International Journal of Combinatorial Optimization Problems and Informatics, 14(3), Sep-Dec 2023, 157-175. ISSN: 2007-1558.
https://doi.org/10.61467/2007.1558.2023.v14i3.399

www.editada.org

Manuscripts Design Algorithm for Sequential Pneumatic and Electropneumatic Systems

Mario Oscar Ordaz Oliver [1], *Evelin Gutiérrez-Moreno* [1], *Jesús Patricio Ordaz Oliver* [2], *Javier Hernández Pérez* [1], *Omar-Jacobo Santos-Sánchez* [2]
[1]Mechatronic Engineering, Polytechnic University of Pachuca, Pachuca, Hidalgo, México
[2]Research Center on Technology of Information and Systems, Autonomous University of Hidalgo State, Pachuca, Hidalgo, México

E-mails: mario.ordaz@upp.edu.mx, evgutierrez@upp.edu.mx, jesus_ordaz@uaeh.edu.mx, jahdez@upp.edu.mx, omarj@uaeh.edu.mx

Abstract. This article presents an alternative automation algorithm for pneumatic sequences that enables the solution of complex sequential problems without the need to organize detection phases into groups. By not using groups to organize movements in detection phases, two benefits can be achieved: firstly, all the drawbacks associated with using groups are eliminated, such as design complexity, difficulty in debugging, increased cost, and difficulty in optimizing the circuit; secondly, it enables the use of logical functions to define each motion in pneumatic actuators. This opens up the possibility to solve logical functions using Boolean algebra techniques, and even their generalization into an algorithm that can be programmed in any language. To validate the proposed algorithm, MATLAB pseudocode is presented. This pseudocode allows obtaining the logical functions needed to build the ladder diagram of any sequence involving up to four double-acting cylinders with any number of phases in a work cycle.

Keywords: pneumatics, automation, control, processes, sequences, complexity.

Article Info
Received Sep 26, 2023
Accepted Dec 11, 2023

1 Introduction

In the industry, the use of pneumatic and hydraulic systems has represented an efficient solution for automation problems and process control that require moving loads and performing sequential operations over the past seven decades (Groover, 2016). Pneumatics has become a key technology in industrial automation and is used in a wide range of applications, such as manufacturing, heavy machinery, construction, mining, food industry, energy production, and many other areas. Pneumatic technology has evolved over the years, and today advanced and sophisticated pneumatic systems are used that allow for greater automation and efficiency in industrial processes (Parr, 2011). Pneumatic sequences are commonly used in industrial automation to drive the movement of machines or systems. For instance, in a production line, pneumatic sequences can control the movement of a machine's components to perform a specific task. They are also used in process automation systems, such as controlling a flow valve or opening and closing a safety gate (Bartelt, 2012). In general, pneumatic sequences offer a reliable and precise way to control motion in automated systems and are widely used in a variety of industries, from manufacturing to the food and pharmaceutical industries (Ilango & Soundararajan, 2011).

The solution of a pneumatic sequence can be achieved by different methods, which generally depend on the number of detection phases or the complexity of the sequence (Ponomareva, 2006). Among the classic methods commonly implemented in practice, the intuitive method is a traditionally-used approach that relies on the specialist's experience. Its main characteristic is that it is a heuristic method, which is a disadvantage in complex circumstances. There are also systematic methods that can represent a general solution to a large number of problems in pneumatic sequences. One such method is the step-by-step method, which consists of a control circuit for the pneumatic sequence and can be performed in various ways, all of which have in common the way in which the distributor and signal valves (both 3/2) are controlled. In some cases, when the number of detection phases in the sequence is considerable or the complexity is high, valves with simultaneity logic (AND) are used to prevent backflow of air in the paths of end-of-stroke valves or signal valves and avoid pressure loss in the air (Henao Castañeda & Monroy Jaramillo, 2012).

For the systematic step-by-step method, four fundamental steps are followed: grouping, identifying conditions or states (end-of-stroke valves) in each phase and group, identifying valves and components to be used in the implementation, and designing and simulating the pneumatic circuit. This method is applied to sequences with a minimum of 3 groups, characterized by using 3/2 directional control valves for switching and group change. The cascade method, on the other hand, involves four fundamental steps: grouping, identifying the conditions or states (such as end-of-stroke valves) in each phase and group, selecting the appropriate valves and components, and designing-simulating the pneumatic circuit. There are three variants of the cascade method, known as: the traditional method, in which the use of logical valves predominates and is used for sequences that generate a maximum of five groups; the simplified method, which is used for sequences that generate a maximum of three groups; and the modern method, which eliminates most of the problems of the previous variants and is used for sequences that generate a maximum of five groups, characterized by using 5/2 directional control valves for switching and group change (Guenther, Perondi, DePieri & Valdiero, 2006).

The choice of method for solving pneumatic sequential problems is generally related to the complexity and number of detection phases in the sequence. However, regardless of the method implemented to solve the problem, the use of groups can present some disadvantages in both the step-by-step and cascade methods (Cundiff, Kocher, 2019; Barber, 1997):

- Design complexity: The use of groups can increase the complexity of the pneumatic circuit design, as more components and pipes are needed to connect the different groups.
- Difficulty in debugging: When using groups, it may be more difficult to debug the pneumatic circuit, as a failure in one group can affect other groups.
- Cost increase: The use of groups can increase the cost of the pneumatic circuit, as more components and pipes are needed to connect the different groups.
- Higher energy consumption: The use of groups can increase energy consumption, as more components and pipes are needed to connect the different groups.
- Difficulty in optimizing the circuit: The use of groups can make it difficult to optimize the pneumatic circuit, as the effects of the different groups on the overall performance of the circuit must be considered.

The main solution to most of the disadvantages derived from traditional systematic methods lies in the implementation of such methodologies through electro-pneumatics (Smith & Corripio, 2005). Nowadays, electro-pneumatics has displaced pneumatics, mainly because electro-pneumatics is a technology that combines pneumatics with electronics to achieve more precise and sophisticated control in automated processes (Gea & Lladonosa, 1998). This is because electronics allows the integration of sensors, actuators, and controllers to monitor and adjust processes more accurately, while pneumatics provides the force and speed necessary to perform the required actions in processes. Moreover, electro-pneumatics offers additional advantages in terms of energy efficiency and cost reduction, as electro-pneumatic systems can be programmed to use only the required amount of compressed air and electrical energy, reducing energy consumption and overall system operating costs. In other words, electro-pneumatics is more precise, efficient, and cost-effective than pure pneumatics in process automation, making it a preferred option in many industrial applications (Swider, Wszotek & Carvalho, 2005).

Recent methodologies propose novel algorithms and sophisticated procedures based on the use of Boolean algebra, Karnaugh maps, flip-flops or memory states, Petri nets, genetic algorithms and evolutionary algorithms to solve pneumatic sequential problems, as presented in the works developed by (Bayoumi, 2007, May; Santos & Silva, 2017; Venkatesh, Zhou & Caudill, 1994; Sajaysurya & Kumar, 2011; Ganesh & Gurunathan, 2017). For instance, the use of Petri nets, which is a contribution from the programming and computer systems field, proposes the use of Petri nets to control discrete events in automated industrial systems, such as pneumatic and hydraulic sequences of complex processes. Some of the main results compare the effectiveness of their implementation with that of ladder diagrams (Venkatesh, Zhou & Caudill, 1994). On the other hand, Petri nets have been used in industrial automation to calculate a plant model. This model allows applying an algorithm that determines the set of equations for the ladder diagram, thus ensuring the tracking of the trajectory of an electro-pneumatic sequence (Vázquez, Gómez-Castellanos & Ramírez-Treviño, 2018). However, most of these methodologies lead to a similar pneumatic circuit, which is typically defined by the cascade method, where 5/2 directional control valves are used to control the group change or require the application of extremely complex algorithms, such as those proposed that propose the use of Petri nets. In this work, an alternative methodology is proposed that provides an effective solution to complex pneumatic sequences, in which the idea of grouping the detection phases and therefore the use of 3/2 or 5/2 directional control valves for group changes is employed, avoiding step-by-step and cascade methods, respectively. This eliminates the problems caused by the existence of the aforementioned groups. The main contributions of this work can be summarized as follows:

