen und den Aktuatoren oder deren Steuergeräten (Relais, Kontaktoren, Magnetventile usw.).

Das Bild zeigt einen Auszug aus einem typischen Schaltplan,[2] der dem Programmierer in der Anfangsphase der Entwicklung des Werks zur Verfügung gestellt wurde.



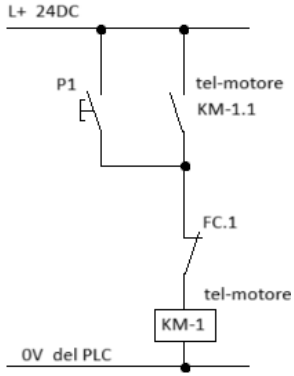
Knopf P1 bringt das Steuersignal zum Eingang %I0,2,

dann das dritte Bit des ersten Eingangsklemmenblocks, während der elektromechanische Endschalter mit dem nächsten digitalen Eingang %I0.3 verbunden ist.

Der Programmierer realisiert wahrscheinlich einen "selbstverschließenden Befehl", der die Bewegung bis zur zugewiesenen Grenze aufrechterhält.

[2] Im Schaltplan werden die Ein- und Ausgänge als Rechtecke dargestellt, organisiert in Reihen zu je acht oder einem Byte. Sie haben ein Adressfeld und ein Feld für die Tags (symbolisches Label der Variablen). Auf den ersten Blick gilt: Wenn das Rechteck am unteren Rand der Seite steht, sind es Eingaben, wenn es oben ist, sind es Ausgaben.
Interne Variablen wie Merker und die in den Datenbanken definierten sind in den Schaltplänen nicht dargestellt.

Was in dieser Zeichnung gezeigt wird, sollte nicht im Schaltplan erscheinen:

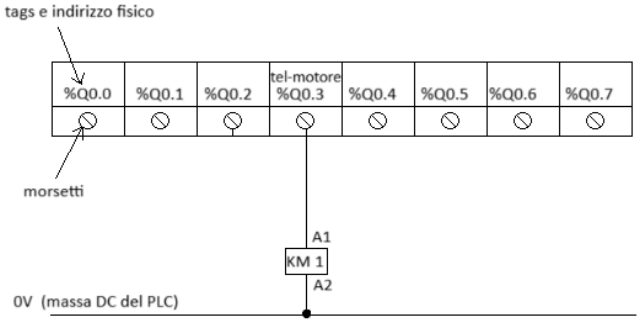


Dies erscheint im Schema nicht, da es die intern implementierte Logik darstellt[3].
Es handelt sich um einen elektromechanischen Flip-Flop, der in der Lage ist, ein Bit bis zum Unhooking-Ereignis zu speichern.
In diesem Fall wird der monostabile[4] Laufwerksbefehl im internen KM-1-Bit gespeichert, vorzugsweise innerhalb eines Data Block deklariert, statt als Merker in der Tags-Tabelle.

Wenn die P1-Taste gedrückt wird, wird das positive Signal bei 24V DC zum darunterliegenden Knoten übertragen.
Am Knoten trifft es auf ein normalerweise geschlossenes, verdrahtetes elektromechanisches Gerät, dann passiert das Signal und aktiviert die Spule.
Die unter Strom stehende Spule schließt ihren Kontakt, der parallel zum Steuerknopf angebracht ist.
Der Bediener kann den Knopf loslassen, aber die Rolle wird von ihrem eigenen Kontakt gehalten, der parallel angebracht ist.
Ein beweglicher Teil der Maschine fängt FC.1 ab, der durch das Durchtrennen der Leitung das gehaltene Auto freisetzt.
Wie wir in mehreren Teilen des Textes sehen werden, ist diese Symbolik, obwohl sie starke Ähnlichkeiten aufweist, kein elektrisches Diagramm und trägt den Namen funktionales Diagramm.
Viele Dinge, die für Schaltpläne verpflichtend wären, sind in Funktionsdiagrammen vernachlässigbar, zum Beispiel viele Nummerierungen.

[3] In der Informatik ist Implementierung gleichbedeutend mit Realisieren, also Programmierung.
[4] Eingangssignale können monostabil oder bistabil sein. Monostabile Signale sind impulsiv, wie sie von Knöpfen erzeugt werden, die den WAHREN Zustand nur so lange erzeugen, wie der Bediener mit dem Finger bleibt. Bistabile Signale hingegen sind typisch für Schalter. Jede Aktion entspricht einem Übergang von einem stabilen Zustand zum komplementären, wobei sie bis zu einer weiteren Aktion des Operators bleibt. In der Automatisierung wird die monostabile Handlung bevorzugt und jede in der Software geschaffene Selbstbeschränkung, um die Automaten bei Eintrittsbedingungen selbstfunktionsfähig zu machen. Darüber hinaus eignen sich selbsthaltende Systeme besser für die automatische Freigabe während Sicherheitsmaßnahmen.

Es macht keinen Sinn, ein Kabel zu nummerieren, das tatsächlich nicht existiert, da es durch Software dargestellt wird.
Die gleichen Verbindungen haben eine direkte Entsprechung zur Software, zum Beispiel stellt die Parallele die Boolesche Operation[5] ODER dar und die Reihe die Operation AND.
Das Verdrahtungsschema der Ausgänge gibt der Verbindung der Kontaktorspule Bedeutung.



In diesem Diagramm muss die Leitungslänge zwischen Anschluss %Q0,3 und Anschluss A1 des Förderers nummeriert werden, auch wenn sie hier der Einfachheit halber nicht enthalten ist.
Die gleichen Terminals A1 und A2 werden im realen Diagramm im Standardmodus als "durchgestrichener Punkt" dargestellt, zu dem die mit dem Rest des Diagramm übereinstimmende Formulierungen kombiniert werden, zum Beispiel X1-1 usw.

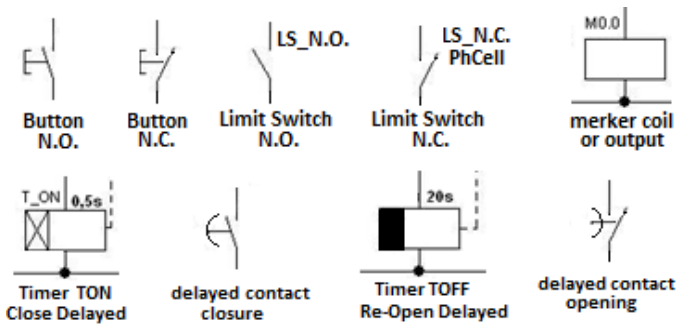
[5] Boolesche Operationen sind solche, die in der Boolesche Algebra betrachtet werden. George Boole war ein englischer Mathematiker und Logiker, der 1815 geboren wurde, lange vor dem Aufkommen der Computer. Er wandte die Normalalgebra auf Basis-Zwei-Nummerierungssysteme an und postulierte einige fehlende und wesentliche Elemente, wie die Tautologie, die die Vereinfachung redundanter Signale ermöglicht. Um ein Beispiel zu geben: Der Ausdruck "heute regnet es oder es regnet nicht" ist eine Tautologie, die durch das parallele "ODER" der beiden Möglichkeiten erreicht werden kann, die mit demselben Kontakt in der normal offenen oder normal geschlossenen Situation ausgedrückt werden können. Da diese Parallele immer wahr ist (TRUE), kann sie eliminiert oder besser durch eine Leitung ersetzt werden, die von der +24V-GLEICHSTROMLEITUNG stammt. Sie ist direkt mit der Spule des Ausdrucksschließrelais verbunden. Die wichtigste der Boolesche Minimierungen wird somit implementiert, wie man in der Karnaugh-Kartentechnik finden kann. Ein Kompendium der Grundregeln der booleschen Logik ist der Beitrag eines anderen Wissenschaftlers, De Morgan, der die Gleichungen und damit die Austauschbarkeit zwischen ODER und UND einführt, wenn sie den entsprechenden Signalnegationsbedingungen unterliegen.

Die grundlegenden Elemente und Symbole der funktionalen Logik.

