On the Improvement of R-TNCESs Verification using Distributed Cloud-based Architecture

Choucha Chams Eddine[1], Mohamed Oussama Ben Salem[2], Mohamed Khalgui[1,3], Laid Kahloul[4]
and Naima Souad Ougouti[5]
[1]*LISI Laboratory, National Institute of Applied Sciences and Technology (INSAT),*
University of Carthage, Tunis 1080, Tunisia
[2]*Team Project IMAGES-ESPACE-Dev, UMR 228 EspaceDev IRD UA UM UG UR, University of Perpignan Via Domitia,*
Perpignan 66860, France
[3]*School of Electrical and Information Engineering, Jinan University, Zhuhai Campus, Zhuhai 519070, China*
[4]*LINFI Laboratory, Computer Science Department, Biskra University, Biskra, Algeria*
[5]*LSSD Laboratory, Computer Science Department,*
University of Science and Technology of Oran Mohamd Boudiaf, Algeria

Keywords: Formal Verification, Discrete-event System, Reconfiguration, Petri Net, Ontology.

Abstract: Reconfigurable discrete event control systems (RDECSs) are complex and critical systems, motivating the use of formal verification. This verification consists of two major steps: state space generation and state space analysis. The application of the mentioned steps is usually expensive in terms of computation time and memory. This paper deals with state space generation (accessibility graph generation) during verification of RDECSs modeled with specified reconfigurable timed net condition/event systems (R-TNCESs). We aim to improve model checking used for formal verification of RDECSs by proposing a new aproach of state space generation that considers similarities. In this approach, we introduce the modularity concept for verifying systems by constructing incrementally their accessibility graphs. Furthermore, we set up an ontology-based history to deal with similarities between two or several systems by reusing state spaces of similar components that are computed during previous verification. A distributed cloud-based architecture is proposed to perform the parallel computation for control verification time and memory occupation. The paper's contribution is applied to a benchmark production system. The evaluation of the proposed approach is performed by measuring the temporal complexity of several large scale system verification. The results show the relevance of this approach.

1 INTRODUCTION

Reconfigurable discrete event control systems (RDECSs) are the trend of future systems. RDECSs can be reconfigured in a static way (off-line) or in a dynamic way (automatically at run-time). In the latter, a reconfiguration scenario should be applied automatically and timely as a response related to dynamic environment, or user requirements. Therefore, an RDECS may go through several modes at run-time (Khalgui et al., 2011), increasing verification process complexity. Formal verification represents a reliable method to ensure the correctness of RDECSs. Usually, it consists in generating and analyzing the state spaces of studied systems. However, with the combinatorial growth, the state space size becomes too big, even with small sized systems. Hence, model-checking becomes quite challenging for industry and academia because of the state space explosion problem (Valmari, 1996). Several studies have been done to cope with state space explosion problems. The authors in (Souri et al., 2019) present symbolic model checking that represents the state space symbolically instead of explicitly, by exploiting the state graph regularity using boolean functions. In (Gadelha et al., 2017), bounded model checking (BMC) is proposed to look for a counter-example in executions whose length is limited by an integer k. If no bug is found, then k is increased until a possible bug is found. The above methods can proceed efficiently proceed to complex systems verification. However, they use an implicit representation of state

Eddine, C., Ben Salem, M., Khalgui, M., Kahloul, L. and Ougouti, N.
On the Improvement of R-TNCESs Verification using Distributed Cloud-based Architecture.
DOI: 10.5220/0009836103390349
In *Proceedings of the 15th International Conference on Software Technologies (ICSOFT 2020)*, pages 339-349
ISBN: 978-989-758-443-5

spaces, which present limitation for computation of quantitative properties (e.g., state probabilities in stochastic models) (Camilli et al., 2014).

With the apparition of new complex systems such as reconfigurable manufacturing systems, reconfigurable wireless networks, etc (Ben Salem et al., 2017), techniques and formalisms used for verification must evolve. Petri nets has been extended by many works. Reconfigurable Petri nets presented in (Padberg and Kahloul, 2018), proposed for reconfigurable systems. However, although useful, being non-modular formalism, it can cause confusion to engineers for possible reusing. Timed net condition/event systems (TNCES) formalism prersented in (Hafidi et al., 2018) as modular extension of Petri nets to deal with time constraints. TNCES is used for their particular dynamic behavior, modularity and interconnection via signals. However, dynamic behavior of reconfigurable systems is still not supported. Reconfigurable net condition/event systems (R-TNCESs) are developed as an extension of the TNCES formalism in (Zhang et al., 2013), where reconfiguration and time properties with modular specification are provided in the same formalism while keeping the same semantics of TNCESs. With R-TNCES formalism, physical system processes are easily understood thanks to modular graphic representations. In addition, it can capture complex characteristics of an RDECS. Formally an R-TNCES is a multi-TNCES defined as a couple (B,R), where B is a set of TNCESs, and R is a set of reconfiguration rules(Zhang et al., 2013). A layer-by-layer verification method is proposed where similarities between TNCESs are considered. This method is improved in (Hafidi et al., 2018) where the authors propose a new method for accessibility graph generation with less computing time and less required memory.The previous methods improve classical ones. However, with large scale systems, their application using a unique machine (i.e., a centralized system) may be expensive in terms of time.

