Journal on Interactive Systems, 2026, 17:1
ISSN: 2763-7719 • doi: 10.5753/jis.2026.6644

Research Paper

Interactively Converting Ladder Diagrams into Grafcet Diagrams on a Robotic Neutralization System Case Study

Paulo André Sperandio Giacomin [State University of Santa Cruz | *pasgiacomin@uesc.br*]

Department of Engineering and Computing, State University of Santa Cruz, Campus Soane Nazaré de Andrade, Highway Jorge Amado, km 16, District Salobrinho. CEP 45662-900. Ilhéus-BA, Brazil.

Abstract. *Background:* Programmable Logic Controllers (PLCs) are widely used to implement automation systems based on sequential control. For such systems, Ladder and Grafcet diagrams are standardized, respectively used as programming and specification formalisms. Additionally, a neutralization system is responsible for mixing a solution with a neutralizer so that the resulting liquid has a neutral pH. *Purpose:* This study proposes an interactive approach for converting Ladder Diagrams into Grafcet diagrams, and a robotic neutralization system is considered as a realistic case study. *Methods:* An algorithm is designed and programmed for converting Ladder diagrams into Grafcet diagrams using Binary Decision Diagrams (BDDs), and its time complexity is mathematically determined. For validating the effectiveness of the proposed algorithm, the Ladder diagram is simulated and compared side by side with the resulting Grafcet diagram for the same expected initial state and input sequences. *Results:* The resulting Grafcet diagram is considered equivalent to the original Ladder diagram, and the proposed algorithm is classified according to its time complexity. *Conclusions:* The proposed strategy helps the expert to identify the plant's process control flow in a realistic case study where either transitions AND or OR are addressed. Besides, the exponential growth in running time is softened by the use of BDDs and the resulting Grafcet diagram's flexibility is improved without compromising its equivalence to the Ladder diagram. Moreover, compared to preexisting neutralization systems, the proposed one preserves the solution acidity once its desired preset value is achieved.

Keywords: Discrete Event Systems, Algorithms, Translation, Binary Decision Diagrams.

Edited by: Saul Delabrida | **Received:** 05 October 2025 • **Accepted:** 27 April 2026 • **Published:** 06 May 2026

1 Introduction

Nowadays, PLCs are widely used in the automation of several industrial plants. In the past, PLCs were programmed almost exclusively by using Ladder diagrams. However, afterwards, the standard IEC 61131-3 emerged, along with new PLC programming languages, such as the SFC, whose specification comes from the Grafcet diagram, which is currently regulated by the standard IEC 60848. If compared to the Ladder diagram, the Grafcet has a different paradigm, which is often advantageous since it can explicitly represent the plant's sequential control flow [Falcione and Krogh, 1993; Galeano González and Botero Castro, 2007; Burgos *et al*., 2020][1]. However, since the Ladder diagram predates the standard IEC 61131-3, most systems were written using Ladder diagrams. The industry chose to use the Ladder diagram because it is like the old relay diagrams, for which the workforce was trained and available. However, for several cases, the Ladder diagram is not the best language for programming PLCs, and diverse studies point out its deficiencies, such as [Venkatesh *et al*., 1994] and [Zanma *et al*., 1999].

For instance, the Ladder diagram is not easy to read, and it is difficult to extend and maintain. These drawbacks stem from the fact that the Ladder diagram does not explicitly represent the plant's process control flow [Falcione and Krogh, 1993]. Since the Grafcet diagram has a different paradigm and does not suffer from the same drawback, it is well-known that it can be used as an auxiliary tool for documenting Ladder diagrams. However, few studies describe how to recover the plant's sequential control flow from the Ladder diagram. According to Pressman [2019], the design recovery is interesting because it captures part of the confidence already implemented in older systems, which are validated under real operating conditions.

This study aims at designing and programming an algorithm able to convert Ladder diagrams into Grafcet diagrams, and the two diagrams must be equivalent, i.e., they must produce the same output command sequences for the same expected input event sequences and initial conditions. A robotic neutralization system is considered as a realistic case study. Here, the bibliography review is divided into two parts, where the first one contextualizes the proposed study within the general area of automation based on PLCs, and the second one concentrates specifically on studies about converting Ladder diagrams into Grafcet diagrams.

From a broader perspective, common subjects involving PLC programming are design, language interpretation, simulation, programming language extension, formal verification, performance evaluation, teaching, and conversion between programming languages. Regarding studies on design, an instance is given by Dhanabalan and Selvi [2023], where the Laboratory Virtual Instrument Engineering Workbench (LABVIEW) is used to design a hardware based on Field-Programmable Gate Array (FPGA), from the Ladder diagram, but the Grafcet diagram is not considered. Another example is given by Palaniappan *et al*. [2023], where the Grafcet is employed to design the Ladder diagram, and a

[1] The study of Galeano González and Botero Castro [2007] is translated as 'Grafcet representation of the actuation of synchronous generators in a modernized hydroelectric power plant'

processing station is considered as a case study. In contrast, here the conversion is made in the opposite direction, i.e., from the Ladder diagram to the Grafcet diagram, which has a didactic purpose.

With respect to studies about the way a programming language is interpreted, an example is given by Mroß *et al*. [2023], where an algorithm is proposed for interpreting the Grafcet language without ambiguities. Another study is given by Schnakenbeck *et al*. [2023], where an abstract interpretation is performed, avoiding concurrent behaviour, and an automatic testing machine for quality control of components is considered as a case study. Although a precise interpretation mechanism can make the Grafcet more reliable, more research is necessary to evaluate its relation with the other languages of the standard IEC 61131-3. By contrast, here one studies the relation between the Ladder and Grafcet diagrams.

With respect to studies on simulations, an example is given by Alsamhan *et al*. [2023], where Grafcet is used to simulate a yogurt filling machine. An additional example is given by Hrbček *et al*. [2026], where a digital twin is controlled by a PLC. Although the simulation can be used to reduce risks, it is a prediction. However, reverse engineering can be valuable after the system is in operation, since it can recover part of the confidence already tested by the system's operation [Pressman, 2019].

With regard to studies proposing programming language extensions, an example is given by Abrishambaf *et al*. [2023], where the function block formalism is extended to more effectively manage the transducer level. Another example is given by Roisin *et al*. [2025], in which an extended SFC is used to control a mixing system for two liquids. Although extensions can make a programming language more versatile, the integration between different technologies is a characteristic of the standard IEC 61131-3, and consequently, more research is necessary to evaluate its impact on the other programming languages of the standard IEC 61131-3. On the other hand, in this work the relation between the Ladder and Grafcet diagrams is examined in detail.

Considering studies about formal verification, an example is given by Hu *et al*. [2024], where a program written using structured text is simulated and validated by using a Petri net. Another example is given by Chen *et al*. [2026], where the function block is converted into its equivalent finite automata model to formally verify its correctness, and Chen *et al*. [2026] show that formal verifications may require conversions between two formalisms. However, unlike the studies of Hu *et al*. [2024] and Chen *et al*. [2026], the proposed one focuses on Grafcet, which complies with the standards IEC 61131-3 and IEC 60848.

Considering studies that evaluate the performance of PLC systems, an example is given by Bojnurdi *et al*. [2024], where the distribution of function blocks is optimized by using a greedy algorithm. An additional example is the study of Liu *et al*. [2025], where an approach based on graph neural networks is employed to evaluate the performance of the PLC. Although improving the performance can make the industry more competitive, Ladder and Grafcet diagrams are not considered in [Bojnurdi *et al*., 2024] and [Liu *et al*., 2025]. However, the Ladder diagram is the most used PLC programming language [Godase, 2025], and the Grafcet diagram is a versatile control tool and is becoming more and more popular [Roisin *et al*., 2025]. Hence, these two diagrams are considered in the proposed study.

From a more specific perspective, studies about converting Ladder diagrams into Grafcet diagrams are scarce. An example is proposed by Falcione and Krogh [1993]: to accomplish the objective, the authors employ different kinds of graphs, such as the simultaneity graph, the dependency graph, the condensed simultaneity graph, and the decomposition operations, such as the connected component decomposition and the full connectivity decomposition. Besides, theu authors consider as a case study a neutralization system with a piping and instrumentation diagram. Although the Falcione and Krogh [1993]'s study points out important issues to be investigated, such as the computational complexity of the conversion algorithm, strategies for enhancing the efficiency, and rigorous equivalence criteria, their neutralization system alters the solution acidity while the resulting solution is provided to the system output. In contrast, though here the same neutralization system is considered, the control logic preserves the solution acidity while it is provided to the system output. Besides, though the Boolean function manipulation may have exponential time complexity in the worst case scenario, BDDs are employed here, which mitigate the exponential growth in running time [Bryant, 1986]. Although Bryant [1986] describes the time complexity for the BDD handling operations, a formal proof is not presented. In contrast, it is provided here.

Another study is proposed by Zanma *et al*. [1999], where additional information is included in the conversion process in the form of temporal logic, and a handling-robot is given as an example. Although the Zanma *et al*. [1999]'s approach points out the importance of extracting the plant's process control flow, and though simultaneous tasks are also considered, the Zanma *et al*. [1999]'s case study does not take into account the existence of alternative sequential control flows known as transitions OR. In contrast, this case is covered here.

A third study is proposed by Lopes and Sousa [2017], where an extended state-space approach is employed for the same case study of Falcione and Krogh [1993]. Although the Lopes and Sousa [2017]'s state-machine's flexibility is slightly improved, the number of states may grow exponentially because the state-machine does not have a special formalism for representing concurrent tasks. In contrast, Grafcet diagrams do not suffer from the same drawback. Besides, though it is not always possible to perform the conversion without including extra information in the conversion process [Lopes and Sousa, 2017], the studies [Falcione and Krogh, 1993], [Zanma *et al*., 1999] and [Lopes and Sousa, 2017] include this information in design time, i.e., the additional information must be provided before the conversion process. However, this information may not be available. On the other hand, part of this information is present in the Ladder diagram. In this case, the conversion algorithm helps the expert to identify the possible plant's process control flow, i.e., part of the identification of the plant's process control flow is performed by the conversion algorithm, rather than by the expert.

