Control Engineering Practice 82 (2019) 97–107

Contents lists available at ScienceDirect

Control Engineering Practice

journal homepage: www.elsevier.com/locate/conengprac

An implementation method for the supervisory control of time-driven systems applied to high-voltage direct current transmission grids

Miguel Romero-Rodríguez [a,*], Romain Delpoux [b,a], Laurent Piétrac [b,a], Jing Dai [c,a], Abdelkrim Benchaib [a,d], Eric Niel [b,a]

[a] SuperGrid Institute SAS, F-69611 Villeurbanne, France
[b] Université de Lyon, CNRS, INSA-Lyon, AMPERE, F-69621 Villeurbanne, France
[c] Group of Electrical Engineering — Paris (GeePs), UMR CNRS 8507, CentraleSupélec, Univ. Paris-Sud, Univ. Paris-Saclay, Sorbonne Universités, UPMC Univ. Paris 06, F-91192 Gif-sur-Yvette, France
[d] GE Grid Solutions, F-91300 Massy, France

ARTICLE INFO

Keywords:
Discrete event systems
Supervisory control theory
Control implementation
HVDC transmission systems
EMTP-RV
Computer programming languages

ABSTRACT

In recent years, the growth of renewable energy production has encouraged the development of new technologies, such as High-Voltage Direct Current (HVDC) networks, that enhance the integration of such energy sources to power transmission grids. However, this type of technology introduces new challenges in the way power transmission systems are controlled and operated, as faster and more complex control strategies will be needed in a domain which nowadays relies heavily on human decisions. In this context, Discrete Event Systems (DES) modeling and Supervisory Control Theory (SCT) are powerful tools for the development of a supervisory control to be deployed in the grid. This paper presents an application of the SCT to HVDC grids and proposes an implementation method for the resulting supervisors. The proposed method is capable of integrating decentralized and discrete-event controllers that interact with the continuous-time physical system. The language chosen for the implementation is C code, as it can be easily incorporated in power system simulation software, such as EMTP-RV. The method is validated by the simulation of the start-up of a point-to-point link in the EMTP-RV software.

1. Introduction

The integration of renewable energy sources to the existing electrical grids is a key issue in the domain of energy transportation. The development of large High-Voltage Direct Current (HVDC) networks that bring the power from remote renewable sources to load centers will increase the complexity of power transmission systems, thus introducing new challenges in the way these types of systems are controlled and operated (van Hertem & Ghandhari, 2010; Zhang, Li, & Bhatt, 2010). For instance, in traditional power transmission systems based on widespread Alternating Current (AC) technology, large turbo generators are connected to the grid. In consequence, the inertia of their rotating masses liberates energy that provides resistance against frequency disturbances, allowing the different frequency control actions to be deployed in a timescale from 1 to 2 s to 15 to 30 min after the disturbance (Rebours, Kirschen, Trotignon, & Rossignol, 2007). On the contrary, the lower energy stocked in HVDC systems provides less resistance against voltage disturbances. In consequence, the transient generated by the disturbance will not be compensated in time, and thus the control should react faster (in the order of 100 ms). In addition, new converter topologies such as the Modular Multilevel Converters (MMCs) introduce additional degrees of freedom for control that increase the complexity of grid operation. For all these reasons, the need for an automated and coordinated supervisory control system during grid operation will increase over the years.

In this context, Discrete Event Systems (DES) modeling and the Supervisory Control Theory (SCT), first proposed in Ramadge and Wonham (1987), offer a formal framework for the synthesis of supervisors ensuring that the system under control respects a set of behavioral specifications, imposed by the designer, within its physical limitations. Moreover, the use of an SCT-based modal approach, such as the one presented in Faraut, Piétrac, and Niel (2009), would allow to manage the transition between the different operating modes of an HVDC system: start-up, fault protection, power ramp, shut-down, etc.

Despite the need to ensure that the interaction between the components of complex power transmission networks (highly reconfigurable and composed of many interconnected components) does not impact negatively the behavior of the whole system, this problem has not been treated in the literature. In consequence, the authors proposed a method for the synthesis of a decentralized supervisory control system

* Corresponding author.
E-mail address: miguel.romerorodriguez@supergrid-institute.com (M. Romero-Rodríguez).

https://doi.org/10.1016/j.conengprac.2018.10.002
Received 18 July 2017; Received in revised form 26 March 2018; Accepted 3 October 2018
Available online xxxx
0967-0661/

for HVDC grids in Romero Rodríguez, Delpoux, Piétrac, Dai, Benchaib, and Niel (2017). In the current paper, the aspects regarding the practical implementation of the theoretical supervisors for the start-up of a point-to-point link obtained in Romero Rodríguez et al. (2017) are addressed. Because it is desired to implement the supervisory control in power system's specific simulation software such as EMTP-RV (Mahseredjian, Dennetière, Dubé, Khodabakhchian, & Gérin-Lajoie, 2007) or its real-time simulation counterpart HYPERSIM (Do, Soumagne, Sybille, Turmel, Giroux, Cloutier, & Poulin, 1999), the implementation method presented here is based on common user oriented languages, such as C code.

A number of papers have contributed to the implementation of the supervisors obtained with the SCT over the last decades, for the most part related to the control of manufacturing systems based on widespread Programmable Logic Controllers (PLCs). Consequently, most of the related works in the literature try to adapt the SCT framework to the programming languages defined by the International Electrotechnical Commission in the IEC 61131-3 standard (International Electrotechnical Commission, 2003), especially to the most popular graphical languages: Ladder Diagram (LD) and Sequential Function Chart (SFC). While the methods developed in de Queiroz (2002), Fabian and Hellgren (1998), Gouyon, Pétin, and Gouin (2004), Lauzon, Mills, and Benhabib (1997), Leal, da Cruz, and Hounsell (2012) and Ramirez-Serrano, Zhu, Chan, Chan, Ficocelli, and Benhabib (2002) are all based on LD, those presented in Charbonnier, Alla, and David (1995) and Vieira, Santos, de Queiroz, Leal, Neto, and Cury (2017) are SFC-based. However, the particular syntax of those languages offers little portability for the proposed methods to be applied outside PLC-based environments.

In addition, because the future development of multi-terminal DC (MTDC) grids might imply a combinatorial explosion during the synthesis of the supervisors, a decentralized architecture that localizes the control is suitable. Thus, the implementation method requires that the information communicated between the different controllers should be taken into account, as opposed to previous contributions, where only centralized (Balemi, 1992; Cantarelli & Roussel, 2008) and modular (de Queiroz, 2002; Vieira et al., 2017) architectures with no communication between controllers were contemplated.

The remainder of this paper is organized as follows. Section 2 reviews the fundamentals of DES modeling and SCT. A case study is presented and a decentralized supervisory control for the start-up of an HVDC system is synthesized in Section 3. In Section 4, the proposed implementation method is presented and the simulation results obtained in the EMTP-RV software are shown. At last, conclusions are drawn in Section 5.

2. Background

This section reviews the basic notions of DES modeling, along with the fundamentals of SCT and the control architectures that can be derived from the synthesized supervisors.

2.1. Discrete event systems

A DES is a discrete-state, event-driven system which does not depend on time and whose state evolution depends entirely on the occurrence of asynchronous discrete events (Cassandras & Lafortune, 2008). Based on the property of controllability, it is possible to divide the event set Σ into two subsets, i.e. $\Sigma = \Sigma_c \cup \Sigma_u$, where Σ_c and Σ_u are respectively the set of controllable and uncontrollable events. The occurrence of an event in Σ_c (resp. Σ_u) can (resp. cannot) be prevented by a supervisor S. The concatenation of the events $\sigma_i \in \Sigma$ $(i = 1, \ldots, n)$ forms finite sequences (or strings) which are all represented by the infinite set Σ^*, derived by the operation called Kleene-closure (*):

$$\Sigma^* = \{\varepsilon \cup \sigma_1 \cup \sigma_2 \cup \sigma_3 \cup \sigma_1\sigma_2 \cup \sigma_1\sigma_3 \cup \ldots\}, \tag{1}$$

where ε is the empty string. Thus, a language L, which is a finite set of finite-length strings formed from events in Σ, is a subset of Σ^* ($L \subseteq \Sigma^*$). A language is said to be prefix-closed if any prefix $t \in \Sigma^*$ of any string $s \in L$ is also an element of L ($L = \bar{L}$), with $\bar{L}$ consisting of all the prefixes of all the strings in L:

$$\bar{L} := \{s \in \Sigma^* : (\exists t \in \Sigma^*)\,[st \in L]\}. \tag{2}$$

A deterministic automaton A can be defined as a six-tuple $A = (X, \Sigma, f, \Gamma, x_0, X_m)$, where X is the set of states, Σ is the finite set of events associated to A and $f : X \times \Sigma \to X$ is the partial transition function. This function can be extended to $f : X \times \Sigma^* \to X$ in a natural way. Moreover, $\Gamma : X \to 2^{\Sigma}$ is the active event function representing the set of all events σ for which a transition $f(x, \sigma)$ is defined at state x. Finally, $x_0 \in X$ is the initial state and $X_m \subseteq X$ is the set of marked states that represent the completion of a task.