Funktionale Logik beschreibt die Maßnahmen, die der Controller ausführen muss, und ist eine nützliche Unterstützung für die Softwareentwicklung. Die grundlegenden Elemente sind:

1. Die Stromleitung ist oben horizontal, angezeigt durch L+ oder L, wenn sie durch Gleich- oder Wechselstrom betrieben wird. Da es sich nicht um eine elektrische Schaltung handelt, sondern um einen Algorithmus, der in der internen Logik der SPS über die Software implementiert wird, ist es sinnvoller, immer kontinuierlich zu sein.
2. Rückleitung, angezeigt durch M (Masse als Abschluss einer L+-Leitung) oder N (Neutralleiter als Abschluss einer Wechselleitung).
3. Die den logischen Fluss nach unten ableiten. Im funktionalen reellen Segment beginnt jedes Segment mit einem Abstieg und leitet sich nicht horizontal vom vorherigen Segment ab.
4. Schließender Kontakt (d. h. normalerweise offen verdrahtet), mit einem trockenen Kontakt, der links vom Abstieg gezogen wird.
5. Öffnungskontakt (d. h. normalerweise geschlossen mit Draht), mit einem trockenen Kontakt, der rechts vom Abstieg gezogen wird.
6. Zeitgesteuerte Spule verzögert bei der Anregung und ihrem Kontakt beim Öffnen und Schließen.
7. Die zeitgesteuerte Spule verzögert sich bei der Entspannung und ihrem Kontakt beim Öffnen und Schließen.
8. Zähler, dargestellt als rechteckige Lastspule, aber mit Eingängen nach oben, runter, setzen und zurücksetzen. Der voreingestellte Wert muss ebenfalls dargestellt werden.

Die Symbole, die für das Schreiben eines Funktionals nützlich sind, sind im untenstehenden Bild zusammengefasst. Wie wir sehen können, sind Dreiecke an den Endanschlüssen, wie z. B. die schließenden Stäbe, nicht wesentlich, da die verschiedenen Kontakte nur den Boolesche Zustand eines Bits innerhalb oder im Klemmenblock darstellen, nicht das elektromechanische Objekt im Feld.

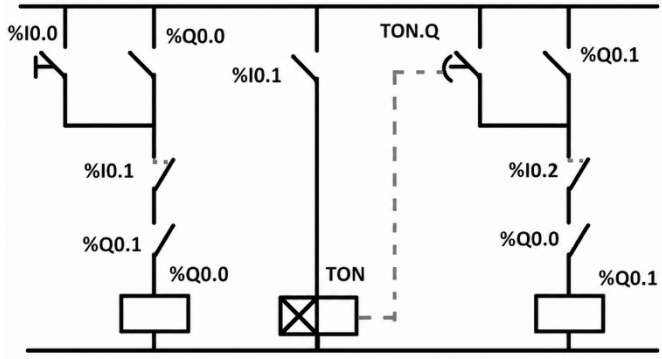


Da in diesem Zusammenhang das Funktionsdiagramm kein Schaltplan, sondern ein Algorithmus ist, der zur Implementierung des Ladder-Programms zu befolgen ist, sind die Anzahl der Symbole und die Zeichnungsregeln stark vereinfacht.

Erstens stellen die im Funktional vorhandenen Symbole die Signale dar, die elektromechanische Objekte liefern.

Das Funktionsdiagramm ist eine schematische und intuitive Darstellung des Programms, die mit der Software implementiert werden muss. Es erfüllt dieselbe Funktion wie das Flussdiagramm [6]in hochrangigen Sprachen.

Nur in einer ersten Näherung stellt die funktional gedrehte 90-Grad-Drehung die Implementierung der Software dar. Im Allgemeinen stimmt das nicht. In diesem Zusammenhang sollte die **Gotthard-Regel berücksichtigt werden.**

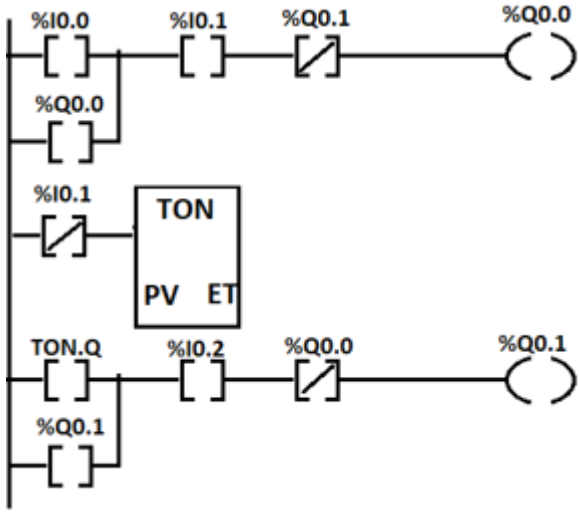


Gottardo-Regel: Wird einem Kontakt* die Öffnung eines Selbsthaltekontakts zugewiesen, wird dies in der Software entgegen der in der Funktion dargestellten Logik erfasst.

*Es ist ein feiner, elektromechanischer Öffnungskontakt gemeint. Wenn dasselbe Signal durch Virtualisierung oder in irgendeiner Form simuliert wird, wird die Regel nicht bestätigt, wenn der eventuelle Öffnungskontakt nicht real vorhanden ist, d. h. es handelt sich nicht um einen elektromechanischen Öffnungskontakt.

[6] Flussdiagramme sollten in der Entwicklungsphase von SPS-Programmen nicht angewendet werden, da sie unzureichend sind. Diese werden, wo möglich, durch Funktionsdiagramme ersetzt. Die Symbolsätze für Flussdiagramme sind oval für Abflüge und Ankünfte, Rechtecke verschiedener Art für Aktionen, Diamanten für IF-typische Entscheidungsträger, Pfeile zur Anzeige der Flussrichtungen. Analog dazu haben funktionale Diagramme ihre eigenen begrenzten Symbole: den offenen Kontakt links vom Abstieg, den geschlossenen Kontakt rechts vom Abstieg, die Spule, die zeitgesteuerten Spulen, während die Logiken mit Parallelen für die ORs und Reihen für die ANDs erstellt werden.

Das vorgeschlagene Funktionsdiagramm führt zur Umsetzung des unten standardisierten Codes IEC 61131.
Die Aufmerksamkeit sollte auf die Kontakte %I0,1 und %I0,2 gelenkt werden, die im Funktional auf der rechten Seite des Abseils angezeigt sind, also normale 1 –> 0 gedrückt.
In diesen Leitersegmenten hingegen sind sie normalerweise offen, was in der erwähnten "Gotthard-Regel" erklärt wird.[7]
Die Notwendigkeit dieser Regel entsteht nach dreißig Jahren Erfahrung im Bildungsbereich und ist nicht weniger trivial als die Behauptung, dass zwei parallele Linien niemals aufeinandertreffen.



Hinweis zur intrinsischen Sicherheit:
Ein selbstverriegelndes System ist von Natur aus sicher, wenn der Bruch oder das Fehlen des Endschalters dem Controller denselben Status wie der Auslöser sendet.

Demonstration mit kontranominaler Technik : Für die Gotthard-Regel wenden wir die kontranominale Beweistechnik an, die darin besteht, mit der Negation der These zu beginnen, um zur Negation der Hypothese zu gelangen, und nicht zu einem Widerspruch zur Hypothese.
Beide **Thesen** sind die Notwendigkeit, den Freigabekontakt des Sicherungswagens in Bezug auf dessen Darstellung im funktionalen Bereich zu verweigern.

[7] Bis das Gegenteil bewiesen ist, nutzt der Autor den Status eines Doktoranden, um dessen Existenz und den hier formulierten Nachweis zu validieren.

Die Negation der These ist, dass der Zustand des physischen Kontakts auf dem Spielfeld nicht geleugnet wird.

Die **Hypothese** besagt jedoch, dass die physischen Kontakte auf dem Feld, die zur Freigabe des Sicherungswagens delegiert sind, normalerweise geschlossen mit Draht verdrahtet sind.

Nehmen wir an, absurd, dass der Staat im Bereich des physischen Kontakts dem Abhaken übergeordnet ist.

Es ist zu beobachten, dass kein Signal das zum Lesen zugewiesene Terminal erreicht.

Da dies gemäß der internen Logik der Software mittels einer Booleschen Kombination AND verbunden ist, wird das kombinatorische Ergebnis, unabhängig vom Zustand des Startknopfs, null sein, was verhindert, dass die Spule sich in selbsthaltend einhakt.

Die **Negation der Hypothese** ist, dass der Zustand des an den Sicherungswagen delegierten Auslöserkontakts nicht in Bezug auf die Darstellung im funktionalen Zustand abgelehnt werden sollte.

Indem der Zustand dieses Kontakts nicht leugnet wird, implementiert er den Boolesche Doppelnegationssatz[8] der Aussage "not(not(unhooked) = unhooked", und daher startet der Motor nicht, weil er unhooked ist.

die doppelte Negation erlaubt es, eine Negation aus dem Ausdruck zu eliminieren, d. h. den Zustand des Kontakts im Leiter Schritt 7 offen statt geschlossen zu platzieren, was die Gotthard-Regel endgültig bestätigt, bis das Gegenteil bewiesen ist.