Regarding the author's previous study, an interactive algorithm was proposed in [Giacomin and Schneebeli, 2010] [2], where a case study was performed on a carrier system based on conveyor belts and pistons, and though the Ladder and Grafcet diagrams are considered equivalent, the plant is simpler than the one considered here, and it was necessary to improve the algorithm for a realistic case study. Moreover, though other robust formalisms exist, such as Petri nets, this paper prioritizes compliance with the current industrial standard IEC 60848. Hence, since Petri nets are significantly different from Grafcet and not a standard for PLC programming, they are not compared here. In the proposed study, the bibliography review covers papers published from 1970 onwards, and though the design of algorithms has been of particular interest in other areas [Barbosa *et al.*, 2016; Tavares *et al.*, 2017; Villela *et al.*, 2023], after an exhaustive search by using indexed databases, such as the Google Scholar, the Scientific Electronic Library Online (SciELO), Scopus, ACM Digital Library, IEEE Xplore, and the SBC Open Library (SOL), other studies focusing on converting Ladder diagrams into Grafcet diagrams were not found in the literature.

The main contribution of this paper is an interactive algorithm based on BDDs for converting Ladder diagrams into Grafcet diagrams. The proposed approach helps the expert to identify part of the plant's process control flow. Additionally, a robotic neutralization system is considered as a realistic case study, where either transitions AND or OR are addressed. For implementing the main conversion algorithm, other sub-procedures are also programmed.

This paper is organized as follows: Section 2 describes the problem, where the robotic neutralization system is adopted as a case study, as well as its Ladder diagram, Section 3 describes the methodology, where a conversion algorithm is proposed, Section 4 determines the time complexity of the proposed algorithm, Section 5 shows the results, and the conclusions are presented in Section 6.

2 The Problem

This paper aims at converting a Ladder diagram into the equivalent Grafcet diagram, where a robotic neutralization system is used as a case study. Hence, the robotic neutralization system is firstly described in Subsection 2.1, and its Ladder diagram is subsequently presented in Subsection 2.2.

2.1 The Robotic Neutralization System

The robotic neutralization system shown in Figure 1 is considered as a case study, which aims at chemically treating the solution coming from the reservoir and sending it to the next tank. The sensors tl and as are activated when the temperature and acidity are respectively considered suitable. Whenever the values of tl and as are simultaneously considered appropriate, the neutralization ends. Additionally, the sensors ls_1, ls_2 and ls_3 indicate that the level is at or above the respective level.

[2]Translated as: An interactive approach for converting Ladder diagrams into Grafcet diagrams. Published in the XVIII Brazilian Congress on Automation, an event promoted by the Brazilian Society of Automation. The paper is available in Portuguese at http://www.sba.org.br/Proceedings/CBA/CBA2010.zip. Accessed on 24 April, 2026.

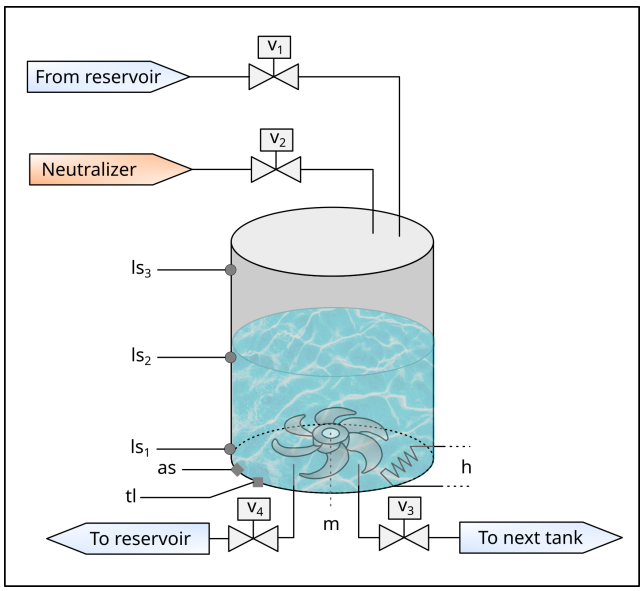


Figure 1. The robotic neutralization system.

The neutralization process proceeds as follows:

- Initially, the tank is empty, the mixer m and the heater h are off and all the valves are closed;
- When the start button s is pressed (not shown), the valve v_1 is opened until the level ls_2 is achieved; this fills the tank with the solution not neutralized;
- The mixer m is turned on if the solution level rises above ls_2, andit is turned off when the level drops below ls_1;
- If the temperature is below a preset point, the heater h is energized;
- If the acidity is not suitable and ls_2 is on, then the valve v_2 is opened, and the neutralizer is added to the tank;
- When the level ls_3 is achieved, and the solution preparation is not finished, the valve v_2 is closed, and the valve v_4 is opened; the valve v_2 is reopened if the level drops below ls_2;
- Whenever both tl and as are on, valves v_1, v_2 and v_4 are closed, the heater h is turned off, and the valve v_3 is opened, thereby sending the neutralized solution to the next tank.

Although the neutralization system shown in Figure 1 is inspired by the study of Falcione and Krogh [1993], here all valves v_1, v_2, and v_4 are kept closed when the valve v_3 is opened, thereby keeping the acidity level constant during the delivery of the neutralized solution to the next tank.

2.2 The Ladder Diagram

Before describing the Ladder diagram of the robotic neutralization system, a short review of the basic concepts of the Ladder diagram is necessary. As shown in Figure 2, the symbols like a circle represent coils (or relays), and since the Ladder diagram is inspired by the old relay contact circuit, it has two lines on its sides, where the first one represents the virtual phase wire, and the later one the virtual ground wire. The symbols in the first rung, having addresses %I0 and %I2, represent the normally open contact (NO) and the normally closed one (NC), respectively. When the address of a normally open contact is on, the contact closes, and when the address of a normally closed contact is on, the contact

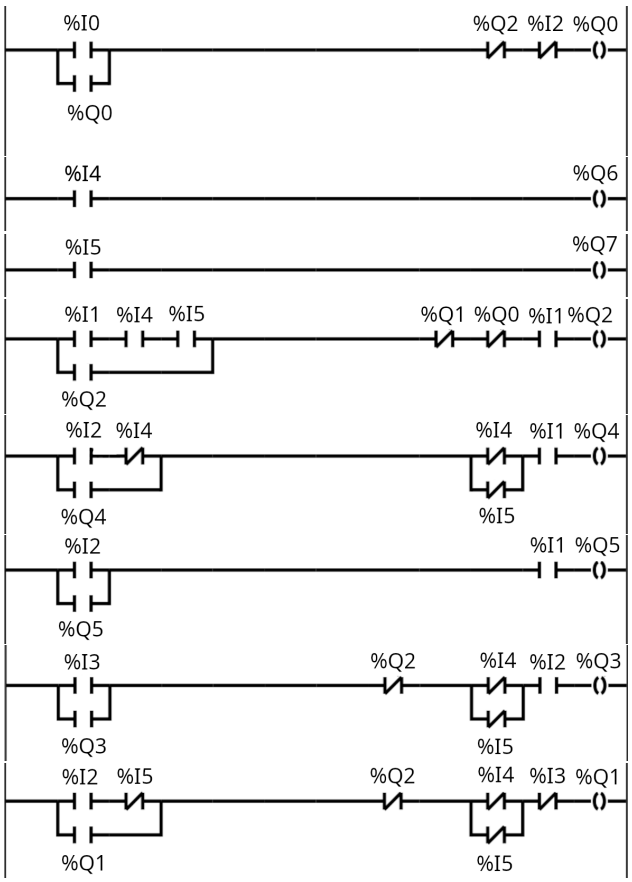


Figure 2. The Ladder diagram for the neutralization system presented in Figure 1.

opens. If a current from the virtual phase wire achieves the coil (relay), and the coil is connected to the virtual ground wire, the coil's address is turned on, and its contacts are updated accordingly. Besides, when the system is in operation and an output coil is on, a physical output acts on the plant, but when an auxiliary coil is on, only an internal address is updated. In contrast, if only a simulation is performed, the output coil can be activated without causing any impact on the plant.

During each cycle, the diagram is processed from the first rung to the last one, which means the first rung is updated first, afterwards the second rung, and so on. As observed in Figure 2, the normally open contact %Q0 is included in the activation logic of the coil having the same address. Here, when this happens, the rung is called persistent, and persistent rungs are usually necessary for making the output command durable until some terminal condition is satisfied. Table 1 shows a description of each input and output address of the proposed Ladder diagram.

3 The Methodology

Before describing how a Ladder diagram can be converted into a Grafcet diagram, a short review of the basic concepts of the Grafcet diagram is necessary.

3.1 The Grafcet Diagram

In the standardized representation, a Grafcet is a graph with two types of nodes, i.e., steps and transitions (a Grafcet has at least one step and one transition). A directed arc can connect a step to a transition or a transition to a step, but never two

Address	Description
%I0	Start button
%I1	Level at ls_1
%I2	Level at ls_2
%I3	Level at ls_3
%I4	Ideal temperature achieved.
%I5	Ideal pH reached.
%Q0	Open valve v_1.
%Q1	Open valve v_2.
%Q2	Open valve v_3.
%Q3	Open valve v_4.
%Q4	Turn on the heater.
%Q5	Turn on the mixer.
%Q6	Turn on the light tl.
%Q7	Turn on the light as.

Table 1. Input and output addresses of the proposed Ladder diagram.

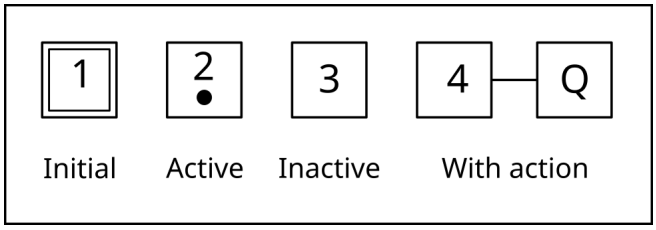


Figure 3. Types of steps. If a step is active, the variable Q is turned on, see [David, 1995] for details.

steps or two transitions [David, 1995].

