# On Switchable Languages of Discrete-Event Systems with Weighted Automata

Michael Canu, Naly Rakoto-Ravalontsalama

23rd International Symposium on Mathematical Theory of Networks and Systems
Hong Kong University of Science and Technology, Hong Kong, July 16-20, 2018

# On Switchable Languages of Discrete-Event Systems with Weighted Automata

Michael Canu and Naly Rakoto-Ravalontsalama

*Abstract*— **The notion of switchable languages has been defined by Kumar, Takai, Fabian and Ushio in [11]. It deals with switching supervisory control, where switching means switching between two specifications. In this paper, we first extend the notion of switchable languages to $n$ languages, ($n \geq 3$). Then we consider a discrete-event system modeled with weighted automata. The use of weighted automata is justified by the fact that it allows us to synthesize a switching supervisory controller based on the cost associated to each event, like the energy for example. Finally the proposed methodology is applied to a simple example.**

*Keywords*: *Supervisory control; switching control; weighted automata.*

## I. Introduction

Supervisory control initiated by Ramadge and Wonham [15] provides a systematic approach for the control of discrete event system (DES) plant. There has been a considerable work in the DES community since this seminal paper. On the other hand, from the domain of continuous-time system, hybrid and switched systems have received a growing interests [12]. The notion of switching is an important feature that has to be taken into account, not only in the continuous-time domain but for the DES area too.

As for non-blocking property, there exist different approaches. The first one is the non-blocking property defined in [15]. Since then other types of non-blocking properties have been defined. The mutually non-blocking property has been proposed in [5]. Other approaches of mutually and globally nonblocking supervision with application to switching control is proposed in [11]. Robust non-blocking supervisory control has been proposed in [1]. Other types of non-blocking include the generalised non-blocking property studied in [13]. Discrete-event modeling with switching max-plus systems is proposed in [17], an example of mode switching DES is described in [6] and finally a modal supervisory control is considered in [7].

M. Canu is with Univ. los Andes, Bogota, Colombia, e-mail: *m.canu134@uniandes.edu.co*

N. Rakoto-Ravalontsalama is with IMT Atlantique and LS2N, France, e-mail: *naly.rakoto@mines-nantes.fr*

In this paper we will consider the notion of switching supervisory control defined by Kumar and Colleagues in [11] where switching means switching between a pair of specifications. Switching (supervisory) control is in fact an application of some results obtained in the same paper [11] about mutually non blocking properties of languages, mutually nonblocking supervisor existence, supremal controllable, relative-closed and mutually nonblocking languages. All these results led to the definition of a *pair of switchable languages* [11].

In this paper, we first extend the notion of switchable languages to $n$ languages, ($n \geq 3$). Then we consider a discrete-event system modeled with weighted automata. The switching supervisory control strategy is based on the cost associated to each event, and it allows us to synthesize an optimal supervisory controller. Finally the proposed methodology is applied to a simple example.

This paper is organized as follows. In Section II, we recall the notation and some preliminaries. Then in Section III the main results on the extension of $n$ switchable languages ($n \geq 3$) are given. An illustrative example of supervisory control of AGVs is proposed in Section IV, and finally a conclusion is given in Section V.

## II. Notation and Preliminaries

Let the discrete event system plant be modeled by a finite state automaton [10],[4] to which a cost function is added.

*Definition 1:* (Weighted automaton). A weighted automaton is defined as a sixtuple

$$G = (Q, \Sigma, \delta, q_0, Q_m, \mathbb{C})$$

where

- $Q$ is the finite set of states,
- $\Sigma$ is the finite set of events,
- $\delta : Q \times \Sigma \to Q$ is the partial transition function,
- $q_0 \subseteq Q$ is the initial state,
- $Q_m \subseteq Q$ is the set of *marked* states (final states),
- $\mathbb{C} : \Sigma \to \mathbb{N}$ is the cost function.