We distinguish between the language $L(A)$ generated by A and the language $L_m(A)$ marked by A. While $L(A)$ represents all the strings s starting from the initial state and whose transition function f is defined at (x_0, s):

$$L(A) := \{s \in \Sigma^* : f(x_0, s)\} \text{ is defined}, \tag{3}$$

the language marked by A is formed by the strings s that start from the initial state and end at a marked state ($f(x_0, s) \in X_m$):

$$L_m(A) := \left\{s \in L(A) : f(x_0, s) \in X_m\right\}. \tag{4}$$

An automaton is said to be non-blocking when all its states are accessible from x_0 and co-accessible, that is, X_m can be reached from state x. Then, $\overline{L_m(A)} = L(A)$.

2.2. Supervisory control theory

The SCT was first proposed in Ramadge and Wonham (1987). Based on language theory and DES modeling, the SCT aims to synthesize a supervisor that ensures by construction that the behavior of the system (also called plant) under control remains admissible with respect to a set of specifications. The plant is modeled in the form of an automaton G and is independent of the control objectives as it represents the physical process. The designer then models in the same form the control specifications to be imposed on the uncontrolled plant in order to restrict its behavior within the subset $K \subseteq L_m(G)$. Then, conforming to the SCT, a non-blocking supervisor S exists such that $L_m(S/G) = K$ and $L(S/G) = \overline{K}$, with $K \subseteq L_m(G)$ and $K \neq \emptyset$, if and only if the controllability condition ($\overline{K}\Sigma_u \cap L(G) \subseteq \overline{K}$) and the $L_m(G)$-closure condition ($K = \overline{K} \cap L_m(G)$) are respected. If K is not controllable, the largest sublanguage of K that is controllable, with $L_m(G)$-closure condition, can be computed. Formally, the supervisor S for the plant G is a function that maps each word of the language of G to the set of controllable events which are enabled after the occurrence of that word. Meantime, the set of feasible uncontrollable events cannot be disabled by the supervisor S. So for a string $s \in L(G)$, $S(s)$ is defined according to Cassandras and Lafortune (2008):

$$S(s) = [\Sigma_u \cap \Gamma(f(x_0, s))] \cup \left\{\sigma \in \Sigma_c : s\sigma \in \overline{K}\right\}. \tag{5}$$

In the first term of (5), the supervisor enables after string s all uncontrollable events that are feasible in G. In this way, a feasible uncontrollable event is never disabled. In the second term of (1), all the controllable events that extend s inside of K are allowed. The language marked by the closed-loop S/G is defined as follows:

$$L_m(S/G) := L(S/G) \cap L_m(G), \tag{6}$$

where $L_m(S/G) \subset L(G)$ is strictly contained in the language generated by G and it corresponds to the optimal behavior of G under the supervision of S. In a centralized or monolithic control architecture (Fig. 1), the automaton representing a supervisor is typically automaton S/G itself.

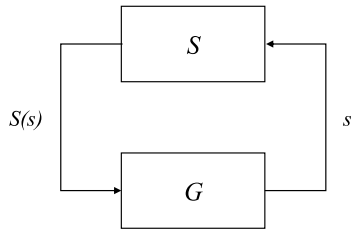


Fig. 1. Centralized control architecture (Cassandras & Lafortune, 2008).

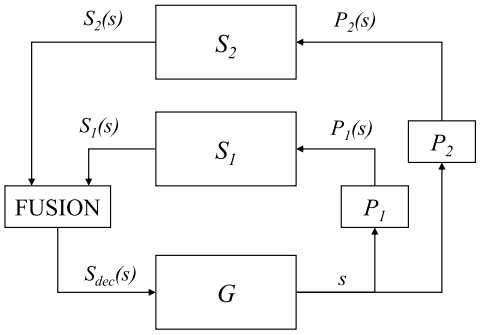


Fig. 2. Decentralized control architecture (Cassandras & Lafortune, 2008).

In a decentralized architecture (Fig. 2), however, each decentralized controller S_i $(i = 1, \dots, n)$ receives an observed string obtained through a projection operation $P_i : \Sigma_G^* \to \Sigma_i^*$, which takes a string formed from the larger event set Σ_G and deletes the events in it that do not belong to the smaller event set Σ_i:

$$P_i(\varepsilon) := \varepsilon,$$
$$P_i(\sigma) := \begin{cases} \sigma & \text{if } \sigma \in \Sigma_i, \\ \varepsilon & \text{if } \sigma \in \Sigma_G \setminus \Sigma_i, \end{cases} \tag{7}$$
$$P_i(s\sigma) := P_i(s)P_i(\sigma) \text{ for } s \in \Sigma_G^*,\ \sigma \in \Sigma_G.$$

The projection P_i is then extended to the language $L(G) \subset \Sigma_G^*$ by simply applying it to all the strings in the language.

Because several controllers may have common events in their alphabets, thus establishing an implicit communication with other controllers via the system, the decentralized control structure is built on the joint action of all n decentralized supervisors. In Yoo and Lafortune (2002), two fusion rules for merging the control actions of the individual supervisors are defined. In this paper, a conjunctive fusion rule based on the intersection of enabled events (or equivalently, disjunction of disabled events) is considered. The control action given as a result of the fusion of the individual behaviors of both supervisors is $S_{dec}(s)$. Thus, for n decentralized supervisors, the global control map $S_{dec}(s) : L(G) \longrightarrow 2^{\Sigma}$ is defined as follows:

$$S_{dec}(s) = \bigcap_{i=1}^{n} S_i(s). \tag{8}$$

According to Yoo and Lafortune (2002), such a supervisory control for G exists and is non-blocking if:

- K is controllable with respect to $L(G)$ and Σ_u,
- K is $L_m(G)$-closed,
- K is CP-coobservable with respect to $L(G)$, the set of locally observable events $\Sigma_{o,i}$ and the set of locally controllable events $\Sigma_{c,i}$ $(i = 1, \dots, n)$.

The language K is said to be CP-coobservable if for all string $s \in \overline{K}$ and for all event $\sigma \in \Sigma_c = \bigcup_{i=1}^{n} \Sigma_{c,i}$, at least one of the supervisors that can control σ knows unambiguously that it must disable σ. Moreover, the synthesized supervisors are minimally restrictive if their closed-loop language is equivalent to that of a monolithic supervisor ($L(S/G) = \bigcap_{i=1}^{n} L(S_i/G)$ and $L_m(S/G) = \bigcap_{i=1}^{n} L_m(S_i/G)$). If this is not the case, a solution is given in Overkamp and van Schuppen (2000).

3. Supervisory control of an HVDC link

Existing HVDC systems have been generally limited to point-to-point links (Fig. 3) operated by human action. The future integration of multi-terminal DC (MTDC) grids with multiple interconnected MMC stations represents a huge change in the way power systems are operated. It is thus important to correctly abstract the physical behavior of the grid components in order to develop a discrete control structure capable of interacting with the controlled station that evolves in continuous time. Then, later in this section and following a modular approach, the grid plant is constructed as the composition of several stations and a centralized supervisor is synthesized. Finally, the centralized supervisor is decentralized into local supervisors that communicate between them, in order to reduce the size of the automata to be implemented. Thus, a supervisory control for the start-up of a point-to-point link is obtained. The material in this section is largely borrowed from Romero Rodríguez et al. (2017).

3.1. Modeling of a controlled station

Due to the MMC topology (Lesnicar & Marquardt, 2003) and the DC cables nature, the HVDC grids have predominantly a capacitive behavior. In consequence, any variation of the DC voltage can be interpreted as the charging/discharging of an equivalent grid capacitor. Therefore, in the case of finite-time voltage variations, the voltage always reaches a steady state. Based on this steady-state behavior, the system can be naturally abstracted into discrete models. Also, we consider that all events are observable, since the state of the system can be inferred from the current and voltage measurements.

Fig. 3 shows the controllable components in a point-to-point link of a symmetrical monopole topology (DeBoeck, Tielens, Leterme, & Hertem, 2013). On the DC side, the converter can be connected or disconnected from the HVDC cables (one for each pole) through the DC Circuit Breakers (DCCB). The same function is realized on the AC side by the AC Circuit Breakers (ACCB). The MMC is described in Lesnicar and Marquardt (2003). It consists of a set of capacitive submodules connected in six parallel arms. In order for the MMC to operate properly, these capacitors need to be pre-charged. When the control of the capacitors is active, the MMC is said to be deblocked. Otherwise, it stands in blocked state. As the current cannot be controlled in blocked state, the Pre-Insertion Resistor (PIR) module is active during the pre-charging in order to restrict the surge current within the safety limits. When the converter is controlled, the PIR module is deactivated.

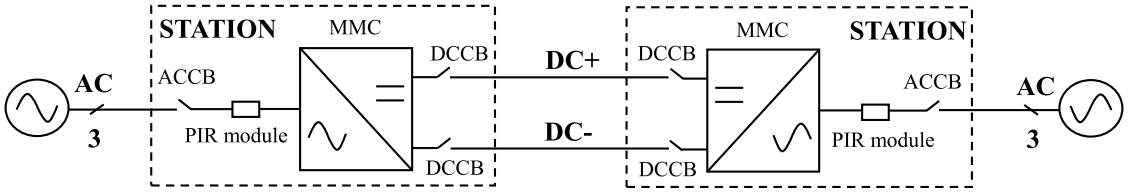


Fig. 3. Point-to-point HVDC architecture.

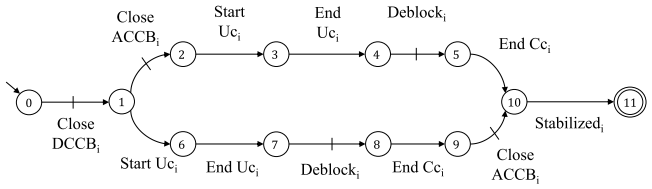


Fig. 4. Controlled station model G_i ($i \in \{1,2\}$).