In this paper, we are interested in reconfigurable systems, modeled with the R-TNCES formalism where the RDECS behavior is represented by the behavior of control components (CCs) and the communication between them (synchronization) (Zhang et al., 2013). We propose a new verification method that aims to improve R-TNCES formal verification. Indeed, state space generation is considered the most complex verification step, thus we focus on its improvement. The verification of an R-TNCES requires checking of each configuration, namely each TNCES. TNCESs which describe configurations often contain similarities called internal similarities. On another hand, some RDECSs share the same system components, so their model contains similarities called external similarities, which implies redundant calculation during checking of these systems. Thus, in order to avoid many repetitive computation due to previous problems, we propose in this paper the following contributions:

1. An ontology-based history to facilitate the detection of external similarities: Ontologies allow us to describe the RDECSs (components, work process, component relationships ..., etc.) in an abstracted way than the formal model. Thus, we can efficiently detect the similarities between RDECSs with less computing time and resources, thank the ontology alignment method (Ougouti et al., 2017). Each model must be accompanied by a system ontology, which describes the system to be verified. The system ontology is aligned to the ontology-based history, which contains descriptions of already verified systems. The detected similarities allow reusing state spaces computed during previous verification.
2. Incremental construction of the accessibility graphs to deal with similarities: The verification of R-TNCES requires the verification of each TNCES that composes the R-TNCES model. In order to deal with similarities that TNCESs contain (similar control components), we construct the accessibility graph in an incremental way in two steps: (i) Fragmentation: During this step, we proceed to the decomposition of the R-TNCES models into a set of CCs. Then, we generate an accessibility graph for each different CC, while preserving semantics. (ii) Accessibility graph composition: Accessibility graphs recovered thanks to ontology alignment, and those computed during the fragmentation step are composed following an established composition plan based on priority order.
3. An adequate distributed cloud-based architecture to perform parallel executions for formal verification: This distributed architecture is composed of computation units organized in three hierarchical levels that are: Master, workers, and sub-workers. Data storage is ensured by Amazon simple storage service S3 (Murty, 2008).

The main objective of this paper is to propose a new formal verification method that improves the classical ones by controlling complexity. As a running example, we use the FESTO MPS benchmark system presented in (Koszewnik et al., 2016), to demonstrate the relevance of the proposed contributions. The obtained results are compared with different works. The comparison shows that the sate spaces generation is improved in terms of computed states and execution time (i.e., less complexity to compute state spaces). The remainder of the paper is organized as follows. Section 2 presents some required concepts. The distributed formal verification is presented in Section 3. The method and the proposed algorithms are presented in Section 4. Section 5 presents the evaluation of the proposed method. Finally, Section 6 concludes this paper and gives an overview about our future work.

2 BACKGROUND

In this section, we present required concepts to follow the rest of the paper.

2.1 Reconfigurable Timed Net Condition/Event System

R-TNCES represents an extension of TNCESs (Ramdani et al., 2018), based on Petri nets and control components CCs. R-TNCES is used for formal modeling and verification of RDECSS.

2.1.1 Formalization

An R-TNCES is defined in (Zhang et al., 2013) as a couple $RTN = (B,R)$, where R is the control module and B is the behavior module. B is a union of multi TNCES-based CC modules, represented by

$$B = (P;T;F;W;CN;EN;DC;V;Z_0) \qquad (1)$$

where, 1. P (resp, T) is a superset of places (resp, transitions), 2. $F \subseteq (P \times T) \cup (T \times P)$[1] is a superset of flow arcs. 3. W: $(P \times T) \cup (T \times P) \rightarrow \{0,1\}$ maps a weight to a flow arc, $W(x,y) > 0$ if $(x,y) \in F$, and $W(x,y) = 0$ otherwise, where $x,y \in P \cup T$, 4. $CN \subseteq (P \times T)$ (resp, $EN \subseteq (T \times T)$) is a superset of condition signals (resp, event signals), 5. $DC : F \cap (P \times T) \rightarrow \{[l_1,h_1],..,[l_{F\cap(P\times T)},h_{F\cap(P\times T)}]\}$ is a superset of time constraints on input arcs of transitions, where $\forall$ i$\in$ [1, $|F \cap (P \times T)|$], l_i, h_i $\in \mathbb{N}$ and $l_i < h_i$. 6. $V : T \rightarrow \wedge,\vee$ maps an event-processing mode (AND or OR) for every transition. 7. $Z_0 = (M_0,D_0)$, where $M_0 : P \rightarrow \{0,1\}$ is the initial marking, and $D_0 : P \rightarrow \{0\}$ is the initial clock position. R consists of a set of reconfiguration functions, formalized as follows. $R = \{r_1,..,r_n\}$ where: $r = (Cond,s,x)$ such that:1. $Cond \rightarrow$ {true, false} is the pre-condition of r, which means specific external instructions, gusty component failures, or the arrival of certain states. 2. $s : TN(^*r) \rightarrow TN(r^*)$ is the structure modification instruction such that $TN(^*r)$(resp. $TN(r^*)$) is the original (resp. target) TNCES before (resp. After) r application. 3. x : $last_{state}(TN(^*r)) \rightarrow initial_{state}(r^*)$ is the state processing function, where $last_{state}(TN(^*r))$ (resp. $initial_{state}(TN(r^*)))$ is the last (resp. the initial) state of $TN(^*r)$ (resp. $TN(r^*)$). The application of r makes a modification of the R-TNCES structure by the mean of instructions presented in Table 1. we denote by x a place, y a transition, CC a control component module, and "$+$" the AND of instructions to represent complex modification instructions.

Table 1: Fundamental structure modification instructions of an R-TNCES.

Instruction	Symbol
Add condition signals	$Cr(cn(x,y))$
Add event signals	$Cr(ev(y,y))$
Add control component	$Cr(CC)$
Delete condition signals	$De(cn(x,y))$
Delete event signals	$De(ev(y,y))$
Delete control component	$De(CC))$

2.1.2 R-TNCES Dynamics

The dynamics of R-TNCESs is represented by: 1. The reconfiguration between TNCESs in module behavior B, by applying a reconfiguration function r when its pre-condition is fulfilled. 2. The firing transition in each TNCES, depends on the rules of firing transitions in TNCESs and the chosen firing mode. Reconfiguration changes the system from a configuration to another, however, the initial and the new configurations can contain similarities.

Definition 1. Internal similarity is the property of sharing the same physical process between different configurations of a unique RDECS. Thus, the model contains similar parts. It is caused by the fact that a reconfiguration is rarely radical.

