

Development of a Syntax for Representing Regular Constraints on the Behavior of Distributed Systems with Output

Grygoriy Zholtkevych, Pavlo Kovalev, Victoriya Kuznietcova and Anastasiia Morozova

V.N. Karazin Kharkiv National University, 4 Svobody Sq., Kharkiv, 61022, Ukraine

Abstract

The paper addresses the method of processing and recognizing sequences of false and non-false signals from a distributed system with an output. It is proposed to adapt existing algorithms for non-distributed systems. The aim of the research is to obtain new approaches to detect and synthesize signal patterns from a distributed system. It allows us to identify more false signals than are initially known, and thus recognize more possible errors during the operation of a distributed system. The developed algorithm must meet certain criteria, such as the number of states, the percentage of false assumptions, etc. The relevance of the work lies in the need to recognize erroneous behavior in distributed systems, where the theoretical number of error patterns is proportional to the number of system elements, and it is not always possible to list all erroneous states, but it is possible to specify certain patterns of such states and allow the system to determine specific erroneous states based on known patterns. The paper uses many well-known approaches, including a similar solution for non-distributed systems, the combination and adaptation of which gives the required result.

Keywords

pattern, state, transition, configuration, regular language, detector, synthesis

1. Introduction

In the evolving landscape of technology, distributed systems have become a cornerstone in a myriad of applications ranging from industrial automation to cyber-physical systems like modern transportation and smart cities [1]. These systems are characterized by their decentralized nature, where a network of nodes operates in concert to perform complex tasks [2]. However, this complexity also brings forth significant challenges, particularly in monitoring and detecting erroneous states that can critically impact system performance and reliability [3].

The need for robust error detection in distributed systems is not just a matter of operational efficiency but safety and reliability [4]. While effective in simpler, non-distributed systems, traditional error detection methods often must be revised to address the intricate dynamics of distributed environments [5]. These systems demand a more nuanced approach to recognize and interpret the signals from various nodes [6].

The advent of advanced technologies in distributed systems has further complicated the landscape of error detection [7]. With the integration of IoT devices, cloud computing, and artificial intelligence, these systems are becoming more complex and critical to manage effectively. The interconnected nature of these technologies means that a single erroneous signal can have cascading effects across the entire network [8]. While offering enhanced capabilities, this technological integration underscores the need for sophisticated error detection methods


Profit AI 2023: 3rd International Workshop of IT-professionals on Artificial Intelligence (Profit AI 2023), November 20–22, 2023, Waterloo, Canada

✉ g.zholkevych@karazin.ua (G. Zholtkevych); kovalev.pavlo.evgeniyovych@gmail.com (P. Kovalev); vkuznietcova@karazin.ua (V. Kuznietcova); a.morozova@karazin.ua (A. Morozova)

ORCID 0000-0002-7515-2143 (G. Zholtkevych); 0009-0008-4112-5105 (P. Kovalev); 0000-0003-3882-1333 (V. Kuznietcova); 0000-0003-2143-7992 (A. Morozova)



© 2023 Copyright for this paper by its authors.
Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).

 CEUR Workshop Proceedings (CEUR-WS.org)

that can adapt to the evolving nature of distributed systems and handle the diverse range of signals and states they exhibit [9].

Recognizing this gap, our research pivots to developing an algorithm tailored for distributed systems, focusing on synthesizing false signal sequences. This algorithm adapts and extends existing methodologies originally designed for non-distributed systems to suit the complex requirements of distributed architectures. Our primary objective is to facilitate more accurate, efficient, and reliable monitoring of these systems by honing the detection of erroneous states through an innovative approach to signal synthesis.

This work aims to develop a language for representing regular constraints in distributed systems by available finite sets of events, detecting patterns, and directly recognizing these events in a given distributed system. In the context of this paper, a pattern is a template for the state of a distributed system.

Based on the goal, the following tasks were defined: developing a language that would allow describing patterns of sequences of events and a system for recognizing these patterns. Taken together, the solution of these tasks allows us to control the state of a distributed system by declaring the required system states and analyzing the current state.

During the study, it was proposed to adapt existing mechanisms for pattern recognition in non-distributed systems and a new way of declaring patterns.

The relevance of the work is due to the need to control distributed systems, namely to identify sequences of events that may indicate a malfunction in the system, an attack, etc. To summarize, there is a need to recognize false system states. Currently, some approaches can recognize a false state but cannot make assumptions about states that are not defined as false. It is not always possible to list all such system states, so searching for patterns in known false states and recognizing them is necessary.

This paper presents the study's results: a language for specifying event patterns and a system for recognizing them. The algorithm presented in this paper solves the problem of supplementing a known set of patterns.

2. Introduction

2.1. Background

Distributed systems are very common in many areas of human activity, such as cyber-physical systems (e.g., a modern car, airplane, industrial machine) [10] or purely software systems (e.g., a web server or a virtual machine cluster management system) [11]. As a rule, modern distributed systems operate with a large number of nodes that provide information about their current state, based on which the decision-making node forms an appropriate response and changes the system state.

Very often, to make a global decision about the further behavior of the system, it is necessary to have up-to-date information about the states of all nodes or some subset of them. For example, if we consider an airplane as a cyber-physical system, in order to enable landing mode, the airplane control system must collect data on the landing gear, flap angle, elevator angle, engine speed, etc. [12].

Because of this, the problem of parallel information processing arises, i.e., recognizing a set of signals coming from different system nodes that can be interpreted as a sequence of states, where each element of the sequence is the current state of a particular system node. It is important that the system is able to recognize such states that are a system malfunction (hereinafter referred to as an error in the system). Given that there are many events in such systems, we can summarize the concept of an error to an error pattern, i.e. certain sequences of events in certain nodes, the occurrence of which indicates a malfunction of the system.