Let $\Sigma^*$ be the set of all finite strings of elements in $\Sigma$ including the empty string $\varepsilon$. The transition function $\delta$ can be generalized to $\delta : \Sigma^* \times Q \to Q$ in the following recursive manner:

$\delta(\varepsilon, q) = q$
$\delta(\omega\sigma, q) = \delta(\sigma, \delta(\omega, q))$ for $\omega \in \Sigma^*$

The notation $\delta(\sigma, q)!$ for any $\sigma \in \Sigma^*$ and $q \in Q$ denotes that $\delta(\sigma, q)$ is defined. Let $L(G) \subseteq \Sigma^*$ be the language generated by $G$, that is,

$$L(G) = \{\sigma \in \Sigma^* | \delta(\sigma, q_0)!\}$$

Let $K \subseteq \Sigma^*$ be a language. The set of all prefixes of strings in $K$ is denoted by $pr(K)$ with $pr(K) = \{\sigma \in \Sigma^* | \exists\ t \in \Sigma^*; \sigma t \in K\}$. A language $K$ is said to be *prefix closed* if $K = pr(K)$. The event set $\Sigma$ is decomposed into two subsets $\Sigma_c$ and $\Sigma_{uc}$ of controllable and uncontrollable events, respectively, where $\Sigma_c \cap \Sigma_{uc} = \emptyset$. A controller, called a supervisor, controls the plant by dynamically disabling some of the controllable events.

A sequence $\sigma_1\sigma_2 \ldots \sigma_n \in \Sigma^*$ is called a *trace* or a *word* in term of language. We call a *valid trace* a path from the initial state to a marked state ($\delta(\omega, q_0) = q_m$ where $\omega \in \Sigma^*$ and $q_m \in Q_m$). The cost is by definition non negative. In the same way, the cost function $\mathbb{C}$ is generalized to the domain $\Sigma^*$ as follows:

$\mathbb{C}(\varepsilon) = 0$
$\mathbb{C}(\omega\sigma) = \mathbb{C}(\omega) + \mathbb{C}(\sigma)$ for $\omega \in \Sigma^*$

In other words, the cost of a trace is the sum of the costs of each event that composes the trace.

*Definition 2:* (Controllability) [15]. A language $K \subseteq L(G)$ is said to be *controllable* with respect to (w.r.t.) $L(G)$ and $\Sigma_{uc}$ if

$$pr(K)\Sigma_{uc} \cap L(G) \subseteq pr(K).$$

*Definition 3:* (Mutually non-blocking supervisor) [5]. a supervisor $f : L(G) \to 2^{\Sigma - \Sigma_u}$ is said to be $(K_1, K_2)$-*mutually non-blocking* if

$$K_i \cap L_m(G^f) \subseteq pr(K_j \cap L_m(G^f)), \text{ for } i, j \in \{1, 2\}. \quad (1)$$

In other words, a supervisor $S$ is said to be *mutually non-blocking* w.r.t. two specifications $K_1$ and $K_2$ if whenever the closed-loop system has completed a task of one language (by completing a marked trace of that language), then it is always able to continue to complete a task of the other language [5].

*Definition 4:* (Mutually non-blocking language) [5]. A language $H \subseteq K_1 \cup K_2$ is said to be $(K_1, K_2)$-*mutually non-blocking* if $H \cap K_i \subseteq pr(H \cap K_j)$ for $i, j \in \{1, 2\}$.

The following theorem gives a necessary and sufficient condition for the existence of a supervisor.

*Theorem 1: (Mutually nonblocking supervisor existence) [5].* Given a pair of specifications $K_1, K_2 \subseteq L_m(G)$, there exists a globally and mutually non-blocking supervisor $f$ such that $L_m(G^f) \subseteq K_1 \cup K_2$ if and only if there exists a nonempty, controllable, relative-closed, and $(K_1, K_2)$-mutually non-blocking sublanguage of $K_1 \cup K_2$.

The largest possible language (the supremal element) that is controllable and mutually non-blocking exists, as stated by the following theorem.

*Theorem 2: (SupMRC($K_1 \cup K_2$) existence)* [5]. The set of controllable, relative-closed, and mutually non-blocking languages is closed under union, so that the supremal such sublanguage of $K_1 \cup K_2$, denoted $supMRC(K_1 \cup K_2)$ exists.