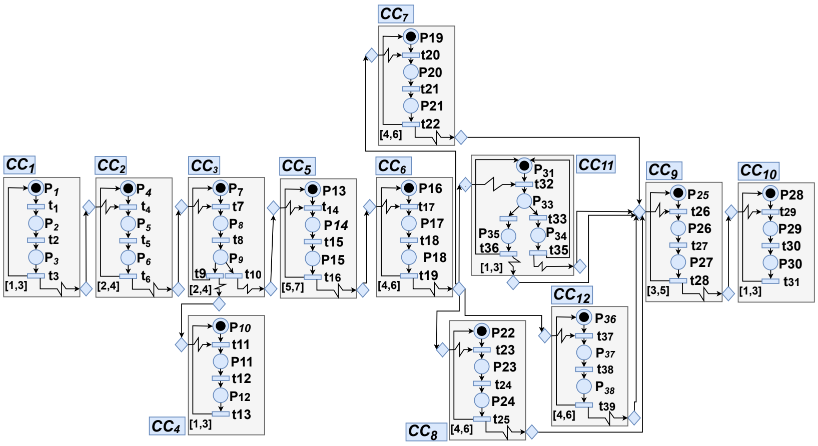
Definition 2. External similarity is the property of sharing the same physical process between configurations of two or several R-TNCESs. It is caused by the fact that some systems share same components or stations.

2.2 Production Systems: FESTO MPS & THREADING HOLE SYSTEM

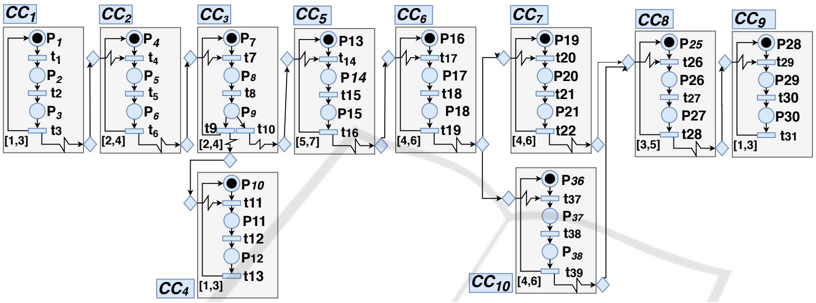
This subsection presents two production systems FESTO MPS and THREADIN HOLE SYSTEM.

---

[1] Cartesian product of two sets: $A \times B = \{(a,b)|a \in A, b \in B\}$.



(a) Behavior module of $RTN_{Sys_{01}}$.



(b) Behavior module of $RTN_{Sys_{02}}$.

Figure 1: Behavior module of $RTN_{Sys_{01}}$ and $RTN_{Sys_{02}}$.

2.2.1 FESTO MPS

FESTO MPS is a well-studied system for research and educational purposes which is defined and detailed in (Hafidi et al., 2018; Ramdani et al., 2018). It is composed of three units. The distribution contains a pneumatic feeder and a converter. It forwards cylindrical workpieces from the stack to the testing unit. The testing unit contains the detector, the elevator and the shift out cylinder. The detection unit performs checks on workpieces for height, material type and color. Workpieces that successfully pass this check are forwarded to the processing unit. The processing unit is composed of a rotating disk, drilling machines, a checker and an evacuator. The drilling of the workpieces is performed as the primary processing of this MPS. The result of the drilling operation is then checked by the checking machine and the workpieces is forwarded for further processing to another mechanical unit. FESTO MPS performs three production modes: (i) High mode: when $Driller_1$ and $Driller_2$ are both activated and ready to work simultaneously, (ii) Medium mode: when $Driller_1$ and $Driller_2$ are both activated but work sequentially, (iii) Light mode: when only one driller is activated at once. We denote $Light_i$, when $Driller_i$ / $i \in \{1,2\}$ works. FESTO MPS is modeled with an R-TNCES $RT_{FESTO}\{B_{FESTO}, R_{FESTO}\}$ such that: $B_{FESTO} = \{High, Medium, Light_1, Light_2\}$ is the behavior module where the combination of CC_s describes the system modes. As shown in Figure 1a. $R_{FESTO} = \{r_{H,L_1}, r_{H,L_2}, r_{H,M}, r_{M,H}, r_{M,L_2}, r_{L_1,L_2}\}$ is a set of different system reconfigurations. The set of control chains describing FESTO MPS control system is presented as follows: $Cchain_1 = CC_1, CC_2, CC_3, CC_4$,
$Cchain_2 = CC_1, CC_2, CC_3, CC_5, CC_6, CC_7, CC_9, CC_{10}$,
$Cchain_3 = CC_1, CC_2, CC_3, CC_5, CC_6, CC_8, CC_9, CC_{10}$,
$Cchain_4 = CC_1, CC_2, CC_3, CC_5, CC_6, CC_{11}, CC_9, CC_{10}$,
$Cchain_5 = CC_1, CC_2, CC_3, CC_5, CC_6, CC_{12}, CC_9, CC_{10}$.
This paper uses the description and the R-TNCES model of FESTO MPS for the construction of the proposed ontology as shown in Figure 4a.

2.2.2 THREADING HOLE SYSTEM

It is modeled using R-TNCES formalism. It is composed of three units: (i) the distribution unit, (ii) the testing unit, and (iii) the processing unit. The first two units are used in FESTO MPS. The processing unit is composed of a rotating disk, threading hole machine, a checker and an evacuator perform the threading of the workpiece holes as the primary processing task of the system. The result of the threading operation is then checked by the checking machine and the workpieces are forwarded for finally further processing to another mechanical unit. Behavior module B_{THS} and ontology O_{THS} are presented in Figure 1b and Figure 4b respectively on page 6. such that:

$B_{THS} = \{High, Light\}$ is the behavior module shown in Figure 1b. $R_{THS} = \{r_{H,L}, r_{H,L}\}$ is a set of different system reconfiguration.

The set of control chains describing THS control system is presented as follows:

$Cchain_1 = CC_1, CC_2, CC_3, CC_4,$
$Cchain_2 = CC_1, CC_2, CC_3, CC_5, CC_6, CC_7, CC_8, CC_9,$
$Cchain_3 = CC_1, CC_2, CC_3, CC_5, CC_6, CC_{10}, CC_8, CC_9.$

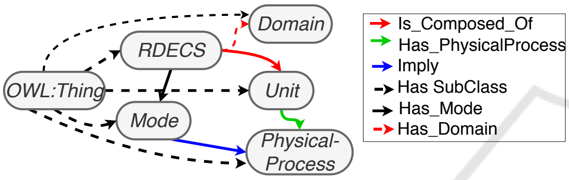


Figure 2: Genric ontology.

