Automatic Properties Classification Approach for Guiding the Verification of Complex Reconfigurable Systems

Mohamed Ramdani[1,3], Laid Kahloul[2] and Mohamed Khalgui[3]
[1]*University of Tunis El Manar, Tunis, Tunisia*
[2]*LINFI Laboratory, Computer Science Department, Biskra University, Algeria*
[3]*University of Carthage, National Institute of Applied Sciences and Technology, Tunis 1080, Tunisia*

Keywords: Discrete-event System, Reconfiguration, R-TNCES, Computation Tree Logic, CTL, Automatic Generation, Formal Verification, Model-checking.

Abstract: This paper deals with reconfigurable discrete event/control systems (RDECSs) that dynamically change their structures due to external changes in environment or user requirements. Reconfigurable Timed Net Condition/Event Systems (R-TNCESs) are proposed as an extension of the Petri nets formalism for the optimal functional and temporal specification of RDECSs. The correct design of these systems continues to challenge experts in both academia and industry, since bugs not covered early can be extremely expensive at the final deployment. The classic model-checking using computation tree logic (CTL) and its extensions (extended CTL, Timed CTL, etc) produces a large number of properties, possibly redundant, to be verified in a complex R-TNCES. To control the complexity and to reduce the verification time, a reduction technique of properties is proposed. The novelty consists in the classification of CTL properties according to their semantic relationships for guiding an efficient verification. An algorithm is proposed for the automatic classification of CTL properties before starting model-checking process. A case study is exploited to illustrate the impact of using this technique. The current results show the benefits of the paper's contribution.

1 INTRODUCTION

Discrete event/control systems (DECSs) are asynchronous and non-deterministic systems which have a discrete nature of states and can change their state, in time, by the occurrence of external/internal events. The control and the coordination of physical devices that compose a DECS become more complex and more critical in large-scale systems. The development of auto-control systems and the integration of reconfigurability aspect in DECSs accelerate the appearance of innovative systems called reconfigurable discrete event/control systems (RDECSs). This kind of systems can work simultaneously on various conditions: concurrency, control, communication, etc. Such systems include manufacturing, real time systems(Ghribi et al., 2018), (Lakhdhar et al., 2018), mobile systems (Idriss et al., 2017), and wireless sensor networks (Grichi et al., 2017), intelligent approaches (Meskina et al., 2017), intelligent applications, and communication systems (Karoui et al., 2017) etc.

RDECSs represent a class of systems that adapt their behavior to any evolution in their environment, dynamically and timely. In reconfigurable systems, any adaptation to the external changes or any response to user requirements is considered as a reconfiguration scenario (Zhang et al., 2015). The reconfiguration can be applied statically at design time or dynamically at run-time (Khalgui and Hanisch, 2011). The reconfiguration is a set of changes in the structure, functionality, and control algorithms which can cause dysfunctions in the system. Therefore, every newly applied reconfiguration of the current state of a system must produce a new state which preserves always the required properties. In spite of its complexity, the verification task of RDECSs is a primordial one. Formal verification methods are applied because they use mathematical specifications and the exhaustive testing. To ensure the correctness of a system by model-checking method, the formal model of the system behavior/structure and the specification of the related properties in a temporal logic are required to be involved in a verification/debugging cycle. The functional and temporal properties are specified by one of the existing temporal logics, such as: Computation tree logic (CTL), extended computation

Ramdani, M., Kahloul, L. and Khalgui, M.
Automatic Properties Classification Approach for Guiding the Verification of Complex Reconfigurable Systems.
DOI: 10.5220/0006863005910598
In *Proceedings of the 13th International Conference on Software Technologies (ICSOFT 2018)*, pages 591-598
ISBN: 978-989-758-320-9

tree logic (eCTL), and timed computation tree logic (TCTL).

To cope with the reconfigurability, Petri nets has been extended and developed by many works (Padberg and Kahloul, 2018). Timed net condition/event systems (TNCESs) formalism is a modular extension of Petri nets presented in (Khalgui and Hanisch, 2011). In order to deal with reconfigurable systems, TNCESs formalism was extended towards R-TNCESs in (Zhang et al., 2013). The R-TNCES can capture the complex characteristics of an RDECS and facilitate the good understanding of the physical process of the system by the modular graphical representation. The layer-by-layer verification of R-TNCES proposed in (Zhang et al., 2013) is applied directly to the set of TNCESs layers without any analysis of relationships between behaviors to be checked. This direct and blind checking, of the set of properties, makes the verification process too complicated and harder in the case of a complicated reconfigurable system. This complication is justified by the system complexity and the high number of properties to be checked by the designer.