Q.E.D.

Bitte beachten Sie: Die Hypothese erwähnt "geschlossene Kontakte im Feld", daher sollte dies nicht zutreffen, wenn die genannten Kontakte nicht existieren, also wenn sie in einem HMI-Panel virtualisiert werden.

Beweis des Gegenteils: Der ineinandergreifende Kontakt sollte nicht umgekehrt werden, da er die ursprünglichen Annahmen durch den Beweis der These nicht erfüllt.

[8] Nachweisbar sowohl durch Hilberts deduktive Systeme als auch in der klassischen Aussagenlogik, selbst mit alleiniger Hilfe von Tautologien. Nach den Regeln der doppelten Negation, die in booleschen und kombinatorischen Algebren theoremisiert werden, erlaubt die doppelte Negation es, eine Negation aus dem Ausdruck zu eliminieren und so die Gotthard-Regel endgültig zu validieren, bis das Gegenteil bewiesen ist.

Korrekte technische Aussage

Im Falle eines **physischen Kontakts eines normalerweise geschlossenen verdrahteten (NC) elektromechanischen Endschalters**, der der **Freigabe eines selbsthaltenden** Geräts zugeordnet ist, gilt folgende Regel:

In der SPS-Software muss der Kontakt als negiert in Bezug auf seine funktionale Darstellung programmiert werden.

In der Praxis:

- Im Feld → NC-Kontakt (im Ruhezustand geschlossen).
- In Ladder muss → als normal offener Kontakt (—| |—) eingefügt werden, die mit dem Eingabebit verbunden ist.

Das liegt daran, dass die SPS **elektrische Logikpegel ausliest**, nicht funktionale Symbolik.

Analyse der Notwendigkeit der Inversion

Realer physikalischer Zustand (NC-Endschalter)

Physikalischer Zustand	**PLC-Anschluss**	**Bit di ingresso**
Nicht gepresst	1	TRUE
Gepresst	0	FALSE

Aber funktional gesehen:

- Gedrückt = Trip-Bedingung
- Nicht gedrückt = Maschine aktiviert

Daher wird die funktionale Bedeutung bezüglich der logischen Ebene, die gelesen wird, umgekehrt.

Korrekte Form der Selbstbeherrschung

Korrekte Gleichung:

Q_{n+1}= (START ∨ Q_n) ∧ EINGABE

Wobei:

- INPUT ist das Bit, das von der SPS (verkabelte NC) gelesen wird.
- Wenn der Endschalter → INPUT = 0 → gedrückt wird, hebt das AND alle garantierten Freigaben → auf.

Wenn der Programmierer dies in der Software ablehnt:

Q_{n+1}= (START ∨ Q_n) ∧ ¬EINGABE

dann:

- Ruhe → EINGABE = 1 → ¬1 = 0 → Die Maschine springt nie an.
- Gepresst → EINGABE = 0 → ¬0 = 1 → Die Maschine kann mit aktiviertem Endschalter starten (schwerwiegender Fehler).

Und genau das ist das Herzstück der im Text beschriebenen kontranominalen Demonstration.

Korollar: Q_{n+1} repräsentiert den Zustand, den die Ausgabe im Scanzyklus nach der Auswertung des logischen Zustands des Speicherstandorts annimmt, wie von der SPC-Laufzeit vorgesehen. In SPS-Systemen, die dem IEC 61131-3 Sprachausführungsmodell entsprechen, werden Eingaben zu Beginn des Scanzyklus abgetastet und in das Prozessbild der Eingaben kopiert, während Ausgabezuweisungen das Prozessbild der Ausgaben während der Programmausführung aktualisieren. Erst am Ende des Zyklus wird das Bild der Ausgänge an die physischen Anschlüsse übertragen. Das bedeutet, dass das Programm den aktualisierten Zustand einer Spule im selben Zyklus beobachten kann, während das Feld ihn im nächsten Zyklus sieht.

Logische Übersetzung des Beweises

Hypothese: Körperkontakt ist im Feld kein Kontakt.

Thesis: Sie sollte in der Software nicht abgelehnt werden.

Gegennominale Demonstration:

- Wenn du es in der Software ablehnst,
- Dann würde man verhindern, dass der Verriegel einrastet, oder die Maschine im Freigabezustand aktivieren,
- daher wäre die Hypothese korrekter Funktionsweise falsch,
- daher darf es nicht geleugnet werden.

Der Verweis auf doppelte Negation:

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

Sie dient lediglich dazu, zu formalisieren, dass:

Physischer NC + KEIN Kontakt auf der Leiter = implizite doppelte Negation → konsistentes Verhalten.
Wenn es NICHT zutrifft

Die Anmerkung im Buch ist korrekt:

Die Regel gilt **nur, wenn der Kontakt physisch, elektromechanisch, NC oder im Feld verdrahtet ist**.

Sie gilt nicht, wenn:

- Das Signal ist virtuell (HMI).
- Das Signal wird simuliert.
- Körperkontakt ist NEIN.
- Das sind interne Logikverriegelungen.

Betriebszusammenfassung (vom Anlagentechniker)

Wenn die Sicherheitsvorrichtung:

- Verkabelte physische NC → NICHT in der SPS umkehren.
- KEINE physische → nach Funktion bewertet.
- Virtuell → keine Herrschaft des Gotthard.

Wir können nun die folgenden Punkte entwickeln:

- den Beweis in einer rigorosen Form reiner propositionaler Logik (ohne Erzählung) umzuschreiben,
- Zeigen Sie den Fall mit dem TON-Timer wie im oben dargestellten Diagramm
- Analysieren Sie den typischen Fehler im TIA Portal, wenn funktionale Darstellung und elektrischer Zustand verwechselt werden.

ohne Beeinträchtigung der Gotthard-Regel, die bereits im Text mit ISBN enthalten ist: 9798345038970, lassen Sie uns die Analyse aller drei genannten Punkte weiterentwickeln.

Lassen Sie uns als Axiom die Formulierung der Gotthard-Regel beibehalten, die bereits in vielen Texten vorhanden ist, und gehen wir strukturiert mit den drei erforderlichen Punkten fort.

Beweis in der strengen Aussagenlogik

Formale Definitionen

Lass es so sein.:

- F = physikalischer Zustand des NC-Kontakts im Feld
  F=1 → geschlossener Stromkreis (Normalzustand)
  F=0 → offener Stromkreis (Endschalter gedrückt)
- I = von PLCENC verkabelt gelesenes Bit:

$$I=F$$

- Q = Selbstverriegelnder Ausgang
- S = start

Korrekte Selbstbeherrschung:

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

These (Gotthard-Regel)

Es lässt sich das "I" in Software nicht leugnen.

Gegennominale Demonstration

Angenommen, die These ist falsch:
Du verweigerst den Kontakt in der Software.

Dann:

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

Fall 1 — Normalzustand (F = 1 → I = 1):

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

Die Fixierung kann nicht → nicht konformes Verhalten angebracht werden.

Fall 2 — Aktiver Endschalter (F = 0 → I = 0):

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

Die Maschine kann in einem auslösenden Zustand → schwerwiegenden Funktionsverletzung starten.

Daher impliziert die Negation der These die Negation der Hypothese der korrekten Funktionsweise.

Q.E.D.

Fall mit TON (wie im beigefügten Diagramm)

In deinem Diagramm **erscheint eine MENGE,** die das Andocken oder Aktivieren verzögert.

Wir erinnern dich daran:

TON.Q=1 erst danach PT mit IN = 1

Wenn der NC-Endschalter in der Software abgelehnt wird:

- Unter normalen Bedingungen startet → IN = 0 → TON nie.
- Bei Reisebedingungen → IN = 1 → TONN, obwohl es nicht sein sollte.

Folgen:

- Umgekehrte Zeiten.
- Selbstverriegelung, die mit dem betätigten Endschalter einrastet.
- Reiseverzögerungen statt Startverzögerungen.

Bei Timern wird der Verstoß noch heimtückischer, weil er inkonsistente Zeitdynamiken einführt.

Typischer Fehler im TIA-Portal

Häufiger Fehler bei Schülern:

Verwirrend:

- Funktionales Grafiksymbol (NC gezeichnet)
- Logikstatus, der von SPS gelesen wird

Im TIA:

1. Kontakt —|/|— bedeutet nicht "physischen Kontakt mit Kontaktkontakt".
2. Es bedeutet "verweigerte logische Bedingung".