2.3 Ontology Concept

As defined in (Ougouti et al., 2018) an ontology is an explicit description of concepts or classes in a certain domain that constitutes a knowledge base. An ontology is defined mathematically as quadruple $O = (C, S, Re, I)$ where: 1. $C = c_1, .., c_m$ is a set of concepts that refer to a real world objects. 2. $S = s_1, .., s_n$ is a set of properties that refer to a property of a concept, which is a value of a simple type such as Integer, String or Date. 3. $Re = Re_1, .., Re_p$ is a set of relationships defined between concepts. 4. $I = i_1, .., i_q$, where each i_w is an instance of some concept $c_x \in C$. It include a value for every property s_y associated to c_x or its ancestors. An ontology can be presented graphically as a formed graph $O = G(C, E)$ where C is a set of concepts linked by a set of directed edges E which specifies concept relations. The function y defines the type of edges, i.e., $y : E \rightarrow T$ where T is the set of possible edge types (transitivity, symmetry and reflexivity). We define an generic ontology $Gen = (C, S, Re, I)$, which is instantiated to model the verified RDECS. Table 2 shows the defined concepts $\in C$ and their properties include in S. Figure 2 shows the relations $\in Re$.

3 NEW STATE SPACE GENERATION METHOD

We present in this section the proposed method for state space generation during formal verification of R-TNCESs. Using our proposed approach, we minimize temporal complexity by proposing a distributed architecture on cloud server (Hayes, 2008). Thus, we improve model-checking of reconfigurable systems and make it more efficient.

3.1 Motivation

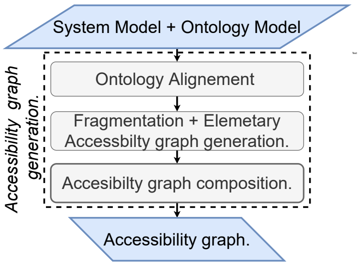


Figure 3: Global idea for State Space Generation.

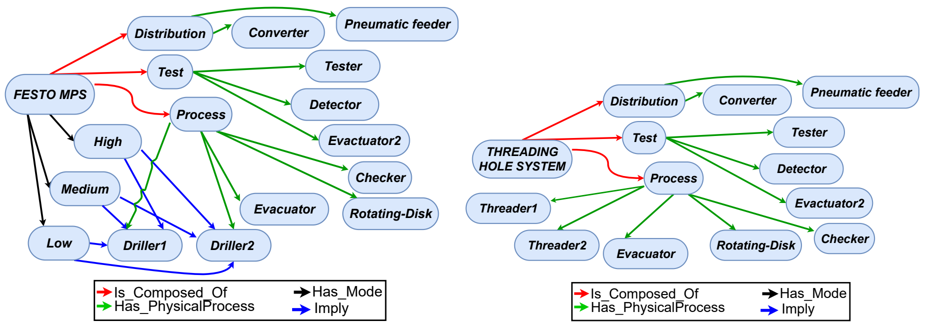
The correctness of RDECSs can be ensured by a formal verification. The exploration of the state space is widely used for analyzing models formalized with R-TNCES, or related formalisms. The complexity of R-TNCES makes the verification task complex, because of combinatorial growth of the state space according to the model size. The verification of an R-TNCES requires the checking of each configuration, namely each TNCES. TNCESs that describe the configurations often present similarities which implies redundant calculation during checking of these systems. Thus we propose an adequate approach that avoids many repetitive computations. To ensure this objective, this paper proposes a new method where verification is executed in a distributed architecture to control R-TNCESs complexity. The formal verification is performed through the following tasks: fragmentation, ontology alignment and accessibility graph composition. Figure 3 presents the main steps of the proposed method.

3.2 Formalization

In this section, we present accessibility graph generation steps according to our proposed method.

3.2.1 Ontology Alignment

According to the definition presented in (Ougouti et al., 2017), aligning two ontologies is to find a set of



(a) Ontology $Osys_{01}$ (b) Ontology $Osys_{02}$

Figure 4: Ontologies that describe Sys_{01} and Sys_{02}.

Table 2: Generic ontology which modeled RDECSs.

Concepts $\in C$	RDECS	Domain	Unit	Physical Process	Mode
Properties $\in S$	Id: String Name: String Description: Text	Id: String Name: String	Id: String Name: String Description: Text	Id: String Name: String Description: Text Control chain: String	Id: String Name: String Description: Text

correspondences, where each correspondence is described by: a unique identifier Id, the concept $c_i \in O_1$, the concept $c_j \in O_2$ and σ_{ij} the degree of similarity between c_i and c_j evaluated in the interval [0,1]. Formally, it is to find $|O_1| \times |O_2|$ correspondences $(Id_{ij}, c_i, c_j, \sigma_{ij})$. A threshold τ is defined and compared with σ_{ij}. The correspondence is established only if $\sigma_{ij} > \tau$. Global similarity σ_{ij} is computed through the following steps:

1. Compute semantic similarity by comparing concepts neighbors using Tversky measurement: $Tm_{ij} = \frac{|(n_i \cap n_j)|}{|(n_i \cap n_j)| + \alpha|(n_i - n_j)| + \beta|(n_j - n_i)|}$, where:
   n_i (resp. n_j): Neighbor set of c_i (resp. c_j).
   $n_i \cap n_j$: Number of common neighbors between c_i and c_j.
   $n_i - n_j$ (resp. $n_j - n_i$): Number of neighbors that exist $\in n_i$ and $\notin n_j$ (resp. $\in n_j$ and $\notin n_i$).
2. Compute lexical similarity, a weighted sum of *normalized Leveinstein* and *n-gram* similarities: $SimLex_{ij} = \alpha * LevNorm(i,j) + \beta * g(i,j)$.
3. Compute partial similarity of concept descriptions using the cosinus function:
   $SimDes_{(A,B)} = cos(\theta) = \frac{A.B}{|A||B|} = \frac{\sum A \times B}{\sqrt{\sum A^2} \times \sqrt{\sum B^2}}$.
4. Compute linguistic similarity by combining *SimLex* and *SimDes*: $SimLing_{(i,j)} = \alpha SimLex_{(i,j)} + \beta SimDes_{(i,j)}$. with $\alpha = 0.4$ and $\beta = 0.6$.
5. Calculate the global similarity which is a weighted sum of linguistic and semantic similarity: $\sigma_{ij} = \alpha SimLing_{ij} + \beta Tm_{ij}$, with $\alpha = \beta = 0.5$.