The objective of this work is to improve the verification process of RDECSs. In this paper, we propose to classify automatically properties and to introduce a priority in verification in order to control the high number of properties to be verified in such systems. The proposed approach takes as input a set of formulas, and gives as output an automatic classification according to defined relations between behaviors. Thus, the goal of this research is to automatically range, filter and reorganize the CTL and its extension formulas for a complex verification of an R-TNCES. We argue that using this arrangement process, one can reduce the number of properties to be verified. The purpose of this paper is to identify, automatically, relationships between formulas. These relationships will be used by an algorithm to make the suitable order in which properties will be checked. In fact, we can recapitulate the whole process in four steps, as follows. (i) A syntactic sorting of all properties according to their CTL class, (ii) A grouping of properties by creating sub-sets of properties according to their configuration and their goals form (simple/composed), (iii) An automatic generation by a simple research in the above sub-sets, and (iv) A guiding of verification according to the established relationships.

To demonstrate the feasibility of the proposed approach, a formal case study is provided. We detail the application of the proposed verification process step by step to show how the guiding method works from the syntactic sorting to the guidance arrangement of verification. To illustrate the gain of this task as a preprocessing before the model-checking process, we give some simulations. The results show that the proposed technique reduces effectively the number of properties to be verified in reconfigurable systems. The organization of this paper is as follows. Section II gives backgrounds. Section III presents the formalization and the automatic generation based algorithm. The papers contribution is applied to a formal case study in Section IV. Finally, Section V concludes this paper and describes future works.

2 BACKGROUND

In this section, we present the concerned formalism and some related concepts of CTL and its extensions which will be used in this paper.

R-TNCES is a modeling formalism based on Petri nets and control components *CC*. A component is a logical software unit (Khalgui et al., 2011), which represents the data-flues and actions of sensors/actuators (algorithms, extraction or activation). Every *CC* resumes the physical process in three actions: activation, working, and termination. An R-TNCES *RTN* is a composed structure that contains two components, thus $RTN = (B,R)$. The first component B is the behavior module, composed of n configurations $B = (C_1,\ldots,C_n)$, each one is a TNCES, possibly redundant. The second component R is the control module which represents the set of reconfiguration functions $R = (r_1,\ldots,r_m)$ with $n,m \in \mathbb{N}$. Formally, the behavior module of an R-TNCES is a place/transition net model specified as proposed in (Zhang et al., 2013).

$$B = (\mathbb{P},\mathbb{T},\mathbb{F},W,\mathbb{CN},\mathbb{EN},\mathbb{DC},V,Z_0) \qquad (1)$$

where, $\mathbb{P}$ (resp. $\mathbb{T}$) is a superset of places (resp. transitions), $\mathbb{F}$ is a superset of arcs, $W : (P \times T) \cup (T \times P) \rightarrow \{0,1\}$ maps a weight to a flow arc, $\mathbb{CN}$ (resp. $\mathbb{EN}$) is a superset of condition signals (resp.event signals), $\mathbb{DC}$ is a superset of clocks on output arcs, V: $T \rightarrow \{AND, OR\}$ maps an event processing mode for every transition, and $Z_0 = Z_{0i} = (M_0, D_0)$, where M_0 is the initial marking, and D_0 is the initial clock position. An example of R-TNCES will be detailed in section 4.1. The dynamic and the structure modification instructions of this formalism are well detailed in (Zhang et al., 2013).

According to the way in which the system reacts due to the environment changes and the user requirements, any adaptation is assumed as a reconfiguration scenario. R-TNCESs include many reconfiguration scenarios in one model. Each reconfiguration represents a switch between two TNCESs, possibly redun-

dant. Since behavior can be repeated in many TNCESs, its verification can be quantified from an execution to another execution. The redundancy between TNCESs creates relationships through different behaviors which may express dominance, equivalence, and composition of execution order.

Definition 1. A dominance behavior is an execution which respects the order of precedence between physical processes included in the same configuration or in different ones.

Definition 2. An equivalent behavior is an execution which includes the same physical process in different configurations.

Definition 3. A composition behavior is a global execution composed of many physical processes which are in the same configuration or in different ones.

The model-checking of R-TNCESs is an automatic technique for the verification of properties correctness of finite-state systems based on their reachability graph. The properties are specified using the temporal logic CTL or its extensions. Computation tree logic (CTL) (Baier et al., 2008) and extended computation tree logic (eCTL) (Starke and Roch, 2002) are used to specify the functional properties, and Timed CTL (Boucheneb et al., 2009) is used to specify the reasoning about the truth values of the time constraint of an RDECS over a time interval. Using CTL and its extensions, the time is not explicitly expressed, but we have the possibility of saying that a property will frequently/infrequently be verified or will never be verified. CTL offers facilities for the specification of properties that must be fulfilled by the system, like safety, liveness, reachability, etc.