As shown in Figure 3, a step, represented by a square, can be in one of two states, i.e., active or inactive, and a token in the step indicates that it is active. Besides, the initial step, which is activated when the system is started, is represented by a double square.

As shown in Figure 4, transitions can be classified in terms of their kind of connections or by the logic it represents. With regard to the kind of connections, a transition can be simple, a conjunction or a disjunction. According to the logic it represents, the transition can be of type AND or OR. Normally, transitions going from top to bottom are represented by continuous lines, while transitions going from bottom to top have an additional arrow. Hence, there are six possible combinations between these two classifications.

A transition AND is preceded or followed by double bars; in the first case, it must wait until all of its input steps are active before firing the transition. In the second case, all output steps are activated when the transition is fired. Each receptivity R_i is a function of Grafcet variables and it is associated with each transition i. One says that a transition is fireable if, firstly, all steps preceding the transition are active, and secondly, the transition's receptivity is active. In a disjunction OR, at least one transition is fired if the transition conditions are active, as well as the preceding step, see [David, 1995] for details. Hence, a junction OR activates the output step if at least one of the transitions is fired.

Transition conditions can also be a function of auxiliary and output variables, and actions can activate auxiliary variables, since these rules are allowed in the standard IEC 60848 [IEC, 2013], and they are adopted here for simplifying the conversion process and for making the Grafcet diagram

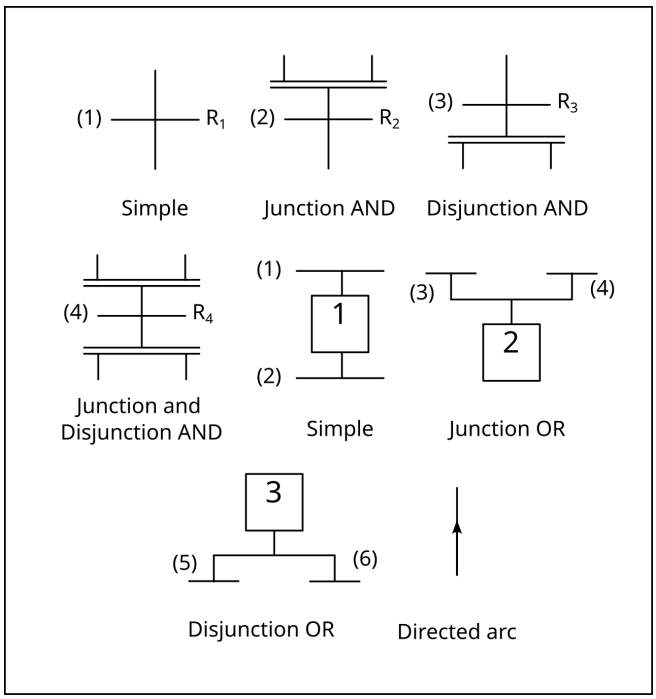


Figure 4. Six kinds of transitions in a Grafcet diagram. One says that the transition fires if its receptivity R_i is enabled and all previous steps are active, see [David, 1995] for details.

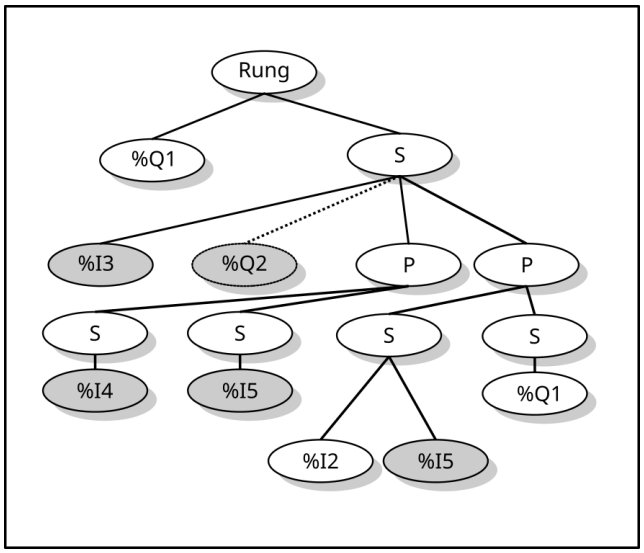
equivalent to the Ladder diagram.

3.2 The Data Structures

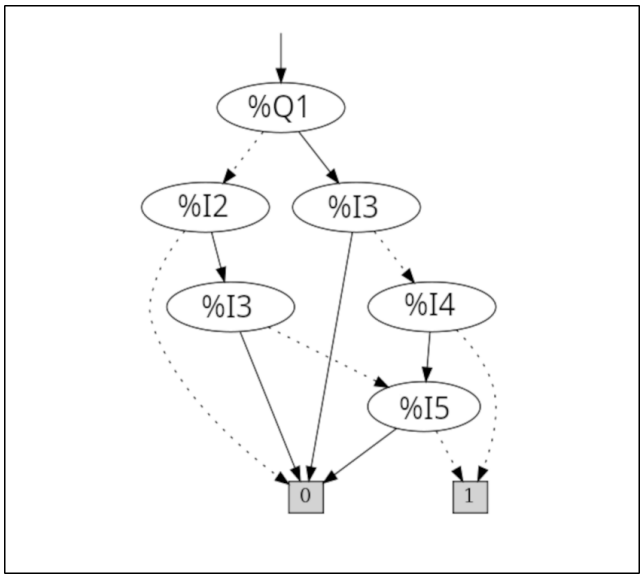
It is well-known that a sequential control system, which is composed of two main parts, i.e., the controller and the plant, typically has several states. Here, rungs having normally open feedback contacts always represent a state and these rungs are considered here persistent rungs because they are kept active for some time even when some of their activation conditions cease. In the proposed study, non-persistent rungs represent conditional actions or activation conditions of persistent rungs and a state is always represented in the Grafcet diagram by a step, which also has an activation condition, a deactivation condition and optionally an action. However, not every state is related to a persistent rung in the Ladder diagram since the Grafcet diagram may require additional steps. Besides, two steps are different if they are placed at distinct places of the Grafcet diagram, even if they have the same name.

Before discovering the sequence among the states, a syntactic tree is built for each rung in the Ladder diagram. Figure 5a shows the syntactic tree of the rung having output coil $\%Q1$. As shown in Figure 5a, the syntactic tree has different nodes, i.e., the root, which represents the rung, and the serial and parallel nodes, which represent the serial and parallel connections, respectively, and the leaf nodes, which represent NO and NC contacts (the NC contacts are in gray). Besides, each root node always has at least one child node representing the respective coil and each parallel node has at least two serial nodes.

Afterwards, BDDs are built from the syntactic trees, because they smooth the exponential growth in running time related to the logic function manipulation, see [Bryant, 1986] for details. Basically, the BDD representing the NC contacts are produced by applying the not operation (the logic-not



(a) The syntactic tree. The dotted line points out that the node is temporally deactivated.



(b) The activation BDD.

Figure 5. a) The syntactic tree representing the activation logic of the output coil %Q1. The nodes P and S represent parallel and serial connections, respectively, and the darkened nodes represent NC contacts. b) The activation BDD obtained from the syntactic tree, where dotted and continuous lines respectively represent that the logical values **0**(false) and **1** (true) are attributed to their respective variables.

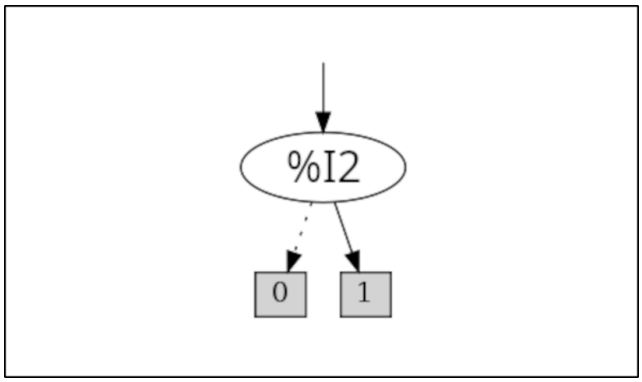


Figure 6. The deactivation BDD of the output coil %Q0.

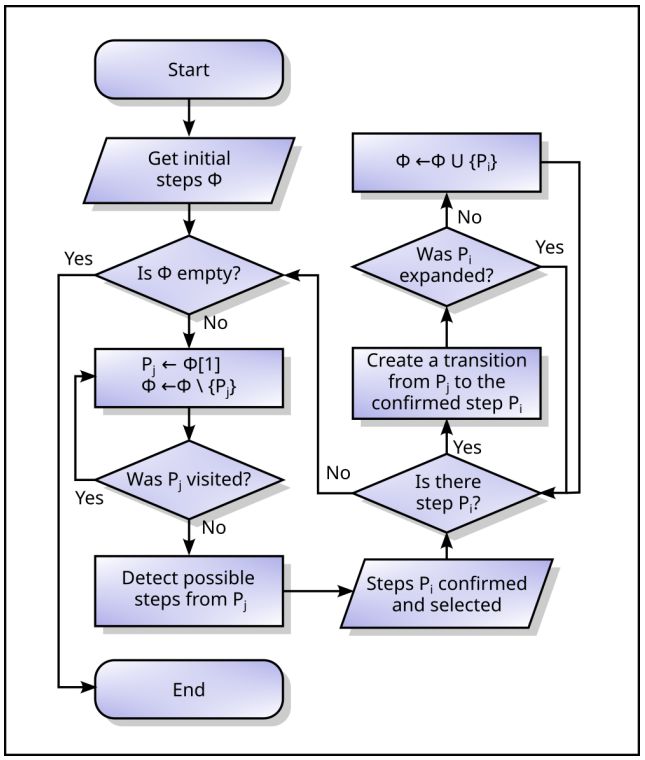


Figure 7. The general overview of the conversion algorithm. The rectangles, rounded rectangles, diamonds, and trapezoids respectively represent processes, start or end, decisions, and data input from the expert.

or complement) to its respective NO representation and new BDDs are recursively built by applying valid logical operations to the previous BDD, where $\wedge$ (the logical **and**) and $\vee$ (the logical **or**) logic operations are applied when the serial and parallel nodes appear in the syntactic tree, respectively, until all syntactic tree's nodes are processed. As a result, each persistent rung has an equivalent activation BDD produced from its syntactic tree. Figure 5b represents the activation BDD produced from the syntactic tree shown in Figure 5a.