The problem is also complicated by the fact that each node of the system can be in many states, and also has its own sequences of states that should be considered a malfunction. To summarize, the task is to recognize disturbances in the distributed system as a whole, which means that it is

necessary to build an algorithm for recognizing errors in a two-dimensional vector of system states.

2.2. The state of the art

Currently, an algorithm for recognizing and completing error patterns exists and was described in [13]. The goal is to propose a similar algorithm for recognizing a two-dimensional set of patterns, and to optimize the algorithm according to important metrics.

This paper uses and develops the ideas proposed in the article, so here are the main definitions used in the paper.

Definition 1. A regular pattern detector is a tuple

$$R = \langle \Sigma, Q, q_0, \Pi, \delta \rangle,$$

where Σ is the alphabet of input signals; Q is a finite set of states; $q_0 \in Q$ is a fixed state called the initial state; Π is a finite alphabet whose elements denote the corresponding recognized patterns; $\delta: Q \times \Sigma \rightarrow \Pi + Q$ is a decision function.

An alphabet is a finite set of characters. A finite sequence of characters is called a word and is denoted by Σ^+ .

Definition 2. A partial mapping from a set X to a set Y is a subset $f \subset X \times Y$ such that: if $(x, y') \in f$, where $x \in X$ and $y', y'' \in Y$ means that $y' = y''$.

$f(x) \downarrow = y$ means that $(x, y) \in f$

$f(x) = \uparrow$ means that $(x, y) \notin f \forall y \in Y$

$f(x) \downarrow$ means that $\exists y \in Y: (x, y) \in f$

To determine the Π -indexed family $R^+ = \{R_s^+ | s \in \Pi\}$ of sequence sets recognized by a regular pattern detector R , we need to supplement the decision function δ with a partial mapping $\delta^+: Q \times \Sigma \rightarrow \Pi + Q$ which is defined recursively as follows:

$\delta^+(q, a) \downarrow = \delta(q, a)$ for any $q \in Q$ and $a \in \Sigma$

$\delta^+(q, ua) = \uparrow$ if $\delta^+(q, u) \downarrow = s$ or $\delta^+(q, u) \uparrow$

$\delta^+(q, ua) \downarrow = \delta(q', a)$ or $\delta^+(q, ua) \downarrow = \delta(q', a)$ where $q' \in Q$

Definition 3. The word $u \in \Sigma^+$ is recognized by a regular pattern detector R as a pattern $s \in \Pi$ (denoted as $u \in R_s^+$) if $\delta^+(q_0, u) \downarrow = s$.

In other words,

$$R_s^+ = \{u \in \Sigma^+ | \delta^+(q_0, u) \downarrow = s\},$$

for any $s \in \Pi$.

Thus, the Π indexed family of $R^+ = \{R_s^+ | s \in \Pi\}$ a subset Σ^+ is associated with any regular pattern detector R . The properties of this family are defined in the following statement.

Statement 1. For any regular pattern detector R , Π the indexed family R^+ has the following properties:

1. Sets with R^+ are non-overlapping sets.
2. Each set in R^+ is regular.
3. The common plural prefix $\bigcup_{s \in \Pi} R_s^+$ is an empty word.

The proof of this statement is given in [14].

Algorithm 1. Synthesis algorithm for regular pattern detectors.

The idea of the algorithm is as follows:

Having any finite alphabet of signals and a finite alphabet of recognizable patterns, we fix Π as indexed family of mutually non-overlapping sets $\{E_s | s \in \Pi\}$ from finite subsets, which do not have Σ^+ of the common prefix; a finite subset $C \subset \Sigma^+$ of events that are considered unrecognizable and moreover, $E = \bigcup_{s \in \Pi} E$, where $E \cap C = \emptyset$.

The output of the algorithm is a regular pattern detector $\langle \Sigma, Q, q_0, \Pi, \delta \rangle$ so that:

1. $E_s \subset R_s^+$ for all $s \in \Pi$.
2. $\bigcap_{s \in \Pi} R_s^+ \cap C = \emptyset$.
3. The number of elements in Q is less or equal to the number of states for any detector satisfying conditions 1 and 2.

Data: Π is an indexed family of E finite subsets Σ^+ that do not share a common prefix and a finite subset $C \subset \Sigma^+$.

Results: regular pattern detector R for all $q \in Q$ and $a \in \Sigma$:

if $\delta(q, a) = \perp$ then:

choose randomly X from $(Q + \Pi) \setminus \{q, \perp\}$;

redefine $\delta(q, a) = x$;

if $u \in R^+_s$ for any $s \in \Pi$ and $u \in C$

cancel the override;

continue;

minimize the resulting detector by the Hopcroft method.

The symbol \perp is used to indicate a trash state that is not a member of Q . Also, the state that the detector takes after successful pattern recognition is called the terminal state.

Tree-shaped detector $\langle \Sigma, Q, q_0, \Pi, \delta \rangle$ is determined by the following way:

$$Q = \left\{ u \in \Sigma^* \mid uv \in \bigcup_{s \in \Pi} E_s, v \in \Sigma^* \right\} \cup \{\perp\},$$

$q_0 = \epsilon$;

$\delta(\perp, a) = \perp$ for all $a \in \Sigma$;

$\delta(u, a) = ua$ if $ua \in Q$;

$\delta(u, a) = \perp$ or else.