Recall that a pair of languages $K_1, K_2$ are *mutually nonconflicting* if $pr(K_1 \cap K_2) = pr(K_1) \cap pr(K_2)$ [18]. $K_1, K_2$ are called *mutually weakly nonconflicting* if $K_i, pr(K_j)$ $(i \neq j)$ are mutually nonconflicting [5].

Another useful result from [5] is the following. Given a pair of mutually weakly nonconflicting languages $K_1, K_2 \subseteq L_m(G)$, the following holds ([5], Lemma 3). If $K_1, K_2$ are controllable then $K_1 \cap pr(K_2), K_2 \cap pr(K_1)$ are also controllable.

The following theorem is proposed in [11] and it gives the formula for the supremal controllable, relative-closed, and mutually nonblocking languages.

*Theorem 3: (SupMRC($K_1 \cup K_2$))* [11]. For relative-closed specifications $K_1, K_2 \subseteq L_m(G)$, $supMRC(K_1 \cup K_2) = supRC(K_1 \cap K_2)$.

The following theorem, also from [11] gives another expression of the supremal controllable, relative-closed, and mutually nonblocking languages.

*Theorem 4:* [11] Given a pair of controllable, relative-closed, and mutually weakly nonconflicting languages $K_1, K_2 \subseteq L_m(G)$, it holds that $supMRC(K_1 \cup K_2) = (K_1 \cap K_2)$.

And finally the following theorem gives a third formula of the supremal controllable, relative-closed, and mutually nonblocking languages.

*Theorem 5:* [11] For specifications $K_1, K_2 \subseteq L_m(G)$, $supMRC(K_1 \cup K_2) = supMC(supRC(K_1 \cap K_2))$.

In order to allow switching between specifications, a pair of supervisors is considered, such that the supervisor is switched when the specification is switched. The supervisor $f_i$ for the specification $K_i$ is designed to enforce a certain sublanguage $H_i \subseteq K_i$. Suppose a switching in specification from $K_i$ to $K_j$ is induced at a point when a trace $s \in H_i$ has been executed in the $f_i$-controlled plant. Then in order to be able to continue with the new specification $K_j$ *without reconfiguring the plant*, the trace $s$ must be a prefix of $H_j \subseteq K_j$. In other words, the two supervisors should enforce the languages $H_i$ and $H_j$ respectively such that $H_i \subseteq pr(H_j)$. Hence the set of pairs of such languages are defined to be *switchable languages* as follows.

*Definition 5:* (Pair of switchable languages) [11]. A pair of specifications $K_1, K_2 \subseteq L_m(G)$ are said to be *switchable languages* if
$SW(K_1, K_2) := \{(H_1, H_2) | H_i \subseteq K_i \cap pr(H_j), i \neq j,$ and $H_i$ controllable$\}$.

The supremal pair of switchable languages exists and is given by the following theorem.

*Theorem 6:* (Supremal pair of switchable languages) [11]. For specifications $K_1, K_2 \subseteq L_m(G)$,
$supSW(K_1, K_2) =$
$(supMC(K_1 \cup K_2) \cap K_1, supMC(K_1 \cup K_2) \cap K_2)$.

## III. Main Results

We now give the main results of this paper. First, we define a triplet of switchable languages. Second we derive a necessary and sufficient condition for the transitivity of switchable languages ($n = 3$). Third we generalize this definition to a n-uplet of switchable languages, with $n > 3$. And fourth we derive a necessary and sufficient condition for the transitivity of switchable languages for $n > 3$.

### A. Triplet of Switchable Languages

We extend the notion of pair of switchable languages, defined in [11], to a triplet of switchable languages.

*Definition 6:* (Triplet of switchable languages). A triplet of languages $(K_1, K_2, K_3)$, $K_i \subseteq L_m(G)$ with $H_i \subseteq K_i,\ i = \{1, 2, 3\}$ are said to be a *triplet of switchable languages* if they are pairwise switchable languages, that is,

$SW(K_1, K_2, K_3) := SW(K_i, K_j),\ i \neq j,\ i, j = \{1, 2, 3\}$.