Differently from the activation BDD, the BDD representing the deactivation logic of a given rung is not obtained directly from the syntactic tree. Instead, it is obtained by handling the respective activation BDD, where the NO feedback contact is made equal to **1** (true in a Boolean representation), thereby producing a temporary BDD. Additionally, though interlocking contacts are indispensable for preventing two output coils from being simultaneously activated when they represent conflicting commands, if the two rungs in the sequence are not conflicting, then the interlocking contacts also are temporarily removed from the temporary BDD, which is later complemented by using a $\neg$ operation (the logical **not**), thereby resulting in the deactivation BDD. For instance, Figure 6 shows the deactivation BDD of the rung having output coil %Q0, resulting from the respective syntactic tree and activation BDD. One observes that the path from %I2 to 1 in Figure 6 is also a subpath in a path from %Q1 to 1 in Figure 5b. Hence, %I2 may deactivate %Q0 and activate %Q1, and Algorithm 1 described in Subsection 3.3 is based on this principle.

3.3 The Conversion Algorithm

The proposed algorithm's general overview is shown in Figure 7, and its running is illustrated in Figure 8. Firstly, the

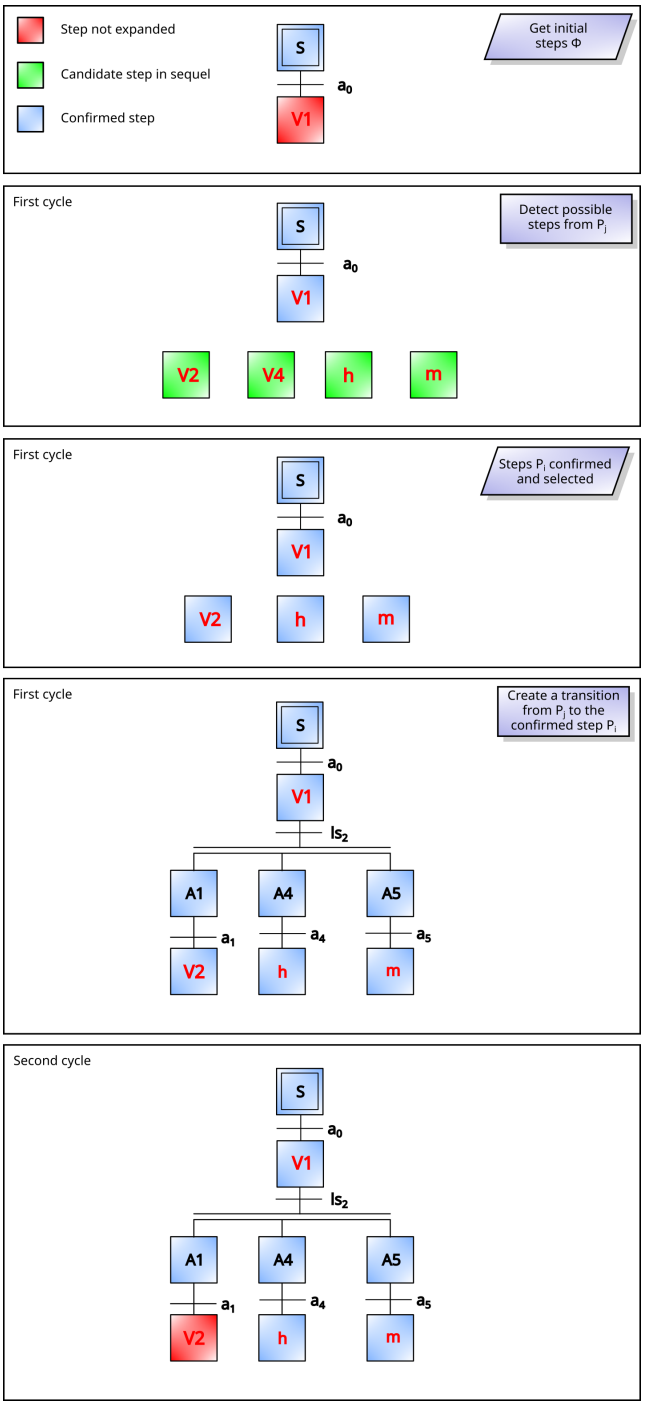


Figure 8. The first stage-by-stage iteractions of the conversion algorithm.

conversion algorithm selects the initial step v_1, which is by default considered not expanded. Secondly, the candidate steps following v_1 are identified by using BDDs. Thirdly, some candidate steps are confirmed by the expert. Fourthly, the transitions are created from v_1 to each confirmed step. Finally, the second cycle starts by considering v_2 as the next not expanded step, since it has not been expanded or visited yet. The algorithm continues until all steps have been expanded. For simplicity reasons, the addresses of the coils are omitted.

The conversion algorithm's flowchart, shown in Figure 7, is presented in detail in Algorithm 1. Algorithm 1 receives as input the set Λ of steps, where each step is associated with each persistent rung in the Ladder diagram. As a consequence, each step in Λ has its activation and deactivation

conditions computed by using its activation and deactivation BDDs. Afterwards, Algorithm 1 requests that the expert inform the steps in Λ following the initial step, as well as the kind of transition. Afterwards, while the set Φ is not empty, Algorithm 1 runs the external loop.

Within the external loop, the first two actions consist of getting the first step from the set Φ and removing it from the same set. Afterwards, the first **if** checks if step P_j was not expanded, i.e., if P_j is not in the set Δ. If this condition is satisfied, the step is considered expanded, i.e., it is included in Δ. Afterwards, the steps in the set Θ possibly following step P_j are computed according the rule: a step P_i is possibly followed by a step P_j if the same subgraph Γ_1 (possibly disconnected) activating the P_j's deactivation BDD also activates the P_i's activation BDD (a vertex is also a subgraph). Although the subgraph Γ_1 can be disconnected, the equivalent connected one Γ_2, which is a BDD, can be produced by using the same variables and BDD operations. However, since this rule is a heuristic, the possible sequence must be confirmed by the expert, who interacts with the algorithm. This confirmation results in the confirmed steps $\{P_1, ..., P_m\}$, which may also contain auxiliary steps and the expert must also inform the kind of transition α employed for the confirmed sequence.

Algorithm 1 ends by iterating through the set $\{P_1, ..., P_m\}$. For each step P_i, the kind of transition α is requested for creating the sequence $P_j \rightarrow D_j \rightarrow P_i$[3], and if step P_i has not been expanded, it is included in the set Φ, which is the set of unexpanded steps. Each step D_j is an auxiliary step and it is necessary because the deactivation condition of step P_j generally is not exactly equal to the activation condition of step P_i. Besides, the transition condition from P_j to D_j is represented by subgraph Γ_2's BDD and the one from D_j to P_i is represented by P_i's activation BDD.

Since all Algorithm 1's instructions are defined, its time complexity is determined in Section 4.

Algorithm 1 The Conversion Algorithm.

```
1:  procedure L2G(Λ)                                   ▷ Λ is the set of steps.
2:      Φ ← initial_steps(Λ)                                          ▷ c1 · |Λ|
3:      while |Φ| ≠ 0 do                                              ▷ c2
4:          P_j ← Φ[1]                                                ▷ c3 · |Φ|
5:          Φ ← Φ \ {P_j}                                             ▷ c4 · |Φ|
6:          if P_j ∉ Δ then                                           ▷ c5 · |Φ| · |Δ|
7:              Δ ← Δ ∪ {P_j}                                         ▷ c6 · |Φ|
8:              Θ ← ∅                                                 ▷ c7 · |Φ|
9:              for i ← 1 to |Λ| do                                   ▷ c8 · |Φ| · |Λ|
10:                 if |f_D(P_j, P_i)| ≠ 0 then
11:                                                                   ▷ c9 · |Φ| · |Λ| · T_D
12:                     Θ ← Θ ∪ {P_i}                                 ▷ c10 · |Φ| · |Λ|
13:                 end if                                            ▷ c11 · |Φ| · |Λ|
14:             end for                                               ▷ c12 · |Φ| · |Λ|
15:             [{P_1, ..., P_m}, α] ← conf(Θ)
16:                                                                   ▷ c13 · |Θ| · |Φ|
17:             for i ← 1 to m do                                     ▷ c14 · |Φ| · m
18:                 Make P_j → D_j → P_i by using α [2]
19:                                                                   ▷ c15 · |Φ| · m
20:                 if P_i ∉ Φ then                                   ▷ c16 · m · |Φ|^2
21:                     Φ ← Φ ∪ {P_i}                                 ▷ c17 · |Φ| · m
22:                 end if                                            ▷ c18 · |Φ| · m
23:             end for                                               ▷ c19 · |Φ| · m
24:         end if                                                    ▷ c20 · |Φ|
25:     end while                                                     ▷ c21 · |Φ|
26: end procedure                                                     ▷ c22
```

Algorithm 2 The Discover Function.

```
1:  procedure f_D(P_1, P_2)                            ▷ P_1 and P_2 are steps.
2:      f_1 ← P_1.get_desativation()                                  ▷ c30
3:      f_2 ← P_2.get_ativation()                                     ▷ c31
4:      V_1 ← get_vars(b, f_1, V_1)                                   ▷ c32 · T_V
5:      V_2 ← get_vars(b, f_2, V_2)                                   ▷ c33 · T_V
6:      V ← V_1 ∩ V_2                                                 ▷ c34 · |V_1| · |V_2|
7:      if V ≠ ∅ then                                                 ▷ c35
8:          C_1 ← get_cubes(b, f_1)                                   ▷ c36 · T_C
9:          C_2 ← get_cubes(b, f_2)                                   ▷ c37 · T_C
10:         Ψ ← ∅                                                     ▷ c38
11:         Ω ← ∅                                                     ▷ c39
12:         return f_CO(V, C_1, C_2, Ψ, 0, Ω)                         ▷ c40 · T_CO
13:     end if                                                        ▷ c41
14: end procedure
```

4 The Time Complexity

Algorithm 1's time complexity depends on Algorithms 2-5. Hence, the last ones are determined first, in a bottom-up fashion. In the following lines, if A is some set, then $|A|$ represents its size, and as it is usually done the symbol $\square$ indicates the end of each demonstration.