We denote that every two properties sharing similar goals or a part of goals can be involved in a bi-part relationship. The bi-part relationship can be: (i) Equivalence relationship which means that two similar properties represent an equivalent behavior which concerns two different configurations. (ii) Composition relationship which means an argumentative relation of two properties that have a conjunction factor representing a composition behavior (precedence of goals), and (iii) a dominance relationship between two properties means that the final result of the whole formula depends only on the dominant part of this formula because of the dominant behavior of the physical process to be verified.

Definition 4. A goal means a simple locality in the model which can be a place, a transition, a state or a formula written in CTL or one of its extensions.

The classical verification ignores the redundancy effect on behaviors that increase the number of properties to be verified and complicates the verification process. Therefore, the classical verification is incapable of controlling the number of properties to be verified and so that incapable to identify the relationships between them. This work tries to give a verification strategy to avoid this problem.

3 PROPERTIES CLASSIFICATION

Considering the above drawbacks of formalisms and methods in reconfigurable systems verification process, the verification time increases as well as the number of properties to be verified increases. We propose in this section an optimal strategy for the guidance of CTL formulas verification. Using our proposed approach, we minimize the validation time by reducing the number of properties to be verified, and we improve model-checking of reconfigurable systems and make it more efficient by automatizing the classification and properties order generation. The **reduction** is obtained by extracting the possible relationships. Then, expressing those relations using a set of new proposed operators. Figure 1 summarizes the global idea of the automatic generation of relation order between the CTL formulas.

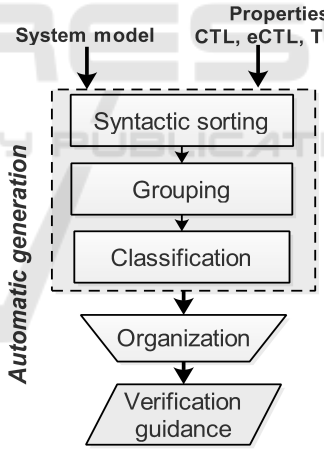

Figure 1: Global idea of the automatic generation process.

3.1 Formalization

The syntax of new proposed operators is presented in this section. Let us denote by ϕ^{TL} a formula written in one of these logics. The classification operation introduces a set of new operators to express relationships between every couple of properties.

- Dom^{c_i}: Internal dominance operator between two distinguished formulas expressed on the same configuration $C_i \in B$,

- Dom^{c_i,c_j}: External dominance operator between two formulas expressed on two distinguished configurations C_i and C_j,
- $Equi^{c_i,c_j}$: Equivalence operator between two formulas expressed on two distinguished configurations C_i and C_j in B,
- $Comp^{c_i}$: Internal composition operator between two formulas expressed on the same configuration $C_i \in B$,
- $Comp^{c_i,c_j}$: External composition operator between two formulas expressed on two distinguished configurations C_i and C_j in B,

Assumption 1. A classified formula deals only with one type of extension, i.e., it is not possible to classify two formulas from two different extensions with the same operator.

The dominance (ϕ_h^{TL} Dom^{C_i} ϕ_k^{TL}) (resp. ϕ_h^{TL} Dom^{C_i,C_j} ϕ_k^{TL}) means formula ϕ_h^{TL} on the configuration C_i dominates formula ϕ_k^{TL} on the same configuration (resp. on the configuration C_j) because of the existence of a precedence relation between states, an order of temporal logics connectors or a functional dependency between the left part and the right part.

The equivalence (ϕ_h^{TL} $Equi^{C_i,C_j}$ ϕ_k^{TL}) means that formula ϕ_h^{TL} on configuration C_i is equivalent with formula ϕ_k^{TL} on configuration C_j as a result of an equivalence between goals in CTL or time factor in TCTL, or sequence factor $\tau - sequences$ in eCTL.

The composition (ϕ_h^{TL} $Comp^{C_i}$ ϕ_k^{TL}) (resp. ϕ_h^{TL} $Comp^{C_i,C_j}$ ϕ_k^{TL}) means the conjunction of two properties on the same configuration C_i which have a precedence order in their goals (resp. between two properties belonging to two different configurations C_i and C_j).

In order to facilitate properties verification, we propose a new method for the automatic classification of basic CTL, TCTL, eCTL formulas according to relationships order. The proposed method passes through two principal operations: (i) the sorting and grouping operation, (ii) and the automatic generation.