At the initial state, the voltage in the submodules is zero, and so they need to be pre-charged to their rated voltage through a start-up procedure for the proper operation of the MMC. It is assumed in this paper that no fault occurs during the start-up. Among different start-up methods (Das, Nademi, & Norum, 2011; Gao, Li, Xu, Chu, Wang, & Li, 2014; Li, Xu, Zhang, Yang, Wang, Wang, & Xu, 2015; Yu, Ge, Lei, Wang, Yang, & Gou, 2013), we consider a self-excited charging strategy, similar to the one presented in Yu et al. (2013). Two stages can be identified in the start-up procedure: (i) an open-loop charging phase where the MMC is uncontrolled and the capacitors control is blocked; and (ii) after the capacitor has been charged sufficiently to deblock the MMC, a closed-loop charging phase where the MMC capacitors are controlled. Since both MMC are charged from one AC grid, they have different roles depending on their position: when the MMC is connected to the supplying AC side, it is called source converter; on the contrary, when the MMC is charged passively from the DC link and disconnected from its AC side during start-up, it is called remote converter.

Following the formalisms in Section 2, we define the model G_i ($i \in \{1,2\}$) in Fig. 4 for the station in source role (string of events in the upper path) and remote role (string of events in the lower path). The transitions crossed by a line are activated by the occurrence of a controllable event, while those uncrossed are labeled by an uncontrollable event. It is therefore possible to derive from the state transition diagram in Fig. 4 the set of events $\Sigma_{G_i} := \left\{\Sigma_{c\,G_i}, \Sigma_{uc\,G_i}\right\}$, with $\Sigma_{c\,G_i}$ = (Close DCCB$_i$, Close ACCB$_i$, Deblock SM$_i$) and $\Sigma_{uc\,G_i}$ = (Start Uc$_i$, End Uc$_i$, End Cc$_i$, Stabilized). The initial state is $0 \in X_{G_i}$ with X_{G_i} being the set of states. As both paths in the state transition diagram direct to the marked state 11, the language marked by G_i is non-blocking: $L(G_i) = \overline{L_m(G_i)}$. The list of transitions in G_i for each role, along with their physical meaning, is presented next:

- $f(0,\ \text{Close DCCB}_i) = 1$. In state 0, all the circuit breakers of the station are open. This transition represents the closing of the local DCCB. The closure of the DCCB is necessary for the DC cables to be charged by the source converter, or for the remote converter to be charged by the DC cables.
- $f(1,\ \text{Close ACCB}_i) = 2$. This transition represents the connection of the source MMC to its AC side. It is up to the operator to determine which AC grid is to be used to charge the HVDC system. The activation of the PIR module can also be associated to this event.
- $f(2,\ \text{Start Uc}_i) = 3$. After the closure of the ACCB, the current enters the HVDC link from the feeding AC grid. This current creates a voltage rise in the MMC capacitors and, upon its detection, the uncontrollable event Start Uc$_i$ is generated.
- $f(1,\ \text{Start Uc}_i) = 6$. If there is a current circulating through the remote MMC capacitors while it stands in blocked state and that the ACCB has not been closed, the measured voltage rise means a distant station has been connected to the corresponding AC grid, and the converter is being charged from the DC link.
- $f(3,\ \text{End Uc}_i) = 4$ and $f(6,\ \text{End Uc}_i) = 7$. The End Uc$_i$ event is generated when the voltage measured in the MMC capacitors reaches a steady state at the end of the open-loop energization and that a certain voltage threshold is attained. This voltage threshold is previously fixed and is different between the source and remote roles.

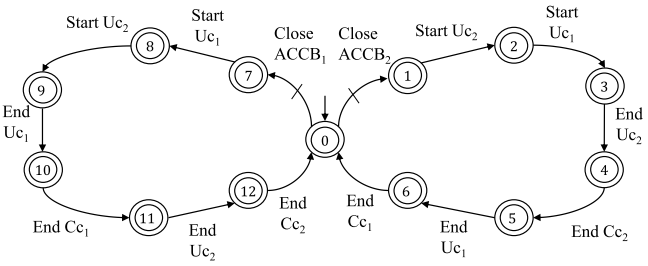


Fig. 5. Physical constraints G_c.

- $f(4,\ \text{Deblock SM}_i) = 5$ and $f(7,\ \text{Deblock SM}_i) = 8$. The voltage level reached at the end of the uncontrolled charging allows the MMC capacitors to be controlled, which effectively starts the closed-loop energization. The deactivation of the PIR module can also be associated to this event.
- $f(5,\ \text{End Cc}_i) = 10$ and $f(8,\ \text{End Cc}_i) = 9$. This event is generated upon arrival of the MMC capacitors voltage to the steady state at the end of the closed-loop energization. The voltage threshold is the same in both roles, and it corresponds to the rated voltage of the converter.
- $f(9,\ \text{Close ACCB}_i) = 10$. This transition represents the connection of the remote MMC to its AC grid, at the end of the controlled charging.
- $f(10,\ \text{Stabilized}_i) = 11$. The station is not fully operational until the voltages in the MMC and the DC cables are stabilized. Once the measured voltages correspond to the rated value and no large oscillations are observed, the event Stabilized is generated and the marked state 11 is reached.

3.2. Centralized architecture for start-up control

From the G_i model of each station, an automaton for the entire grid can be built through parallel composition. The parallel composition is an operation between automata denoted by $\|$ (Cassandras & Lafortune, 2008). Using the Supremica software (Akesson, Fabian, Flordal, & Malik, 2006), we are able to generate a new plant automaton $G' = G_1 \parallel G_2$ for the entire point-to-point system, where $\Sigma_{G'} = \Sigma_{G_1} \cup \Sigma_{G_2}$. The parallel composition of G_1 and G_2 is the automaton $G' = G_1 \parallel G_2 := Ac(X_{G_1} \times X_{G_2}, \Sigma_{G_1} \cup \Sigma_{G_2}, f, \Gamma_{1\|2}, \left(x_{0G_1}, x_{0G_2}\right),\ X_{mG_1} \times X_{mG_2})$, where $f\left(\left(x_{G_1}, x_{G_2}\right), \sigma\right) :=$

$$\begin{cases} \left(f_1\left((x_{G_1}), \sigma\right), f_2\left((x_{G_2}), \sigma\right)\right) & \text{if } \sigma \in \Gamma_1(x_{G_1}) \cap \Gamma_2(x_{G_2}), \\ \left(f_1\left((x_{G_1}), \sigma\right), x_{G_2}\right) & \text{if } \sigma \in \Gamma_1(x_{G_1}) \setminus \Sigma_{G_2}, \\ \left(x_{G_1}, f_2\left((x_{G_2}), \sigma\right)\right) & \text{if } \sigma \in \Gamma_2(x_{G_2}) \setminus \Sigma_{G_1}, \\ \text{undefined} & \text{otherwise,} \end{cases}$$

and therefore $\Gamma_{1\|2}\left(x_{G_1}, x_{G_2}\right) = [\Gamma_1(x_{G_1}) \cap \Gamma_2(x_{G_2})] \cup \left[\Gamma_1(x_{G_1}) \setminus \Sigma_{G_2}\right] \cup \left[\Gamma_2(x_{G_2}) \setminus \Sigma_{G_1}\right]$.

Since its event set only includes local event, G_i is valid for both the source and the remote converter. However, even though local models are valid for local control, they do not cover all the behaviors that appear in the system when several stations are coupled. Thus, the language $L(G')$ contains illegal strings either because they lead to states that are physically impossible or because they violate some safety constraints that we wish to impose. The dependence in the ordering of events between the linked stations is not captured in $L(G_i)$ and it is thus necessary to define a plant G_c that models the physical constraints introduced by the DC link (Fig. 5) and that completes the grid plant modeling obtained by composition of the station automata.

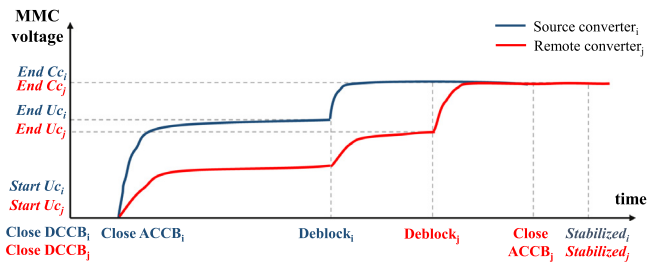


Fig. 6. MMCs voltage behavior during start-up.

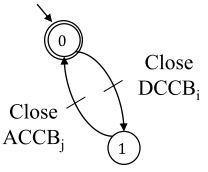


Fig. 7. Specification H_{G_i} $(i = 1, 2;\ j = 1, 2;\ i \neq j)$.

Each string in G_c represents the order in which the MMC reaches the different steady-state voltages, depending on its role as the source or remote converter. For example, if MMC_1 is the source converter, the voltage increase in its capacitors (Start Uc_1) will naturally be detected first in its arms and later in the remote MMC arms. Then, as MMC_2 is dependent on the HVDC grid, it will logically reach the end of the open-loop and closed-loop (End Uc_2, End Cc_2) charging later than the source MMC. Finally, it is necessary to distinguish the $Stabilized_i$ event between the two stations in local model G_i because in a complex network all voltages might not stabilize at the same time. In our case, as there is only one line, the two stations stabilize simultaneously once both AC grids are interconnected and the power balance is established; so $Stabilized_1$ and $Stabilized_2$ are merged into one common event Stabilized. All the states are marked because the plant models the physical behaviors imposed by the HVDC link. Then the global grid model G is obtained as $G = G' \parallel G_c$.