The running time of Algorithm 5 is determined by the summation of the running time of its lines. Additionally, each line's running time depends on some constant c_i and the number of times the line is run. Since the summation's highest order term is the most important for large input sizes, all costs c_i can be rounded to 1 without compromising the analysis [Cormen *et al.*, 2009]. Thus, let the input size n be the number of variables of f. Then, the Algorithm 5's time complexity can be determined by the recurrence $T_C(n)$ according to (1)-(2) as follows

$$T_C(1) = 1 \tag{1}$$
$$T_C(n) = 2 \cdot T(n-1) + 1 \tag{2}$$

Lemma 1. *The solution of $T_C(n)$ is $O(2^n)$, which is also the time complexity of Algorithm 5.*

Proof. If the recurrence definition is applied several times, the next expansions are determined as follows

$$\begin{aligned} T_C(n) &= 2 \cdot T_C(n-1) + 1 \\ T_C(n-1) &= 2 \cdot T_C(n-2) + 1 \\ T_C(n-2) &= 2 \cdot T_C(n-3) + 1 \\ &\cdots \\ T_C(2) &= 2 \cdot T_C(1) + 1 \\ T_C(1) &= 1 \end{aligned}$$

If the recurrence expansions are substituted in the pre-

[3] $P_j \rightarrow D_j \rightarrow P_i$ means that the step P_j is followed by the step D_j, which is followed by the step P_i.

Algorithm 3 The Common Function.

```
1:  procedure f_CO(V, C_1, C_2, Ψ, i, Ω)
2:      if |Ω| ≠ 0 then                                   ▷ c_50
3:          return Ω                                      ▷ c_51
4:      else if i < |V| then                              ▷ c_52
5:          v_T ← {V[i], T}                               ▷ c_53
6:          Ψ ← Ψ ∪ {v_T}                                 ▷ c_54
7:          Ω ← F_CO(V, C_1, C_2, Ψ, i + 1, Ω)            ▷ c_55 · T_CO
8:          Ψ ← Ψ \ {v_T}                                 ▷ c_56
9:          v_F ← {V[i], F}                               ▷ c_57
10:         Ψ ← Ψ ∪ {v_F}                                 ▷ c_58
11:         Ω ← F_CO(V, C_1, C_2, Ψ, i + 1, Ω)            ▷ c_59 · T_CO
12:         Ψ ← Ψ \ {v_F}                                 ▷ c_60
13:     else                                              ▷ c_61
14:         c ← Cube(Ψ)                                   ▷ c_62
15:         if c ∈ C_1 ∧ c ∈ C_2 then                     ▷ c_63
16:             Ω ← Ω ∪ {c}                               ▷ c_64
17:         end if                                        ▷ c_65
18:     end if                                            ▷ c_66
19:     return Ω                                          ▷ c_67
20: end procedure
```

Algorithm 4 Get variables.

```
1:  procedure get_vars(b, f, V)
2:      if f = b.one() ∨ f = b.zero() then                ▷ c_70
3:          return V                                      ▷ c_71
4:      else                                              ▷ c_72
5:          v ← b.getvar(f)                               ▷ c_73
6:          if v ∉ V then                                 ▷ c_74
7:              V ← V ∪ {v}                               ▷ c_75
8:          end if                                        ▷ c_76
9:          V ← get_vars(b, b.low(f), V)                  ▷ c_77 · T_V
10:         V ← get_vars(b, b.high(f), V)                 ▷ c_78 · T_V
11:     end if                                            ▷ c_79
12:     return V                                          ▷ c_80
13: end procedure
```

vious ones, one obtains

$$T_C(n) = 2 \cdot (2 \cdot (2 \cdot T_C(n-3) + 1) + 1) + 1$$
$$T_C(n) = 2^3 \cdot T_C(n-3) + 2^2 + 2 + 1$$

The last equation above is generalized according to the pattern it obeys, and afterwards, one considers that the expansion ends when $n - k = 1$. Then, one obtains

$$T_C(n) = 2^k \cdot T_C(n-k) + 2^{k-1} + 2^{k-2} + \cdots + 2 + 1$$
$$T_C(n) = 2^{n-1} \cdot T_C(1) + 2^{n-2} + 2^{n-3} + \cdots + 2 + 1$$
$$T_C(n) = 2^{n-1} + 2^{n-2} + 2^{n-3} + \cdots + 2 + 1$$
$$T_C(n) = 2^n - 1$$
$$T_C(n) \in O(2^n)$$

□

In a pessimistic scenario, the Algorithm 4's time complexity is determined by the solution of $T_V(n, 1)$ similarly as follows

$$T_V(1, n) = n$$
$$T_V(n, n) = 2 \cdot T_V(n-1, n+1) + n$$

Algorithm 5 Get cubes.

```
1:  procedure get_cubes(b, f)
2:      Φ ← ∅                                             ▷ c_81
3:      Ω ← ∅                                             ▷ c_82
4:      return get_cubes_(b, f, Φ, Ω)                     ▷ c_83
5:  end procedure
6:  procedure get_cubes_(b, f, Φ, Ω)
7:      if f ≠ b.one() ∧ f ≠ b.zero() then                ▷ c_84
8:          x ← b.getVar(f)                               ▷ c_85
9:          v ← [x, T]                                    ▷ c_86
10:         Φ ← Φ ∪ {v}                                   ▷ c_87
11:         Ω ← get_cubes_(b, b.high(f), Φ, Ω)            ▷ c_88 · T_C
12:         Φ ← Φ \ {v}                                   ▷ c_89
13:         v ← [x, F]                                    ▷ c_90
14:         Φ ← Φ ∪ {v}                                   ▷ c_91
15:         Ω ← get_cubes_(b, b.low(f), Φ, Ω)             ▷ c_92 · T_C
16:         Φ ← Φ \ {v}                                   ▷ c_93
17:     else if f = b.getOne() then                       ▷ c_94
18:         c ← Cube(Φ)                                   ▷ c_95
19:         Ω ← Ω ∪ {c}                                   ▷ c_96
20:     end if                                            ▷ c_97
21:     return Ω                                          ▷ c_98
22: end procedure
```

Lemma 2. *The solution of $T_V(n, 1)$ is $O(n \cdot 2^n)$, which is also the time complexity of Algorithm 4.*

Proof. If the recurrence definition is recursively applied several times, the next expansions are determined as follows

$$T_V(n, 1) = 2 \cdot T_V(n-1, 2) + 1$$
$$T_V(n-1, 2) = 2 \cdot T_V(n-2, 3) + 2$$
$$T_V(n-2, 3) = 2 \cdot T_V(n-3, 4) + 3$$
$$T_V(n-3, 4) = 2 \cdot T_V(n-4, 5) + 4$$
$$\cdots$$
$$T_V(1, n) = 1$$

If the expanded recurrences are substituted in the previous ones, one obtains

$$T_V(n, 1) = 2 \cdot [2 \cdot [2 \cdot [2 \cdot T_V(n-4, 5) + 4] + 3] + 2] + 1$$
$$T_V(n, 1) = 2^4 \cdot T_V(n-4, 5) + 2^3 \cdot 4 + 2^2 \cdot 3 + 2 \cdot 2 + 1$$
$$T_V(n, 1) = 2^k \cdot T_V(n-k, k+1) + 2^{k-1} \cdot k + \cdots + 2^1 \cdot 2 + 1$$

The recurrence above ends when $n - k = 1$. Hence, by making $k = n - 1$ one obtains

$$T_V(n, 1) = 2^{n-1} \cdot n + 2^{n-2} \cdot (n-1) + \cdots + 1$$

It will be shown by induction that $T_V(n, 1) = 2^n \cdot (n-1) + 1$ for all n greater than or equal to 1.

Base: suppose that $n = 1$. Then $2^n \cdot (n-1) + 1 = 2^1 \cdot (1-1) + 1 = 1$, and $2^{n-1} \cdot n = 2^0 \cdot 1 = 1$. Hence, the hypothesis is true when $n = 1$.

Induction: suppose that $2^n \cdot (n-1) + 1$ is equal to $2^{n-1} \cdot n + 2^{n-2} \cdot (n-1) + \cdots + 1$. Hence,

$2^n \cdot (n+1) + 2^{n-1} \cdot n + \cdots + 1 =$
$= 2^n \cdot (n+1) + 2^n \cdot (n-1) + 1 =$
$= 2^n \cdot n + \cancel{2^n} + 2^n \cdot n - \cancel{2^n} + 1 =$

$= 2^{n+1} \cdot n + 1$

Conclusion: $T_V(n,1) = 2^n \cdot (n-1) + 1$ for all n greater than or equal to 1. Therefore, $T_V(n,1) \in O(n \cdot 2^n)$. □

The Algorithm 3's time complexity can be determined by the recurrence $T_{CO}(n)$ as follows

$$T_{CO}(1) = 1$$
$$T_{CO}(n) = 2 \cdot T_{CO}(n-1) + 1$$

which has the same structure as $T_C(n)$. Hence, Algorithm 3's time complexity is $O(2^n)$.