Viele Programmierer:

1. NC-Verkabelung im Feld.
2. Sie setzen —|/|— auf die Leiter.
3. Sie bekommen doppelt falsche Umkehr.

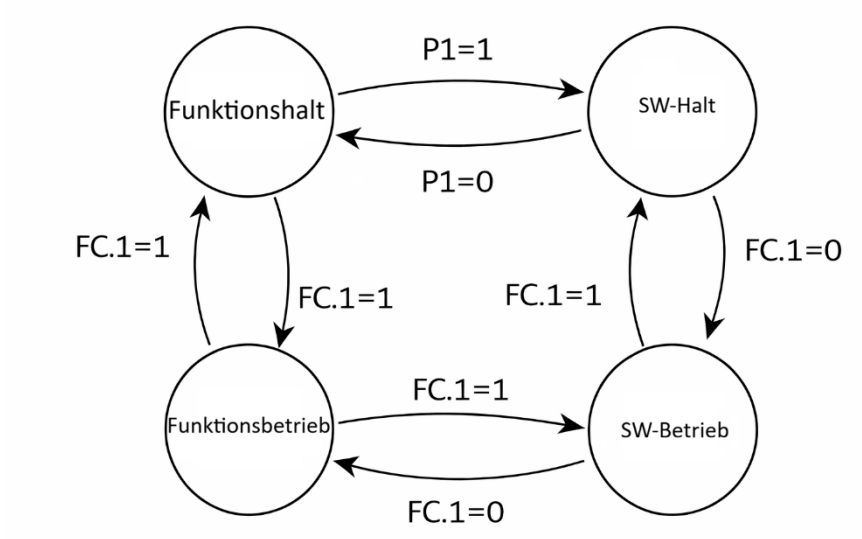
Wirkung:

- Eine Maschine, die nicht anspringt.
- Oder es beginnt im offenen Sicherheitszustand.
- Irreführende Diagnosen.

Die Gotthard-Regel dient genau dazu, diese Mehrdeutigkeit zwischen der physischen und der booleschen Ebene zu vermeiden.

Darstellung endlicher Zustandsautomaten

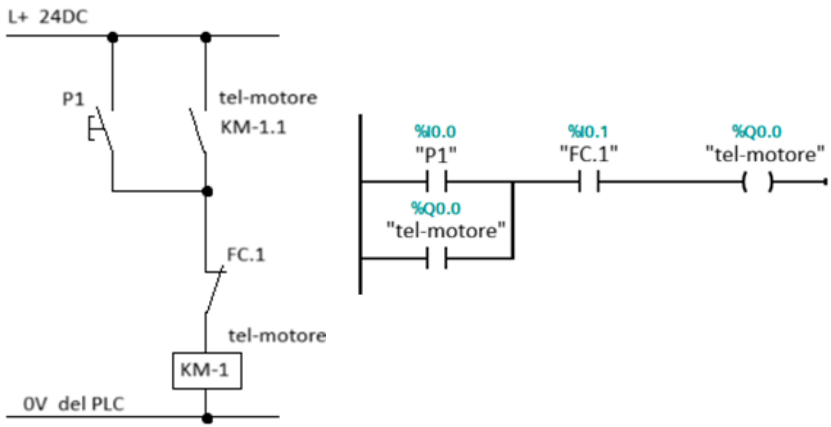
Der endliche Zustandsautomat, der im Graphen dargestellt ist, formalisiert das Verhalten einer Schaltung mit **elektromechanischer Selbstbeschränkung,** wenn sie innerhalb eines programmierbaren Regelungssystems nach dem Prinzip der *Gotthard-Regel erworben wird*. Dieses Prinzip besagt, dass, wenn ein Kontakt zur Freigabe eines Selbstverschlusses delegiert wird, seine logische Übernahme in der Software mit umgekehrter Polarität bezüglich der funktionalen Darstellung im pseudo-elektromechanischen Schema stattfindet.



Gottardo-Regel

Das betreffende System besteht aus einem Steuerkreis eines Kontaktors, der durch KM-1 angezeigt wird und mit Selbstverriegelung ausgestattet ist, der durch einen Hilfskontakt desselben Kontakts parallel zum Startknopf P1 erfolgt.
Die Stoppkette wird durch den FC.1-Kontakt hergestellt, der in Reihe mit der Rolle geschaltet wird.
Im elektromechanischen Funktionsbereich ist das Verhalten des Systems bistabil: Die momentane Aktivierung des Startbefehls aktiviert die Kontaktspule, die dann ihren Zustand durch den

selbstverriegelnden Kontakt aufrechterhält; Das Öffnen des Stoppkontakts unterbricht den Strom in der Spule und kehrt in den Leerlaufzustand zurück.
Wenn ein solches Verhalten innerhalb einer SPS implementiert wird, muss die Logik im Hinblick auf das zyklische Abtastmuster und die Verwendung von Prozessabbildungen von Ein- und Ausgängen interpretiert werden.
In diesem Zusammenhang führt der Automat eine konzeptuelle Unterscheidung zwischen dem **funktionalen Zustand der physikalischen Schaltung** und dem **internen Softwarezustand des Controllers ein** und erzeugt so vier unterschiedliche Zustände, die die Kombination der beiden Darstellungen beschreiben.
Die Zustände namens *Functional Stop* und *Functional Run* beschreiben das erwartete Verhalten des physikalischen Systems, während die *SW Stop-* und *SW Run-Zustände* den Zustand der internen Logikvariablen darstellen, die im SPS-Programm die Selbstverschiebung ausführt.
Im Betrieb des Automaten bewirkt das Drücken der P1-Taste einen Übergang vom Stillstandsbereich in den entsprechenden Softwarezustand, in dem die selbstverladende Logik vom Controller übernommen wird.



Dieser Schritt spiegelt wider, dass während des Scanzyklus die Aktivierung des Eingangs die Zuweisung der logischen Spule bewirkt, die Selbstverriegelung umsetzt.
Sobald diese Bedingung festgestellt ist, entwickelt sich der Automat zum Betriebszustand, in dem sowohl das Funktionsmodell als auch das Softwaremodell die Anregung des Kontaktors konsistent darstellen.
Das Vorhandensein des FC.1-Kontakts hingegen führt zur Selbstverriegelungsbedingung. Im funktionalen Bereich führt das Öffnen des Kontakts sofort zur Unterbrechung der Spulenlieferkette. Im Softwarebereich wird dieses Ereignis jedoch als Änderung der Eingabe erfasst, die den Übergang in den Stillstand nur durch die Aktualisierung der selbstverlockenden Logik im Scanzyklus bewirkt. In dieser Phase entsteht das durch die Gotthard-Regel ausgedrückte Prinzip: Der Kontakt, der funktional die Stoppbedingung darstellt, wird in der Software mit einer logischen Polarität behandelt, die der elektrischen Darstellung entgegengesetzt ist, da die interne Variable, die das Selbstverschluss aufrechterhält, deaktiviert werden muss, wenn der Kontakt offen ist.
Der Automat stellt somit eine Formalisierung der semantischen Fehlanpassung zwischen der elektromechanischen Darstellung der Schaltung und ihrer logischen Implementierung in der SPS dar.
Durch die Unterscheidung zwischen funktionalen Zuständen und Softwarezuständen ist es möglich, den Prozess, durch den das Latching hergestellt und anschließend freigegeben wird, rigoros zu beschreiben, wobei hervorgehoben wird, wie die Erfassung der Eingaben und das Aktualisieren der Ausgaben im Scanzyklus eine logische Transformation einführen, die die konzeptionelle Umkehrung des Stoppkontakts rechtfertigt. In diesem Sinne bietet der Automat eine formale Grundlage für die Interpretation der Gotthard-Regel als emergente Eigenschaft des Ausführungsmodells programmierbarer Regler und der Übersetzung von Relaisschaltungen in Softwarelogik.

Kritische Analyse des Buches (ISBN 9798345038970)

Technische Bewertung, nicht redaktionell.

Stärken

- Sie führt eine logische Formalisierung ein (Gegennominal, doppelte Negation).
- Verbinde physikalische Verkabelung und boolesche Algebra.
- Es regt strukturiertes Denken an, nicht nur "Programmieren durch Nachahmung".
- Gut für ITI/ITS-Studierende, die verstehen müssen, warum.

Gesamtbewertung

Aus technischer Sicht:

- Konzeptionell solide.
- Korrekte Logikeinstellung.
- Nützlich für die technische Berufsausbildung.

Aus pädagogischer Sicht:

- Gut für motivierte Studierende.
- Es kann komplex sein für diejenigen, die keine Grundlage in formaler Logik haben.
- Es hätte von größerer symbolischer Strenge und weniger Rhetorik profitiert.