- An effective solution for complex sequential pneumatic circuits: Conventional methods like step-by-step and cascade for solving sequential pneumatic circuits have limitations in terms of efficiency and accuracy in more complex circuits. The solution presented in this document aims to overcome these limitations, as it eliminates the issues caused by the 5/2 valves in these two methods and is generalized into an algorithm that can be programmed in any structured text or high-level language to obtain the Boolean expressions associated with the control stage.
- Flexible implementation in pneumatic and electro-pneumatic circuits: One of the advantages of the proposed method is its flexibility in implementation, whether in pneumatic or electro-pneumatic circuits. The method proposed in this document allows for its application in various significant scenarios, such as Automation education, microfluidic hardware design, and other Automation applications, where process efficiency might be compromised by circuit complexity.
- Practical applications in industrial automation: The proposed method can be used in different applications, particularly in industrial automation, where improving efficiency and accuracy in the resolution of sequential pneumatic circuits is desired. Moreover, by enhancing the efficiency and accuracy of the process, it contributes to reducing the time and costs associated with resolving complex circuits.

The document is organized as follows: Section 1 provides an introduction to the topic addressed in this work and describes previously published related literature. In Section 2, we present some characteristics of traditional methods for solving sequential circuits, which establish their main disadvantages in implementation, and from which we define the set of steps to apply the proposed method in the Section 3, as a case study, we apply the proposed method to a complex sequence consisting of 16 phases and develop the method to determine a solution for both pneumatic and electropneumatic problems. Section 4 establishes the conditions that allow transferring the algorithm resulting from the proposal in this document to a programming language such as "M" code. Finally, Section 5 provides the conclusions.

It's important to mention that the pneumatic and electro-pneumatic circuit diagrams for the sequences presented in this document were created using **FluidSIM 4.5** from **FESTO**. This computer-aided engineering and design software utilizes the standard symbology associated with the international standards **ISO 1219-1** and **ISO 1219-2**. These standards define the symbols to be used in diagrams for representing valves and elements of pneumatic and hydraulic circuits, as well as the international standard **IEC 61131** for Programmable Logic Controllers (PLCs). Additionally, this software enables virtual validation of programs loaded onto PLCs from various manufacturers.

2 Minimum-Essential-Parts (MEP) Algorithm

In any pneumatic sequence, it is possible to encounter repeated detection phases during a working cycle. This means that the instantaneous states of the 3/2 roller-operated valves that detect the states of the actuator stems may coincide more than once during a cycle. Using heuristic methods to solve this type of sequence can be complicated since the control logic must make the correct decision each time the instantaneous states occur with multiplicity in space-time and execute the corresponding action.

Several solutions and methodologies have been proposed for pneumatic sequences, including those put forward by [6,13,14]. These methods address the problem by either applying the principles of RS and JK flip-flops pneumatically or by creating memorizable variables to differentiate between multiple states and to determine the appropriate logical conditions for decision-making in the outputs with the aid of Boolean algebra. However, implementing these methods often results in a circuit whose characteristics are similar to those of the cascade method. This means that the circuit will have as many switchable conduits for compressed air as there are states with multiplicity in the sequence, minus one.

Using this cascading configuration typically results in increased complexity in circuit design, difficulty in debugging errors, higher implementation costs, increased energy consumption, and difficulty in optimizing the circuit. For instance, consider the sequence A+B+B-A- in an automatic process similar to the one depicted in Figure 1:

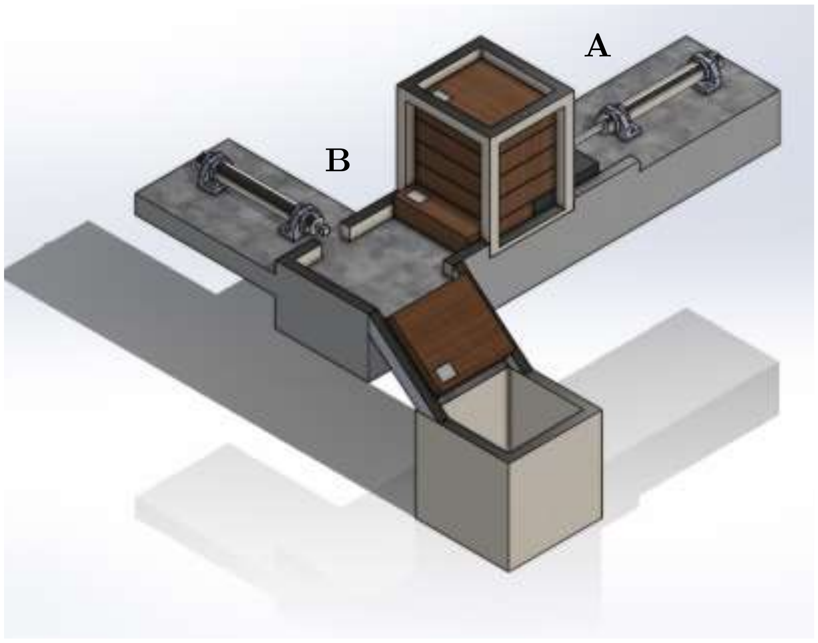

Figure 1. Movement sequence for packing (Ebel, 2013).

Careful observation of the displacement step diagram of this sequence depicted in Figure 2 reveals the presence of multiplicity in two different detection phases. The first phase occurs when executing action, *A+*, while the second phase occurs when executing *B-*, resulting in the simultaneous occurrence of conditions A_1 and B_0 at the end of the first and third phases, respectively.

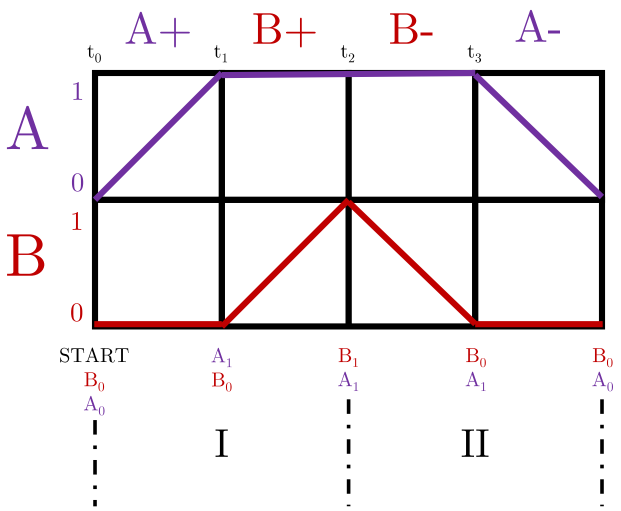

Figure 2. Displacement-step diagram for A+B+B-A- sequence and groups.

As a result, the cascade method and its variants require the creation of groups that incorporate detection phases to handle sequences with multiplicity. This leads to the rule of the cascade method, which states that within a group, two or more actions of the same double-acting pneumatic actuator should not occur (Santos and Silva, 2017), in their work define groups I and II as variables X and Not X respectively, with the purpose of establishing different Boolean functions from states with multiplicity $A_1 \cdot B_0$ (End-of-stroke sensors). By simplifying the corresponding expressions through Karnaugh maps. It is possible to discriminate variable B_0 in the second phase and A_1 in the third phase. This distinction between variables is evident since the Boolean expressions include only those variables that represent a change at the end of each executed action. To switch between groups, I and II in each cycle, a 5/2 directional control valve is used. The resulting pneumatic circuit is depicted in Figure 3.