3.1.1 Sorting and Grouping operation

This first operation proceeds in two steps: (i) First step, a syntactic sorting of properties according to the temporal logic class, i.e., we have the initial set of all properties including CTL, eCTL, TCTL properties, like an input denoted by $Prop\_Initial$ and we get sets of properties sorted according to each kind of logics in the output denoted by $Prop\_CTL$, $Prop\_eCTL$, $Prop\_TCTL$. Algorithm 1 resumes the sorting of $Prop\_initial$ assumed not empty to the sets $Prop\_CTL$, $Prop\_eCTL$, $Prop\_TCTL$ initially empty.

Algorithm 1: Syntactic sorting.

Input: $Prop\_Initial$;
Output: $Prop\_CTL$, $Prop\_eCTL$, $Prop\_TCTL$;
for *each* $\phi \in Prop\_initial$ **do**
  if *(ϕ_i is a CTL formula)* **then**
    Insert ϕ in $Prop\_CTL$;
  end
  if *(ϕ_i is a eCTL formula)* **then**
    Insert ϕ in $Prop\_eCTL$;
  end
  if *(ϕ_i is a TCTL formula)* **then**
    Insert ϕ in $Prop\_TCTL$;
  end
  Return ($Prop\_CTL$, $Prop\_eCTL$, $Prop\_TCTL$)
end

(ii) The second step projects each generated set in the first step on the system model to group the properties according to their configurations and their goals type (simple or composed). For each class of logic and for each configuration of the system, the projection gives two sets of properties: *Composed Goal Set* on the configuration i (CGS_i) and *Simple Goal Set* on the configuration i (SGS_i).

Definition 5. *Simple Goal Set* is the set of all properties that have a simple goal represented by a simple place or simple transition. e.g., $EF\ p_4$, $AG\ t_9$, etc.

Definition 6. *Composed Goal Set* is the set of all properties that have a composed goal represented by a conjunction of places or transitions or liveness properties .e.g., $EF(p_4 \wedge p_5)$, $AG(t_9 \longrightarrow EF\,p_9)$, etc. Algorithm 2 projects $Prop\_CTL$ set and groups their properties.

Algorithm 2: Grouping operation.

Input: $Prop\_CTL$;
Output: $\bigcup_{i=1}^{n} SGS_i$, $\bigcup_{i=1}^{n} CGS_i$;
for *each* $\phi \in Prop\_CTL$ **do**
  for *each configuration* C_i,*i=1..n* **do**
    if ϕ *is expressed on configuration* C_i **then**
      if *(ϕ_i is with a simple goal)* **then**
        Insert ϕ in SGS_i;
      end
      if *(ϕ_i with a composed goal)* **then**
        Insert ϕ in CGS_i;
      end
    end
  end
end
Return ($\bigcup_{i=1}^{n} SGS_i$, $\bigcup_{i=1}^{n} CGS_i$)

Figure 2 depicts the sorting and the grouping operation of properties.

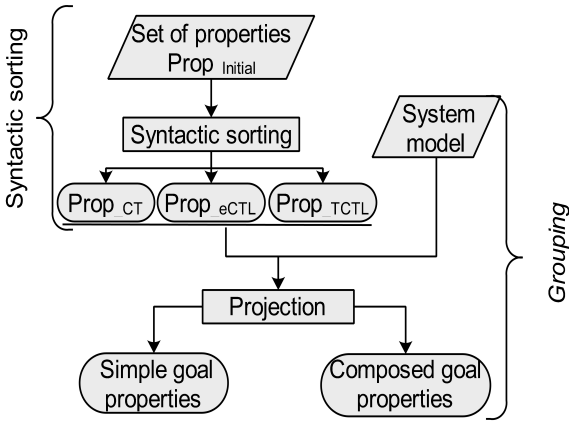

Figure 2: Sorting strategy of properties.

3.1.2 Automatic Generation

The algorithm of automatic classification exploits the structure of properties to be verified and identifies the possible relationships between them. After that, the model-checking algorithms assure an optimal guided and global verification using these relations. In this work, we focus on the automatic classification of basic formulas ϕ^{CTL}, ϕ^{TCTL}, and ϕ^{eCTL}. For one formula ϕ, if it is not possible to identify any relationships on any set, then the property will be called the non-involved property and it will be added to a new set called *non_involved*. Algorithm 3 identifies the relationships between properties through five steps. Each step corresponds to one kind of property identification. The negative result of each step takes the process to the next step. Those steps are:

- *step1*: Identify the internal dominance by checking the adhesion of simple goal to another composed goal property in the same configuration.
- *step2*: Identify the external dominance by checking the membership of simple goal to another with a composed goal in a different configuration.
- *step3*: Check a property with the same simple goal from another simple goal set.
- *step4*: Identify the internal composition by checking the precedence order in the same simple goal set.
- *step5*: Identify the external composition by checking the precedence order in the simple goal set from another configuration.