Running Example 1. Let O_{FESTO} and O_{THD} two ontologies, which describe the production systems presented in subsection 2.2. Given two concepts $Process \in O_{FESTO}$ and $Process \in O_{THS}$. Table 3 shows an application of ontology alignment where, we compute: i) lexical similarity, which concerns the concepts property "Name", ii) semantic similarity, which concerns concepts neighbors, iii) description similarity, which concerns the concepts property "Description", iv) linguistic similarity, which is the combination of lexical and description similarities, and v) global similarity by combining the said similarities. $\sigma(Process, Process) = 0.61$ (low value) and the threshold $\tau = 0.8$ (fixed). We conclude that $Process \in O_{Sys_{FESTO}}$ and $Process \in O_{Sys_{THS}}$ are non-similar. Thus, the non-similar and similar parts are efficiently distinguished and redundant calculations are avoided.

3.2.2 Fragmentation

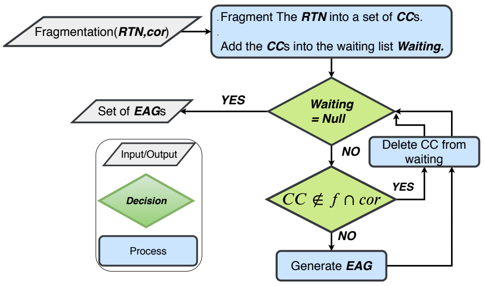


Figure 5: Operative steps of the fragmentation function where $Waiting$ is the list of CCs to be computed.

Fragmentation consists on decomposing an R-TNCES into a set of CC and generating elementary accessibility graph $EAGs$ for CCs that are not concerned by the correspondences computed in the previous step.

Table 3: Application of ontology alignment on running example where $Concept_1 \in O_{FESTO}$ and $Concept_2 \in O_{THS}$.

Properties / Concepts	Name	Neighbors	Descriptions	
Concept 1	Process	{Driller1, Driller2, checker, Evacuator Rotating disk}	Workpieces that pass the test unit successfully are forwarded to the rotating disk of the processing unit, where the drilling of workpieces is done. It is assumed that in this work there exist two drilling machines Drill1 and Drill2 to drill workpieces. The result of the drilling operation is next checked by a checker and finally the finished product is removed from the system by an evacuator.	
Concept 2	Process	{Threader1, Threader2, checker, Evacuator Rotating disk}	Workpieces are received by rotating disk of the process unit, where the threading of workpieces is done. It is assumed that in this work there exist one threading hole machine to thread workpieces. The result of the is next checked by a and finally the finished product is removed from the system by an evacuator.	
Similarities	$SimLex = 1$	$Tm = 0.46$	$Simdes = 0.6$	$SimLing = 0.76$ $\sigma = 0.61$

Running Example 2. To show the application of fragmentation, we consider production systems presented in Subsection 2.2. They are modeled by RT_{FESTO} (to be verified) and RT_{THS} (already verified). Let cor be a set of correspondences computed during alignment of O_{FESTO} and O_{THS}. Table 4 and Figure 6 show application of fragmentation on RT_{FESTO}. It runs in two steps: 1. decomposing RT_{FESTO} into a set of CC $f = \{CC_1, .., CC_{12}\}$, and 2. computing elementary accessibility graphs EAGs of each $CC \notin f \cap cor$. During fragmentation, CCs synchronization transitions are stored for reuse when composing the accessibility graph AG. Real RDECSs encompass millions of transitions, which increases accessibility graph generation complexity. Fragmentation allows us to control complexity. Moreover, it allows us to deal with internal similarities.

Table 4: Application of fragmentation on FESTO MPS.

System	FESTO MPS
f	$\{CC_1, .., CC_{12}\}$
cor	$\{CC_1, CC_2, CC_3, CC_4, CC_5, CC_6, CC_{10}\}$
$EAGs$	$EAG_{CC_7}, EAG_{CC_8}, EAG_{CC_9}, EAG_{CC_{10}}, EAG_{CC_{11}}, EAG_{CC_{12}}$

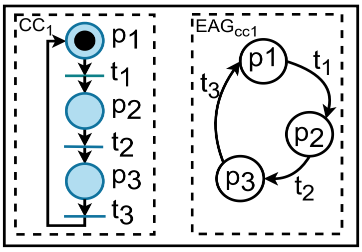


Figure 6: CC_1 with its elementary accessibility graph.

3.2.3 Planning

We set up a priority order for accessibility graph composition. Let RTN be a system modeled by R-TNCES and described by ontology O_{sys}. We extract from O_{sys} control chains $Cchains$. $Cchains$ are then en-queued to a queue Q depending on their length such as the smallest one is en-queued firstly.

Running Example 3. By using the behavior module B of RTN_{FESTO}, the composition plan to be followed for AG_{FESTO} generation for test failure case described by C_{chain_1} is presented as follows:
$EAG_{CC_1} \times EAG_{CC_2} > PAG_{CC_{12}} \times CC_3 > PAG_{123} \times CC_4$.

3.2.4 Accessibility Graph Composition

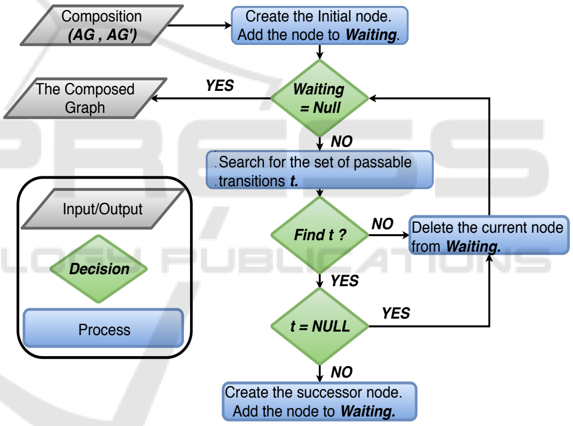


Figure 7: Operative steps of the graph composition function, where ***t*** is the set of the fixed passable transition and ***Waiting*** is the list of nodes to be computed.