Similarly, Algorithm 2's time complexity can be determined by $T_D(n)$ as follows

$$T_D(n) = 2 \cdot T_V(n,1) + 2 \cdot T_C(n) + T_{CO}(n) + 1$$
$$T_D(n) \in 2 \cdot O(n \cdot 2^n) + 2 \cdot O(2^n) + O(2^n) + 1$$
$$T_D(n) \in O(n \cdot 2^n)$$

Since the time complexities of all Algorithm 1's subroutines were found, the next theorem can be postulated as follows

Theorem 1. *The solution of* $T(|\Lambda|, n)$ *is* $O(|\Lambda| \cdot (|G| \cdot log_2|G| + |\Lambda|))$*, which is also the time complexity of Algorithm 1.*

Proof. Since $T_D(n)$ is determined, the time complexity of Algorithm 1 can be calculated as follows

$$T(|\Lambda|, n) = c_1 \cdot |\Lambda| + \cdots$$
$$\cdots + (c_3 + c_4 + c_6 + c_7 + c_{20} + c_{21}) \cdot |\Phi| + \cdots$$
$$\cdots + |\Phi| \cdot |\Lambda| \cdot (c_5 + c_8 + c_{10} + c_{11} + c_{12}) + \cdots$$
$$\cdots + (c_2 + c_{22}) + |\Phi| \cdot |\Lambda| \cdot T_D(n) + c_{13} \cdot |\Theta| \cdot |\Phi| + \cdots$$
$$\cdots + |\Phi| \cdot m \cdot (c_{14} + c_{15} + c_{17} + c_{18} + c_{19}) + c_{16} \cdot m \cdot |\Phi|^2$$

Since the highest order term of $T(|\Lambda|, n)$ is the most important for large input sizes, all summations of all constants c_i can be rounded to 1 without compromising the analysis, as follows

$$T(|\Lambda|, n) = |\Lambda| + |\Phi| + |\Phi| \cdot |\Lambda| + 1 + |\Phi| \cdot |\Lambda| \cdot T_D(n) + \cdots$$
$$\cdots + |\Theta| \cdot |\Phi| + |\Phi| \cdot m + m \cdot |\Phi|^2$$

In a pessimistic scenario, $|\Phi|$, $|\Theta|$ and m can be rounded to $|\Lambda|$. Besides, since $T_D(n) \in O(n \cdot 2^n)$, one obtains

$$T(|\Lambda|, n) = |\Lambda| + |\Lambda| + |\Lambda|^2 + 1 + |\Lambda|^2 \cdot n \cdot 2^n + \cdots$$
$$\cdots + |\Lambda|^2 + |\Lambda|^2 + |\Lambda|^3$$

Since Algorithm 1 interacts with the expert $|\Phi|$ times, all non-constant terms are reduced by a factor of $|\Phi|$. Since $|\Phi|$ is rounded to $|\Lambda|$, one has

$$T(|\Lambda|, n) = 1 + 1 + |\Lambda| + 1 + |\Lambda| \cdot n \cdot 2^n + |\Lambda| + |\Lambda| + |\Lambda|^2$$

Since the highest order terms of $T(|\Lambda|, n)$ are $|\Lambda| \cdot n \cdot 2^n$ and $|\Lambda|^2$, it follows that

$$T(|\Lambda|, n) \in O(|\Lambda| \cdot (n \cdot 2^n + |\Lambda|)) \quad (3)$$

□

In contrast, let $|G|$ be the number of nodes the BDD would have if it were not reduced, i.e., if it were a complete binary tree. Then, $|G|$ would be equal to $2^{n+1} - 1$. Hence, since $2^n < 2^{n+1} - 1$, (3) can be rewritten as

$$T(|\Lambda|, n) \in O(|\Lambda| \cdot (|G| \cdot log_2|G| + |\Lambda|)) \quad (4)$$

However, the reduce operation optimizes the BDD so that $|G|$ is minimized, which softens the exponential growth, since $|G|$ is significantly lower than $2^{n+1} - 1$ [Bryant, 1986] and each node can be visited only once for determining the number of variables. Furthermore, n is the number of variables of f, which is frequently much smaller than the number of variables of the whole Ladder diagram. Usually, n is lower than 20, while BDDs can efficiently handle functions having up to one hundred variables [Bryant, 1986]. Hence, Algorithm 1 can be considered efficient in practice.

5 The Results

The Ladder diagram shown in Section 2 was provided as input to Algorithm 1, which converted it into the equivalent Grafcet diagram shown in Figure 9. The videos showing the tests performed are available at https://github.com/pagiacomin/ladder-grafcet, and during the interactions, the slowest response was given in less than 3 s. Furthermore, several scenarios were taken into account, either for the control software or the conversion algorithms, and the tests were considered satisfactory for a proof-of-concept version. Since the resulting original Grafcet diagram was considered graphically too large, it was translated to the equivalent one (preserving its logic, but it was just redrawn for a better layout) by using the updated version of the software Inkscape [Harrington *et al*., 2003].

As shown in Figure 9, all steps beginning with the letter A represent auxiliary steps, i.e., they are not directly related to persistent rungs in the Ladder diagram. As observed in Figure 9, steps v_2, m, and h can be active simultaneously. This is compatible with the neutralization system's specification since the heater, the mixer and the neutralization valve can be turned on concurrently. The diagram shown in Figure 2 was simulated stage by stage and side by side with the diagram shown in Figure 9 after submitting them to the same initial state, when all input and coils are off, and to the input events shown in Tables 2-4, and they produced the same output actions shown in Tables 2-4, where **on**$\{\bullet\}$ and **off**$\{\bullet\}$ means that the coils or inputs in $\{\bullet\}$ are on and off, respectively. Hence, they can be considered equivalent.

As observed in Figure 9, different steps can have the same name because they are directly related to the same persistent rung in the Ladder diagram, though they are activated at different moments. This behaviour is also observed in the Ladder diagram, and this was possible in practice because each step identifier has two parts: firstly, the number of the related rung, and secondly, an identifier within the set of steps related to the same rung. Hence, in Line 18 of Algorithm 1, two steps with the same name are considered different if they have a different instance number. Additionally, as shown in Figure 9, if each rung were related to only one step in the Grafcet diagram, the diagram would be smaller. However, breaking this rule was necessary to improve the Grafcet

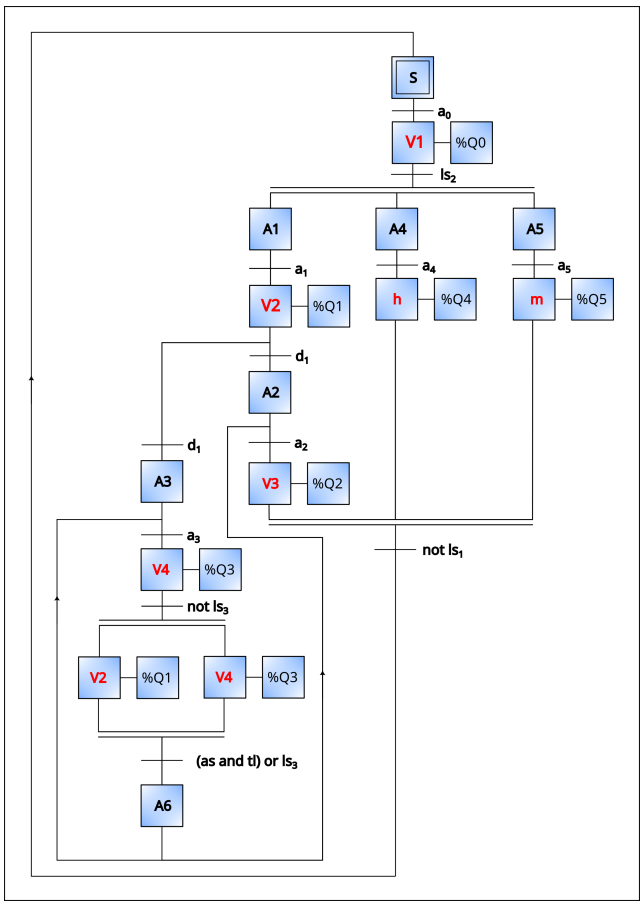


Figure 9. The resulting Grafcet diagram.

	Input	Output	Effect
1	**on**$\{s\}$	**on**$\{v_1\}$	**on**$\{ls_1\}$
2	**on**$\{ls_2\}$	**on**$\{m\}$	**off**$\{v_1\}$
3	**on**$\{ls_2\}$, **off**$\{tl\}$	**on**$\{h\}$	
4	**on**$\{ls_2\}$, **off**$\{as\}$	**on**$\{v_2\}$	
5	**on**$\{ls_3\}$	**on**$\{v_4\}$	**off**$\{ls_3, , v_2\}$
6	**off**$\{ls_3\}$	**on**$\{v_2\}$	**on**$\{as\}$
7	**on**$\{as, tl, ls_1\}$	**on**$\{v_3\}$	**off**$\{ls_3, ls_2, ls_1, v_4, v_2\}$
8	**off**$\{ls_1\}$		**off**$\{m, h, v_3, as, tl\}$

Table 2. The expected sequence when solution is mixed and returned to the reservoir; this happens when more neutralizer is required than expected and the flow in v_4 is approximately equal to the flow in v_2.

diagram's flexibility, which was able to process the greatest number of the plant's process control flows, without compromising the equivalence relation to the Ladder diagram. This advantage is not observed in the study of Zanma *et al.* [1999] and Falcione and Krogh [1993], where each rung is associated with at most only one step in the Grafcet diagram. As observed in Figure 9, v_3 remains active until the tank is empty, while the other valves remain closed. Hence, the proposed neutralization system keeps the solution's acidity constant during its delivery to the next tank, differently from what happens in the studies of Falcione and Krogh [1993] and Lopes and Sousa [2017], thereby improving the quality of the neutralized solution.

As shown in Table 6, a quantitative comparison is performed among the approaches based on graphs [Falcione and Krogh, 1993], state-space [Lopes and Sousa, 2017], and the proposed one (interactive), by taking into account the number of vertices, transitions, and combinations between the activations and deactivations of the valves, where actions are not considered vertices. As observed in Table 6, the state-space approach presented the greatest number of vertices and transitions. This stems from the fact that the state-space approach does not provide a specific formalism for representing parallel tasks, which may lead to the well-known state-space explosion.

	Input	Output	Effect
1	**on**$\{s\}$	**on**$\{v_1\}$	**on**$\{ls_1\}$
2	**on**$\{ls_2\}$	**on**$\{m\}$	**off**$\{v_1\}$
3	**on**$\{ls_2\}$, **off**$\{tl\}$	**on**$\{h\}$	
4	**on**$\{ls_2\}$, **off**$\{as\}$	**on**$\{v_2\}$	
5	**on**$\{as, tl, ls_1\}$	**on**$\{v_3\}$	**off**$\{ls_3, ls_2, ls_1, v_2, v_1\}$
6	**off**$\{ls_1\}$		**off**$\{m, h, v_3, v_4, as, tl\}$

Table 3. The expected sequence when the solution is mixed without returning to the reservoir; this happens when little neutralizer is required.