Another expression of the triplet of switchable languages is given by the following lemma.

*Lemma 1:* (Triplet of switchable languages). A triplet of languages $(K_1, K_2, K_3)$, $K_i \subseteq L_m(G)$ with $H_i \subseteq K_i,\ i = \{1, 2, 3\}$ are said to be a *triplet of switchable languages* if the following holds:

$SW(K_1, K_2, K_3) = \{(H_1, H_2, H_3) \mid H_i \subseteq K_i \cap pr(H_j), i \neq j,$ and $H_i$ controllable$\}$.

### B. Transitivity of Switchable Languages ($n = 3$)

The following theorem gives a necessary and sufficient condition for the transitivity of switchable languages.

*Theorem 7:* (Transitivity of switchable languages, $n = 3$) . Given 3 specifications $(K_1, K_2, K_3)$, $K_i \subseteq L_m(G)$ with $H_i \subseteq K_i,\ i = \{1, 2, 3\}$ such that $SW(K_1, K_2)$ and $SW(K_2, K_3)$.
$(K_1, K_3)$ is a pair of switchable languages, i.e. $SW(K_1, K_3)$, if and only if

1) $H_1 \cap pr(H_3) = H_1$, and
2) $H_3 \cap pr(H_1) = H_3$.

*Proof:* The proof can be found in [3]. ■

### C. N-uplet of Switchable Languages

We now extend the notion of switchable languages, to a n-uplet of switchable languages, with ($n > 3$).

*Definition 7:* (N-uplet of switchable languages, $n > 3$). A n-uplet of languages $(K_1, ..., K_n)$, $K_i \subseteq L_m(G)$ with $H_i \subseteq K_i,\ i = \{1, ..., n\}, n > 2$, is said to be a *n-uplet of switchable languages* if the languages are pairwise switchable that is,

$SW(K_1, ..., K_n) := SW(K_i, K_j),\ i \neq j,\ i, j = \{1, ..., n\}, n > 2$.

As for the triplet of switchable languages, an alternative expression of the n-uplet of switchable languages is given by the following lemma.

*Lemma 2:* (N-uplet of switchable languages, $n > 3$). A n-uplet of languages $(K_1, \ldots, K_n)$, $K_i \subseteq L_m(G)$

with $H_i \subseteq K_i,\ i = \{1, ..., n\},\ n > 3$ are said to be a *n-uplet of switchable languages* if the following holds:

$SW(K_1, ..., K_n) = \{(H_1, ..., H_n) \mid H_i \subseteq K_i \cap pr(H_j), i \neq j$, and $H_i$ controllable$\}$.

### D. Transitivity of Switchable Languages $(n > 3)$

We are now able to derive the following theorem that gives a necessary and sufficient condition for the transitivity of $n$ switchable languages.

*Theorem 8:* (Transitivity of n switchable languages, $n > 3$) . Given $n$ specifications $(K_1, ..., K_n)$, $K_i \subseteq L_m(G)$ with $H_i \subseteq K_i,\ i = \{1, ..., n\}$. Moreover, assume that each language $K_i$ is at least switchable with another language $K_j,\ i \neq j$.
A pair of languages $(K_k, K_l)$ is switchable i.e. $SW(K_k, K_l)$, if and only if

1) $H_k \cap pr(H_l) = H_k$, and
2) $H_l \cap pr(H_k) = H_l$.

*Proof:* The proof is similar to the proof of Theorem 6 and can be found in [3]. ■
It is to be noted that the assumption that each of the $n$ languages be at least switchable with another language is important, in order to derive the above result.

## IV. Example: Switching Supervisory Control of AGVs

The idea of switching supervisory control is now applied to a discrete-event system, modeled with weighted automata. We take as an illustrating example the supervisory control of a fleet of fleet automated guided vehicles (AGVs) that move in a given circuit area. The example is taken from [9]. A circuit is partitioned into sections and intersections. Each time an AGV moves in a new intersection or a new section, then the automaton will move to a new state in the associated automaton. An example of an area with its associated basic automaton is depicted in Figure 1.