Algorithm 3: Automatic generation.

Input: $\bigcup_{i=1}^{n} SGS_i$, $\bigcup_{i=1}^{n} CGS_i$;
Output: *Set_ClassCTL*, *non_involved*;
Set_ClassCTL $\leftarrow \emptyset$;
non_involved $\leftarrow \emptyset$;
for *each configuration* C_i,i=1..n **do**
  for *each* ϕ_i *from* SGS_i **do**
    if $(\exists \psi \in CGS_i / \phi_i \in \psi)$ **then**
      *Insert $\phi_i Dom^{C_i} \psi$ into Set_ClassCTL;
      Update(*Set_ClassCTL*, $\phi_i Dom^{C_i} \psi$);
      Remove(ϕ_i, ψ);
      Break;
    end
    if $(\exists \psi \in CGS_j / \phi_i \in \psi)$ **then**
      Update(*Set_ClassCTL* , $\phi_i Dom^{C_i,C_j} \psi$);
      Remove(ϕ_i, ψ);
      Break;
    end
    if $(\exists \psi \in SGS_j / \phi_i \equiv \psi)$ **then**
      Update(*Set_ClassCTL*, $\phi_i Equt^{C_i,C_j} \psi$);
      Remove(ϕ_i, ψ);
      Break;
    end
    if $(\exists \psi \in SGS_i / \phi_i$ *above* $\psi)$ **then**
      Update(*Set_ClassCTL*, $\phi_i Comp^{C_i} \psi$);
      Break;
    end
    if $(\exists \psi \in SGS_j / \phi_i$ *above* $\psi)$ **then**
      Update(*Set_ClassCTL*, $\phi_i Comp^{C_i,C_j} \psi$);
      Break;
    end
    Update (*non_involved*, ϕ_i);
  end
end
Return (*Set_ClassCTL*, *non_involved*);

For a large-scale system of n configurations, the automatic classification algorithm has the cost of $O(|n| \times |K|)$, with K is the maximum number of projected properties in *SGS* set.

4 EXPERIMENTAL STUDY

4.1 Case Study

To demonstrate the performance and the gain of the proposed contribution, we use an R-TNCES to model a sequential system *Sys* (for example a production chain), to be denoted by $RTN_{Sys} = (B_{Sys}, R_{Sys})$. *Sys* is composed of 10 physical processes represented by 10 *CC*s. The behavior module of the system (B_{Sys}) is modeled graphically as shown in Figure 3. This model covers three configurations (Cf_1, Cf_2, Cf_3). It is assumed that every configuration has one control chain