Given the capacitive behavior of the MMC and the HVDC cables, a voltage evolution similar to the one presented in Fig. 6 is expected for the start-up of a point-to-point link for the considered strategy. There exist multiple strings of events that could occur during the start-up but, for the considered strategy, only those shown in Fig. 6 originate an acceptable voltage behavior in the capacitors of the MMCs. In order to obtain the desired behavior, the set of strings of $L_m(G)$ must be restricted within the subset $K \subseteq L_m(G)$ according to the control specifications that we wish to enforce on the language generated by G.

Conforming to the SCT, these specifications are declared in the form of specification automata H_{G_i} $(i \in \{1, 2\})$. The automata H_{G_i} in Fig. 7 prevent the two stations from connecting to their respective AC grid before all the DCCBs in the HVDC link are closed, thus ensuring a safe start and a complete energization of the DC cables and both MMCs. The global specification automaton $H_G = H_{G_1} \parallel H_{G_2}$ is declared, with Σ_{H_G} = (Close $DCCB_1$, Close $DCCB_2$, Close $ACCB_1$, Close $ACCB_2$).

According to the Supervisory Control Theory (Cassandras & Lafortune, 2008), to enforce the correct alternation of events during the start-up imposed by H_G with respect to the global grid plant G, we synthesize a Centralized Grid Supervisor CGS (Fig. 8), that dynamically enables or disables controllable events of $L_m(G)$ to respect the specification H_G. The automaton $R_{CGS} = G \parallel H_G$ in Fig. 9 is a realization of CGS, such that $L_m(R_{CGS}) = L_m(CGS/G)$ and $L(R_{CGS}) = L(CGS/G)$. In our case the admissible marked language is obtained by forming $K = \overline{L(H_G)} \cap L_m(G)$ which is guaranteed to be $L_m(G)$-closed. In H_G, all forbidden events are controllable, so the controllability condition is satisfied.

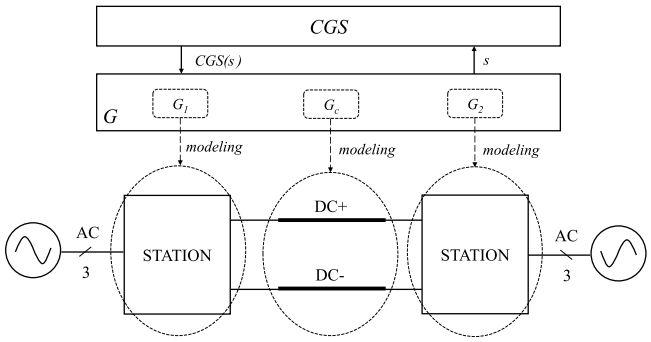


Fig. 8. Feedback loop for centralized supervisory control.

3.3. Decentralized architecture for start-up control

As the grid size increases, so does the size of the centralized supervisor. Given that the implementation of such automata might be difficult, other control architectures minimizing the size of the supervisory control automata need to be considered. Because of the need for coordination between stations in HVDC systems, a decentralized control (Lin & Wonham, 1988; Rudie & Wonham, 1992) that enforces local control and minimizes the number of events to be transmitted should be taken into account. In this way, the events that are not critical for the tracking of the system state can be removed from the supervisor alphabet while ensuring a correct operation.

The decentralized control structure is built on the joint action of two local Decentralized Grid Supervisors (DGS). Formally, given admissible supervisors DGS_1 and DGS_2, each defined for G, we define the decentralized supervisor S' corresponding to the intersection of all DGS_i as follows: $S'(s) := DGS_1(s) \cap DGS_2(s)$ (Fig. 2). The controlled system generates $L(S'/G) = L(DGS_1/G) \cap L(DGS_2/G)$ and marks $L_m(S'/G) = L_m(DGS_1/G) \cap L_m(DGS_2/G)$. The strings observed in the local station suffice for the control of the local station.

However, given that the specification H_G must be respected in order to obtain an admissible start-up, the events in the alphabet Σ_{H_G} must be communicated between DGS_1 and DGS_2 so that the CP-coobservability property is satisfied. Therefore the local alphabets $\Sigma_1 = \Sigma_{G_1} \cup \Sigma_{H_G}$ and $\Sigma_2 = \Sigma_{G_2} \cup \Sigma_{H_G}$ are determined. On these alphabets we calculate DGS_1 and DGS_2 such that $L(DGS_1) = P_1(L(CGS/G))$ and $L_m(DGS_1) = P_1(L_m(CGS/G))$. Similarly, $L(DGS_2) = P_2(L(CGS/G))$ and $L_m(DGS_2) = P_2(L_m(CGS/G))$. P_1 and P_2 are obtained by the projection operation $P_i : \Sigma_G^* \to \Sigma_i^*$ $(i \in \{1, 2\})$.

The synthesized supervisors are minimally restrictive in that $L(DGS_1/G) \cap L(DGS_2/G) = L(CGS/G)$ and that $L_m(DGS_1/G) \cap L_m(DGS_2/G) = L_m(CGS/G)$. In practice, the realizations R_{DGS_1} and R_{DGS_2} of DGS_1 and DGS_2 are two observers (Cassandras & Lafortune, 2008) for R_{CGS} with the partitions $\Sigma_{R_{CGS}} = \Sigma_G = \Sigma_1 \cup \Sigma_{uo1}$ and $\Sigma_{R_{CGS}} = \Sigma_G = \Sigma_2 \cup \Sigma_{uo2}$, where $\Sigma_{uo1} = \Sigma_R \setminus \Sigma_1$ and $\Sigma_{uo2} = \Sigma_R \setminus \Sigma_2$. As the control actions of DGS_1 and DGS_2 are limited to the first states, where a local event is forbidden, it is therefore possible to reduce the local supervisors (Vaz & Wonham, 1986), thus obtaining as a result the realizations $R_{DGS_{1r}}$ and $R_{DGS_{2r}}$ that minimize the number of states such that $L(R_{DGS_{ir}}/G) = L(R_{DGS_i}/G)$ and $L_m(R_{DGS_{ir}}/G) = L_m(R_{DGS_i}/G)$ for $i \in \{1, 2\}$. Fig. 10 presents the automaton $R_{DGS_{ir}}$, where the events forbidden by the supervisor are labeled by the dashed transitions.

The number of events in the supervisor's alphabet have been effectively limited while ensuring an admissible start-up. Also, even if there exist communication delays, grid stability is not affected by control decentralization as the voltage is in steady state when the control actions associated to the 4 controllable events included in Σ_{H_G} are generated. Finally, in the case a short-circuit fault occurs, fast protection algorithms (Loume, Bertinato, Raison, & Luscan, 2017) would react locally to bring the system to a safe steady state. Then, the supervisory control could resume its control actions.

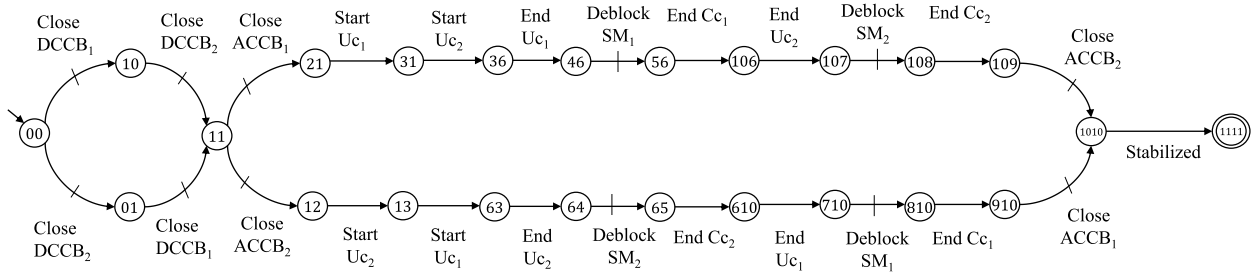


Fig. 9. Automaton R_{CGS} such that $L_m(R_{CGS}) = L_m(CGS/G)$ and $L(R_{CGS}) = L(CGS/G)$.

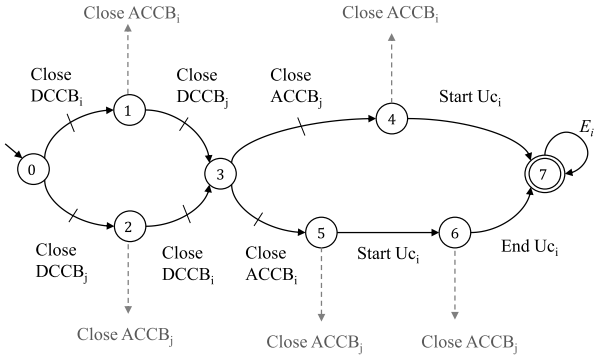


Fig. 10. Automaton R_{DGS_i} for supervised start-up ($i \in \{1, 2\}$).

4. Supervisory control implementation method

As we wish to integrate the obtained theoretical supervisors in power system simulation software such as EMTP-RV, some transformations have to be made before they are implemented as C code. Indeed, the gap between the hypothesis made in the SCT and the real conditions in computer-based environments has to be filled. First in this section, a literature review on the difficulties encountered when implementing SCT-based supervisors is presented. Then, the implementation method proposed by the authors is detailed and applied to the automata for the start-up control of an HVDC link obtained in the previous section. Finally, the simulation results for the considered case study are shown.