The area to be supervised is the square depicted in Figure 1 (left). The flow direction with the arrows are specified the four intersections $\{A, B, C, D\}$ and the associated basic automaton are given in Figure 1 (right). The basic automaton is denoted $G_{basic} = (Q_b, \Sigma_b, \delta_b, \emptyset, \emptyset)$ where the initial state and the final state are not defined. The initial state is defined according to the physical position of the AGV and the final state is defined according to its mission, that is his position target. A state represents and intersection or a section. Each state corresponding to a section is named

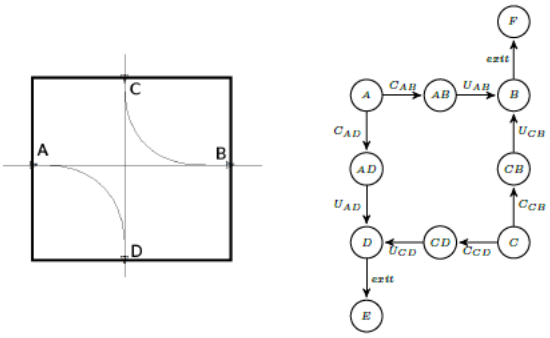


Fig. 1. An AGV circuit (left) and its basic automaton (right)

$XY_i$ where $X$ is the beginning of the section, $Y$ its end and $i$ the number of the AGV. For each section, there are two transitions, the first transition $C_{XY}$ is an input which is controllable and represents the AGV moving on the section from $X$ to $Y$. The second transition is an output transition $U_Y$ which is uncontrollable and represents the AGV arriving to the intersection $Y$.
For example the basic automaton depicted in Figure 1 (right) can be interpreted as follows. If $AGV_i$ arrives at section $A$, then it has two possibilities, either to go to section $B$ with the event $C_{ABi}$, or the go section $D$ with the event $C_{ADi}$. If we choose to go to section $B$, then the next state is $AB_i$. From this state, the uncontrollable event $U_{AB}$ is true so that the following state is $B_i$. And from $B_i$, the only possibility is to exit to Point $F$ with the uncontrollable event $exit_i$.
Now consider for example that 2 AGVs are moving in the circuit of Figure 1 (left). Assume $AGV_1$ is in $D$ and $AGV_2$ is in $AB$ so that the state is in $(D_1, AB_2)$. $AGV_1$ is leaving the area when the event $exit_1$ is true so that the system will be in state $(E_1, AB_2)$. And since $AGV_1$ is out of the considered area, then the new state will be $(E_1, AB_2) = (\emptyset_1, AB_2) = (AB_2)$ since $AGV_1$ is out of the area.

We give here below the synthesis algorithm for calculating the supervisor $S_c$ as it aws proposed by Girault et Colleagues in [9]. For more details on the synthesis algorithm, the reader is referred to the above paper.

**Algorithm 1** – Synthesis algorithm of $S_C$ [9]

**Data:** $G_{w,1}, \dots G_{w,n}$
**Result:** Supervisor $S_C$

$G_w \leftarrow \{G_{w,1}, \dots G_{w,n}\}$
$G_u \leftarrow \{\emptyset\}$
**forall** $G_{w,i} \in G_w$ **do**
  $G_u \leftarrow G_u \cup U_{\gamma_i}(G_{w,i})$
**end**
$S_C \leftarrow S(G_{u,i})$
$G_u \leftarrow G_u \backslash \{G_{u,1}\}$
**while** $G_u \neq \emptyset$ **do**
  $x \leftarrow get(G_u)$
  $S_C \leftarrow S(S_C || x)$
  $G_u \leftarrow G_u \backslash \{x\}$
**end**

## V. Conclusions

The notion of switchable languages has been defined by Kumar and Colleagues in [11]. It deals with switching supervisory control, where switching means switching between two specifications. In this paper, we have extended the notion of switchable languages to a triplet of languages ($n = 3$) and we gave a necessary and sufficient condition for the transitivity of two switchable languages. Then we generalized the notion of switchable languages of a n-uplet of languages, $n > 3$ and we gave also necessary and sufficient condition for the transitivity of two (out of $n$) switchable languages. Finally the proposed methodology is applied to a simple example for the supervisory control of a fleet of AGVs. Ongoing work deals with a) the calculation of the supremal of n-uplet of switchable languages, and b) the optimal switching supervisory control of DES exploiting the cost of the weighted automata for the synthesis strategy.