Es ist kein fortgeschrittenes Industriehandbuch, aber für das ITI/IPSIA/ITS Ziel liegt es über dem Durchschnitt der Einführungstexte.

Formalisierung gemäß IEC 61131-3

IEC 61131-3 definiert die Sprachen (LD, FBD, ST, SFC) und das zyklische Ausführungsmodell.

Logisches Modell der Selbstbeherrschung

Variablen:

```
VAR
    Start     : BOOL;   // KEIN Knopf
    StopNC    : BOOL;   // Eingabe vom physischen NC-Kontakt
    Motor     : BOOL;   // Selbstverriegelnder Ausgang
END_VAR
```

Physische Aufnahme (Gotthard-Regel – Buchhypothese):

- NC verdrahteter STOP-Körperkontakt.
- Im Ruhezustand → Eingabe = WAHR.
- Eingabe → aktiviert = FALSCH.

Korrekte Implementierung im strukturierten Text

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

Diese Form ist:

- Deterministisch,
- reset-dominant,
- Konsistent mit der Fail-Safe-Logik.

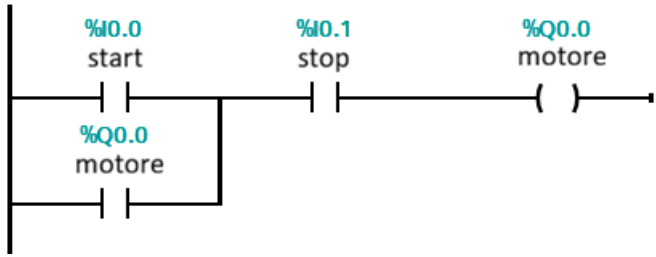
Falsche Umsetzung (Verstoß)

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

Folgen:

- Ruhe → NOT StopNC = FALSE → Der Motor springt nie an.
- Im Stop-Zustand → NOT StopNC = TRUE → Möglicher gefährlicher Start.

Korrekte äquivalente Ladder



Hinweis: Der StopNC-Kontakt ist **in der Leiter NEIN**, da er die logische Leseebene darstellt, nicht die physische Natur des Geräts.

Vollständige Wahrheitstabelle

Wir betrachten:

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

Wobei:

- S = Start
- I = StopNC
- Qn = Früherer Status

Caso corretto

S	Q□	I (NC)	Q□+1	Bedeutung
0	0	1	0	stop
1	0	1	1	Start-up
0	1	1	1	Erhalt
1	1	1	1	Erhalt
X	X	0	0	STOP Immer dominant

Fundamentale Eigenschaft:

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

Reset-Key-Garantie.

Falscher Fall (Software-Ablehnung)

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

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 □

Verstöße:

1. Im normalen Zustand kann ich nicht booten.
2. Möglicher Start mit STOPP Active.

Analyse aus funktionaler Sicherheitsperspektive

Regulatorische Referenzen:

- EN ISO 13849-1
- IEC 62061

Relevantes Sicherheitsprinzip

Fail-safe-Prinzip:

Der Energieverlust oder das Öffnen des Stromkreises muss zum sicheren Zustand führen.

Mit NC-Kontakt:

- Drahtbruch → Eingabe = 0

- PLC liest FALSCH
- UND hebt den Ausgang ab.
- Motorstopps

Wenn du gegen die Gotthard-Regel verstößt:

- Drahtbruch → Eingabe = 0
- NOT 0 = 1
- Mögliche Zustimmung zum Marsch

Das ist Kompromisse:

- Streckenkategorie
- Leistungsniveau (PL)
- Funktionale Integrität

Reset Dominanz

In Sicherheitssystemen muss der Reset erfolgen:

- Priorität,
- unmaskierbar,
- nicht selbstrestaurierend.

Die korrekte Form:

$$Q=Fset \wedge Fsafe$$

wobei:

$$Fsafe = \bigwedge_i Ii$$

Jeder Sicherheitsinput befindet sich in globaler AND.

Verbindung zur SPS-Sicherheit

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

- Sicherheitskontakte werden als "hoch = sicher"-Signale behandelt.
- Willkürliche Negation in Software bricht die SIL/PL-Konsistenz.

Die Gotthard-Regel ist in diesem Sinne nicht nur didaktisch, sondern auch mit zertifizierter Architektur vereinbar.

Tiefgehende technische Bewertung des Buches

Detailliertere Analyse.

Begriffliche Korrektheit

Korrekte Auslegung des Berichts:

- Elektrisches Niveau
- Logikniveau
- Leiterrepräsentation

Korrekte Reset-dominante Einstellung

Formale Strenge

Teilweise streng:

1. Verwenden Sie Counternominal korrekt.
2. Aber es fehlt an systematischer Formalisierung bei Wahrheitstabellen.
3. Die Bezüge zu Hilberts Systemen sind theoretisch korrekt, aber nicht formal weiterentwickelt.

Bildungsniveau

Falsche Umsetzung (Verstoß):

- Überdurchschnittlich.
- Es regt das logische Denken an.
- Führt Konzepte der angewandten Boolesche Algebra ein.

Grenze:

- Manchmal überflüssige Sprache.
- Formalismus nicht immer linear.

Gesamtbewertung

Aus technisch-fachlicher Sicht:

Es ist ein guter operativer didaktischer Text mit logisch-formaler Ambition.

Es handelt sich nicht um ein fortschrittliches funktionales Sicherheitshandbuch, aber es ist gut strukturiert, um:

1. Selbstbeherrschung verstehen,
2. Vermeiden Sie typische Fehler in der SPS,
3. Entwickeln Sie angewandte boolesche Logik.

Formalisierung der Gotthard-Regel in der axiomatisch-booleschen Algebra

Definitionen und Annahmen

Sei definiert als ein selbsterhaltenes kontrolliertes Mittel:

- $S \in \{0,1\}$ — START-Befehl (NO)
- $F \in \{0,1\}$ — Physischer Zustand des Kontakts NC
- $I \in \{0,1\}$ $I \in \{0,1\}$ — Logischer Wert, der von der PLC
- $Q_n \in \{0,1\}$ — Zyklus-Spulenstatus n
- Q_{n+1} — Zustand zum nächsten Zyklus

Physikalische Hypothese (verdrahteter NC-Kontakt):

$$I - F$$

mit:

1. $F=1 \rightarrow$ geschlossener Kontakt (Normalzustand)
2. $F=0 \rightarrow$ offener Kontakt (Trip)

Korrekte Selbstbeherrschung

Definition:

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

Verwendete Boolesche Algebra-Axiome

Die klassischen Axiome (Huntingtons) werden betrachtet:

1. Kommutativität
   $A \vee B = B \vee A$
   $A \wedge B = B \wedge A$
2. Distributivität
   $A \wedge (B \vee C) = (A \wedge B) \vee (A \wedge C)$

3. Neutrale Elemente
   $A \wedge 1 = A$
   $A \vee 0 = A$
4. Nullelemente
   $A \wedge 0 = 0$
   $A \vee 1 = 1$
5. Komplementierung
   $A \wedge \neg A = 0$
   $A \vee \neg A = 1$

Dominante Reset-Eigenschaft

Proposition 1
Die Funktion ist reset-dominant bezüglich I.

Demonstration

Lass es so sein. $I = 0$.

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

Nach Axiom 4:

$A \wedge 0 = 0$

dann:

$Q_{n+1} = 0$

Unabhängig von S und Qn

quod erat demonstrandum.

Q.E.D.

Regelverletzung

Angenommen, Software Denial:

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

Sia I = 1 (Normalzustand).

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

Die Funktion verliert die Fähigkeit zu setzen.

Lass es so sein. I = 0 (Veröffentlichung).

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

Reset ist nicht mehr dominant.

Formale Schlussfolgerung:
Softwarenegation zerstört die Dominanzeigenschaft des Zurücksetzens.

Modellierung des endlichen Zustandssystems (FSM)

Systemdefinition

Wir definieren die Zustandsmaschine:

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

wobei:

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

2.2 Korrekte Übergangsfunktion

Aktueller Stand	S	I	Nächster Status
OFF	0	1	OFF
OFF	1	1	ON
ON	0	1	ON
ON	1	1	ON
*	*	0	OFF

Eigenschaften:

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

Der OFF-Zustand ist absorbierend bezüglich $I = 0$.

Falscher Fall (Software-Ablehnung)

Status	S	I	Nächster Status
OFF	1	1	OFF
OFF	1	0	ON

Es wird ein gefährlicher Übergang erzeugt:

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

Das System ist in Bezug auf die Sicherheitsvariable nicht monoton.