Full accessibility graph AG is computed by composing $EAGs$ computed during fragmentation step and partial accessibility graphs $PAGs$ retrieved during ontology alignment step as shown in Figure 7. The composition is done according to the established plan.

Running Example 4. During AG_{FESTO} generation, several composition of EAGs are executed. Indeed, we run $Composition(EAG_{CC_1}, EAG_{CC_2})$ function to obtain PAG_{12} shown in Figure 8. It proceeds as follows:

1. Creates initial state S_0 by concatenating initial states S'_0 and S''_0 of both EAG_{CC_1} and EAG_{CC_2}, 2. searches the set of enabled transitions from S'_0 and S''_0, and 3. checks whether the transition t is a common transition. If yes, then we create a new state S_1

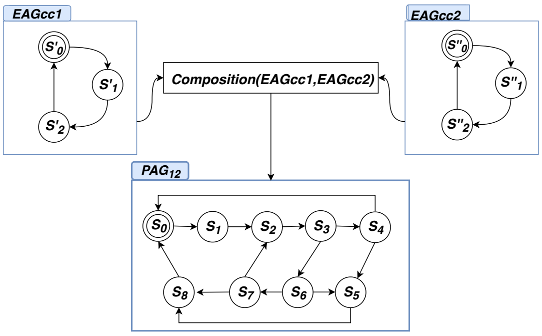


Figure 8: Compostion of EAGcc1 & EAGcc2.

by concatenating the current target states from S'_0 and S''_0. Otherwise, if t belongs only to EAG_{CC_1}, then a new state S_1 is obtained by concatenating the current state S''_0 from EAG_{CC_2} and the current target state S'_1 from EAG_{CC_1} and vice versa.

We repeat these steps for the remaining states until we get the whole state space.

4 DISTRIBUTED CLOUD-BASED STATE SPACE GENERATION

This section presents Cloud-based distributed architecture and how to perform formal verification on it.

4.1 Distributed Architecture for State Space Generation

In this subsection, we present the proposed hierarchical and distributed architectures shown in Figure 9. The idea that motivates the development of this architecture is to increase computation power and storage availability. It is composed of computational and storage resources. To develop the architecture shown in Figure 9 we need the following units.

- Computational Units: Execute tasks defined in subsection 3.2 by means of M+n machines where:

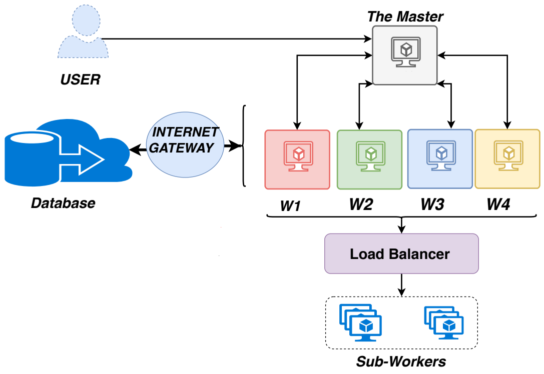


Figure 9: Distributed architecture for formal verification.

  (i) M represents the number of machines (i.e., 5 machines in our approach). The set of machines are composed of a master and four workers $W_1, ..., W_4$ that have specific tasks. (ii) n is the number of sub-workers that execute the high complex tasks (i.e., *EAGs* generation and *PAGs* composition). n depends on system size.

- Storage Unit: represents the allocated cloud database that stores domain ontologies, *EAGs* temporary and *PAGs* permanently.

4.2 Distributed State Space Generation

This subsection presents the process of distributed Formal verification on a cloud based architecture.

Running Example 5. The user sends a verification request $req(R_{FESTO}: R\text{-}TNCES, O_{FESTO}: Ontology)$. The master ensures tasks coordination by receiving the verification request and sending R_{FESTO} and O_{FESTO} to workers to carry out their tasks as follows.

1. sending simultaneously ontology O_{FESTO} to workers W_1, W_4 and R_{FESTO} to worker W_2,
2. waiting signals from W_1 and W_2 and to receive the composition plan from W_4 to forward it to W_3.
3. waiting signal from W_3 to allow beginning ontology fusion by W_1.

W_1 has two main tasks: (i) Ontology alignment to extract correspondences and (ii) Ontology fusion to update domain ontology-based history, we merge O_{FESTO} AND O_D.
W_2: At the reception of R_{FESTO}, it proceeds to the fragmentation, sends CCs to sub-workers after applying a load balancer algorithm and sends a signal to master which announces the end of these two tasks: fragmentation and generation of *EAGs*.
W_3 receives the composition plan and collects the elements that it needs from the database for the *AG* composition. Finally, it sends a signal to master which announces the end of its task.
W_4 is responsible for planning compositional order for full accessibility graph generation. It extracts the control chains concepts from O_{FESTO}. Then the plan is sent to the master.

4.3 Implementation

In this subsection, we present the main algorithms used in our approach.

Algorithm 1 describes the fragmentation task. It decomposes the R-TNCES in a set of CCs and generates their accessibility graphs EAGs.

Algorithm 2 describes the steps for the full accessibility graph composition AG. It composes the accessibility graphs recovered thanks to the ontology alignment and the ones computed during fragmentation to return the full accessibility graph of the verified model.