	Input	Output	Effect
1	**on**$\{s\}$	**on**$\{v_1\}$	**on**$\{ls_1\}$
2	**on**$\{ls_2\}$	**on**$\{m\}$	**off**$\{v_1\}$
3	**on**$\{ls_2\}$, **off**$\{tl\}$	**on**$\{h\}$	
4	**on**$\{ls_2\}$, **off**$\{as\}$	**on**$\{v_2\}$	
5	**on**$\{ls_3\}$	**on**$\{v_4\}$	**off**$\{v_2\}$
6	**off**$\{ls_3\}$	**on**$\{v_2\}$	
7	**on**$\{ls_3\}$	**on**$\{v_4\}$	**off**$\{v_2\}$
8	**off**$\{ls_3\}$	**on**$\{v_2\}$	
9	**on**$\{as, tl, ls_1\}$	**on**$\{v_3\}$	**off**$\{ls_3, ls_2, v_4, v_2, v_1\}$
10	**off**$\{ls_1\}$		**off**$\{m, h, v_3, as, tl\}$

Table 4. The expected sequence when the flow in v_4 is lower than the flow in v_2; in this case, v_4 is on while v_2 toggles between on and off until the ideal acidity is reached.

With respect to the number of combinations between the activations and deactivations of the valves, the state-space approach provided only one valid combination, i.e., $v_1 \rightarrow v_2 \rightarrow v_3$, because there is no transition from v_4 to the initial state, see [Lopes and Sousa, 2017] for details. Additionally, the approach based on graphs suffered from the same problem, since the only possible combination was v_1 followed by the parallel activation of v_2, v_3, and v_4. Although the approach based on graphs exhibited the smallest number of vertices and transitions, the number of combinations between the activations and deactivations of the valves was also considered small (only one). In contrast, the interactive approach yielded three possible combinations, as shown in Tables 2-4, thereby improving the system's flexibility.

A video was recorded showing the algorithm interactively converting the Ladder diagram shown in Figure 2 into the Grafcet diagram shown in Figure 9, and it is also available at https://github.com/pagiacomin/ladder-grafcet. As shown in Figure 10, of the 13 real sequences (5+8), 8 were detected and confirmed by the expert and 5 were not detected, i.e., they were directly informed by the expert. Besides, of the 22 sequences (14+8) the system detected and suggested to the expert, 8 were confirmed. Hence, the proposed strategy helped the expert to confirm most of the real sequences (more than 60%). If the interactive approach were not used, these

Name	Condition
a_0	$(v_1 \wedge \neg ls_2) \vee (\neg v_1 \wedge s \wedge \neg ls_2)$
a_1	$(v_2 \wedge \neg ls_3 \wedge tl \wedge \neg as) \vee (v_2 \wedge \neg ls_3 \wedge \neg tl)...$ $... \vee (\neg v_2 \wedge ls_2 \wedge \neg ls_3 \wedge \neg as)$
a_2	$(v_3 \wedge ls_1) \vee (\neg v_3 \wedge ls_1 \wedge tl \wedge as)$
a_3	$(\neg v_3 \wedge v_4 \wedge ls_2 \wedge tl \wedge \neg as) \vee (\neg v_3 \wedge v_4...$ $... \wedge ls_2 \wedge \neg tl) \vee (\neg v_3 \wedge \neg v_4 \wedge ls_2 \wedge ls_3 \wedge ...$ $... \wedge tl \wedge \neg as) \vee (\neg v_3 \wedge \neg v_4 \wedge ls_2 \wedge ls_3 \wedge \neg tl)$
a_4	$(h \wedge ls_1 \wedge \neg tl) \vee (\neg h \wedge ls_1 \wedge ls_2 \wedge \neg tl)$
a_5	$(m \wedge ls_1) \vee (\neg m \wedge ls_1 \wedge ls_2)$
d_1	$ls_3 \vee (\neg ls_3 \wedge tl \wedge as)$

Table 5. The activation and deactivation conditions extracted from the Ladder diagram shown in Figure 2 considering the symbols shown in Tables 1.

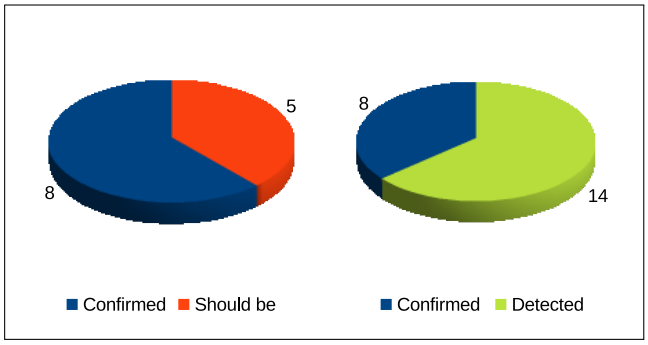


Figure 10. On the left, the circle represents the total number of desirable sequences. On the right, the circle represents the total number of sequences detected, from which a small part was confirmed.

Approach	Transitions	Vertices	Combinations
Graphs	9	9	1
State-space	41	24	1
Interactive	12	15	3

Table 6. The number of transitions, vertices, and combinations between the activations and deactivations of the valves, resulted from the approaches based on graphs, state-space, and interactive.

sequences must have been detected by the expert alone, and the expert should have analysed 64 possible sequences (the number of rungs raised to the power of two), instead of 22 (a reduction of almost 3 to 1). Therefore, the interactive approach substantially reduced the expert's workload.

Some resulting transition conditions are too large to be shown in Figure 9, and they are presented in more detail in Table 5. As shown in Table 5, the transition conditions were correctly recovered from the activation and deactivation conditions of the rungs of the Ladder diagram shown in Figure 2. Additionally, with the help of the BDD, the system was able to convert $\neg(\neg tl \vee \neg as)$ into $(tl \wedge as)$, which was necessary to detect the transition from v_2 to v_3; the symbols $\neg$, $\wedge$ and $\vee$ respectively represent the logical operators **not**, **and** and **or** and the logical **not** has the highest precedence compared to the other operators; for a conversion between more complex logic expressions, see [Giacomin and Schneebeli, 2010].

As can be observed in Figures 2 and 9, the Ladder and Grafcet diagrams produced the same output command sequences for the same input events and initial conditions. Besides, some videos were also recorded showing the Ladder diagram simulation for the sequences shown in Tables 2-4, see https://github.com/pagiacomin/ladder-grafcet for details.

Although the Ladder and Grafcet diagrams shown in Figures 2 and 9 are equivalent, they have different advantages. On one hand, though the Ladder diagram shown in Figure 2 can be considered more flexible, since it can handle some input sequences not handled by the Grafcet diagram, these sequences may sometimes be considered undesirable. This is what happens when the flow in v_4 is greater than the flow in v_2 and the Ladder diagram turns on the valves v_1 and v_2 concurrently, which make the solution achieve the acidity more slowly. On the other hand, the Grafcet diagram shown in Figure 9 explicitly represents the plant's process control flow, which is not directly presented in the Ladder diagram, thereby making the plant's process control flow easier to understand. Hence, the Ladder and Grafcet diagrams shown in Figures 2 and 9 can be considered complementary.

6 Conclusions

An interactive algorithm for translating Ladder diagrams into Grafcet diagrams was proposed, and a robotic neutralization system was considered as a realistic case study. Basically, the Ladder diagram was programmed for controlling the robotic neutralization system, and Algorithm 1 converted it into the equivalent Grafcet diagram. For accomplishing this purpose, several subroutines were also implemented.

Some conclusions were reached from this study. Firstly, the interactive approach is capable of helping the expert to detect and confirm a significant number of condition actions or sequences without including any additional information before the conversion process. This didactic purpose can be valuable when only the Ladder diagram is available.

Secondly, though in [Falcione and Krogh, 1993] and [Zanma *et al.*, 1999] a Ladder diagram's rung is related to at most one step in the equivalent Grafcet diagram (in the study of Zanma *et al.* [1999] steps with the same name do not represent the same rung because the activation conditions are different), *one concludes that if this rule is broken, a more flexible Grafcet diagram can be obtained without compromising its equivalence to the original Ladder diagram.*

Thirdly, though Bryant [1986] describes the time complexities for the BDD operations, a formal proof is not given. In the proposed study, the Algorithm 1's time complexity was mathematically determined, and its analysis showed that the use of BDD can soften the exponential growth in running time, which helps to extend the scope to where the proposed approach can be applied.

Fourthly, in contrast to the state-space approach, the number of vertices and transitions grows more slowly when the proposed approach is considered, and though they are slightly greater than those obtained when the approach based on graphs is employed, this fact contributes to a more flexible Grafcet diagram.

Fifthly, on one hand, though the Ladder diagram can sometimes be considered more flexible, since it was able to handle some sequences not handled by the Grafcet diagram, these sequences may eventually be considered undesirable. On the other hand, the Grafcet diagram was capable of representing the plant's process control flow, which was not explicitly shown in the Ladder diagram. Since this information can help the expert to understand the control logic, here the

two diagrams can be considered complementary. These facts are in accordance with the standard IEC 61131-3, where integration among programming languages is an objective, and they have a didactic purpose, which helps to consolidate the value of the reverse engineering tool. This advantage was not provided by the approaches based on graphs or state-space.

Sixthly, though Falcione and Krogh [1993] have presented the same robotic neutralization system, the solution acidity is not preserved while the solution is delivered to the next tank, unlike what happens here, which is usually desirable for the sake of quality.

6.1 Difficulties

The first version of the conversion algorithm was programmed using JDK 6, which has some libraries that are not compatible with several Java IDEs nowadays. In order to program the current version, it was necessary to test several library versions until some of them made the compilation possible. Fortunately, the software Maven was used (https://maven.apache.org/), which is free, easy to configure and able to compile the source code by using several JDK versions, including JDK 6.