2.3. Task statement

The aim of this paper is to develop a recognizer for certain sequences of events in a distributed system, as well as a language that could be used to describe false and non-false sequences. The recognizer should also supplement the set of false sequences by identifying patterns in known sequences.

In the context of this work, these sequences are called regular constraints because they are the language of the detector. This language is also called the syntax of regular constraints on the behavior of a distributed system with an output. Thus, it is necessary to develop an algorithm that, given a finite set of constraints and words that are not constraints, will synthesize a syntax to represent the regular constraints of a certain system with an output.

The constructed detector should have the following properties:

1. Parallel recognition of signals from nodes of a distributed system
2. Minimum number of states of the resulting detector

3. Methods

3.1. Reasoning for choosing the algorithm

The main difference, and the reason why it is impossible to use the existing synthesis algorithm to solve the problem, is that the system is divided, and at one moment of time signals can come from many channels. If we assume that there are channels, and the 3d channel receives messages constructed over the alphabet, then if we collect data from all channels at a certain point in time, we get a vector of length, which will represent the current state of the system.

Theoretically, such a word could be considered a regular constraint, but constraints in a system exist not only globally (for the entire system), but also for its individual nodes. If you declare constraints in this way, it will be difficult to distinguish and understand, looking at the constraints in this form, how the system as a whole should actually behave, and how its individual elements should behave.

In this case, it is possible to use the principle of "divide and conquer" [15], in the sense that it is possible to solve a larger number of simple problems, and only then combine their solutions to solve the problem.

Since the global state of the system is completely dependent on its parts, and the parts of the system are ordinary message generators, we can set constraints for each part of the system separately and check the state of a particular element of the distributed system. The state of each individual element of the system can be generalized to the state of normal operation or to the state of error, which can be represented by an alphabet of only two letters, for example, 0 and 1, accordingly.

It is worth mentioning that the survey of system elements on their condition occurs with a certain periodicity, so we will consider time as discrete.

Thus, the task comes down to first recognizing the signals from each element of the system separately, and for this purpose, there is already a detector algorithm described in [14] and a synthesis algorithm (1).

If a detector is created for each of the system elements, then the trash state, or any non-terminal state, can be considered as 0, and the terminal state as 1. In this context, the existence of a terminal state indicates the recognition of an error pattern in a system element.

After implementing detectors for individual elements of a distributed system, the output vector obtained from the elements is simply a sequence of 0s or 1s. This sequence can be further simplified by considering it as a binary representation of a non-negative integer. Since we need to recognize a word again (though now only over a two-character alphabet), we can use the idea of a detector again. We will call this detector a coordinating detector because it recognizes the word obtained from the current recognition results of the detectors on each node of the system.

Now the main problem is the number of states in such a recognizer - that is, the number of possible transitions equal to because from each state it is possible to transfer to any other state. This number of states is not satisfactory, because the number of states grows very quickly with the number of elements of the distributed system.

This means that it is not necessary to generate all theoretically possible states, but only those that can appear with a high probability. It is also necessary to determine the total number of generated states, or the steps that the state generation algorithm will take because the algorithm must stop at some point.

To optimize the number of states, it is worth considering the following concepts the intensity of each signal source (system element). Based on the information about the frequency of receiving messages, the next state can be predicted from a given state, and then there is no need to create transitions from this state to unlikely states, which can save both the number of states and the number of transitions. Additionally, there is a limit to the length of each word of the coordinating detector. In fact, given that there are a total of channels, the length of the coordinating detector word is always equal to n , since the i -th character of this word corresponds to the current state of the i -th channel.

Therefore, it is recommended to use:

1. Alphabet for the coordinating detector. Restriction - words are built over this alphabet.
2. Any alphabet is necessary for an element of the system to describe its states (in the context of this work, this is not important, because we only need the result of the node detector)
3. An algorithm for synthesizing new states based on the frequency of messages (other approaches for synthesizing states will also be considered in the work).

3.2. Implementation of algorithms

First, let us explain some notations that will be used in the following algorithms.

D is a set of words to be recognized by the detector. Hereinafter, the words consist of the alphabet $\Sigma = \{0, 1\}$; N is a set of words that should not be recognized by the detector; n is number of channels (system elements); $|D|$ is the number of words to be recognized by the detector; $|N|$ is the number of words to be recognized by the detector.

Algorithm 2: Calculating the intensity of each channel.

Data: channels $Ch = \{S_1, \dots, S_n\}$

limitations $|Ch| = n, |D_i| = n \forall i = 1 \dots n$

Result: Tuple $\langle f_1, \dots, f_n \rangle$ of frequencies of each channel.
 For each Ch from

$$f_{ch} = \sum_{i=1}^{|D|} D_{s_i},$$

$$f_{ch} = \frac{f_{ch}}{|D|}.$$

The graph below shows the expected number of signals from each element of the system at each moment of time, the number of channels = 5. Values received per channel:

$\{0, 1, 1, 1, 0\}, \{1, 1, 1, 1, 0\}, \{1, 1, 1, 0\}, \{0, 0, 1, 1, 1\}, \{0, 1, 0, 1, 0\}$

The vertical line divides the chart into two halves: the left half shows the actual number of signals (cumulative), and the right half shows the expected number of signals.