4.1. Implementation problems

Fabian and Hellgren (1998) identified the main problems encountered during the implementation of abstract supervisors into PLCs. Recently, a thorough review on the synthesis and implementation of logic controllers has been made in Zaytoon and Riera (2017). Given that PLCs are industrial digital computers, these problems are extensible to all computer-based environments and can be resumed in the following:

- *Avalanche effect*: occurs when a change in the value of an input signal activates an event that triggers many successive transitions; the controller then jumps through an arbitrary number of states.
- *Simultaneity*: relates to the incapability of the controller to distinguish the order of occurrence of two events within the same scan cycle, because the input signals are read at the beginning of the scan cycle.
- *Causality*: arises when the implemented supervisor is forced to generate commands so that the plant can evolve in response, as opposed to the SCT, where the plant is supposed to generate all the events in its event set.
- *Choice*: appears when the controller has to choose between the generation of two or more concurrent commands, thus introducing a non-determinism issue.
- *Inexact synchronization*: is due to the communication delay between the plant and the controller. Thus, the occurrence of an event in the former is not immediately reproduced in the latter, unlike in SCT (Kumar, Garg, & Marcus, 1991). This could lead to a wrong control action by the controller.

While the avalanche effect issue can be resolved by means of clever programming and resetting the controller signals (Fabian & Hellgren, 1998), the simultaneity and inexact synchronization problems are inherent to this type of implementation. Although the *interleave insensitivity* and the *delay insensitivity* properties are respectively introduced in Fabian and Hellgren (1998) and Balemi (1992) as a solution to the simultaneity and the inexact synchronization issues, such properties are not often satisfied in practice. In Leal et al. (2012), the authors propose to associate the events that may pose such problems to PLC variables that force an interruption in the hardware, and interrupting the processing of further inputs in consequence. However, many PLCs do not have this kind of variables, as pointed out by the authors. In this paper, because the implementation in a simulation software such as EMTP-RV is considered, the scan cycle of the control program is fixed to be significantly smaller than the time constants of the system. Nevertheless, because of the fast dynamics of HVDC grids, other hardware such as microcontroller may be more suitable than PLCs for real-time applications.

Regarding the causality problem, in a real system, as opposed to the SCT, the plant does not generate all the events in its event set, but it rather evolves in response to given commands. In consequence, Balemi (1992) introduces the *forcing event* approach: the controller generates the commands (identified as the controllable events by the author), while the plant generates the responses of the system (i.e. the uncontrollable events). Because the controller has to generate commands in accordance with the supervisor's control map but also with the physical limitations of the plant, the automaton recognizing the supremal controllable language is implemented. This implies that only centralized architectures can be realized. In opposition, Charbonnier et al. (1995) introduces the *supervised control* approach, which clearly differentiates the supervision and command generation tasks within the controller. The supremal controllable language is ensured by a closed-loop control, as an automaton modeling the plant generates all the events from the I/O signals, while the supervisor disables them according to its control policy. This separation of tasks allows to modify the supervisor structure independently of the command generation, thus offering the possibility to integrate different architectures, such as the modular control in de Queiroz (2002), Leal et al. (2012), Lopes, Leal, Rosso, and Harbs (2012) and Vieira et al. (2017).

Furthermore, as previous works in the literature consider the case where the controller sends to the plant all generated commands that may be concurrent themselves, different criteria have been proposed for the extraction of a deterministic controller in order to resolve the choice issue. In other words, such criteria erase from the controller the command generation paths that could introduce non-determinism (Dietrich, Malik, Wonham, & Brandin, 2002; Lopes et al., 2012; Malik, 2002). In Leal et al. (2012), the authors propose the command generation to be selected randomly by an external routine. In power transmission systems, however,

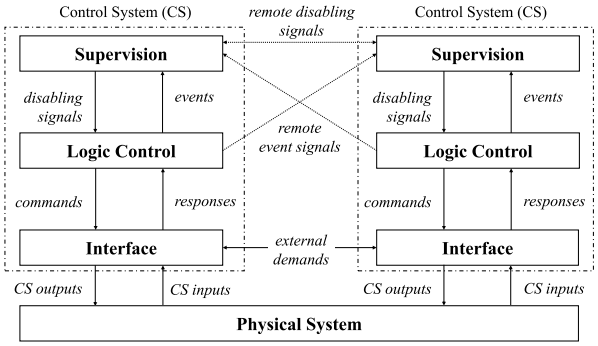


Fig. 11. Control structure in a decentralized architecture.

it is preferred to let the TSO decide which of the alternative commands is generated depending on each scenario. Such decision could be the result of an offline study done by the operator before the grid is put into operation. Thus, previous solutions would limit the range of decision or could eventually lead to unstable situations if chosen randomly. In consequence, the notion of *triggering* is introduced later in this paper.

Finally, the implementation of a supervisory control for power transmission systems, which are continuous-time systems, implies an additional difficulty with respect to prior art, which mainly treats manufacturing systems. Indeed, in manufacturing systems, because the controller is typically connected to the sensors and actuators of the physical system through Boolean-valued signals, the responses of the system are generated when a change in the value of a Boolean variable is detected. In time-driven systems in general, and in power transmission networks in particular, this is no longer valid, as the responses are generated from continuous-time signals, such as voltage and current measurements. In this case, the occurrence of an event is detected when the state trajectory of the considered variables crosses a certain threshold (Zhao, Mi, & Ren, 2006). The controller should then integrate a low-level interface that details within the control system the continuous-time conditions indicating the occurrence of a discrete event.

4.2. Proposed implementation method

As stated in Section 3, a decentralized control architecture is best suited for the control of HVDC systems and so the *supervised control* approach (Charbonnier et al., 1995) is considered. The decentralized control system proposed in this paper is formed by local controllers in each station. Each controller is composed of the Supervision, Logic Control and Interface levels (Fig. 11), which contain the supervisory control system under the form of C code functions. These control levels communicate through signals between them within the local controller and with the remote controllers. The rest of the section details the tasks shown in Fig. 12.

4.2.1. Model preparation

In order for the different control levels to communicate, an input and output alphabet is attached to the automata, which transforms them into Moore or Mealy machines. However, the transition functions of the theoretical automata may need to be modified so that the language recognized by the automata remains unchanged and the state-machines remain deterministic when the I/O alphabets are added (Task 1 of Fig. 12). The algorithms proposed by Vieira et al. (2017) for the conversion of the plant automata in view of their transformation into Moore machines (Moore, 1956) are considered in this paper. A Moore machine has the property that its output function ω depends only on the current state of the machine, but not on the transition reaching the state, i.e. $\omega: X \rightarrow O$. The supervisor automata can be directly transformed into Moore machines, because the disabling signals

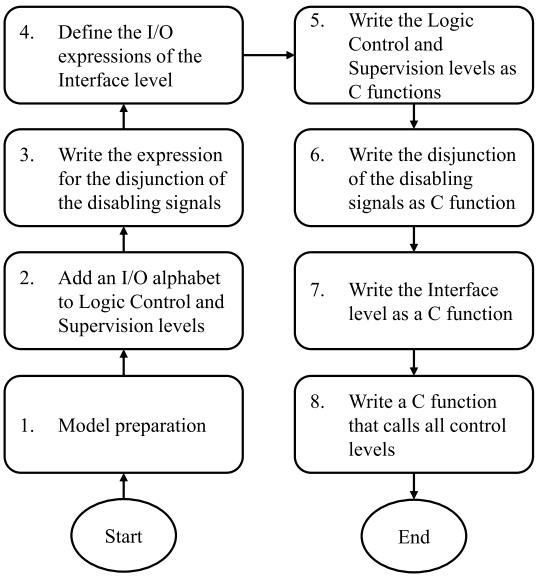


Fig. 12. Workflow of the implementation method.

generated in the states, which form the output alphabet O of the Moore machine S_i^M ($i \in \{1, \ldots, n\}$), are independent of the transition's input alphabet I. This is not the case for the plant automata G_i, and thus the two algorithms proposed in Vieira et al. (2017) convert the plants G_i into equivalent automata G_i'' where each state is reached by only one event. Thus, the Moore machines G_i^M remain deterministic when adding the I/O alphabets as the outputs generated at each state are known and differentiated for a given set of inputs.

Case study: the algorithms defined in Vieira et al. (2017) are applied to the theoretical controlled station model of Fig. 4. Because the plant G_i does not present any self-loop transitions, the first algorithm does not apply, as $G_i = G_i'$ and only the second algorithm is used. The converted plant G_i'' is shown in Fig. 13. As can be observed, the state 10 of G_i is now decoupled into states (10, EndCc_i) and (10, CloseACCB_i). In this way, each state is reached by a unique event, except for the initial state (0,∅), which is not associated to any event. The theoretical decentralized supervisor of Fig. 10 does not need any transformation before the addition of an input and output alphabet.