6.2 Future Works

The proposed problem is challenging, and some points still deserve more research. Firstly, it would be interesting to implement the persistence during the interactive conversion. In this way, the expert would not have to restart the process from the beginning, which could help to save work. Secondly, though the proposed case study is realistic, more research is necessary, considering new case studies, to investigate the algorithm's capability of converting larger Ladder diagrams into Grafcet diagrams, possibly by using a new graphical interface. Thirdly, though the diagram's symbols considered here are quite common in practice, other symbols could be considered.

Declarations

Authors' Contributions

The present author is the sole contributor, who performed the tasks of conceptualization, data curation, formal analysis, investigation, methodology, project administration, resources, software, validation, visualization, writing - original draft, and writing - review and editing.

Competing interests

The author declares that he has no competing interests.

Availability of data and materials

As soon as this paper is published, some auxiliary materials will also be available at https://github.com/pagiacomin/ladder-grafcet, such as the XML file representing the considered Ladder diagram, the DTD file defining the syntax of the Ladder diagram, the DOT files representing the BDD and the layout of the resulting Grafcet diagram, some videos showing the interactive process, simulations, and a web page describing the tests performed.

Further relevant information

Despite the fact that the related literature is scarce, this study acknowledges the importance of inclusivity and diversity in academic citations. **Citation Diversity Statement**: An effort was made to include references from a range of researchers with different backgrounds, institutions, and geographic locations, reflecting contributions from diverse perspectives in the field. By including this statement, one aims at raising awareness of citation bias and encouraging equitable referencing practices in future research on computing.

References

Abrishambaf, R., da Rocha, H., Gomes, D. E., and Espírito-Santo, A. (2023). Enhancing iec 61499 with an ieee 1451 tim function block. In *IECON 2023-49th Annual Conference of the IEEE Industrial Electronics Society*, pages 1–6. IEEE. DOI: https://doi.org/10.1109/IECON51785.2023.10312113.

Alsamhan, A. M., Saleem, W., Khan, R., Salah, B., and Soliman, A. T. A. (2023). 3d simulation of a yogurt filling machine using grafcet studio and factory io: realization of industry 4.0. *Transactions of FAMENA*, 47(3):15–30. DOI: https://doi.org/10.21278/TOF.473049922.

Barbosa, M. d. B., Barbosa, C., and Barbosa, A. (2016). Proposing a practical approach to extract and read xml meta-data from sprite sheets using blob detection algorithm. *Journal on Interactive Systems*, 7(1):17–27. DOI: https://doi.org/10.5753/jis.2016.666.

Bojnurdi, V. E., Lyu, T., Atmojo, U. D., and Vyatkin, V. (2024). Optimal function block distribution for iec 61499 distributed control application. In *2024 IEEE 3rd Industrial Electronics Society Annual On-Line Conference (ONCON)*, pages 1–6. IEEE. DOI: https://doi.org/10.1109/ONCON62778.2024.10931597.

Bryant, R. E. (1986). Graph-based algorithms for boolean function manipulation. *IEEE Transactions on Computers.*, 35(8):677–691. DOI: https://doi.org/10.1109/TC.1986.1676819.

Burgos, A., Alvarez, M. L., Iriondo, N., and Sarachaga, I. (2020). Metodología para la transformación de diseños en grafcet a código iec 61131-3 [methodology for transforming grafcet designs into iec 61131-3 code]. *Información tecnológica*, 31(6):133–146. DOI: http://dx.doi.org/10.4067/S0718-07642020000600133.

Chen, Y.-R., Hsu, C.-H., Li, T.-F., Lin, C.-Y., Weng, S.-C., and Tsai, M.-Y. (2026). Automatic model transformation and formal verification for function block of iec 61499. *Software and Systems Modeling*, 25:615–633. DOI: https://doi.org/10.1007/s10270-025-01316-y.

Cormen, T. H., Leiserson, C. E., Rivest, R. L., and Stein, C. (2009). *Introduction to Algorithms*. MIT Press, Cambridge.

David, R. (1995). Grafcet: A powerful tool for specification of logic controllers. *IEEE Transactions on Control Systems Technology*, 3(3):253–268. DOI: https://doi.org/10.1109/87.406973.

Dhanabalan, G. and Selvi, S. T. (2023). Software design of veriloghdl code generation for ladder diagram and data acquisition using labview. *Wireless Personal Communications*, 128:1087–1115. DOI: https://doi.org/10.1007/s11277-022-09990-7.

Falcione, A. and Krogh, B. H. (1993). Design recovery for relay ladder logic. *IEEE Control Systems Magazine*, 13(2):90–98. DOI: https://doi.org/10.1109/37.206990.

Galeano González, D. A. and Botero Castro, H. A. (2007). Representación mediante grafcet del accionamiento de generadores sincrónicos en una central hidroeléctrica modernizada [grafcet representation of the drive of synchronous generators in a modernized hydroelectric power plant]. *Dyna*, 74(153):325–332.

Giacomin, P. A. S. and Schneebeli, H. A. (2010). Um método interativo de recuperação de diagramas grafcet a partir de diagramas ladder [an interactive method for retrieving grafcet diagrams from ladder diagrams. In *XVIII Congresso Brasileiro de Automática [XVIII Brazilian Congress of Automation]*, pages 1749–1754. SBA.

Godase, V. (2025). Comparative study of ladder logic and structured text programming for plc. *Journal of Electronics Design and Technology*, 2(2):34–44. `https://matjournals.net/engineering/index.php/JEDT/article/view/2117`, Accessed on 03 May 2026.

Harrington, B., Gould, T., and Hurst, N. a. (2003). Inkscape. Available at https://inkscape.org/. Accessed in 03 May 2026.

Hrbček, J., Ždánsky, J., Macko, D., and Bubeníková, E. (2026). Development of a digital twin controlled by plc in a simulation environment. *Transportation Research Procedia*, 93:723–728. DOI: https://doi.org/10.1016/j.trpro.2025.11.109.

Hu, X., Liang, Y., Zhu, S., Li, H., and Zhu, S. (2024). St-petri: A visual executable semantic model for plc structured text language. In *2024 IEEE 22nd International Conference on Industrial Informatics (INDIN)*, pages 1–6. IEEE. DOI: https://doi.org/10.1109/INDIN58382.2024.10774532.

IEC (2013). *GRAFCET specification language for sequential function charts*. International Electrotechnical Commission, Rue de Varembé 3, PO Box 131, CH-1211 Geneva 20, Switzerland, 3.0 edition. https://webstore.iec.ch/en/publication/3684. Accessed on 03 May 2026.

Liu, Y., Zhang, H., and Liu, Z. (2025). Research on performance evaluation and optimization algorithm of plc control system based on graph neural network. In *4th International Conference on Artificial Intelligence and Autonomous Robot Systems (AIARS 2025)*, volume 2025, pages 53–57. IET. DOI: https://doi.org/10.1049/icp.2025.2927.

Lopes, V. and Sousa, M. d. (2017). Algorithm and tool for ld to sfc conversion with state-space method. In *2017 IEEE 15th International Conference on Industrial Informatics (INDIN)*, pages 565–570. IEEE. DOI: https://doi.org/10.1109/INDIN.2017.8104834.

Mroß, R., Schnakenbeck, A., Völker, M., Fay, A., and Kowalewski, S. (2023). Unambiguous interpretation of iec 60848 grafcet based on a literature review. In *2023 IEEE 28th International Conference on Emerging Technologies and Factory Automation (ETFA)*, page 1–8. IEEE. DOI: https://doi.org/10.1109/ETFA54631.2023.10275504.

Palaniappan, R., Nataraj, S. K., Noaman, N. M., and Ismail, Z. (2023). Grafcet based modelling of processing operation in modular production system. In *2023 IEEE 8th International Conference on Engineering Technologies and Applied Sciences (ICETAS)*, pages 1–5. IEEE. DOI: https://doi.org/10.1109/ICETAS59148.2023.10346373.

Pressman, R. S. (2019). *Software Engineering: A Practitioner's Approach*. McGraw-Hill, 9th edition edition.

Roisin, M., Renard, D., Annebicque, D., Riera, B., and Yvars, P. (2025). From algebraic synthesis and GRAFCET to logical controller design in ST code (IEC 61131-3). In Gini, G., Precup, R., and Filev, D. P., editors, *Proceedings of the 22nd International Conference on Informatics in Control, Automation and Robotics, ICINCO 2025, Marbella, Spain, October 20-22, 2025, Volume 1*, pages 494–501. SCITEPRESS. DOI: https://doi.org/10.5220/0013817800003982.

Schnakenbeck, A., Mroß, R., Völker, M., Kowalewski, S., and Fay, A. (2023). A control flow based static analysis of grafcet using abstract interpretation. In *2023 IEEE 21st International Conference on Industrial Informatics (INDIN)*, pages 1–7. IEEE. DOI: https://doi.org/10.1109/INDIN51400.2023.10218176.

Tavares, A. R., Zuin, G. L., Azpúrua, H., and Chaimowicz, L. (2017). Combining genetic algorithm and swarm intelligence for task allocation in a real time strategy game. *Journal on Interactive Systems*, 8(1):4–19. DOI: https://doi.org/10.5753/jis.2017.671.

Venkatesh, K., Zhou, M., and Caudill, R. J. (1994). Comparing ladder logic diagrams and petri nets for sequence controller design through a discrete manufacturing system. *IEEE Transactions on Industrial Electronics*, 41(6):611–619. DOI: https://doi.org/10.1109/41.334578.

Villela, H. F., Corrêa, F., Ribeiro, J. S. d. A. N., Rabelo, A., and Carvalho, D. B. F. (2023). Fake news detection: a systematic literature review of machine learning algorithms and datasets. *Journal on Interactive Systems*, 14(1):47–58. DOI: https://doi.org/10.5753/jis.2023.3020.

Zanma, T., Suzuki, T., Inaba, A., and Okuma, S. (1999). Transformation algorithm from ladder diagram to sfc using temporal logic. *Electrical Engineering in Japan*, 129(1):74–81. DOI: `https://doi.org/10.1002/(SICI)1520-6416(199910)129:1<74::AID-EEJ9>3.0.CO;2-B`.