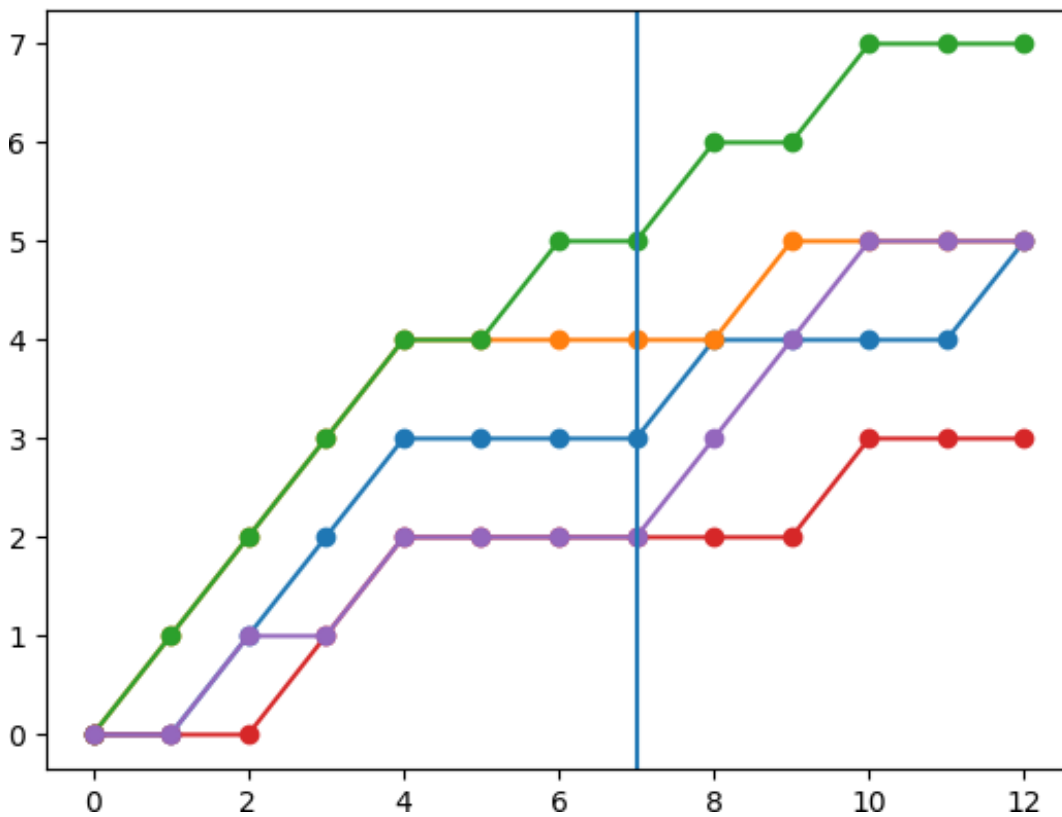


Figure 1: Expected frequency of messages from the channels of a distributed system

Algorithm 3: Frequency-based state generation.

Data: Tuple $freq = \langle f_1, \dots, f_n \rangle$ of frequencies created with algorithm (2).

Result: the random state of the coordinating length detector n that meets the frequency requirements.

For each i with $1 \dots n$

choose a random number r from the segment $[0; 1]$
 add 0 to the current state if $r < f_i$, otherwise add 1.

Algorithm 4: Tree-based detector design based on frequency characteristics of channels, known false and non-false states.

Data: $Ch = \{S_1, \dots, S_n\}$, constrains D, N , and also tuple $freq = \langle f_1, \dots, f_n \rangle$ of frequencies generated with algorithm (2)

Result: a tree-like detector R , a set of states Q specified by constraints, and a set of generated states Q_g .

```

Initialize the current state cur
For each r with D:
 $\delta(\text{cur}, r) = \text{cur} + r$ 
 $\text{cur} = \text{cur} + r$ 
For each sequence  $s_d \in D$ 
For each state q in  $S_d$ 
    Initialize the current state  $\text{cur} = q$ 
    Initialize the current transition  $\text{oldNext} = \delta(q)$ 
    Generate  $|D|$  of new states G using algorithm (3)
    For each state g with G:
         $\delta(\text{cur}, g) = \text{cur} + g$ 
         $\text{cur} = \text{cur} + g$ 
If recognized  $s \in N$ 
Remove new states  $\delta(q) = \text{oldNext}$ 
Minimize the detector with the Hopcroft algorithm [16].

```

When generating new states, it is important not to significantly change the theoretical frequency of each channel. That's why a sequence of random states is generated, because the initial frequency is measured by the characteristic of the states. Also, the algorithm needs to be stopped, although theoretically it can be generated until the states satisfy the frequency characteristics. Of course, the generation of states does not guarantee that the resulting sequences will have a frequency close to the initial one.

However, it is possible to filter out sequences whose frequency characteristics differ from the initial ones, and the degree of difference can be adjusted using the parameter, which can be used to set the maximum frequency deviation from the initial frequency characteristic. Next, we propose an algorithm for removing generated sequences whose frequency differs from the initial one by more than.

Algorithm 5. Removing sequences that do not meet the frequency characteristics.

Data: a tree-like detector *R* with states $Q + Q_g$, frequency characteristics $freq = \langle f_1, \dots, f_n \rangle$, parameter of absolute frequency difference ϵ .

For each sequence $s \in R$

Determine the expected frequency f_e of each channel for *s*

Remove the last states $|D|$ from *s*, if $\exists i \in 1 \dots n: |f_i - f_e| < \epsilon$.

The algorithm removes only the last $|D|$ states of sequences whose frequency characteristics do not meet the requirements. It is obvious that since algorithm (4) generated exactly $|D|$ new states, if the frequency response has changed, it is due to the new states, not the existing ones, on the basis of which the initial frequency was calculated. Moreover, deleting the entire sequence reduces the number of initial sequences.