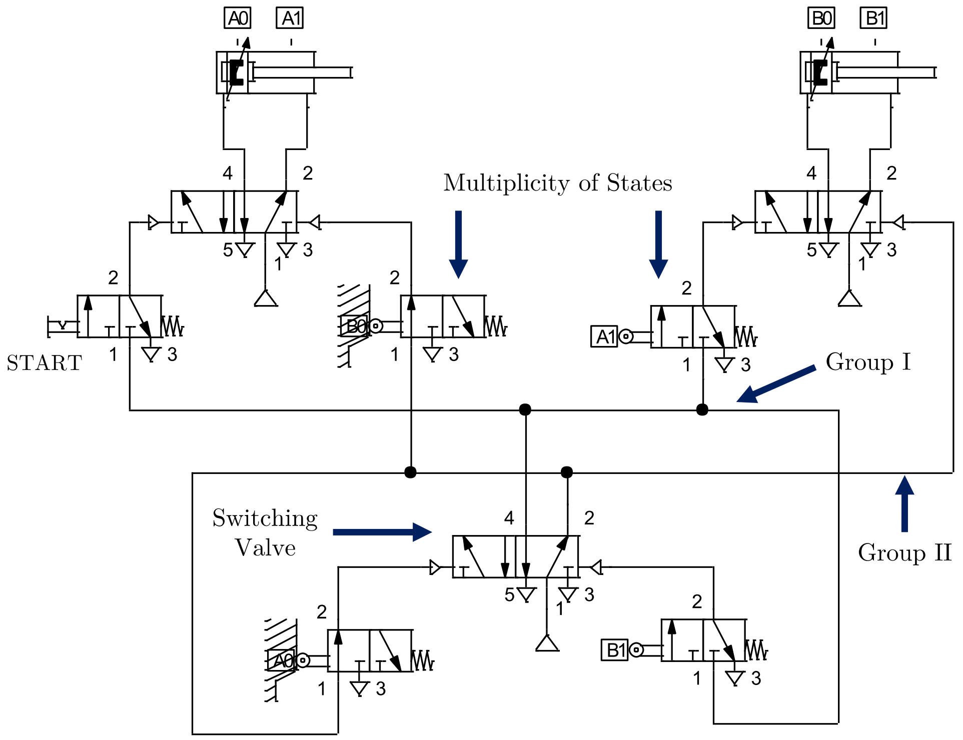

Figure 3. Pneumatic approach or the *A+B+B-A-* sequence with the cascade method.

In the cascade method, it is generally assumed that the conditions that trigger a group change are those that execute actions that create states with multiplicity in the working cycle. In the example shown in Figure 1, A_0 changes from group II to group I, while B_1 changes from group I to group II. The primary characteristics of a pneumatic sequence with multiplicity in its states that require the establishment of groups (memories) in the cascade method can be summarized in the displacement-step diagram presented in Figure 4.

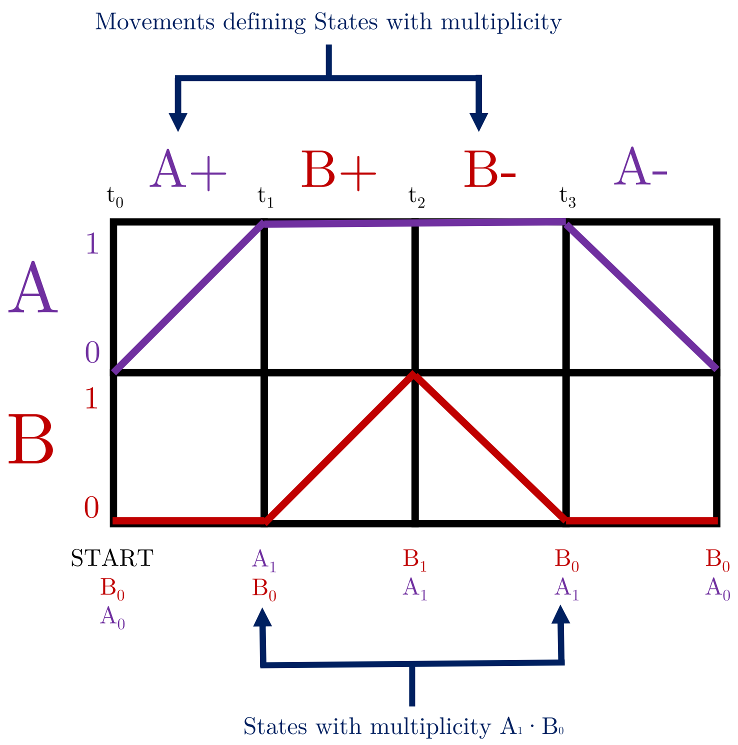

Figure 4. Characteristics of displacement-step diagram for A + B + B - A- sequence.

Given the analysis of this particular case, it is possible to establish a set of generalizations and consequently formulate the procedure of the MEP algorithm for the design of pneumatic sequential circuits as follows:

1. **Draw the displacement-step diagram**: when drawing the displacement step diagram, the set of conditions that are met for the immediate action execution must be indicated at each time. It is important to start with the end-of-stroke sensors that changed state just before the time being analyzed.
2. **Minimization or optimization of algebraic functions for each action**: The set of conditions that are met prior to the execution of the immediate action at each time defines the product of the end-of-stroke sensor signals to produce the activation signals of the double-acting actuator strokes. These products (minterms) represent logical states and can therefore be simplified using Boolean algebra techniques, such as the use of rules, Karnaugh maps, Quine McCluskey algorithm, among others.
3. **Creation of memories for states with multiplicity**: from the set of conditions identified in the first step, it is necessary to recognize those that present multiplicity throughout the sequence, i.e., those sets of signals in the end-of-stroke sensors that occur simultaneously more than once during a work cycle. Once these conditions are recognized, they must be numbered from left to right as they appear in the displacement step diagram as M_1, M_2, M_3, ..., M_n. Where n indicates the total number of repeated states. The M_i memories in the circuit are represented by means of 3/2 distributor valves with SET and RESET states (for pneumatic case). The SET (high) state of each memory is defined by the set of conditions from the previous time, while the RESET (low) state is defined by the set of conditions from the time of each state with multiplicity in the detection phases.
4. **Add the memory variables in the algebraic functions**: the high state of each memory is included as a product (either by means of simultaneity valves or with series connections) in each algebraic function from step 2, where states with multiplicity are considered.

After completing this process, the design of the pneumatic sequential circuit is transferred to CAD software for simulation, which provides an idea of the components, connections, and operation that it will present when implemented. Typically, in pneumatic and hydraulic circuits, simulation provides a reliable approximation of the results that will be obtained at a practical level.

When applying the MEP algorithm to the pneumatic sequence defined in the process shown in Figure 1, the following is obtained:

step 1: the displacement-step diagram is drawn as shown in Figure 4, indicating below each time the end-of-stroke sensors that are active.

step 2: The following logical functions for the rod movements of each cylinder are obtained directly as the product of the end-of-stroke sensors marked at the beginning of each phase, as shown in Figure 4:

At time t_0, $A+=START \cdot B_0$.
At time t_1, $B+=A_1 \cdot B_0$.
At time t_2, $B-=B_1$.
At time t_3, $A-=A_1 \cdot B_0$.

Here, the multiplicity of states at times t_1 and t_3 is observed.

step 3: the memories are set to a HIGH state just before the multiplicity, i.e., at t_0 and t_2, and return to their LOW state when the multiplicity occurs, i.e., at t_1 and t_3.