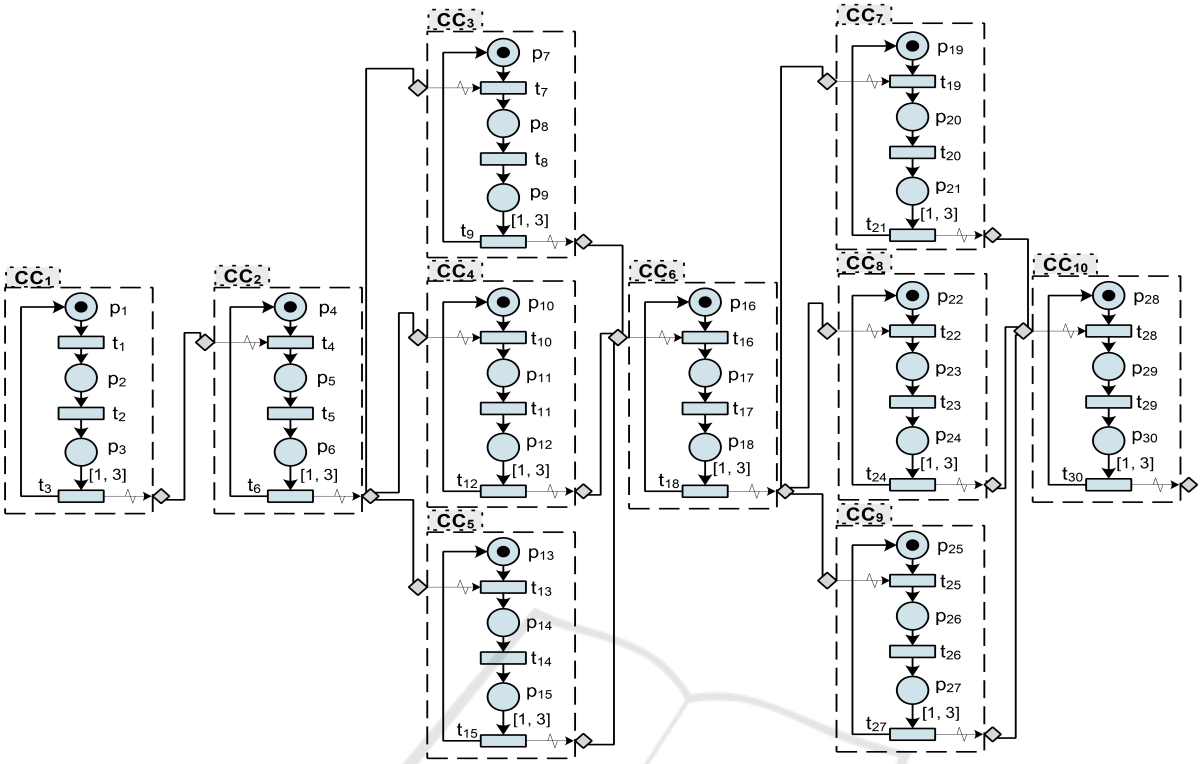

Figure 3: Model with three configurations process.

(C_{chain_i}). The control chains describing the physical process are:

- C_{chain_1}=(CC_1,CC_2,CC_3,CC_6,CC_7,CC_{10}),
- C_{chain_2}=(CC_1,CC_2,CC_4,CC_6,CC_8,CC_{10}),
- C_{chain_3} =(CC_1,CC_2,CC_5,CC_6,CC_9,CC_{10}).

The system model contains 30 states-transitions and 10 input/output events signals. This behavior module is able to be reconfigured automatically and timely between the three configurations ($Cf_i, i = 1,..,3$), according to the environment changes or to user requirements. RTN_{Sys} can apply six different reconfiguration scenarios according to the control module R_{Sys}, which are described as follows:

$R_{Sys} = \{(Cf_1,Cf_2); (Cf_1,Cf_3); (Cf_2,Cf_1); (Cf_2,Cf_3); (Cf_3,Cf_1); (Cf_3,Cf_2)\}$.

In the next section, we investigate the relationships between CCs in chains to prepare the automatic generation.

4.2 Example of Relationships

In order to validate the basic behavior of the system and to guarantee that *Sys* model satisfies the good requirements, we must ensure the CTL and eCTL based functional properties. In particular, to ensure the safety, the liveness and the non-blocking of the system. For the safety, the system allows only one process to be executed at any time, i.e., no activation of two CCs from two different configurations at the same time and the system must respect the potentiality of each CC. For the liveness, whenever any process wants to change the configuration, it will eventually be allowed to do so. For the non-blocking, any active CC is eventually ended, i.e., a process can always request to change the current CC. Each control component is characterized by time constraints, which are lower and upper limits constraining the execution. Indeed, we investigate that every CC respects its activation constraint ensured by the TCTL based temporal properties.

The above mentioned properties are specified using CTL, eCTL and TCTL logics, initially not organized. We apply Algorithm 1 to perform the syntactic sorting and grouping of those properties, and Algorithm 2 for the projection on *Sys* configuration. Table 1 resumes the result of these two operations.

By applying Algorithm 3, we give the classification of CTL, eCTL and TCTL properties in tables 2, 3 and 4. From the classification of the CTL properties presented in Tables 2, the guiding of verification of those properties can be like:

$$P_2 > P_8 > P_1 > P_9 > P_6 > P_{11} > P_5 > P_3 > P_4 > P_{10} \quad (2)$$

Table 1: The syntactic sorting, grouping and projection of $Prop_{Initial}$ set.