4.2.2. Supervision level

The Supervision level contains a realization of the supervisor automaton. It prevents the Logic Control level from generating prohibited strings of commands, so that the real plant respects the behavioral specifications defined by the developer. The Supervision level takes as inputs the Boolean *events* signals generated by the Logic Control (Fig. 11). For this, the input and output alphabets are added to the supervisor automaton (Task 2 of Fig. 12). The input expressions labeling the transitions of the Moore machines S_i^M ($i \in \{1, \ldots, n\}$) are defined by the expression (E1):

(E1) $\sigma = 1$

This condition is fulfilled whenever the associated event signal σ is activated by the Logic Control. On the other hand, the output expression (E2) is formed by the local disabling signals σdi in the set of disabling signals D_M ($\sigma di \in D_M$), generated in any given state whenever an event $\sigma \in \Sigma$ must be disabled next by S_i^M:

(E2) $\sigma di = 1$

In a decentralized architecture, the set of remote controllers with σ in their alphabet will similarly generate in their states a remote disabling signal $\sigma dj \in D_M$, with $j \in \{1, \ldots, n\}$ and $j \neq i$. Then, a logical function that merges the *remote disabling signals* and the *local disabling signals* associated to an event σ is defined by following the conjunctive fusion

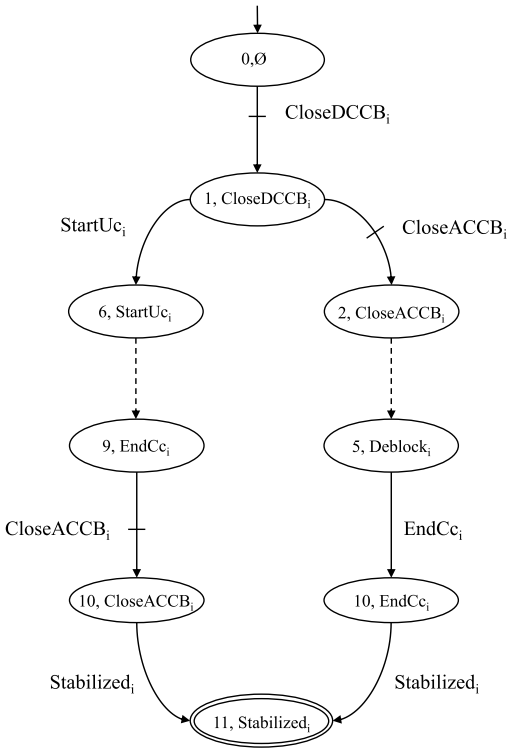


Fig. 13. Converted plant G_i'' ($i \in \{1,2\}$).

```
void supervisor(...){
  static int state = 0;

  switch(state){
      case 1:
          *CloseACCBjd_i = 1;
          *CloseACCBid_i = 1;
          if(CloseDCCBi){
          state = 1;
          }
          if(CloseDCCBj){
          state = 2;
          }
          break;
        ...
  }
}
```

Listing 1: Supervision function

rule described in Section 2.2 (Task 3 of Fig. 12). For this, a signal $\sigma d \in D_M$ is associated in each control system to the controllable events that are generated locally at the considered control system. The merged disabling signal σd is activated when at least one of the disabling signals ($\sigma d1, \ldots, \sigma dn$) deactivates the occurrence of the corresponding event σ. This is shown in expression (E3) for the case of two supervisors:
(E3) IF ($\sigma d1 = 1$ OR $\sigma d2 = 1$) THEN $\sigma d = 1$

Therefore, in order to achieve CP-coobservability, the common events in the supervisor's alphabets are transmitted from the Logic Control level of the local control system to the Supervision level of the remote control system so that at least one of the controllers knows unambiguously that it has to disable a given event (*remote event signals* in Fig. 11). The expressions (E1), (E2) and (E3) are defined as generic expressions that can be interpreted and implemented as C code.

Case study: the expressions (E1), (E2) and (E3) are associated to the supervisor automaton in Fig. 10 under the form of C code by following

```
void disabling_signals(...){

  *CloseACCBid = CloseACCBid_i || CloseACCBid_j;

  *CloseDCCBid = CloseDCCBid_i || CloseDCCBid_j;

  ...

}
```

Listing 2: Disabling signals disjunction function

the criteria of controllability and CP-coobservability described before, as shown in Listing 1. First, an integer variable υ_σ is associated to each event σ in the automaton $R_{DGS_{ir}}$ ($i \in \{1,2\}$) and the expression (E1) is verified for the υ_σ corresponding to a given transition in $R_{DGS_{ir}}$. A set of integer variables $\upsilon_{\sigma di}$ ($i \in \{1,2\}$) has also been associated to the set of events of $R_{DGS_{ir}}$. A value of 1 is assigned to each of the $\upsilon_{\sigma di}$ corresponding to the events to be disabled next by the supervisor (E2). In Listing 1, the states of $R_{DGS_{ir}}$ are expressed as switch cases in a switch statement, while the transitions are represented by the if statements that allow to modify the switch case value. All these statements are encompassed in a function called *supervision*. While the value of $\upsilon_{\sigma di}$ is modified in *supervision*, these variables are also input of another function and so are coded as pointers. This is not the case for υ_σ as they are generated at the Logic Control level and cannot be modified by the *supervision* function. The *state* variable in Listing 1, however, is local to the function and so no pointer is needed.

In order for the decentralized architecture to meet the CP-coobservability requirement, the *disabling signals disjunction* function is created, as shown in Listing 2. Here, the variables $\upsilon_{\sigma di}$ representing the disabling signals to be transmitted between the two decentralized controllers are merged by means of the disjunction of the disabling signals of each individual controller (E3). The result of this disjunction is assigned to an integer variable $\upsilon_{\sigma d}$ that is later transmitted to the Logic Control level and so it is a pointer.

The *supervisor* and *disabling signals disjunction* functions constitute what has been called the Supervision level of the local control system (see Fig. 11).

4.2.3. Logic control level

The Logic Control level contains a realization of the plant automaton. It is in charge of forcing controllable events into the real plant through Boolean-valued *commands* signals, while taking into account the control map of the Supervision level and the physical plant evolution through the *disabling signals* and the *responses* signals, respectively. In view of the complexity of power systems operation, the supervisory control should not be isolated from the TSO so that the latter is able to choose, by means of controllable events, which of the control strategies is best to be triggered at a given state. Obviously, the TSO's requests should be expressed before the system is put into operation in order to meet the fast response requirement due to the DC voltage dynamics. Therefore, the implemented controller should allow for the generation of some controllable events and the associated commands following external demands, as opposed to previous works, where it is assumed that the commands are autonomously generated by the controller in a predetermined manner (Balemi, 1992; Charbonnier et al., 1995; de Queiroz, 2002; Vieira et al., 2017).

Hence, the notion of *triggering* is introduced. The set of events triggered internally by the controller, denoted by Σ_{it}, is distinguished from the set of externally triggered events, denoted by Σ_{et}. Furthermore, the notion of triggering allows the implemented controller to be deterministic at all times since it is up to the TSO to choose which command should be generated at a given state when two or more controllable events are enabled. As uncontrollable events Σ_u are generated by the

physical system itself, they are externally triggered by definition ($\Sigma_u \subseteq \Sigma_{et}$). On the other hand, a subset of the controllable events Σ_c is internally triggered ($\Sigma_{it} \subseteq \Sigma_c$) and it is to be specified by the developer during implementation. Thus, for a set of events $\Sigma = \Sigma_u \cup \Sigma_c$, the following relation is satisfied:

$$(\Sigma_u \subseteq \Sigma_{et}) \wedge (\Sigma_{it} \subseteq \Sigma_c) \wedge (\Sigma_{it} \cup \Sigma_{et} = \Sigma).$$

Thus, the input condition associated with a transition in the Moore machine G_i^M ($i \in \{1, \ldots, n\}$) is defined by the controllability and triggering conditions of the event labeling them (Task 2 of Fig. 12). Whenever an event $\sigma \in \Sigma$ associated with such a transition is controllable ($\sigma \in \Sigma_c$) and triggered locally by the controller ($\sigma \in \Sigma_{it}$), the only condition for the transition to occur is that the disabling signal σd of the event σ that belongs to the set of disabling signals D_M ($\sigma d \in D_M$) is not activated, giving as a result the input expression (E4):

(E4) NOT σd

Whenever the event σ is controllable ($\sigma \in \Sigma_c$) and triggered by a request external to the controller ($\sigma \in \Sigma_{et}$), the transition is crossed if the response signal $rsp\sigma$ in the set of the response signals R_M ($rsp\sigma \in R_M$), which indicates the occurrence of an external request, is activated, and that the corresponding disabling signal σd is deactivated. The Boolean expression (E5) is then defined:

(E5) $rsp\sigma = 1$ AND NOT σd

Finally, whenever the event σ that labels the transition is uncontrollable ($\sigma \in \Sigma_u$), and thus externally triggered by definition, the transition is crossed when the response signal $rsp\sigma \in R_M$ indicates the occurrence of an uncontrollable event, after the continuous-time *CS inputs* fulfill certain predefined conditions. Expression (E6) is used in this case:

(E6) $rsp\sigma = 1$

On the other hand, the output actions associated to each state of G_i^M are defined by the controllability of the unique event reaching the state, with the exception of the initial state x_0, which is not associated to any event. In all cases, the Logic Control generates an event signal σ included in the set Σ_M ($\sigma \in \Sigma_M$) so that the implemented supervisors S_i^M are updated to the current system state. Furthermore, if the event reaching a given state is controllable, the command signal $cmd\sigma$ in the set C_M ($cmd\sigma \in C_M$) is sent to the Interface level so that the *CS outputs* signals are modified according to the desired control action, as defined in expression (E7):

(E7) $\sigma = 1$ AND $cmd\sigma = 1$

On the contrary, no command signal is generated in the case of uncontrollable events, as can be seen in expression (E8):

(E8) $\sigma = 1$

Similarly to the expressions defined in the previous section, expressions (E4) to (E8) have been defined as generic expressions that can be implemented as C code.