## References

[1] S.E. Bourdon, M. Lawford, and W.M. Wonham "Robust nonblocking supervisory control of discrete event systems," In *IEEE Trans. on Automatic Control*, vol. 50, N.12 pp. 2015–2021, 2005.

[2] M. Canu and N. Rakoto-Ravalontsalama, *From mutually non-blocking to switched non-blocking DES*. Presented at MSR'13 Workshop (Poster Session), Rennes, France, Nov 13-15, 2013.

[3] M. Canu and N. Rakoto-Ravalontsalama. *On Switchable Languages of Discrete-Event Systems with Weighted Automata*, Technical Report Mines Nantes, March 2017.

[4] C.G. Cassandras and S. Lafortune, "Introduction to Discrete Event Systems," 2nd Edition, Springer Verlag, 2008.

[5] M. Fabian and R. Kumar. "Mutually non-blocking supervisory control of discrete-event systems," In *Automatica*, 36(12) pp. 1863–1869, 2000.

[6] G. Faraut, L. Pietrac, and E. Niel, "Formal Approach to Multimodal Control Design: Application to Mode Switching", In *IEEE Trans. on Industrial Informatics*, vol.5, N.4 pp. 443–453, Nov 2009.

[7] G. Faraut, L. Pietrac, and E. Niel, "Process Tracking by Equivalent States in Modal Supervisory Control", Proc. of *IEEE ETFA*, Sep. 2011, Toulouse, France.

[8] J. Girault, J.J. Loiseau, O.H. Roux "Synthese en ligne de superviseur compositionnel pour une flotte de robots mobiles." European Journal of Automation, MSR'13, vol 47/1-3, pp. 195–210, 2013.

[9] J. Girault, J.J. Loiseau, O.H. Roux "On-line optimal compositional controller synthesis for AGV by unfolding," Proc. of DCDS 2015, IFAC-PapersOnLine 48-7 (2015) pp. 167–173.

[10] J.E. Hopcroft and J.D. Ullman, "Introduction to Automata Theory, Languages, and Computation," Addison-Wesley, Reading, MA, USA, 1979.

[11] R. Kumar, S. Takai, M. Fabian, and T. Ushio, "Maximally Permissive Mutually and Globally Nonblocking Supervision with Application to Switching Control," In *Automatica*, 41(8) pp. 1299–1312, 2005.

[12] D. Liberzon, "Switching in Systems and Control", ser. Systems and Control: Foundations and Applications. Boston: Birkhauser, 2003.

[13] R. Malik and R. Leduc, "Generalised nonblocking"', in Proc. 9th Int. Workshop on Discrete Event Systems, WODES 2008, Goteborg, Sweden, May 2008, pp. 340–345.

[14] N. Rakoto-Ravalontsalama. "Supervisory control of switched discrete-event systems," in Proc. of 17th Symp. on MTNS 2006, Kyoto, Japan 2006, pp. 2213–2217.

[15] P.J. Ramadge and W.M. Wonham. "Supervisory control of a class of discrete-event processes," In *SIAM J. Control and Optimization*, vol.25 pp. 206–230, 1987.

[16] P.S. Roop, A. Girault, R. Sinha, and G. Goessler.

”Specification Enforcing Refinement for Convertibility Verification,” Proc. of ACSD 2009, pp. 148–157, IEEE, 2009.

[17] T.J.J. van den Boom and B. de Schutter. ”Modelling and control of discrete event systems using switching max-plus-linear systems,” *Control Engineering Practice*, vol. 14 N.10, pp. 1199-1211, 2006.

[18] W.M. Wonham and P.J. Ramadge. ”Modular supervisory control of discrete-event systems,” In *Mathematics of Control, Signals and Systems*, vol.1 (1) pp. 13–30, 1988.