At time t_0, $M_1=START \cdot B_0$.
At time t_1, $M_1'=A_1 \cdot B_0$.
At time t_2, $M_2=B_1$.
At time t_3, $M_2'=A_1 B_0$.

step 4: the memories are added to the logical functions obtained in step 2 as follows:

At time t_0, $A+=START \cdot B_0$, $M_1=START \cdot B_0$.
At time t_1, $B+=A_1 \cdot M_1$, $M_1'=A_1 \cdot M_1$.
At time t_2, $B-=B_1$, $M_2=B_1$.
At time t_3, $A-=B_0 \cdot M_2$, $M_2'=B_0 \cdot M_2$.

Finally, the design is transferred to CAD software to validate its operation and later be implemented. The result of the pneumatic sequential circuit design is shown in Figure 5.

In the different logical functions, both for actions and memories, the use of AND logical operators are observed, which in pneumatic circuits are equivalent to simultaneous valves. The use of these valves can be omitted in practice by using series connections.

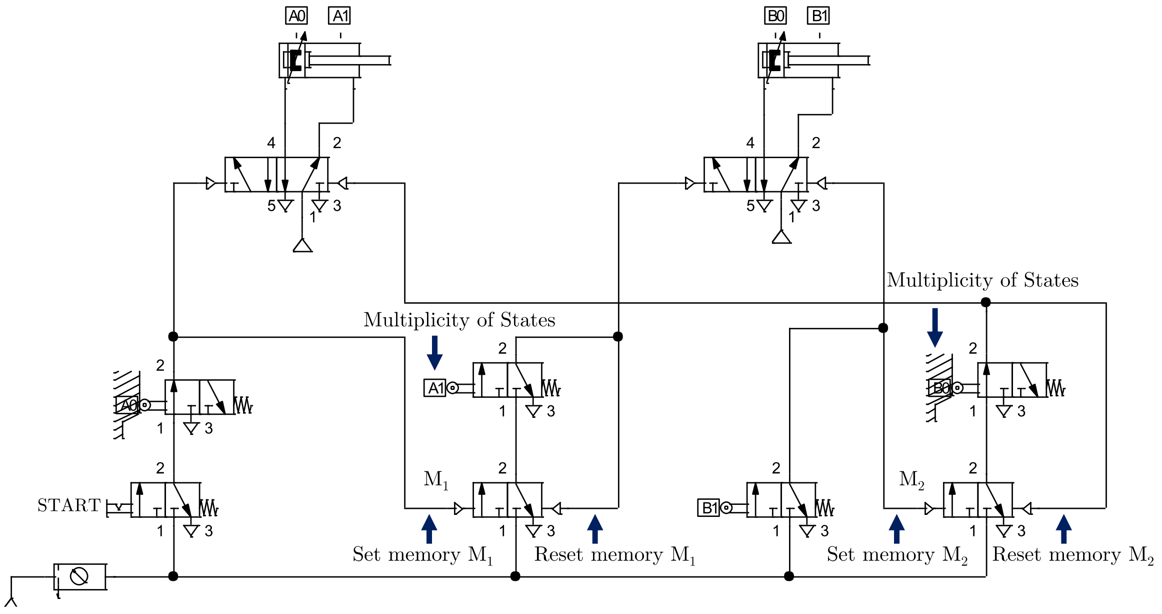

Figure 5: Pneumatic approach of the *A+B+B-A-* sequence with the MEP algorithm.

The algorithm proposed in this document can be applied to any programming language. For example, if you want to control a pneumatic sequential process electrically, programmable logic controllers (PLC's) are generally used, where the standard language is ladder diagrams. In this diagram, *A+*, *A-*, *B+*, *B-* are outputs, M_1, M_2 are flags (internal coils), the end-of-stroke sensors A_0, A_1, B_0, B_1 and the START button are inputs. The RESET of M_1 and M_2 are not defined as flags, but as connections that deactivate the SET state of these. Thus, the ladder language control circuit and the power circuit for the sequence defined by the displacement-step diagram in Figure 4, applying the MEP algorithm, result as shown in Figure 6.

The electric control circuit of the pneumatic cylinders based in PLC is obtained directly from the equations of motion of the pneumatic sequence derived from the application of the steps of the MEP algorithm.

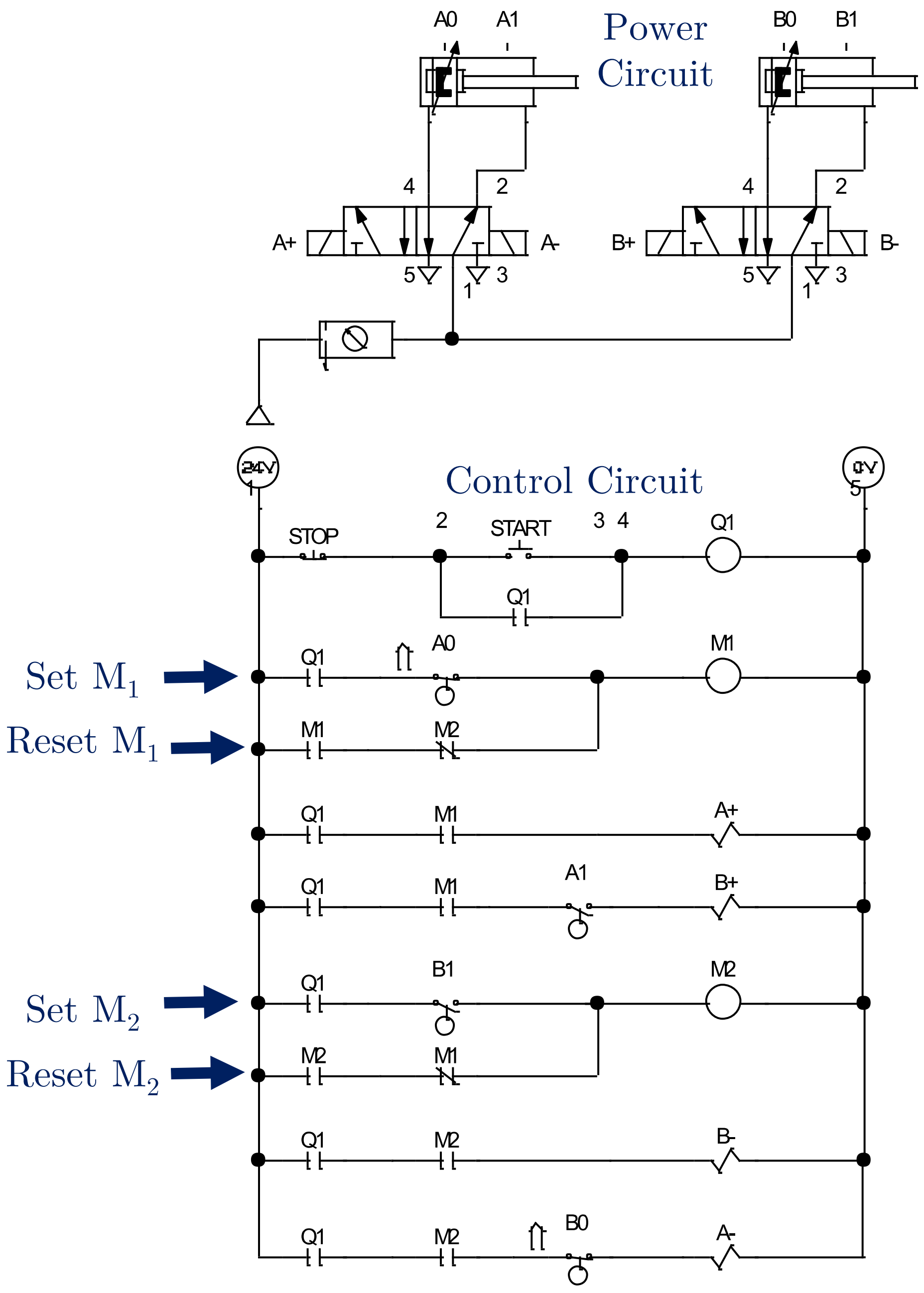

Figure 6. Electro-pneumatic approach of the *A+B+ B-A-* sequence with the MEP algorithm.