As a result of algorithms (4) and (5), we have a system of states with transitions (in fact, a deterministic automaton) that recognizes error patterns and does not recognize patterns of normal system behavior, which is what we needed to do. The set of error patterns is supplemented by generating new states and transitions based on information about the frequency of events in each element of the system. Each generated sequence corresponds to the given sequences in terms of frequency characteristics; at best, the number of completed transitions is $|D|^2$; at worst, 0 if all generated states and transitions differ from the specified characteristics.

Additionally, it should be noted that for elements of a distributed system, it is assumed that the transition to the terminal state does not mean the transition to the terminal state of the entire system, and when such a state appears, the system element signals it and transitions to the initial state.

Figures 2, 3 shows states and transitions before and after running the synthesis algorithm (4).

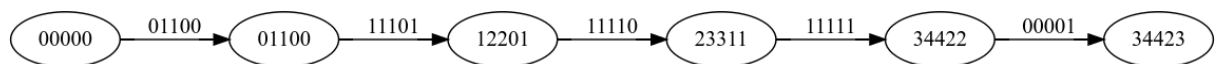


Figure 2: Graph of initial constrain

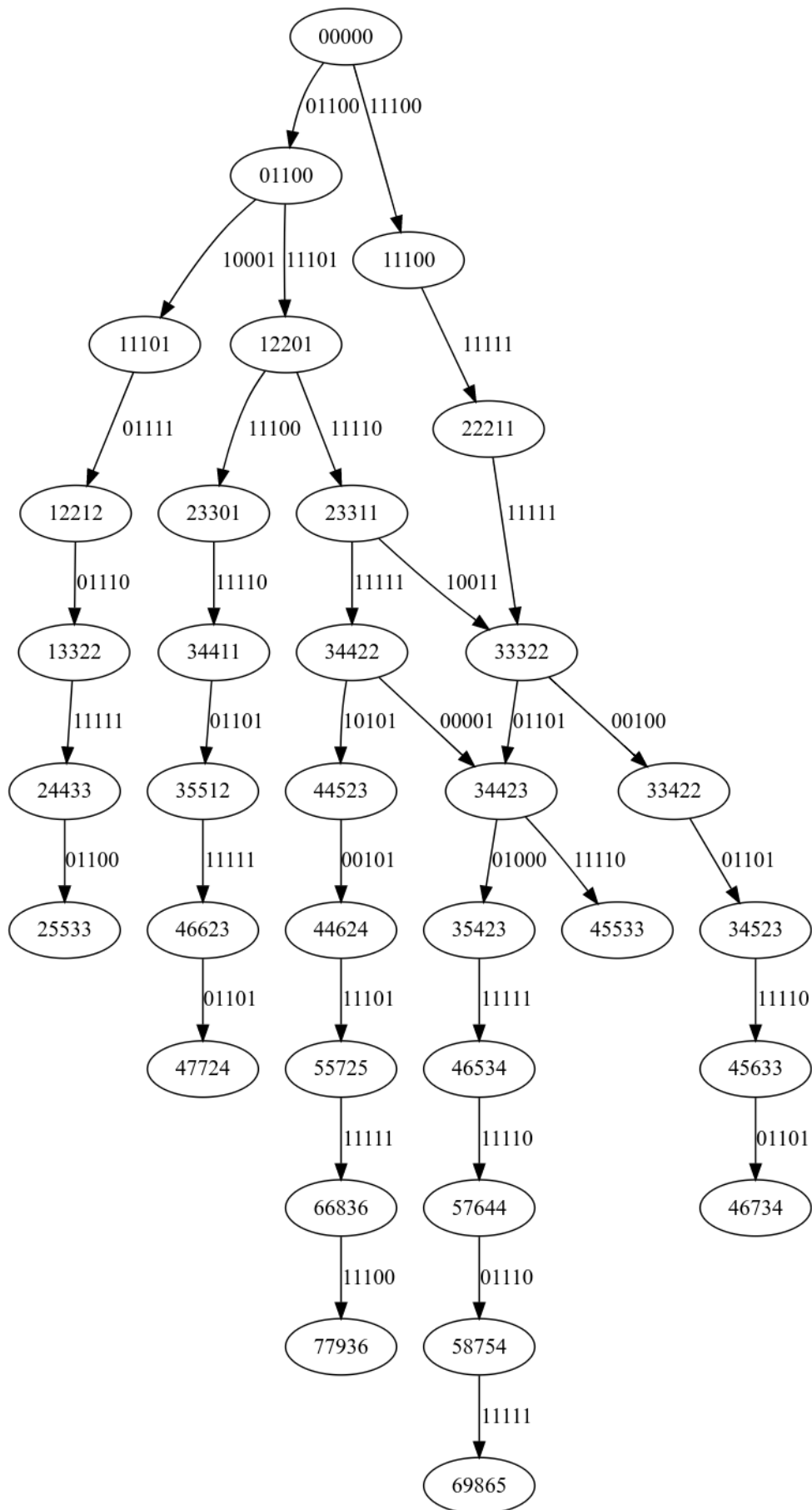


Figure 3: Graph of the enumerations of the algorithm (4)

4. Study of algorithms and analysis of results

4.1. Study of the influence of parameters on the results

As mentioned in the previous section, the algorithm for generating new states and sequences should preserve the frequency characteristics of the channels. We also proposed a sequence filtering algorithm based on the absolute frequency difference parameter. The optimal value of the parameter remains an open question ϵ . It is worth noting that the value of this parameter does not affect the accuracy of the detector, but only affects the number of states of the resulting automaton. At any value of the parameter, the algorithm does not recognize normal (not false) sequences, but only changes the number of probable false sequences.