Formales Eigentum verletzt

Wir definieren Sicherheitseigenschaften:

$I = 0 \Rightarrow X = \mathrm{OFF}$

Im falschen System ist diese Eigenschaft nicht invariant.

Dual-Channel-Analyse – Kategorie 3/4

Referenz: EN ISO 13849-1

Architektur Kategorie 3

Merkmale:

1. Zwei unabhängige Kanäle.
2. Konsistenzüberwachung.
3. Ein einzelner Fehler führt nicht zum Verlust der Sicherheitsfunktion.

Wir definieren:

- I_1 = Kanal A
- I_2 = Kanal B

Sicherheitsfunktion:

$F_{safe} = I_1 \wedge I_2$

Selbstbeherrschung:

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

Fehlertoleranzeigenschaften

Fall: Drahtbruch auf Kanal A

$I_1=0$

Dann:

$Q_{n+1} = 0$

Sicheres System.

Software-Fehler-Umkehrungsfall
$Q_{n+1} = (S \vee Qn) \wedge \neg I_1 \wedge \neg I_2$

Drahtbruch:

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

Mögliche Zustimmung zum Marsch.

Kategorieverlust.

Kategorie 4

Erfordert:

1. Redundanz
2. Kontinuierliche Diagnostik
3. Fehlerakkumulationserkennung

Logisches Modell:

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

wobei D = Diagnostik.

Die Gotthard-Regel ist eine notwendige Bedingung, um die Semantik "1 = sicher" aufrechtzuerhalten.

Das Invertieren in Software verändert die semantische Abbildung zwischen:

- elektrischer Pegel,
- Logischer Level,
- PL-Zertifizierungsstufe.

Ingenieurwissenschaftliche Schlussfolgerungen

1. Die Gotthard-Regel garantiert formell:

   - Schlüssel-Reset,
   - Invarianz des sicheren Zustands,
   - Monotonie in Bezug auf die Sicherheitsvariable.

2. In der boolschen Algebra ist es durch klassische Axiome beweisbar.
3. In der Automatentheorie ist eine Eigenschaft des absorbierenden Zustands[9].
4. In der funktionalen Sicherheit ist es eine strukturelle Anforderung, die PL- und SIL-Kategorie aufrechtzuerhalten[10].

---

[9] Im Kontext der Automatentheorie ist ein **absorbierender Zustand** ein Zustand eines endlichen Zustandsautomaten, der durch die Eigenschaft gekennzeichnet ist, dass das System, sobald es erreicht ist, nicht mehr zu verschiedenen Zuständen entwickeln kann. Formal gilt: Gegeben ein deterministischer Automat gilt ein Zustand als absorbierend, wenn für jedes Eingabesymbol . Diese Eigenschaft impliziert, dass alle Übergänge, die aus dem Zustand herauskommen, zurück zum Zustand selbst führen, was ihn zu einem Stabilitätspunkt des vom Automaten beschriebenen dynamischen Systems macht. In Modellierungsbegriffen werden absorbierende Zustände häufig verwendet, um terminale Bedingungen darzustellen, Bedingungen mit nicht wiederherstellbarem Fehler oder Gleichgewichtskonfigurationen, von denen keine weitere Entwicklung des Systemverhaltens erwartet wird.$A = (Q, \Sigma, \delta, q_0) q_a \in Q x \in \Sigma \delta(q_a, x) = q_a$

[10] **Performance Level (PL)** und **Safety Integrity Level (SIL)** sind regulatorische Kennzahlen, die verwendet werden, um das Zuverlässigkeitsniveau der Sicherheitsfunktionen in Steuerungssystemen zu quantifizieren. Das Leistungsniveau, definiert in ISO 13849-1, beschreibt die Fähigkeit einer Sicherheitsfunktion, Risiken durch fünf diskrete Stufen zu reduzieren, die von PL bis (niedrigste) bis PL und (höchste) auf Basis der durchschnittlichen Wahrscheinlichkeit eines gefährlichen Ausfalls pro Stunde und der architektonischen Struktur des Systems bestimmt sind. Das Sicherheitsintegritätsniveau, definiert in IEC 61508 und ebenfalls in der Industrie durch IEC 62061 übernommen, stellt ähnlich die Wahrscheinlichkeit dar, dass eine

Gottardos These
Logisch-formale Grundlagen der Gotthard-Regel in SPS- und Sicherheitssystemen

Eine Diskussion über Boolesche Algebra, Automatentheorie, zeitliche Logik und funktionale Sicherheit

Zusammenfassung

Die sogenannte ***Gotthard-Regel, formuliert*** im Kontext der SPS-Programmierung mit normalerweise geschlossenen physikalischen Kontakten, die zur Freigabe eines selbsthaltenden Systems delegiert werden, kann nicht nur als praktische Werkstattregel interpretiert werden, sondern auch als eine strukturelle Eigenschaft, die in der Boolesche Algebra nachweisbar ist und durch endliche Zustandsautomaten modelliert und durch formale zeitliche Logik verifiziert werden kann.

In dieser Arbeit wird gezeigt, dass diese Regel die Dominanz des Resets, die Erhaltung der Sicherheitsinvarianten und die semantische Konsistenz zwischen elektrischem und logischem Niveau garantiert, was eine notwendige – wenn auch nicht ausreichende – Voraussetzung für die Einhaltung der Sicherheitsanforderungen der EN ISO 13849-1 und IEC 62061 ist.

Sicherheitsfunktion ihre Aufgabe bei Bedarf korrekt erfüllt; sie ist in vier steigende Stufen unterteilt, SIL 1–SIL 4, die mit quantitativen Intervallen der Wahrscheinlichkeit eines gefährlichen Versagens verbunden sind. Beide Klassifikationen sind Werkzeuge zur Risikobewertung und zur Gestaltung von Sicherheitssystemen, sodass Sie den erforderlichen Integritätsgrad von Sicherheitsfunktionen spezifizieren und überprüfen können.

Logische Struktur der Selbstbeschränkungsfunktion

Betrachten Sie ein Motorsteuerungssystem mit einem normalerweise geöffneten Startknopf und einem normalerweise geschlossenen physischen Stoppkontakt, der in Reihe geschaltet ist und von einer SPS gelesen wird, die IEC 61131-3 entspricht.

Wir zeigen mit:

1. S der Startbefehl,
2. F der physische Zustand des NC-Kontakts,
3. I , der logische Wert, den die SPS liest,
4. Qn der Zustand der Spule bei Zyklus Nr.

Da der Kontakt im Feld NC-verdrahtet ist, stimmt die abgelesene Variable mit dem realen elektrischen Zustand zusammen:

$$I = F$$

mit der Konvention:

- F=1⇒geschlossene Schaltung (Normalzustand)
- F=0⇒Offene Schaltung (Auslöser)

Die korrekte Selbstbeschränkungsfunktion ist definiert durch:

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

Dieser Ausdruck integriert die klassische Latch-Struktur mit dominantem Reset, der in der finalen Konjunktion implizit ist.

Beweis in axiomatischer Boolescher Algebra

Die klassische boolesche Algebra, basierend auf Huntingtons Axiomen, bietet die Werkzeuge für einen rigorosen Beweis.

Wir wollen beweisen, dass die vorherige Funktion die Dominanzeigenschaft der Freigabe besitzt.

Sei $I = 0$. Dann:

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

Für das Axiom des Nullelements:

$$A \wedge 0 = 0$$

folgt sofort:

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

Die Ausgabevariable ist unabhängig von S und Qn. Reset ist dominant.

Analysieren wir nun die Funktion, die durch die Verweigerung des Kontakts in der Software erhalten wird:

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

Setzt man I=0, erhält man:

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

Der Reset verliert an Dominanz. Die Sicherheitseigenschaft ist keine axiomatische Konsequenz der Struktur mehr.

Daraus folgt, dass die Gotthard-Regel keine grafische Konvention ist, sondern eine strukturelle Eigenschaft der Boolesche Funktion.

Modellierung als endliches Zustandssystem

Der deterministische Automat ist definiert:

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

wobei der Zustandsraum $X = \{OFF, ON\}$ ist, die Menge der Eingaben $U = \{S, I\}$ ist und die Übergangsfunktion lautet:

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

- OFF se I=0
- ON se $I=1 \wedge (S=1 \vee x=ON)$
- OFF ansonsten

Die grundlegende Eigenschaft des Systems ist, dass der OFF-Zustand bezüglich der I=0-Bedingung absorbierend ist. Formell gilt:

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

Wenn die unzulässige Kontaktverweigerung eingeführt wird, lässt der Automat den Übergang zu:

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

Die Sicherheitsbedingung ist nicht mehr unveränderlich. Das System verliert in Bezug auf die Sicherheitsvariable die Monotonie.