CTL formulas	
SGS_1	P_1: $EF(p_3)$; P_3: $EF(p_9)$;P_4: $EF(p_{18})$.
SGS_2	P_2: $AF(p_4)$; P_8: $AF(p_{18})$.
SGS_3	P_7: $AF(p_4)$; P_9: $AF(p_{18})$; P_{10}: $EF(p_{30})$.
CGS_1	P_5: $AG(p_{18} \longrightarrow AFp_{21})$.
CGS_2	P_6: $AG(p_{18} \longrightarrow AFp_{24})$.
CGS_3	P_{11}: $AG(p_3 \longrightarrow EFp_{30})$.
eCTL formulas	
CGS_1	P_1: $AGAt_1XAFp_3$; P_3: $AGAt_{18}XEFp_{21}$; P_2:$AGAt_{17}XAFEt_{18}Xp_{19}$.
CGS_2	P_4: $AGAt_2XAFp_3$; P_5: $AGAt_{18}XEFp_{24}$
CGS_3	P_6: $AGAt_{18}XEFp_{24}$; P_7: $AFAt_{18}XAFp_{30}$.
TCTL formulas	
SGS_1	P_1: $EF[1,3]p_3 = 1$; P_4: $EF[2,4]p_6 = 1$; P_9: $EF[4,12]p_{18} = 1$; P_5: $EF[3,9]p_9 = 1$;
SGS_2	P_2: $EF[1,3]p_3 = 1$; P_6: $EF[5,15]p_{24} = 1$; P_7: $EF[4,12]p_{18} = 1$;
SGS_3	P_3: $EF[1,3]p_3 = 1$; P_8: $EF[5,15]p_{27} = 1$ P_{10}: $EF[6,18]p_{30} = 1$;

Table 2: Classification of the CTL properties.

Internal Dominance	$P_8Dom^{C_2}P_6$
External Dominance	$P_1Dom^{C_1,C_3}P_{11}$; $P_9Dom^{C_3,C_1}P_5$.
Equivalence	$P_2Equi^{C_2,C_3}P_7$;
Internal composition	$P_3Comp^{C_1}P_4$.
Non_involved properties	
P_{10}.	

Table 3: Classification of the eCTL properties.

Equivalence	$P_1Equi^{C_1,C_2}P_4$.
Internal composition	$P_6Comp^{C_3}P_7$.
External composition	$P_3Comp^{C_1,C_2}P_5$.
Non_involved properties	
P_2.	

Or like:

$$P_7 > P_8 > P_1 > P_9 > P_6 > P_{11} > P_5 > P_3 > P_4 > P_{10} \quad (3)$$

From the classification of the eCTL properties in presented Table 3, the guiding of verification of those properties can be like:

$$P_1 > P_6 > P_7 > P_3 > P_5 > P_2 \quad (4)$$

$$P_4 > P_6 > P_7 > P_3 > P_5 > P_2 \quad (5)$$

From the classification of the TCTL properties in Table 4, the guiding of verification of those properties can be like:

$$P_1 > P_9 \mid P_7 > P_4 > P_5 > P_3 > P_8 > P_6 > P_{10} \quad (6)$$

Or like:

$$P_2 > P_9 \mid P_7 > P_4 > P_5 > P_3 > P_8 > P_6 > P_{10} \quad (7)$$

Let us assume that we have 100 properties to be verified here, and let us assume that the faulty property can be in different ranges, 0-25%, 25-50%, 50-75%, and 75-100% respectively. Figure 4 shows a quantitative difference for detection of the faulty property between the proposed approach and the verification of layer-by-layer. The benefit of this approach resides in the early detection of the faulty property.

Table 4: Classification of the TCTL properties.

Equivalence	$P_1Equi^{C_1,C_2}P_2$; $P_9Equi^{C_1,C_2}P_7$.
Internal composition	$P_4Comp^{C_1}P_5$; $P_3Comp^{C_2}P_8$;
External composition	$P_6Comp^{C_2,C_3}P_{10}$.

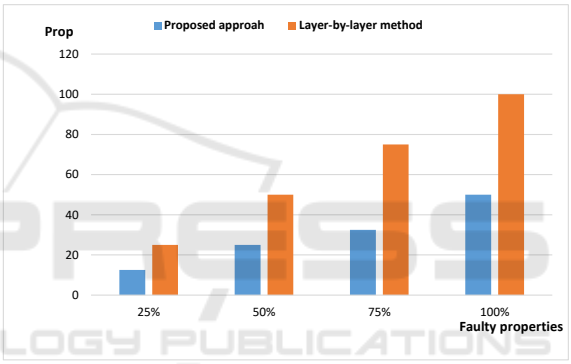

Figure 4: Proposed approach vs Layer-by-layer approach.

As shown in Figure 4, the number of properties to be checked using layer-by-layer method is average of two times bigger than using the proposed method. Thus, avoiding redundancies and ordering relations decreases the number of properties in the automatic generation method.