Case study: the converted plant G_i'' of Fig. 13 is transformed into a corresponding Moore machine G_i^M by means of the defined Boolean input and output expressions ((E4)-(E8)). The G_i^M Moore machine is then implemented as a C code function within the control program, named *logic control*, as shown in Listing 3.

In Listing 3, the Boolean expression of the type (E5) for the event $\text{CloseDCCB}_\text{i}$ labeling the transition from the state (0,∅) to the state (1, $\text{CloseDCCB}_\text{i}$) is expressed by means of two types of integer variables: one for the response signal ($v_{rsp\sigma}$) and the other for the disabling signal ($v_{\sigma d}$). Similarly, according to expression (E7), two types of integer variables are associated to the output signals generated at the state (1, $\text{CloseDCCB}_\text{i}$): one for the command signal ($v_{cmd\sigma}$), as the event reaching the state is controllable; the other for the event signal (v_σ). Finally, it should be noted that it suffices to rank first the input conditions of an uncontrollable event to indicate its priority over a controllable event, as the code is executed line by line during a scan cycle. In graphical languages (LD, SFC), on the other hand, because an additional variable is created in order to prevent the processing of controllable events during a scan cycle (Vieira et al., 2017), two cycles for the same task are required.

```
void logic_control(...){
  static int state = 0;

  switch(state){
      case 0:
      if(rspCloseDCCBi && !CloseDCCBid){
          state = 1;
      }
      break;

      case 1:
      *CloseDCCBi = 1;
      *cmdCloseDCCBi = 1;
      if(rspStartUci){
          state = 6;
      }
      if(rspCloseACCBi && !CloseACCBid){
          state = 2;
      }
      break;
      ...
  }
}
```

Listing 3: Logic control function

As in the *supervision* function, the states of G_i^M are expressed by means of a switch statement, while the transitions are represented by means of if statements. Also, given that the value of variables v_σ and $v_{cmd\sigma}$ is modified in the *logic control* function and shared between several C code functions, they are declared as pointers. The *logic control* function constitutes what has been called the Logic Control level of the local control system (see Fig. 11).

4.2.4. Interface level

In addition to the two levels defined in Charbonnier et al. (1995), a third level that interfaces the supervisory control system with the continuous-time physical system is added in Fig. 11. The Interface level is connected to the physical system through its sensors and actuators. Following the activation in the Logic Control module of the command signals associated to a certain sequence of activities, the Interface module will modify the control system outputs (*CS outputs*) sent to the actuators accordingly. On the other hand, when the value of the continuous-time signals (voltage, current, etc.) measured by the sensors and sent to the control system inputs (*CS inputs*) reaches a predefined threshold, a response signal indicating the occurrence of a particular event in the physical system is activated and sent to the Logic Control level. In addition, the Interface identifies if an external operator makes the choice of a specific control action (*external demands*) and generates the corresponding response. All these input and output expressions are system-specific and have to be written by the developer (Task 4 of Fig. 12).

Case study: the C function responsible for the interaction with the continuous-time system, named *interface*, is shown in Listing 4. A set of variables v_{in} of type double is associated to the continuous-time input signals of the local control system, such as the current and voltage measurements in Listing 4. As the actuators of the system are typically commanded through Boolean-valued signals, a set of integer variables v_{out} is associated to them.

In Listing 4, when a rise in the measured current is detected, a value of 1 is assigned to the respective response variable $v_{rsp\sigma}$. Likewise, if the converter voltage (`m_Vmmc`) and the local voltage measurement of the DC cable (`m_Vdc`) have reached their rated value of 320 kV, a value of 1 is assigned to the variable associated to the end of the charging. Finally, when the Interface detects an external demand by the operator to close the circuit breakers, which is an input variable

```
void interface(...){

  *rspStartUci = fabs(m_Idc) > 0;

  *rspEndCci = (fabs(m_Vdc) >= 320e3 && fabs(m_Vmmc)
    >= 320e3);

  *rspCloseDCCBi = reqDCCB;

  *rspCloseACCBi = reqACCB;

  *outDCCB = cmdCloseDCCBi;

  *outACCB = cmdCloseACCBi;

  ...

}
```

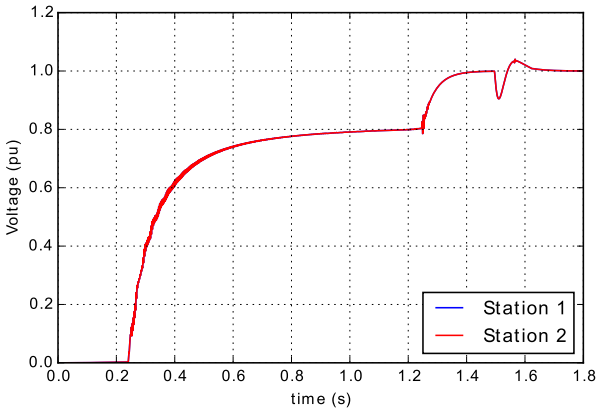
Listing 4: Interface function

```
void main(v_in,v_out){

 //Declare integer variables
 static int v_sigma = 0;
 static int v_sigma_di = 0;
 static int v_sigma_d = 0;
 static int v_cmd_sigma = 0;
 static int v_rsp_sigma = 0;

 //Reset local disabling signals
 v_sigma_di = 0;

 //Call supervision function
 supervision(v_sigma,v_sigma_di);

 //Merge disabling signals
 disabling_signals(v_sigma_di,v_sigma_d);

 //Reset local event signals
 v_sigma = 0;

 //Call logic control function
 logic_control(v_sigma_d,v_rsp_sigma,v_sigma,
     v_cmd_sigma);

 //Reset response signals
 v_rsp_sigma = 0;

 //Call interface function
 interface(v_in,v_cmd_sigma,v_out,v_rsp_sigma);

 //Reset command signals
 v_cmd_sigma = 0;

}
```

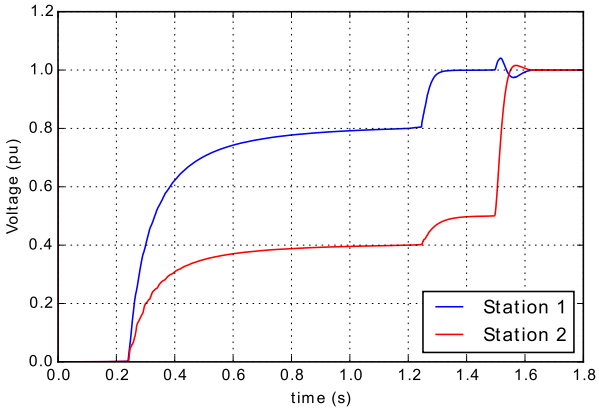
Listing 5: Main function

υ_{in}, a value of 1 is assigned to the associated response variable. This external triggering is awaited by the Logic Control so the command variable value can be modified. Then, the Interface function modifies in consequence the output integer variables sent to the physical breakers (outDCCB, outACCB).

4.2.5. Coordination of the control levels



(a) DC cables voltage during start-up.



(b) MMCs voltage during start-up.

Fig. 14. Simulation results.

The different C code functions written for each level of the supervisory control are now coordinated via the *main* function shown in the pseudo-code of Listing 5. The $\upsilon_{rsp\sigma}$, $\upsilon_{cmd\sigma}$, υ_{σ}, $\upsilon_{\sigma di}$ and $\upsilon_{\sigma d}$ variables used in the previous functions are declared. The term *static* is used so that they are not initialized to zero at each execution cycle. It is important to keep in mind, however, that they must be reset at each execution cycle so that the avalanche effect described in Fabian and Hellgren (1998) does not appear. Then, the functions defined above are called and the corresponding variables are given as parameters. In turn, the *main* function takes as parameters the input and output variables (υ_{in}, υ_{out}) associated to the control system I/O signals exchanged with the HVDC grid. In this way, the *main* C function calls all the functions within the control system at each execution cycle in the order described in Listing 5. All the C functions can then be implemented in the EMTP-RV software for the simulation of electric power systems. Because the software runs in Windows, the program of each decentralized controller can be implemented in EMTP-RV via a Dynamic-Link Library (DLL). Although DLLs cannot be instantiated, multiple copies of the same DLL can be used, which facilitates the integration of the program. The internal variables of the DLLs remain local to each copy if the program is coded with pointers, as it is the case in this paper. Otherwise, the use of global variables would oblige to name differently the variables in each copy.

4.3. Simulation results

The electrical circuit of Fig. 3 is designed in EMTP-RV using the software's specialized library of components. Given the degree of detail of this model and the impossibility to test the obtained controllers in a real system, the simulation is considered as a valid method for their

validation. The control program of each decentralized controller is then implemented via DLLs as explained in Section 4.2.5. The simulation results of the start-up of a point-to-point link are presented in Fig. 14. As it can be observed, the voltage behavior of the MMCs copies that of Fig. 6 as all prohibited actions have been effectively disabled by the decentralized control. Similarly, the voltage of the DC cables remains within the specifications and reaches its rated level in the end of the procedure. Hence, the start-up is effectively realized.