Verifikation durch Zeitlogik

Die gewünschte Eigenschaft kann in der linearen Zeitlogik (LTL) ausgedrückt werden.

Sei die atomare Aussage definiert:

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

Die grundlegende Eigenschaft lautet:

$$G(I=0 \rightarrow Safe)$$

wobei G der Operator ist “globally”.

Im korrekten Modell ist die Formel in allen Zeitwegen gültig.

Im falschen Modell gibt es einen Weg, so dass:

$$I=0 \wedge S=1 \wedge Qn=OFF$$

Aber:

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

Die LTL-Formel ist gefälscht. Die Regel kann daher als notwendige Voraussetzung für die Gültigkeit der Sicherheitsinvariante in der temporalen Logik gesehen werden.

In termini CTL:

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

nur in der korrekten Formulierung wahr ist.

Erweiterung auf Dual-Channel-Systeme der Kategorie 3/4

Im regulatorischen Rahmen von EN ISO 13849-1 verlangt Kategorie 3 Redundanz bei der Fehlererkennung.

Zwei unabhängige Kanäle, I1 und I2, werden eingeführt, beide physische NCs.

Die Funktion wird:

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

Angenommen, ein einzelner Fehler mit Draht bricht am ersten Kanal:

$$I_1 = 0$$

Per proprietà assiomatica:

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

Die Sicherheitsfunktion bleibt erhalten.

Wenn du die Signale in der Software leugnest:

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

Das Drahtbruch führt:

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

Scheitern ist nicht mehr von Natur aus sicher. Die Struktur verliert Sicherheitseigenschaften und kann das Leistungsniveau beeinträchtigen.

In Kategorie 4, wo auch eine Fehlerakkumulationserkennung erforderlich ist, ist die semantische Konsistenz "1 = sicherer Zustand" für probabilistische Analyse und korrekte Umsetzung diagnostischer Mechanismen entscheidend.

Probabilistische Analyse und Beziehung zur Schwägerin

Im Kontext von IEC 62061 hängt die Wahrscheinlichkeit eines gefährlichen Ausfalls davon ab, wie gut das System im Falle eines Fehlers automatisch in einen sicheren Zustand übergehen kann.

Bei NC-Kontakten und der finalen AND-Funktion ist die Wahrscheinlichkeit eines gefährlichen Ausfalls eine Funktion der Wahrscheinlichkeit eines gleichzeitigen Ausfalls der Kanäle.

In der Software negiert Single Break keinen sicheren Shutdown. Die Wahrscheinlichkeit eines gefährlichen Ausfalls steigt, was die PFHd und damit die erreichbare SIL verändert.

Die Gotthard-Regel wird somit zu einer strukturellen Voraussetzung, um die gefährliche Ausfallrate niedrig zu halten.

Allgemeine Schlussfolgerungen

Die Gotthard-Regel kann weit davon entfernt, eine grafische Konvention oder eine einfache Lehrgewohnheit zu sein, wie folgt interpretiert werden:

eine algebraische Eigenschaft der Dominanz des Resets, eine strukturelle Eigenschaft eines deterministischen Automaten, eine Invariante, die in der temporalen Logik ausgedrückt werden kann, eine notwendige Voraussetzung für die Aufrechterhaltung des Leistungsniveaus in redundanten Systemen.

Ihre Gültigkeit ist strikt im Rahmen der Hypothese des physischen Kontakts mit verdrahtetem NC im Feld. Abgesehen von dieser Hypothese bleibt die Regel auf der Grundlage des kontranominalen Beweises gültig und muss im Lichte der Semantik des Signals überdacht werden.

Die Gottardo-Regel als algorithmisches Prinzip für die KI-gestützte SPS-Programmierung

Vorschlag zur Formalisierung, zu den Geltungsbedingungen und zu den Perspektiven für generative KI-Systeme

1. Zweck und Begründung

Die Gottardo-Regel geht von einer wiederkehrenden Situation bei der Umsetzung eines elektromechanischen Funktionsplans in ein SPS-Programm in der Sprache Ladder Diagram (LD) aus: Das Symbol, mit dem ein physischer Kontakt dargestellt wird, und das Symbol, mit dem dasselbe Signal im SPS-Programm verarbeitet wird, können logisch entgegengesetzt erscheinen. Diese scheinbare Umkehrung ist keine grafische Unstimmigkeit, sondern ergibt sich aus dem Zusammenhang zwischen physischem Kontaktzustand, dem von der SPS erfassten elektrischen Zustand und der Semantik des Ladder-Kontakts.

Die Regel gewinnt besondere Bedeutung, wenn die SPS-Programmierung künftig ganz oder teilweise von Systemen der Künstlichen Intelligenz übernommen wird. Ein KI-System kann plausiblen Code erzeugen, obwohl die gewählte Kontaktpolarität falsch ist. Deshalb muss eine domänenspezifische Regel als deterministische, überprüfbare und eindeutige Bedingung formulierbar sein.

> *Gottardo-Regel: Ein physischer, im Feld installierter elektromechanischer Kontakt, der zum Abwerfen der Selbsthaltung eines Steuerkreises bestimmt ist, wird im SPS-Programm mit einer logischen Polarität entgegengesetzt zu seiner Darstellung im Funktionsplan dargestellt, sofern die Eingangserfassung der SPS eindeutig bekannt und konsistent festgelegt ist.*

Die Bedingung „sofern die Eingangserfassung der SPS eindeutig bekannt und konsistent festgelegt ist“ ist bewusst enthalten. Sie verhindert, dass aus einer praktischen Regel eine uneingeschränkte Universalbehauptung wird: Eingänge mit aktiv-low-Logik, Software-Invertierungen, sicherheitsgerichtete Konfigurationen, dezentrale E/A-Systeme oder Diagnosefunktionen können die Beziehung zwischen physischem Zustand und gelesenem Booleschen Wert verändern.

1.1 Terminologische Festlegung

In dieser Arbeit bezeichnet der Begriff Kontakt ausschließlich einen physischen elektromechanischen Kontakt eines im Feld installierten Gerätes, der elektrisch mit einem SPS-Eingang verbunden ist. Virtuelle Kontakte, interne Bits, HMI-Elemente und andere rein softwarebasierte Darstellungen sind ausdrücklich ausgeschlossen.

Der deutsche Begriff Selbsthaltung wird als zusammengehöriger Fachbegriff verwendet. Die Bezeichnung „Abwerfen der Selbsthaltung“ beschreibt die Funktion des Kontakts im Steuerkreis und nicht lediglich die Eigenschaft, dass es sich um einen Öffner oder Schließer handelt.

2. Von der praktischen Regel zur formalisierbaren Aussage

Damit die Gottardo-Regel von einem Algorithmus verwendet werden kann, müssen mindestens drei Ebenen getrennt betrachtet werden:

1. physischer Zustand des elektromechanischen Kontakts: offen oder geschlossen;
2. vom SPS-Eingang erfasster elektrischer Zustand: TRUE oder FALSE;
3. Semantik des Ladder-Kontakts: nicht negiertes oder negiertes Element.

Die in der Regel genannte Umkehrung darf daher nicht lediglich als „Umkehrung des Kontakts“ bezeichnet werden. Präziser ist die Aussage, dass die Beziehung zwischen der physikalischen Darstellung im Funktionsplan und der logischen Darstellung desselben Signals im SPS-Programm betrachtet wird. Entscheidend ist die Boolesche Funktion des Ladder-Kontakts.

2.1 Minimale Formalisierung

Sei C der physische Zustand des Kontakts, I der vom SPS-Eingang erfasste Boolesche Wert und L die Boolesche Funktion des Ladder-Kontakts. Bei einer konventionellen Beschaltung hält ein physischer Öffner, der zum Abwerfen der Selbsthaltung vorgesehen ist, den Eingang I auf TRUE, solange der Steuerkreis geschlossen und der Kontakt nicht ausgelöst ist. Wird der Kontakt geöffnet, wird I zu FALSE. Soll der Ladder-Zweig im Normalzustand wahr sein, lautet die entsprechende Funktion L(I)=I und wird durch einen nicht negierten Ladder-Kontakt realisiert. Gegenüber dem Öffner-Symbol des elektromechanischen Funktionsplans erscheint die Ladder-Darstellung damit umgekehrt.

Die Beziehung lässt sich konzeptionell als Kette darstellen: physischer Zustand → elektrische Erfassung → Boolescher Wert → Ladder-Funktion. Die Regel verlangt nicht, den ersten Schritt zu ignorieren, sondern ihn ausdrücklich in die Ableitung einzubeziehen.