Below is a graph of the dependence of the number of states of the resulting detector on the value of the parameter ϵ . The sequence of initial false states:

$$\{0, 1, 1, 1, 0\}, \{1, 1, 1, 1, 0\}, \{1, 1, 1, 0\}, \{0, 0, 1, 1, 1\}, \{0, 1, 0, 1, 0\}$$

Sequence of normal states $\{0, 1, 1, 1, 0\}, \{0, 1, 0, 1, 0\}$. The parameter ϵ changes from 0 to 0.5 in increments of 0.1. The horizontal line indicates the total number of states generated by the synthesis algorithm (4), but before the filtering algorithm (5) is run. The algorithms were implemented using the Python program language

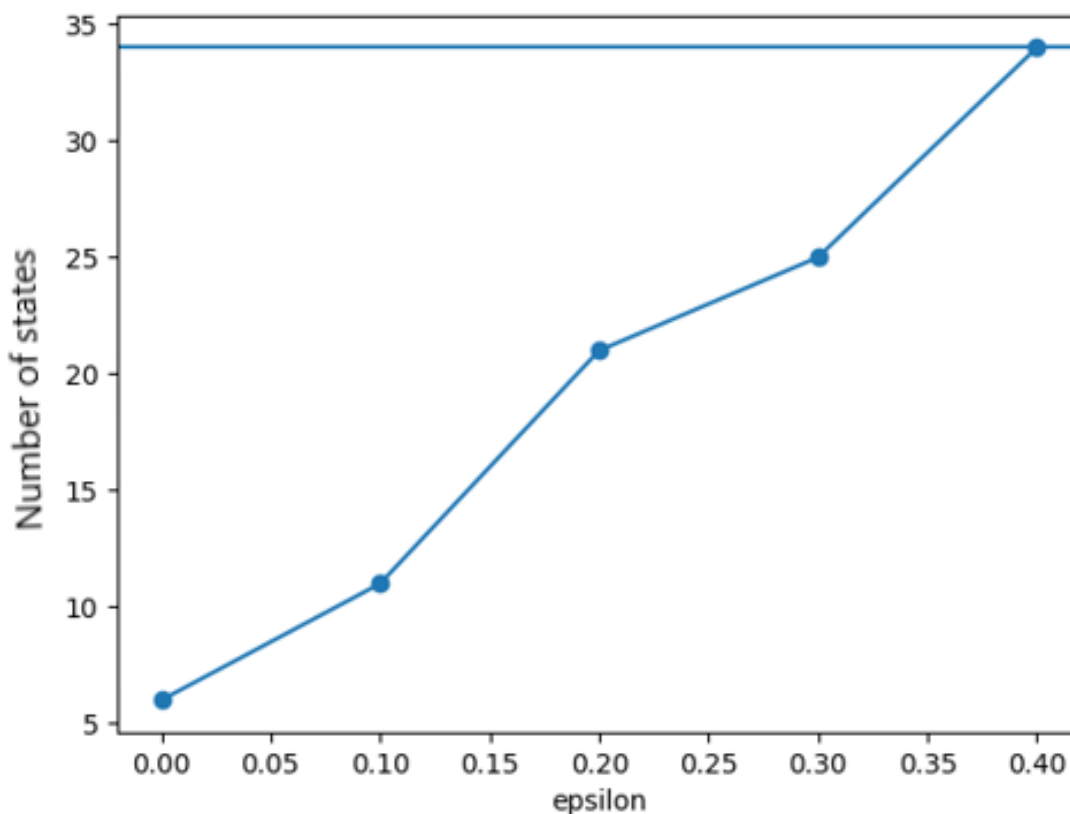


Figure 4: The dependence of the number of states on ϵ

The graph shows that the frequency deviation does not exceed 0.4 for any of the generated sequences. From the graph, we can conclude that the number of states depends on the parameter ϵ almost linearly.

Another important information for understanding the behavior of the algorithm depending on the parameter ϵ is the value of the average deviation of the actual channel frequency from the

expected one. To get the average deviation, we find the average difference between the expected frequency of each channel and the actual one. Below is a graph of the average deviation versus from ϵ .