4.3 Discussion

The performance of this classification approach is directly related to the number of common CCs between TNCESs. Let us assume that the correctness of each model needs to satisfy the safety, the liveness, and the non-deadlock requirements. Let us assume that 10, 11, 29, 40, and 57 represent numbers of CCs in each TNCES for five different systems, each of which contains 10, 39, 70, 169, and 404 TNCESs respectively. We assume that each control component is proven to satisfy at least one property and at most five. Let us denoted by Γ the reduction percentage based on the similarity between TNCESs. We assume that Γ is

randomized between 1% and 30% (the average percentage). Figure 5 reveals the evolution of the emulated number of properties for RDECSs verification with the direct verification method and with the proposed one.

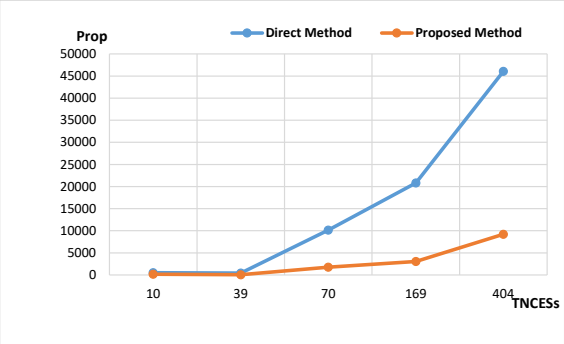

Figure 5: Number of property in the verification.

5 CONCLUSION

This work deals with the automatic classification of properties oriented to the reconfigurable system verification using CTL, eCTL and TCTL specifications.

In this paper, we have presented an arrangement of properties and we have guided model-checking for an improved verification in term of time and complexity (number of properties to be verified). This contribution is considered as a first step before the application of model checking to reduce as possible the number of formulas that specifying RDECS verification. We propose a new technique for the classification of computation tree logic (CTL) properties and the related automatic generation, which significantly reduces the number of properties to be verified in reconfigurable systems models. Besides reducing the number of properties, applying the factorization technique enables efficient model checking of the reconfigurable systems. Classically, we have to check all properties in blind way, but with the classification of CTL properties we identify the redundancies and the factorization between them. Using the contribution of this paper, we apply the model checking technique on a reduced space of properties which results in the gain of designer effort and outperforms time consumption in the verification process.

This work opens several perspectives; first, we plan to apply our approach in verification of real-case studies and to develop an extension of this method for dealing with reconfigurable distributed behaviors. Second, we plan to include the artificial intelligence to guide verification process by using ontologies in the automatic generation phase.

REFERENCES

Baier, C., Katoen, J.-P., and Larsen, K. G. (2008). *Principles of model checking*. MIT press.

Boucheneb, H., Gardey, G., and Roux, O. H. (2009). TCTL model checking of time Petri nets. *Journal of Logic and Computation*, 19(6):1509–1540.

Ghribi, I., Abdallah, R. B., Khalgui, M., Li, Z., Alnowibet, K., and Platzner, M. (2018). R-codesign: Codesign methodology for real-time reconfigurable embedded systems under energy constraints. *IEEE Access*, 6:14078–14092.

Grichi, H., Mosbahi, O., Khalgui, M., and Li, Z. (2017). New power-oriented methodology for dynamic resizing and mobility of reconfigurable wireless sensor networks. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, PP(99):1–11.

Idriss, R., Loukil, A., Khalgui, M., Li, Z., and Al-Ahmari, A. (2017). Filtering and intrusion detection approach for secured reconfigurable mobile systems. *Journal of Electrical Engineering & Technology*, 12(5):2051–2066.

Karoui, O., Guerfala, E., Koubaa, A., Khalgui, M., Tovard, E., Wu, N., Al-Ahmari, A., and Li, Z. (2017). Performance evaluation of vehicular platoons using webots. *IET Intelligent Transport Systems*, 11(8):441–449.

Khalgui, M. and Hanisch, H.-M. (2011). Automatic NCES-based specification and sesa-based verification of feasible control components in benchmark production systems. *International Journal of Modelling, Identification and Control*, 12(3):223–243.

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.

Lakhdhar, W., Mzid, R., Khalgui, M., Li, Z., Frey, G., and Al-Ahmari, A. (2018). Multiobjective optimization approach for a portable development of reconfigurable real-time systems: From specification to implementation. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, PP(99):1–15.

Meskina, S. B., Doggaz, N., Khalgui, M., and Li, Z. (2017). Multiagent framework for smart grids recovery. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 47(7):1284–1300.

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

Starke, P. H. and Roch, S. (2002). *Analysing Signal-net Systems*. Professoren des Inst. für Informatik.

Zhang, J., Khalgui, M., Li, Z., Frey, G., Mosbahi, O., and Salah, H. B. (2015). Reconfigurable coordination of distributed discrete event control systems. *IEEE Transactions on Control Systems Technology*, 23(1):323–330.

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.