Algorithm 1: Fragmentation.

```
Input: $RTN$: R-TNCES; $TN_0$: TNCES;
Output: $S\_EAG$: Set of elementary accessibility graphs;
for int $i = 0$ to $|\sum TN|$ do
    for each $CC \in TN$ do
        if ( !Tagged (CC)) then
            $Insert(S\_EAG, Geneate\_State\_Space(CC))$;
            tag(CC);
        end
    end
end
return $S\_EAG$
```

Algorithm 2: State Space Composition.

```
Input: $S\_AG$: Set of accessibility graphs(EAG, PAG) ;$\sum CChain$: Set of Cchains ;
Output: $AG$: Set Accessibility graphs;
for int $i = 0$ to $|\sum CChain|$ do
    $AG \leftarrow EAG_{CC_i^0}$;
    for int $j = 0$ to $|\sum CC_i|$ do
        $AG \leftarrow Compose(AG, EAG_{CC_i^j})$;
    end
end
return $AG$
```

4.4 Complexity of Distributed State Space Generation

The verification is based on three main functions: (i) the ontology alignment, (ii) the fragmentation, and (iii) the *EAG/PAGs* composition. The ontology alignment complexity on this scale is always polynomial, thus we focus on the two other function presented respectively in Algorithm 1 and 2. As mentioned in (Zhang et al., 2013), TNCES verification complexity is expressed by $O(e^t)$ where t is the number of transition, in our case, we use it for each CC of the verified R-TNCES. For an R-TNCES with $TN = |B|$ the number of TNCESs composing the verified R-TNCES and C the average number of CCs that every TNCES contains, The complexity of Algorithm 1 is $O(TN \times C \times e^t)$. For a composed graph with n' the number of nodes computed by the composition graph function and j the average number of the enabled transitions from each state, Algorithm 2 complexity is expressed by $(n' \times j)$. Thus, verification time complexity is: $O((TN \times C \times e^t) + (n' \times j))$. Therefore, our method complexity is expressed by

$O(\max O(TN \times C \times e^t), O(n' \times j)) = O(TN \times C \times e^t)$.

The complexity of methods presented in (Zhang et al., 2013; Hafidi et al., 2018) is $O(e^m \times TN)$ with $m \times TN = TN \times C \times t$. Thus, to assert that our complexity is better, we have to prove that: $O((TN \times C \times e^t) < O((TN \times e^m)$, which is intuitively correct.

5 EVALUATION

The performance of the proposed verification method is evaluated in this section. We make a comparison between the proposed method, that uses a distributed tool to compute accessibility graphs, and the method reported in (Hafidi et al., 2018) that uses Rec-AG tool. Then we proceed to different evaluations in large scale systems by considering different similarities. The external similarity rate of R-TNCES R_1 with descriptive ontology O_L is given by the following formula.

$$ExternalSimilarity(R_1) = \left(\frac{AlignedConcepts(O_L)}{Concepts(O_L)}\right) \quad (2)$$

where, (i) $AlignedConcepts(O_L)$ returns the number of similar concepts between O_L and the related domain ontology O_D, (ii) $Concepts(O_L)$ returns the total number of concepts that O_L contains. The internal similarity rate is given by the adapted method used in (Hafidi et al., 2018) as follows.

$$InternalSimilarity(R_1) = \left(\frac{Max(\{SimCC(TN_i, TN_j)\}_{i,j \in 0\ldots(n-1)\ and\ i<j})}{Max(NumberOfCC(TN_k))}\right) \quad (3)$$

where, (i) *SimCC*(TN_i, TN_j) is the function that returns the number of similar control components between two TNCESs, (ii) $NumberOfCC()$ takes a TNCES and returns its number of control components, and (iii) *Max*() returns the maximum among a set of natural numbers. We define three degrees of Internal Similarity (resp, External Similarity): High, Medium and low where, *InternalSimilarity* (resp, *ExternalSimilarity*) is 50%-100%, 20%-50% and 0%-20%.

5.1 Evaluation in Large Scale Systems Considering External Similarity

Figure 10 describes the verification result of an R-TNCES model by considering three levels of external

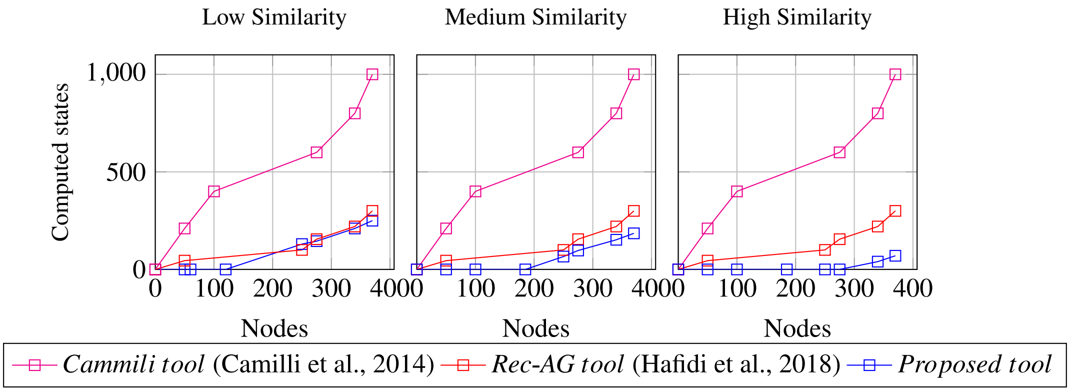


Figure 10: Proposed verification in large scale systems considering external similarity.

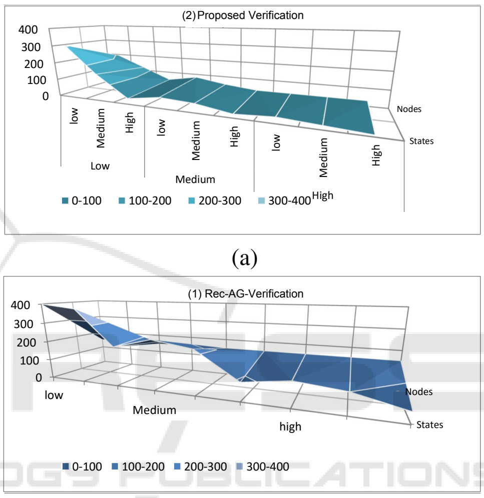
similarity. The model is composed of three TNCESs represented by three parallel control chains of equal length, with $Complexity(CC_{ij}) = 3,\ i \in 1...100$ and $j \in 1...3$ (i.e., each CC contains 3 nodes). By analyzing the plots in Figure 10, we notice that:

In the case of low external similarities, the number of states computed using the proposed method and the one proposed in (Hafidi et al., 2018) in its best case (i.e., in the case of a high internal similarity rate) becomes nearly equal with the ascent of the number of system nodes. It is explained by the fact that the difference in the number of nodes to explore is minimal and becomes non-significant when the system is larger. Nevertheless, low similarity must be exploited because it improves the results in both cases of medium and high internal similarity.