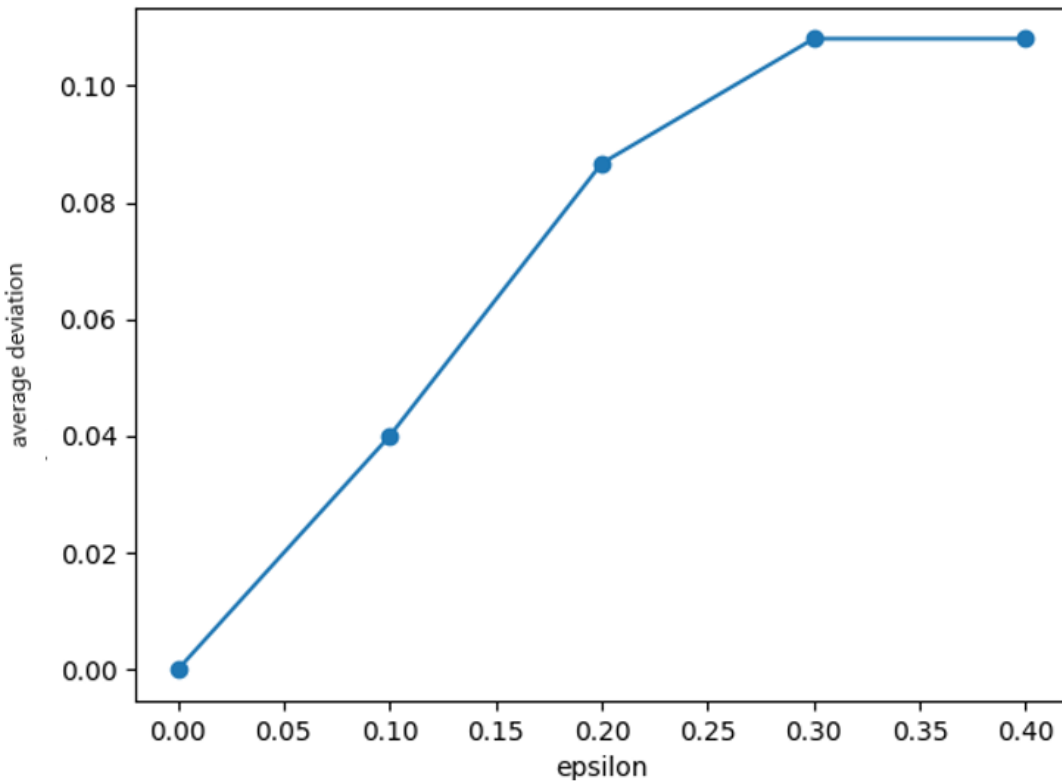


Figure 5: Dependence of the average frequency deviation on the number of states ϵ

The average deviation correlates with the number of states, but does not exceed 0.1.

Next, it is necessary to compare the obtained results with the conventional synthesis method for the detector described in [3]. In the original implementation, the algorithm is not able to correctly generate new states for the coordinating detector, because it randomly changes the length of the recognized words. To eliminate the influence of this factor on the result, we will consider only such sequences of states in which the total length is a multiple of n (the number of channels of the distributed system). This way, we can reproduce $|D|/n$ iterations of the algorithm's recognition of sequences of length.

We also use the same constraints as for the coordinating detector: the sequence of initial false states:

$$\{0, 1, 1, 1, 0\}, \{1, 1, 1, 1, 0\}, \{1, 1, 1, 0\}, \{0, 0, 1, 1, 1\}, \{0, 1, 0, 1, 0\}$$

A sequence of normal states: $\{0, 1, 1, 1, 0\}, \{0, 1, 0, 1, 0\}$. It is important to know the number of states in the generated detector, as well as the average frequency deviation.

The algorithm must be run many times in a row to obtain a reliable average because the algorithm depends on a random number generator.

After running the algorithm on the test dataset twenty times in a row, the average frequency deviation was 0.1895. The number of states does not depend on random numbers and is equal to 26. Below is a graph of frequency deviations depending on the algorithm execution. The minimum average deviation was 0.15, the maximum was 0.2.

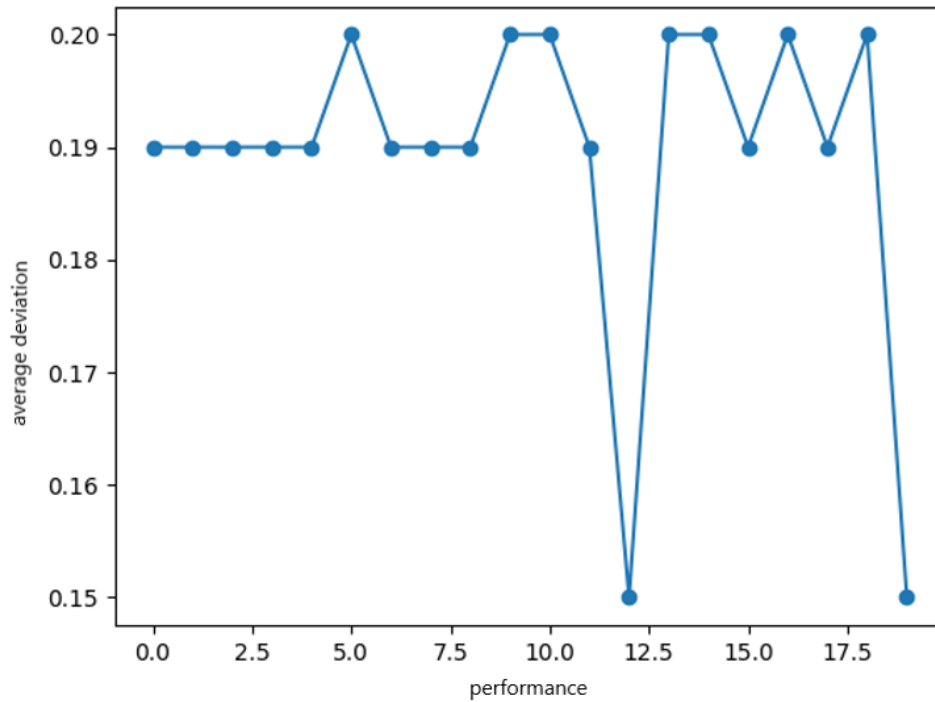


Figure 6: Average deviation in the conventional synthesis method

4.2. Analysis of results

Based on the results of both algorithms, we can conclude that with the same number of states in both algorithms, the accuracy of the algorithm proposed in this paper is exactly twice as high, as evidenced by the difference in the average frequency difference - 0.9 at $\epsilon = 0.25$ for the proposed algorithm, and 0.19 for the existing algorithm. Thus, 0.25 can be considered the optimal value. However, the flexibility of the algorithm allows it to be adapted to different use cases.

To increase the generation accuracy, you need to reduce the value of ϵ , but get fewer generated sequences. If you need as many generated sequences as possible, you can increase the parameter, neglecting the possible error.