5. Conclusions and perspectives

In this paper, a method for the implementation of the supervisory control for HVDC systems is presented. The proposed method allows the controller to be separated into different levels of detail (i.e. Supervision, Logic Control and Interface), which increases the flexibility during the design phase. Moreover, the proposed method allows the implementation of a decentralized architecture, thus adapting to the characteristics of HVDC grids. The method is implemented in C code, as the simulation of the start-up of an HVDC link in the EMTP-RV software was targeted. Future implementation in a real-time simulation software such as HYPERSIM could also be considered with the proposed method. Also, because the I/O expressions of the implemented supervisors have been proposed in a generic form, the proposed method could be extended to Structured Text (ST) language (International Electrotechnical Commission, 2003), given the similarity of its syntax with C code (Basile & Chiacchio, 2007).

The proposed method has been applied to the start-up of a point-to-point link. The use of SCT has imposed the identification of the physical components behavior on the one hand and the requirements on the other hand. It is in the perspectives to develop the supervisory control in a mode-switching structure (Faraut et al., 2009) so that different control procedures, such as start-up, shutdown and post-fault restoration can be treated. Furthermore, the low-level interface that interacts between the discrete-controllers and the continuous-time system could be further refined. For instance, its interaction with complex observers could facilitate the integration of Model Predictive Control (Moradzadeh, Bhojwani, & Boel, 2011) to the supervision system.

Acknowledgment

This work is supported by the French Government under the program Investissements d'Avenir (ANE-ITE-002-01).

References

Akesson, K., Fabian, M., Flordal, H., & Malik, R. (jul 2006). Supremica - An integrated environment for verification, synthesis and simulation of discrete event systems. In *2006 8th international workshop on discrete event systems* (pp. 384–385).

Balemi, S. (1992). Control of discrete event systems: theory and application.

Basile, F., & Chiacchio, P. (2007). On the implementation of supervised control of discrete event systems. *IEEE Transactions on Control Systems Technology*, *15*(4), 725–739.

Cantarelli, M., & Roussel, J. -M. (2008). Reactive control system design using the supervisory control theory: evaluation of possibilities and limits. In *9th international workshop on discrete event systems, 2008* (pp. 200–205).

Cassandras, C., & Lafortune, S. (2008). *Introduction to discrete event systems*. Springer.

Charbonnier, F., Alla, H., & David, R. (1995). The supervised control of discrete event dynamic systems: A new approach. In *Proceedings of the 34th conference on decision and control (Cat.No.95CH35803) 1 (December)* (pp. 913–920).

Das, A., Nademi, H., & Norum, L. (2011). A method for charging and discharging capacitors in modular multilevel converter. In *IECON 2011-37th Annual Conference on* (pp. 1058–1062). IEEE Industrial Electronics Society.

DeBoeck, S., Tielens, P., Leterme, W., & Hertem, D. V. (2013). Configurations and earthing of HVDC grids. In *Power and energy society general meeting* (pp. 1–5).

Dietrich, P., Malik, R., Wonham, W. M., & Brandin, B. A. (2002). Implementation considerations in supervisory control. In *Synthesis and control of discrete event systems* (pp. 185–201). Springer.

Do, V. Q., Soumagne, J. C., Sybille, G., Turmel, G., Giroux, P., Cloutier, G., & Poulin, S. (1999). Hypersim, an integrated real-time simulator for power networks and control systems. In *Transactions of the ICDS 99*.

Fabian, M., & Hellgren, A. (1998). PLC-based implementation of supervisory control for discrete event systems. In *Proceedings of the 37th IEEE conference on decision and control, 1998, Vol. 3* (pp. 3305–3310).

Faraut, G., Piétrac, L., & Niel, E. (2009). Formal approach to multimodal control design: Application to mode switching. *IEEE Transactions on Industrial Informatics*, *5*(4), 443–453.

Gao, F., Li, Z., Xu, F., Chu, Z., Wang, P., & Li, Y. (2014). Startup strategy of VSC-HVDC system based on modular multilevel converter. In *Energy conversion congress and exposition* (pp. 1946–1952).

Gouyon, D., Pétin, J. -F., & Gouin, A. (2004). Pragmatic approach for modular control synthesis and implementation. *International Journal of Productions Research*, *42*(14), 2839–2858.

van Hertem, D., & Ghandhari, M. (2010). Multi-terminal VSC HVDC for the European super grid: Obstacles. *Renewable and Sustainable Energy Reviews*, *14*.

International Electrotechnical Commission (2003). Programmable controllers –Part 3: Programming languages. In *International standard IEC 61131-3*.

Kumar, R., Garg, V. K., & Marcus, S. I. (1991). On controllability and normality of DEDS. *Systems & Control Letters*, *17*, 157–168.

Lauzon, J., Mills, B., & Benhabib, S. (1997). An implementation methodology for the supervisory control of flexible manufacturing workcells. *Journal of Manufacturing Systems*, *16*(2), 91.

Leal, A. B., da Cruz, D. L. L., & Hounsell, M. d. S. (2012). PLC-based implementation of local modular supervisory control for manufacturing systems.

Lesnicar, A., & Marquardt, R. (2003). An innovative modular multilevel converter topology suitable for a wide power range. In *2003 IEEE bologna powertech - conference proceedings, Vol. 3*, (pp. 272–277).

Li, B., Xu, D., Zhang, Y., Yang, R., Wang, G., Wang, W., & Xu, D. (2015). Closed-loop precharge control of modular multilevel converters during start-up processes. *IEEE Transactions on Power Electronics*, *30*(2).

Lin, F., & Wonham, W. M. (1988). Decentralized supervisory control of discrete-event systems. *Information Sciences*, *44*(3), 199–224.

Lopes, Y. K., Leal, A. B., Rosso, R. S. U., & Harbs, E. (2012). Local modular supervisory implementation in microcontroller. In *Proc. of the 9th international conference of modeling, optimization and simulation, Vol. 9*.

Loume, D. S., Bertinato, A., Raison, B., & Luscan, B. (2017). A multi-vendor protection strategy for HVDC grids based on low-speed DC circuit breakers. In *13th IET international conference on AC and DC power transmission AC/DC* (pp. 2–7).

Mahseredjian, J., Dennetière, S., Dubé, L., Khodabakhchian, B., & Gérin-Lajoie, L. (2007). On a new approach for the simulation of transients in power systems. *Electric Power Systems Research*, *77*(11), 1514–1520.

Malik, P. (2002). Generating controllers from discrete-event models. *Proceedings Modelling and Verification of Parallel Processes (MOVEP)*, *33*, 7–342.

Moore, E. (1956). Gedanken-experiments on sequential machines. *Automata studies*, *34*, 129–153.

Moradzadeh, M., Bhojwani, L., & Boel, R. (2011). Coordinated voltage control via distributed model predictive control. In *Chinese Control and decision conference* (pp. 1612–1618).

Overkamp, A., & van Schuppen, J. H. (2000). Maximal solutions in decentralized supervisory control. *SIAM Journal on Control and Optimization*, *39*(2), 492–511.

de Queiroz, J. E. R. C. (2002). Synthesis and implementation of local modular supervisory control for a manufacturing cell. In *Proceedings sixth international workshop on discrete event systems* (pp. 377–382).

Ramadge, P. J., & Wonham, W. M. (1987). Supervisory control of a class of discrete event processes. *SIAM Journal on Control and Optimization*, *25*(1), 206–230.

Ramirez-Serrano, A., Zhu, S. C., Chan, S. K. H., Chan, S. S. W., Ficocelli, M., & Benhabib, B. (2002). A hybrid PC/PLC architecture for manufacturing-system control—theory and implementation. *Journal of intelligent manufacturing*, *13*(4), 261–281.

Rebours, Y. G., Kirschen, D. S., Trotignon, M., & Rossignol, S. (2007). A survey of frequency and voltage control ancillary services–Part I: Technical features. *IEEE Transactions on power systems*, *22*(1), 350–357.

Romero Rodríguez, M., Delpoux, R., Piétrac, L., Dai, J., Benchaib, A., & Niel, E. (2017). Supervisory control for high-voltage direct current transmission systems. *IFAC Proceedings Volumes (IFAC-PapersOnline)*, *50*(1), 12326–12332.

Rudie, K., & Wonham, W. M. (1992). Think globally, act locally: decentralized supervisory control. *IEEE Transactions on Automatic Control*, *37*(11), 1692–1708.

Vaz, A. F., & Wonham, W. M. (1986). On supervisor reduction in discrete-event systems. *International Journal for Control*, *44*(2), 475–491.

Vieira, A. D., Santos, E. A. P., de Queiroz, M. H., Leal, A. B., Neto, A. D. d. P., & Cury, J. E. R. (2017). A method for PLC implementation of supervisory control of discrete event systems. *IEEE Transactions on Control Systems Technology*, *25*(1), 175–191.

Yoo, T. -S., & Lafortune, S. (2002). A general architecture for decentralized supervisory control of discrete event systems. *Discrete Event Dynamic Systems: Theory and Applications*, *12*, 335–377.

Yu, Y., Ge, Q., Lei, M., Wang, X., Yang, X., & Gou, R. (2013). Pre-charging control strategies of modular multilevel converter. In *International conference on electrical machines and systems*.

Zaytoon, J., & Riera, B. (2017). Synthesis and implementation of logic controllers–A review. *Annual Reviews in Control*, *43*, 152–168.

Zhang, P., Li, F., & Bhatt, N. (2010). Next-generation monitoring, analysis, and control for the future smart control center. *IEEE Transactions on Smart Grid*, *1*(2), 186–192.

Zhao, H., Mi, Z., & Ren, H. (2006). Modeling and analysis of power system events. *Proc. Chin. Soc. Electr. Eng.*, *26*(22), 11–16.