3 Case study of pneumatic sequential control circuit design using MEP algorithm

For the particular case analyzed in this section, a configuration of three double-acting cylinders is considered, as shown in Figure 7. The problem to be solved with the proposed algorithm lies in an automatic jam-packing line, where it is intended that cylinder D (not considered to solve the problem) is responsible for the power of the platform. Cylinders B and C are placed through cylinder A, and cylinder B picks up and places a sheet of paper, so that cylinder C finally picks up and places jars of jam (Santos and Silva, 2017). To automatically carry out the described process, it is necessary to ensure the sequence of movements of cylinders A, B, and C: B+B-A+B+C+B-C-A-B+C+B-C-A+B+B-A-. This sequence consists of 16 movements established over 15 times, where the complexity of the sequence guarantees the existence of multiplicity in the detection phases.

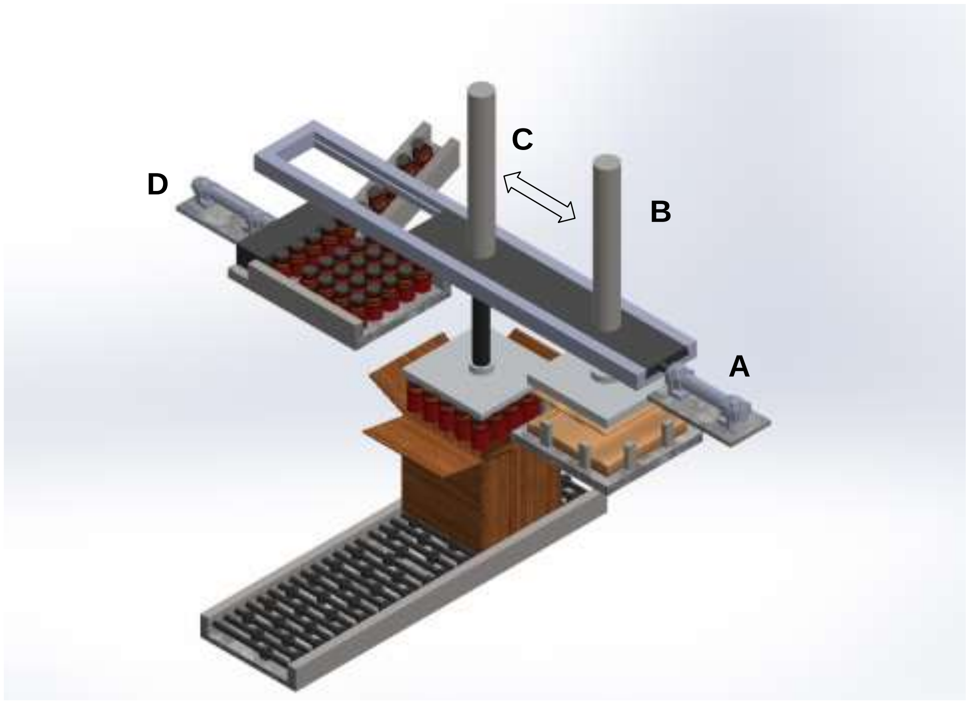

Figure 7. Jam packaging line [14,20].

To visualize this in more detail, the **first step** in the MEP algorithm is applied, and the displacement-step diagram is obtained as shown in Figure 8. The diagram clearly shows that there are 12 instances of multiplicity in the states of the end-of-stroke sensors, which require a minimum of 8 groups to be formed using the cascade method. In addition, multiple simultaneity valves must be connected to the different groups and end-of-stroke sensors to execute the sequence of movements. This case study was selected due to the significant number of multiple states present, which demonstrates the potential of the proposed algorithm.

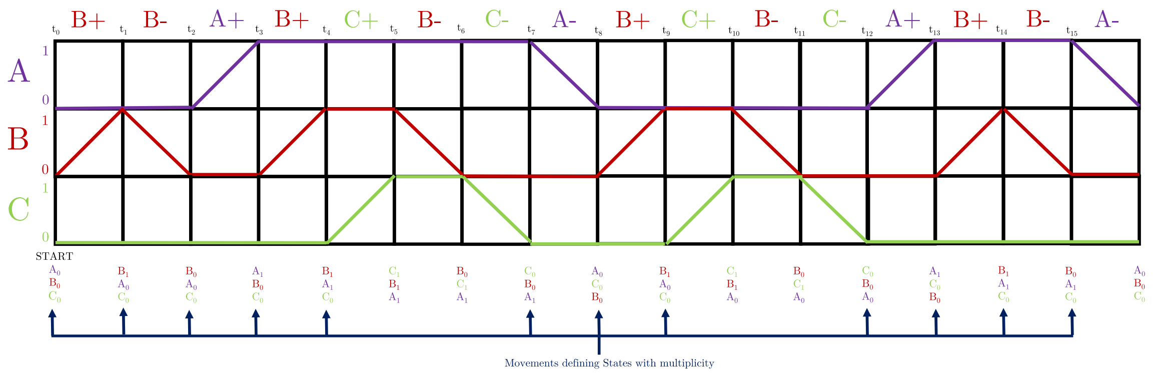
B+ B- A+ B+ C+ B- C- A- B+ C+ B- C- A+ B+ B- A-
A
B
C
START
Movements defining States with multiplicity

Figure 8. Displacement-step diagram for the jam packaging line.

The **second step** of the method consists of obtaining the following logical functions using Boolean algebra and its different factoring methods. For this case study, given the complexity of the sequence, it is possible to choose to omit this step, since the use of 3/2 memory valves allows for only the state of the end-of-stroke sensor that presented a state change at the time that generates an action to be used. For complex cases such as the one analyzed in this section, this represents a series of advantages. Initially, it reduces the time in the design, as it does not require the application of Boolean algebra techniques to obtain the logical functions of each action; secondly, it reduces the complexity of the connections as it only requires an end-of-stroke sensor connected to the corresponding 3/2 memory valve through a simultaneity valve; finally, it reduces the space and costs in the implementation process by reducing the complexity of the connections.

The use of the 3/2 memory valve configuration in conjunction with the end-of stroke sensor is not only used to execute the corresponding action for each time, but also allows for the memory state of the entire circuit to be restored with respect to what has happened in previous times. It should be noted that the high state of each memory valve is defined by the previous action, while the low state is defined by the output of the simultaneity valve of the current action. For sequences that include a considerable number of phases with multiplicity in their states and a reduced number of double-acting cylinders, it is recommended to use the simultaneity valve to make the connection between the memory state and the end of-stroke sensor. However, for sequences that include a reduced number of phases with a small number of states with multiplicity, the simultaneity valve can be omitted and the connection in series of the 3/2 memory valve with the end-of stroke sensor can be made, as shown in the scheme of Figure 5. This practice can represent both a reduction in implementation costs by reducing the number of valves used, as well as a decrease in implementation complexity, due to the reduction of connections made between the inputs and outputs of the distribution valves. Given the above, the **third step** is carried out, which consists of generating memories for states that present multiplicity in the sequence times. For the case study, the use of one memory per time is proposed, simulating a set of connections similar to the step-by-step method.