In the case of high and medium external similarities: the proposed method takes advantage of those presented in (Camilli et al., 2014) and (Hafidi et al., 2018). It is explained by the fact that the number of nodes to explore is reduced. Thanks to the external similarity that allows us to eliminate redundancies. While in the three cases, the proposed method presents better results than the one used in (Camilli et al., 2014), which generates AGs via the classical methods. The proposed method can reduce calculations by more than 50%, depending on model size and similarity rates. This represents the main gain of the paper.

5.2 Evaluation in Large Scale Systems by Considering External and Internal Similarities

The surfaces in Figure 11 describe the results of both the proposed method and the one used in (Hafidi et al., 2018), by using three factors: External similarities, internal similarity and nodes to be explored for a state generation. In their worst case (i.e., $InternalSimilarity = ExternalSimilarity = 0\%$) performance of both methodologies presents limits, with same results using the method reported in (Zhang et al., 2013). However, in the remaining cases, the proposed method always presents better results according to similarity rates. It performs best with: (i) Less computed states, thanks to the external source of partial graphs and elimination of internal redundancies, and (ii) less nodes to be explored for state space generation thus less complexity to generate a state, thanks to the incremental way used when composing the accessibility graph.



(a)

(b)

Figure 11: Proposed verification in large scale systems considering external and internal similarities.

6 CONCLUSION

This paper deals with formal verification of RDECSs that we model with R-TNCES. The proposed method aims to improve the state space generation step by using a distributed architecture. We developed a distributed architecture with three hierarchical levels (Master, worker and sub-worker) and a cloud-based-storage (Amazon Simple Storage S3 (Murty, 2008)). It allows us to increase computational power, data availability and to perform parallel execution. The proposed improvement incorporates ontologies for RDECSs verification. We set up an ontology-based history, which allows us to detect external similarities thanks to an ontology alignment. Thus, we avoid many redundant calculation. In order to deal with internal similarities, we introduce modularity concept by affecting specific tasks to each unit of our architecture, including fragmentation and accessibility graph composition, which allow us to deal with RDECSs fragment by fragment and to construct incrementally accessibility graphs. An evaluation is realized and experimental results are reported. The results prove the relevance of the developed architecture and the improvement of state space generation. Nevertheless, by comparing our work with other verification methods, we identified cases, that provide results which tend toward the works reported in (Zhang et al., 2013) and (Hafidi et al., 2018). Indeed, our method provides less benefits in case of low internal or external similarities. However, despite the minor gain when internal similarity is low, it is important to consider this case the ontology-based history enrichment. Future works will: 1. Deploying the distributed architecture in Amazon Elastic Compute Cloud (EC2) (Murty, 2008). 2. Optimizing the state space analyzing step for RDECSs formal verification. 3. Extending the proposed tool to support other formalism that models RDECSs.

REFERENCES

Ben Salem, M. O., Mosbahi, O., Khalgui, M., Jlalia, Z., Frey, G., and Smida, M. (2017). Brometh: Methodology to design safe reconfigurable medical robotic systems. *The International Journal of Medical Robotics and Computer Assisted Surgery*, 13(3):e1786.

Camilli, M., Bellettini, C., Capra, L., and Monga, M. (2014). Ctl model checking in the cloud using mapreduce. In *Symbolic and Numeric Algorithms for Scientific Computing (SYNASC), 2014 16th International Symposium on*, pages 333–340. IEEE.

Gadelha, M. Y., Ismail, H. I., and Cordeiro, L. C. (2017). Handling loops in bounded model checking of c programs via k-induction. *International Journal on Software Tools for Technology Transfer*, 19(1):97–114.

Hafidi, Y., Kahloul, L., Khalgui, M., Li, Z., Alnowibet, K., and Qu, T. (2018). On methodology for the verification of reconfigurable timed net condition/event systems. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, (99):1–15.

Hayes, B. (2008). Cloud computing. *Communications of the ACM*, 51(7):9–11.

Khalgui, M., Mosbahi, O., Li, Z., and Hanisch, H.-M. (2011). Reconfiguration of distributed embedded-control systems. *IEEE/ASME Transactions on Mechatronics*, 16(4):684–694.

Koszewnik, A., Nartowicz, T., and Pawłuszewicz, E. (2016). Fractional order controller to control pump in festo mps® pa compact workstation. In *2016 17th International Carpathian Control Conference (ICCC)*, pages 364–367. IEEE.

Murty, J. (2008). *Programming amazon web services: S3, EC2, SQS, FPS, and SimpleDB*. " O'Reilly Media, Inc.".

Ougouti, N. S., Belbachir, H., and Amghar, Y. (2017). Semantic mediation in medpeer: An ontology-based heterogeneous data sources integration system. *International Journal of Information Technology and Web Engineering (IJITWE)*, 12(1):1–18.

Ougouti, N. S., Belbachir, H., and Amghar, Y. (2018). Proposition of a new ontology-based p2p system for semantic integration of heterogeneous data sources. In *Handbook of Research on Contemporary Perspectives on Web-Based Systems*, pages 240–270. IGI Global.

Padberg, J. and Kahloul, L. (2018). Overview of reconfigurable petri nets. In *Graph Transformation, Specifications, and Nets*, pages 201–222. Springer.

Ramdani, M., Kahloul, L., and Khalgui, M. (2018). Automatic properties classification approach for guiding the verification of complex reconfigurable systems. In *ICSOFT*, pages 625–632.

Souri, A., Rahmani, A. M., Navimipour, N. J., and Rezaei, R. (2019). A symbolic model checking approach in formal verification of distributed systems. *Human-centric Computing and Information Sciences*, 9(1):4.

Valmari, A. (1996). The state explosion problem. In *Advanced Course on Petri Nets*, pages 429–528. Springer.

Zhang, J., Khalgui, M., Li, Z., Mosbahi, O., and Al-Ahmari, A. M. (2013). R-tnces: a novel formalism for reconfigurable discrete event control systems. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 43(4):757–772.