5. Conclusions

The result of the study is a developed and analyzed algorithm for synthesizing false signal sequences based on finite sets of false and non-false sequences. This algorithm is an adapted version of an existing algorithm for a non-distributed system with an output [14].

As a result of our work, we have achieved:

1. A language of constraints for a distributed system with an output has been developed
2. The optimal number of detector states was achieved, which is less than the maximum theoretically possible number.

3. A mechanism for adjusting the accuracy of transition detection has been developed. The results of this work may allow researchers to develop new automated systems for monitoring the operation of distributed systems, or improve the efficiency of existing solutions. This could allow for greater automation in the operation and monitoring of distributed systems, potentially meaning more reliable systems.

As an intermediate conclusion, this algorithm can be used for experimental purposes, on test samples to investigate the very effectiveness of synthesis algorithms and the approach to processing parallel events in general. Further development of the idea can be carried out in several directions: increasing the probability of guessing a false state, further reducing the total number of states, and further simplifying the constraint language.

References

- [1] P. Chamoso, A. González-Briones, F. De La Prieta, G. K. Venyagamoorthy, and J. M. Corchado, "Smart city as a distributed platform: Toward a system for citizen-oriented management," *Computer Communications*, vol. 152, pp. 323–332, Feb. 2020, doi: 10.1016/j.comcom.2020.01.059.
- [2] W. Basmi, A. Boulmakoul, L. Karim, and A. Lbath, "Distributed and scalable platform architecture for smart cities complex events data collection: Covid19 pandemic use case," *Journal of Ambient Intelligence and Humanized Computing*, vol. 12, no. 1, pp. 75–83, Jan. 2021, doi: 10.1007/s12652-020-02852-9.
- [3] D. Chumachenko, K. Chumachenko, and S. Yakovlev, "Intelligent simulation of network worm propagation using the Code Red as an example," *Telecommunications and Radio Engineering*, vol. 78, no. 5, pp. 443–464, 2019, doi: 10.1615/telecomradeng.v78.i5.60.
- [4] T. Long, X. Ren, Q. Wang, and C. Wang, "Verifying the safety properties of distributed systems via mergeable parallelism," *Journal of Systems Architecture*, vol. 130, p. 102646, Sep. 2022, doi: 10.1016/j.sysarc.2022.102646.
- [5] K. Bazilevych, et al., "Stochastic modelling of cash flow for personal insurance fund using the cloud data storage," *International Journal of Computing*, vol. 17, no. 3, pp. 153–162, Sep. 2018, doi: 10.47839/ijc.17.3.1035.
- [6] N. Dotsenko, et al., "Modeling of the Processes of Stakeholder Involvement in Command Management in a Multi-Project Environment," *2018 IEEE 13th International Scientific and Technical Conference on Computer Sciences and Information Technologies (CSIT)*, pp. 29–32, Sep. 2018, doi: 10.1109/stc-csit.2018.8526613.
- [7] D. Jeanneau, O. Carneiro, L. Arantes, and E. P. Duarte, "An autonomic hierarchical reliable broadcast protocol for asynchronous distributed systems with failure detection," *Journal of the Brazilian Computer Society*, vol. 23, no. 1, Dec. 2017, doi: 10.1186/s13173-017-0064-9.
- [8] D. Chumachenko, "On Intelligent Multiagent Approach to Viral Hepatitis B Epidemic Processes Simulation," *2018 IEEE Second International Conference on Data Stream Mining & Processing (DSMP)*, pp. 415–419, Aug. 2018, doi: 10.1109/dsmp.2018.8478602.
- [9] I. Meniaïlov and H. Padalko, "Application of Multidimensional Scaling Model for Hepatitis C Data Dimensionality Reduction," *CEUR Workshop Proceedings*, vol. 3348, pp. 34–43, 2022.
- [10] V. Sood, M. K. Nema, R. Kumar, and M. J. Nene, "Conceptual Verification of Distributed Cyber Physical Systems using Reference Nets," *Procedia Computer Science*, vol. 171, pp. 81–90, 2020, doi: 10.1016/j.procs.2020.04.009.
- [11] K. Bazilevych, S. Krivtsov, and M. Butkevych, "Intelligent Evaluation of the Informative Features of Cardiac Studies Diagnostic Data using Shannon Method," *CEUR Workshop Proceedings*, vol. 3003, pp. 65–75, 2021.
- [12] M. Fioriti, P. Della Vecchia, and G. Donelli, "Effect of Progressive Integration of On-Board Systems Design Discipline in an MDA Framework for Aircraft Design with Different Level of Systems Electrification," *Aerospace*, vol. 9, no. 3, p. 161, Mar. 2022, doi: 10.3390/aerospace9030161.
- [13] O. Etzion and P. Niblett, *Event Processing in Action*. Manning, 2010. Available: <https://www.manning.com/books/event-processing-in-action>
- [14] G. Zholtkevych, S. Lukyanenko, and N. Polyakovska, "Toward Synthesis of Event-Pattern Detectors for Event Complex Processing with Using Machine Learning," *CEUR Workshop Proceedings*, vol. 2104, pp. 707–715, 2018.
- [15] T. Cormen, C. Leiserson, R. Rivest, and C. Stein, *Introduction to Algorithms*, Third. Cambridge, Mass.: Mit Press, 2009.
- [16] J. E. Hopcroft, "An $n \log n$ algorithm for minimizing states in a finite automation," *Proceedings of an International Symposium on the Theory of Machines and Computations Held at Technion in Haifa, Israel, on August 16–19*, pp. 189–196, Jan. 1971, doi: 10.1016/b978-0-12-417750-5.50022-1.