Each 3/2 memory valve has its output (via 2) connected to one of the inputs of a simultaneity valve. Its right actuation is used to set the memory to high state and its left actuation is used to set it to low state. The remaining input of the simultaneity valve is used to receive the signal coming from an end-of-stroke sensor, which is not chosen arbitrarily, but rather, it must be the one that presented a state change at the time that will trigger an action. Furthermore, the output signal of the simultaneity valve is sent to the 5/2 directional valves corresponding to the double-acting cylinders. Additionally, this signal is used to set the current memory to low state and the next memory to high state, as shown in Figure 10. The connection of each end-of-stroke sensor to the corresponding input of the simultaneity valve defines the **fourth step**, along with the connection of its 3/2 memory valve.

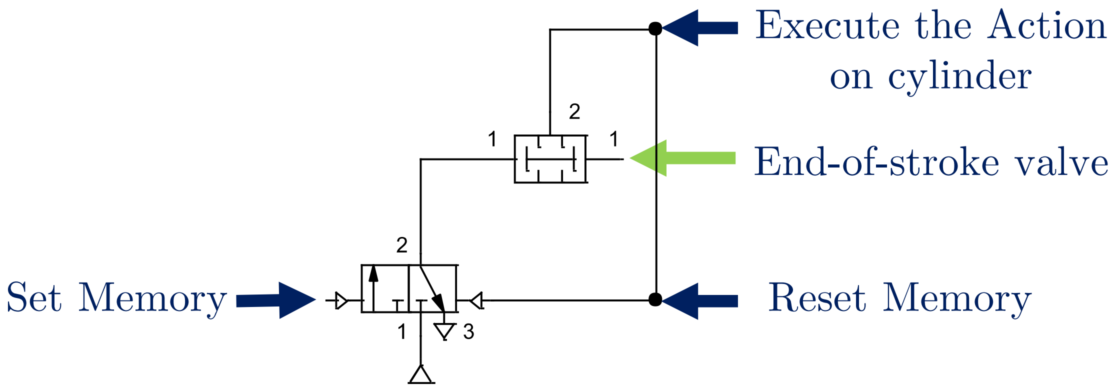

Figure 10. Connection of the 3/2 memory valve for the MEP algorithm.

Finally, the design of the pneumatic sequential circuit is transferred to CAD software for simulation. This simulation provides an idea of the components and their arrangement, as well as the connections and the operation that will be presented when implemented. This design is shown in Figure 9, where, without loss of generality, it is understood that there is only one source of compressed air that, connected to a maintenance unit, supplies compressed air to all components of the automatic system. It is important to note that this circuit follows the design rules for each double-acting cylinder, where the levels from top to bottom are actuators, control elements, logic functions, signal emitters and control signals, pressure intake, and maintenance unit. The order and position of the valves associated with each memory of the proposed method are set according to the phase of motion being executed, meaning that the connections associated with each memory should be located at the bottom of the pneumatic cylinder that will exhibit the corresponding movement in its rod.

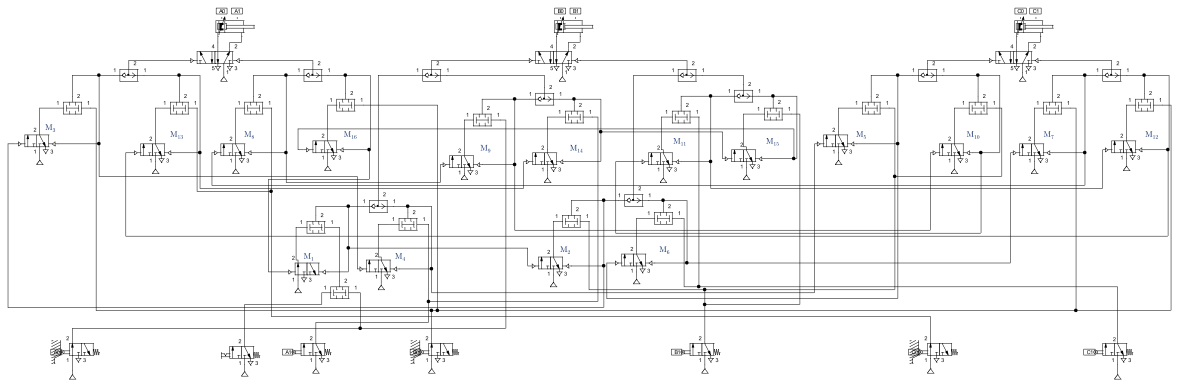

Figure 9. Pneumatic approach of the jam packaging line sequence with the MEP method.

To translate this sequence into ladder language, it is necessary to translate the connections of the 3/2 memory valve shown in Figure 10 into their equivalent electrical connections and components. The 3/2 valve is implemented as an internal coil (flag), where the contacts associated with these coils allow energizing the subsequent coils when connected in series with the corresponding end-of-stroke sensors. To ensure the connection of the coil associated with a memory during the necessary time of the action to be executed, a latching is performed by means of a contact in parallel with the connections that energize it directly to said coil. The disconnection of the memory coil is carried out by means of a normally closed contact (associated with the next memory), which is connected in series with the parallel branch that contains the direct connection and the latching contact, as shown in Figure 11.

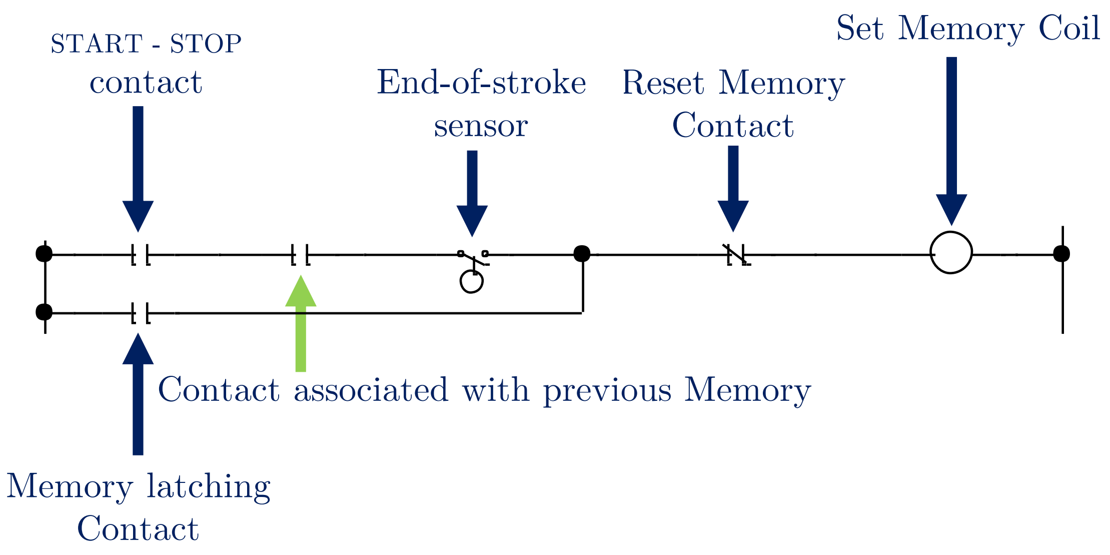

Figure 11. Memory equivalent electrical for the MEP algorithm.

Thus, the ladder language control circuit and the power circuit for the sequence defined by the displacement-step diagram in Figure 8, applying the MEP algorithm, result as shown in Figure 12. Pneumatic and electro-pneumatic circuits of this nature are commonly found in a variety of industrial processes that require precise control of repetitive sequential movements and processes. For example, such circuits are used in the manufacturing of products like fresh food packaging in the food industry. In the automotive industry, these circuits are used for controlling welding robots, painting, and assembly in production lines. Similarly, in the pharmaceutical industry, these circuits are used for automating packaging and drug production processes, while in the textile industry, they are used for automating weaving, dyeing, and finishing processes of fabrics (Deppert & Stoll, 2000).

4 Code Implementation

A This section presents a description of the generalization of the MEP algorithm applied to obtain a program in M language. This algorithm arises from the logical function of the MEP memory, which is described by Figure 11.