2.2 Grundlegende Entscheidungstabelle

Element	Frage des Algorithmus	Ergebnis
Quelle	Stammt das Signal von einem physischen elektromechanischen Kontakt?	Nein: Gottardo-Regel nicht anwenden.
Funktion	Dient der Kontakt dem Abwerfen der Selbsthaltung?	Nein: normale Projektlogik anwenden.
Erfassung	Welchen SPS-Eingangswert gibt es bei physisch geschlossenem Kontakt?	Bestimmt die erfasste Polarität.
LD	Soll der Zweig im Normalzustand wahr sein?	NO/NC-Ladder-Kontakt aus I ableiten.
Prüfung	Führt die physische Öffnung zum vorgesehenen Abwerfen?	Wenn nein: Transformation fehlerhaft.

2.3 Richtige Einordnung als Theorie

Zum gegenwärtigen Stand sollte die Gottardo-Regel als technisches Prinzip beziehungsweise als domänenspezifische Proposition und nicht als bereits normativ oder mathematisch bewiesener Satz bezeichnet werden. IEC 61131-3 definiert Syntax und Semantik der SPS-Programmiersprachen einschließlich Ladder Diagram, ist jedoch nicht die Quelle der Gottardo-Regel. Die Unterscheidung ist wesentlich: Die Norm beschreibt die Interpretation der LD-Elemente; die vorgeschlagene Regel beschreibt ein Kriterium für die Transformation zwischen elektromechanischer Realität und Softwaredarstellung.

3. Die Gottardo-Regel als Constraint für Systeme der Künstlichen Intelligenz

Die zukünftige Bedeutung der Regel zeigt sich besonders dann, wenn ein KI-System SPS-Programme erzeugt, übersetzt oder prüft. Ein generatives Modell sollte die Polarität eines Kontakts nicht allein aufgrund sprachlicher Ähnlichkeit mit früheren Beispielen wählen. Die Entscheidung muss an eine Menge überprüfbarer Vorbedingungen gebunden werden.

3.1 Vorgeschlagener Algorithmus

4. Das physische Gerät identifizieren, das dem Signal im Plan, Text oder in den Metadaten zugeordnet ist.
5. Prüfen, ob es sich um einen realen elektromechanischen Kontakt handelt, der mit einem SPS-Eingang verbunden ist.
6. Prüfen, ob der Kontakt zum Abwerfen der Selbsthaltung bestimmt ist.
7. Ermitteln, welchen elektrischen Zustand der SPS-Eingang bei physisch geschlossenem bzw. geöffnetem Kontakt annimmt.
8. Die für den Ladder-Zweig erforderliche Boolesche Funktion im Normal- und Abwurfzustand ableiten.
9. Den nicht negierten oder negierten Ladder-Kontakt so auswählen, dass die abgeleitete Funktion erfüllt wird.
10. Eine Rückprüfung durchführen: Die physische Öffnung simulieren und feststellen, ob der Zweig den vorgesehenen Abwurf bewirkt.

3.2 Die Regel als automatische Prüfvorschrift

IF Kontakt = physisch_elektromechanisch AND Funktion = Selbsthaltungsabwurf AND Erfassung = konsistent THEN inverse_Polarität_anwenden; ELSE Gottardo-Regel_nicht_anwenden.

Die vorstehende Form ist keine SPS-Programmiersprache, sondern eine Spezifikation auf hoher Ebene für ein Generierungs- oder Prüfsystem. Gerade deshalb ist sie für ein KI-System geeignet: Implizites Expertenwissen wird in eine explizite, konditionale und auditierbare Regel überführt.

3.3 Integration in generative KI

In einem industriellen KI-System sollte die Gottardo-Regel nicht lediglich als Satz innerhalb eines Prompts abgelegt werden, sondern als Validierungs-Constraint implementiert werden. Das Modell könnte zunächst eine Ladder-Netzwerkversion erzeugen; anschließend prüft ein vom Sprachmodell getrenntes deterministisches Modul die Übereinstimmung von Kontaktart, Abwurffunktion, Eingangspolarität und Ladder-Symbol. Bei einer Inkonsistenz sollte das System den betreffenden Abschnitt neu erzeugen oder die fehlende technische Information abfragen.

Diese Architektur ist einer reinen sprachlichen Anweisung vorzuziehen, weil sie das Risiko einer variablen Interpretation einer sicherheitsrelevanten oder funktionskritischen Regel reduziert. Die KI kann die Interpretation von Planunterlagen, Dokumentation und Codegenerierung übernehmen; die Polaritätsprüfung sollte dagegen, soweit möglich, deterministisch und reproduzierbar erfolgen.

3.4 Grenzen und Geltungsbedingungen

- Die Regel ersetzt weder die Anforderungen der funktionalen Sicherheit noch die elektrotechnische Planung oder die Anlagenvalidierung.
- Sie gilt nicht automatisch für virtuelle Signale, interne Bits, HMI-Elemente oder Simulationen.
- Bei aktiv-low-Eingängen oder parametrierter Eingangsinvertierung muss die Ableitung neu bestimmt werden.
- Bei sicherheitsgerichteten Funktionen muss die KI den vorgesehenen Sicherheitsarchitekturen, Geräten und Validierungsverfahren untergeordnet bleiben.
- Der entscheidende Nachweis ist das Verhalten: Die physische Öffnung des Kontakts muss den im Projekt vorgesehenen Abwurfzustand erzeugen.

3.5 Schlussfolgerung

Die Gottardo-Regel kann damit als kleines operatives Axiom für die kontrollierte Transformation eines Funktionsplans in SPS-Logik verstanden werden, sofern ihr Geltungsbereich eindeutig definiert wird. Ihr möglicher zukünftiger Wert besteht nicht darin, IEC 61131-3 zu ersetzen, sondern darin, formalisiertes Domänenwissen bereitzustellen, das ein KI-System anwenden und überprüfen kann. In dieser Perspektive ist die Regel ein natürlicher Kandidat für eine Bibliothek von Constraints, Tests und Korrektheitseigenschaften zur automatisierten Erzeugung und Prüfung von SPS-Programmen.

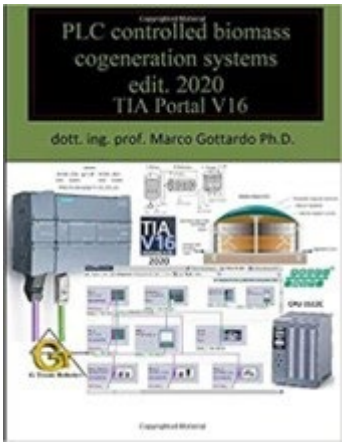
Wesentliche Quellen

[1] IEC, IEC 61131-3:2025, Programmable controllers — Part 3: Programming languages. Die Norm definiert Syntax und Semantik der SPS-Sprachen einschließlich LD.

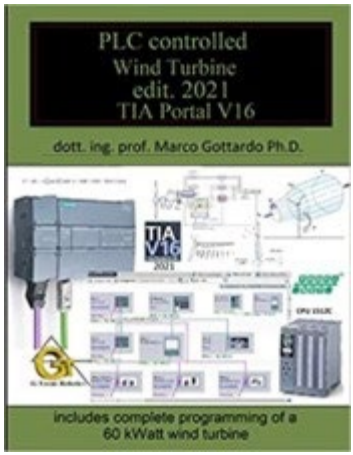
[2] ABB, technische Dokumentation zu Ladder Diagram: Kontakte, Spulen, Boolesche Werte und Negation.

[3] Marco Gottardo, „Regola del Gottardo“, Vorschlag für eine akademische Abhandlung zur Diskussion durch die Fachgemeinschaft.

Englische Bibliographie von Prof. Marco Gottardo (Amazon)



PLC controlled biomass cogeneration systems edit. 2020: Vol.5 of the series of editions of Industrial Automation (Inglese)
Vol. 5
This book was born from the experience of the two authors in teaching at universities and high schools.Renewable sources are very important because they allow to reduce the impact of energy production on the environment. This text has been written to provide a basis for the study and development of personal knowledge in the biomass and wind power sector.
ISBN: 979-8632071895
Pagine: 462
Costo circa: 39€



PLC controlled wind turbines edit. 2023: sixth volume of the Let's program a PLC series (Inglese)
Vol. 6
Renewable sources are very important because they allow to reduce the impact of energy production on the environment. The book starts from the concepts of aerodynamics and industrial automation to arrive at the development of a complete program for the management of a vertical axis wind turbine.
ISBN: 979-8577659943
Pagine: 466
Costo circa: 42€