The main characteristic of the proposed method, which allows for its general application to pneumatic sequences and the obtaining of logical functions for programming on programmable logic controllers using ladder language, is defined by the dependence on contacts associated with past, present, and future states of memories.

This dependence offers the ability to clear the circuit's memory in each phase of the sequence, that is, it eliminates the difficulty in decision making under sequences that present multiple conditions throughout a cycle.

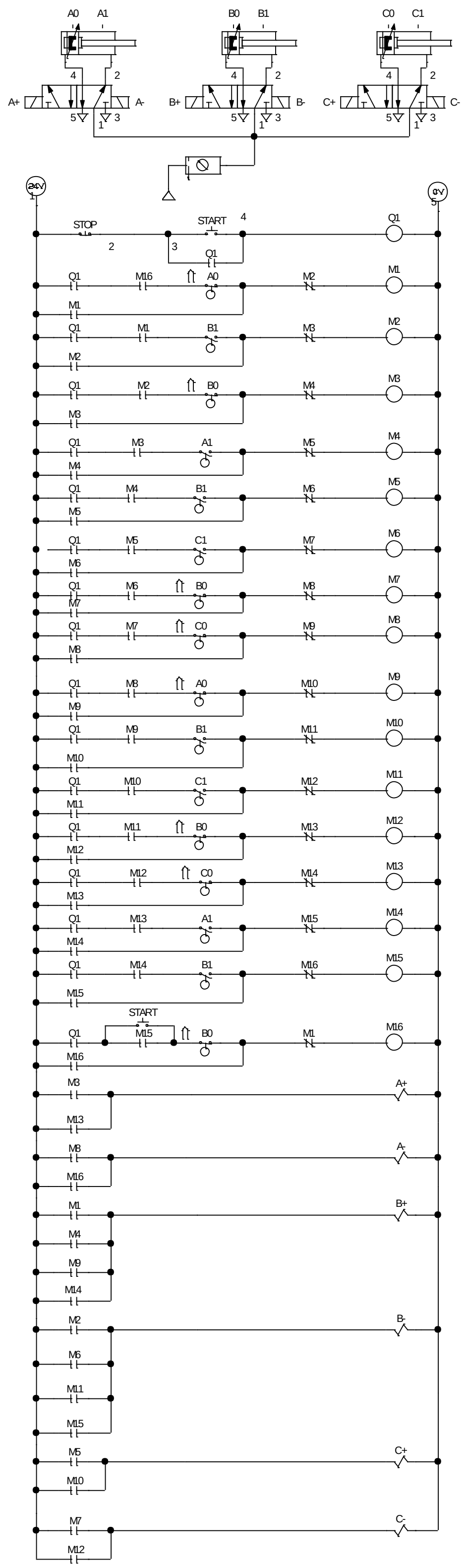

Figure 12. Electro-Pneumatic approach of the jam packaging line sequence with the MEP algorithm.

The generalization of the method includes three logical functions that are invariant to the sequence determined by the program associated with the proposed algorithm. The first one consists of the start-stop function, which is built using the traditional interlocking rule. A closed contact associated with the stop button is connected in series with a parallel connection composed of an open contact associated with the start button and an open contact associated with the internal coil that induces the logical function of this first step. The first step is algebraically described in equation 1:

$$Q_1 = \overline{STOP} \times (START + Q_1) \qquad (1)$$

The second function is the mixed connection that establishes the memory function for the start of the sequence. It is made up of a series connection in which an open contact associated with the step one function is connected, along with a normally open contact associated with the last generated memory (which is built in the last phase of the sequence) and an open contact associated with the end-of-stroke sensor that has presented a change in its state at the beginning of the phase. This series connection is connected in parallel (for interlocking) with an open contact associated with the first memory of the sequence, i.e., the internal coil that induces the logical function of this second step.

The resulting parallel branch of this instruction is connected to an open contact associated with the internal coil of the second memory and leads to the logical function of the first memory. The second step is algebraically described in equation 2:

$$M_k = \left[\left(Q_1 \times M_{n-(k-1)} \times EoS_x\right) + M_k\right] \times \overline{M}_{k+1} \qquad (2)$$

Where: M_k is the *k*th memory the with internal coil, Q_1 is the open contact associated with the step one function, $M_{n-(k-1)}$ is the *n-(k-1)*th normally open contact associated, and *n* is the total number of phases in the sequence, EoS_x is the end-of-stroke sensor activated at the beginning of phase *k*, where *x* takes the logical values 0 or 1 and $\overline{M}_{k+1}$ is a normally closed contact associated with the next memory.

The third logical function that is invariant in this algorithm is related to the internal coil of the last memory for the last phase. This function is similar to the function for the first memory and for all memories in the sequence, except that this logical function includes an open contact associated with the start button connected in parallel with the open contact associated with the penultimate memory. The function of this contact associated with the start button guarantees the interlocking of the last memory and thus allows the start of the sequence. The third step is algebraically described in equation 3:

$$M_n = [(Q_1 \times (M_{n-1} + START) \times EoS_x) + M_n] \times \overline{M}_1 \qquad (3)$$

Where: M_n is the *n*th memory the with internal coil, M_{n-1} is the *n*-1th normally open contact associated and $\overline{M}_1$ is a normally closed contact associated with the first memory.

The main cycle runs in a closed interval, ranging from one to the total number of phases in the sequence *n* minus two, generating the algebraic function of the memories from the second to the penultimate one. Each logic function for these memories complies with the form defined by the first memory, and the indexes of the associated contacts, as well as the end-of-stroke sensors, are defined according to Figure 11. The functions corresponding to the solenoids for executing the positive and negative strokes of each double-acting cylinder are written by selecting the corresponding memory for each action on the phases of the sequence and connecting in parallel the open contacts associated with those memories, as shown in the following pseudocode:

Pseudocode to request the sequence from the user:

```
'clear variables, command window and close windows
'act = define number of actuators in the pneumatic sequence.
'define as symbolic variables the capital letters associated with double-acting cylinders.
'n = define number of phases in the pneumatic sequence.
'build a loop to prompt the user for the capital letter associated with each double-acting cylinder in each phase and the sign associated with the rod stroke in each phase of the sequence.
do for i=1:n
```

```
    sec(1,i)=input(['Enter the cylinder corresponding to the phase ',num2str(i),': ']);
    sig(1,i)=input(['Enter the sign corresponding to phase ',num2str(i),', accompanied by a 1: ']);
end do
'build a cycle to define a vector in which the stroke of the rod is distinguished in each phase of the sequence (one if it is a positive stroke and zero if it is a negative stroke)
do for j=1:n
if sig(1,j)>0
    eos(1,j)=1;
else
    eos(1,j)=0;
end
end do
```

Pseudocode for First step, equation 1:

```
'build the logic function to start and stop the sequential process
display('The logic function corresponding to startup and shutdown is: Q1=Not(stop)*(start+Q1)')
```

Pseudocode for Second step, equation 2:

```
'build the logic function for the first memory of the sequence (internal coil or flag)
display(['The logic function corresponding to the internal coil of memory M1 is M1=((Q1*M',num2str(n),'*',char(sec(1,n)),'_',num2str(eos(1,n)),')+M1)*Not(M2)',])
'build a cycle to generate the logical functions corresponding to the memories of the sequence (internal coil or flag) from the second to the penultimate
do for k=1:n-2
    display(['The logic function corresponding to the internal coil of memory M',num2str(k+1),' is M',num2str(k+1),'=((Q1*M',num2str(k),'*',char(sec(1,k)),'_',num2str(eos(1,k)),')+M',num2str(k+1),')*Not(M',num2str(k+2),')'])
end do
```

Pseudocode for Third step, equation 3:

```
build the logic function for the last memory of the sequence (internal coil or flag)
display(['The logic function corresponding to the internal coil of memory M',num2str(n),' is M',num2str(n),'=((Q1*(M',num2str(n-1),'*',char(sec(1,n-1)),'_',num2str(eos(1,n-1)),'+start))+M',num2str(n),')*Not(M1)'])
```

Pseudocode to generate the solution of the sequence introduced:

```
'build the vector for the pneumatic sequence (capital letter associated with the double-acting cylinder with the sign associated with its rod stroke)
secT=sec.*sig;
'build the logic functions for each solenoid valve (solenoid of each action to be executed), choosing the end-of-stroke sensors that present a change at the beginning of a phase
        display('The parallel branch of the open contacts associated with the memories is connected to solenoid of the valve A+: ')
        Ap=find(secT == A);
        display(Ap)
        display('The  parallel  branch  of  the  open  contacts  associated  with  the
memories is connected to solenoid of the valve A-: ')
        Am=find(secT == -A);
        disp(Am)
        display('The  parallel  branch  of  the  open  contacts  associated  with  the
memories is connected to solenoid of the valve B+: ')
        Bp=find(secT == B);
        display(Bp)
        display('The  parallel  branch  of  the  open  contacts  associated  with  the
memories is connected to solenoid of the valve B-s: ')
        Bm=find(secT == -B);
        display(Bm)
        display('The  parallel  branch  of  the  open  contacts  associated  with  the
memories is connected to solenoid of the valve C+: ')
        Cp=find(secT == C);
        display(Cp)
        display('The  parallel  branch  of  the  open  contacts  associated  with  the
memories is connected to solenoid of the valve C-: ')
        Cm=find(secT == -C);
        display(Cm)
        display('The  parallel  branch  of  the  open  contacts  associated  with  the
memories is connected to solenoid of the valve D+: ')
        Dp=find(secT == D);
        display(Dp)
        display('The  parallel  branch  of  the  open  contacts  associated  with  the
memories is connected to solenoid of the valve D-: ')
        Dm=find(secT == -D);
        display(Dm)
```

The code for this pseudocode can be optimized to work with any number of double-acting cylinders and to accept the sequence as it is defined directly from the command line (without the need to separate the letters associated with the actuators from the signs that represent the stroke of their rod). The code can be translated into any structured text language or high-level programming language, to be loaded onto programmable logic controllers or open-source programmable development boards (such as the Arduino platform), thereby defining the equivalent of a cellular automaton for low-cost sequential pneumatic circuits.

5 Conclusions

The proposed algorithm for automating pneumatic and electropneumatic sequences offers an effective solution for solving complex sequential without the need to organize detection phases into groups. Moreover, it eliminates the use of directional control valves, which are a common component in traditional methodologies. The use of groups to organize movements in detection phases can lead to several disadvantages, including design complexity, difficulty in debugging errors, increased implementation costs, higher energy consumption, and difficulties in optimizing the circuit. By avoiding the use of groups, the proposed algorithm overcomes these issues and offers a simpler and more efficient way of automating pneumatic and electropneumatic sequences.

The proposed automation technique enables the use of logic functions to define each movement in pneumatic actuators. This allows for the possibility of solving logic functions through Boolean algebra techniques, such as Karnaugh maps or the Quine - McCluskey algorithm. Furthermore, by applying the generalization of the proposed algorithm, it is possible to implement in any programming language that allows obtaining the logical functions for pneumatic sequences in complex automation processes.

In conclusion, the proposed automation algorithm in this document offers an effective solution for solving complex sequential problems in pneumatics and electropneumatics, while avoiding the disadvantages associated with traditional methodologies. Additionally, the proposed technique enables the use of logic functions to define each movement, which can be resolved using Boolean algebra techniques.

However, it's important to note that the implementation of this technique requires a significant number of pneumatic devices and systems, along with certain electrical and electronic components. In most cases, these elements can be optimized, even when the implementation challenge involves alternating movements, simultaneous or parallel motions, and even phase shifts.

When employing any traditional method for designing sequential pneumatic circuits, it is crucial to consider the trade-off between the complexity of the procedure and the simplicity of its implementation. In general, very rigorous and extensive methods result in simplified circuits. The method presented here achieves a balance between the procedure and the implementation by allowing for a simple process that does not require the use of Boolean algebra, which can lead to a highly complex circuit. On the other hand, a more complex process that uses algebraic simplifications can result in a significantly reduced circuit.

References

Barber, A. (1997). *Pneumatic handbook*. Elsevier.

Bartelt, T. L. (2012). *Industrial Control Electronics*. Cengage Learning.

Bayoumi, M. S. (2007, May). New method for designing pneumatic sequential logic controller. In *International Conference on Aerospace Sciences and Aviation Technology* (Vol. 12, No. ASAT Conference, 29-31 May 2007, pp. 1-15). The Military Technical College.

Cundiff, J. S., & Kocher, M. F. (2019). *Fluid power circuits and controls: Fundamentals and applications*. CRC Press.
Deppert, W., & Stoll, K. (2000). *Dispositivos neumáticos*. Alfaomega.

Ebel, F. (2013). *Pneumatics Basic Level*. Festo Didactic GmbH & Co KG.

Ganesh, S., & Gurunathan, S. K. (2017). Evolutionary algorithms for programming pneumatic sequential circuit controllers. *Procedia Manufacturing, 11*, 1726-1734.

Gea, J. M., & Lladonosa, V. (1998). *Circuitos básicos de ciclos neumáticos y electroneumáticos*. Marcombo.
Groover, M. P. (2016). *Automation, production systems, and computer-integrated manufacturing*. Pearson Education India.

Guenther, R., Perondi, E. A., DePieri, E. R., & Valdiero, A. C. (2006). Cascade controlled pneumatic positioning system with LuGre model-based friction compensation. *Journal of the Brazilian Society of Mechanical Sciences and Engineering, 28*, 48-57.

Henao Castañeda, É. D. J., & Monroy Jaramillo, M. (2012). Sequence table method for the solution of pneumatic and hydraulic circuits. *Revista Técnica de la Facultad de Ingeniería Universidad del Zulia, 35*(2), 132-140.

Ilango, S., & Soundararajan, V. (2011). *Introduction to hydraulics and pneumatics*. PHI Learning.
Parr, A. (2011). *Hydraulics and pneumatics: A technician's and engineer's guide*. Elsevier.
P
onomareva, E. (2006). Hydraulic and pneumatic actuators and their application areas. En *Mechatronics-Foundations and Applications* (pp. 12-19).

Sajaysurya, G., & Kumar, G. S. (2011, December). Hybrid intelligent systems for pneumatic sequential circuit design: Hybridizing genetic algorithms and rule-based logic programming. In *2011 11th International Conference on Hybrid Intelligent Systems (HIS)* (pp. 241-246). IEEE.

Santos, A. A., & Silva, A. F. D. (2017). Methodology for manipulation of Karnaugh maps designing for pneumatic sequential logic circuits. *International Journal of Mechatronics and Automation, 6*(1), 46-54.
Smith, C. A., & Corripio, A. B. (2005). *Principles and practices of automatic process control*. John Wiley & Sons.
Swider, J., Wszotek, G., & Carvalho, W. (2005). Programmable controller designed for electro-pneumatic systems. *Journal of Materials Processing Technology, 164*, 1459-1465.

Vázquez, C. R., Gómez-Castellanos, J. A., & Ramírez-Treviño, A. (2018). Tracking control of electro-pneumatic systems based on Petri nets. En *Proceedings of the 15th International Conference on Informatics in Control, Automation and Robotics (ICINCO 2018)* - Volume 1 (pp. 344-354). https://doi.org/10.5220/0006861403440354.

Venkatesh, K., Zhou, M., & 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. https://doi.org/10.1109/41.334578