Author's Reply

Ami Sakakibara and Toshimitsu Ushio

December 27, 2019

We are thankful to the reviewers for their fruitful comments. We have revised our manuscript according to the comments. We hope that our revisions have improved the manuscript to their satisfaction.

1 Reply to Reviewer 1 (19275)

Review point 1.1: p.1, abstract, the reviewer recommends that the authors avoid the word "novel" when talking about their own work. You have not discussed novelty in the introduction. At best you can say that you do not know of anyone else who has done this.

Reply. We revised it.

Review point 1.2: p.2, definition of $suc(x)$ is missing quantification? $suc(x) = \{x' \in X :$ there exists $\sigma \in \Sigma(x)$ such that $x' \in \delta(x, \sigma)\}$

Reply. We revised the set as follows (please notice that we discarded descriptions in a nondeterministic manner).

(p. 2, left) $\mathsf{suc}_{\mathcal{T}} = \{x' \in X : \exists \sigma \in \Sigma(x), x' = \delta(x, \sigma)\}$.

Review point 1.3: p.2, please define the semantics of scLTL, i.e., make it clear under what circumstances $w \models \varphi$? Also please discuss the expressive power of this subset of LTL. It seems to capture only one-off liveness conditions such as F phi or (phi U psi), while safety (G phi) or repeated liveness (G F phi) cannot be expressed? This seems a bit weak, at least some of this should be possible with your approach.

Reply. Due to lack of space in the manuscript, we gave up putting the whole semantics. Instead, we referred to [3] and added intuitive explanations for the temporal operators $\bigcirc$ and $\mathbf{U}$ to the revised manuscript as follows.

(p. 2, right) Intuitively, $\bigcirc\varphi$ is true if φ holds on the word from the *next* step and $\varphi_1 \mathbf{U} \varphi_2$ if φ_1 keeps to be satisfied *until* φ_2 turns true at some step.

As you pointed out, the class of scLTL does not have expressibility for safety and repeated liveness. It is known that, however, scLTL specifications have potentials to practical applications [18], including surveillance problems as we show in Section IV. In order to consider safety and repeated liveness formulas, we need to modify the control scheme to some extent because the classes of such formulas are equivalent to words accepted by Büchi automata. Thus, it can be a future direction to extend the specification class, as mentioned in the conclusion in the revised manuscript.

(p. 6, left) It is future work to extend the proposed scheme to control under partial observation or under general LTL constraints.

Review point 1.4: p.2, Definition 2 is not really finite-memory because the supervisor records a step number k which has no upper bound.

Reply. In Definition 2, we require that the state set M be finite. Although a step number is unbounded as you pointed out, we do not implement the list of all returns of the feedback function μ_C. The on-line supervisor refers to the current state $m \in M$ and time step $k \in \mathbb{N}$ and dynamically computes $\mu_C(m, k)$.

Review point 1.5: p.3, it appears that A_φ is the equivalent DFA for formula φ? Please state explicitly. Also X_A and F_A are not defined.

Reply. Yes, A_φ (in the second paragraph of Section IV) is the DFA converted from φ, which has the set X_A of states and the set F_A of accepting states. We revised the sentence to make A_φ, X_A, and F_A clear as follows.

> (p. 3, left) The specification scLTL formula φ is converted into the equivalent DFA A_φ. Then, we obtain the product automaton P of the DES $G = ((X, \Sigma, \delta, x_0), AP, L)$ and the DFA $A_\varphi = ((X_A, 2^{AP}, \delta_A, x_{A,0}), F_A)$, defined as follows.

Review point 1.6: p.3, Definition 4 does not seem to be used; the ranking function defined in the following is stronger than this.

Reply. In the revised manuscript, we define a ranking function by using properties mentioned in Propositions 1 and 2 in the original manuscript. More precisely, we revised Definition 4 as follows.

> (p. 3, left; Definition 4) Let $P = ((X_P, \Sigma_P, \delta_P, x_{P,0}), F_P)$ be a product automaton and $\alpha = |X_P| - |F_P| + 1$. A function $\xi : X_P \to \mathbb{N}$ is a *ranking function* for P if both of the following conditions hold for any $x \in X_P$.
>
> 1) $\xi(x) = 0 \iff x \in F_P$;
>
> 2) $$\xi(x) = \begin{cases} \alpha & \text{if } \Sigma_P(x) = \emptyset, \\ \min\left\{ \min\limits_{\sigma \in \Sigma_{P,c}(x)} \xi(\delta_P(x,\sigma)) + I_{X_P \setminus F_P}(x), \alpha \right\} & \text{if } \Sigma_{P,u}(x) = \emptyset, \\ \min\left\{ \max\limits_{\sigma \in \Sigma_{P,u}(x)} \xi(\delta_P(x,\sigma)) + I_{X_P \setminus F_P}(x), \alpha \right\} & \text{otherwise.} \end{cases}$$

Review point 1.7: p.4-5, it would be nice to have some discussion about the degree of least restrictiveness achieved by this policy. Does it ensure that all states that could be reachable are? Probably not, but please explain.

Reply. The degree of least restrictiveness depends on the permissiveness function. We mention the fact in the revised manuscript as follows.

> (p. 5, left; Remark 1) Our method can be applied to supervisory control problems discussed in a language-based manner. In general, however, the proposed on-line supervisor is not maximally permissive as designed by the conventional method [1]. On the other hand, a maximally permissive supervisor computed by regarding the DFA A_φ as an acceptor for a $*$-language specification does not always satisfy the scLTL specification in the supervised DES due to the existence of livelock situations, which must be avoided in our setting. Despite the difference, the proposed on-line supervisor may determine a maximally permissive control pattern that includes not only all legal transitions but also all neutral ones if the permissiveness level is sufficiently close to α.

Moreover, the reachability of the supervised DES can be explained by the following two rules.

- All states x with $\xi(x) = \alpha$ is never reachable under the control.
- If $\xi(x) < \alpha$, then the reachability to x depends on the permissiveness function. While the permissiveness level is high enough to allow neutral transitions, the supervisor enables more events, which is likely to result in a larger set of reachable states.

It is not guaranteed, therefore, that all states of the supervised DES are reachable in general.

Review point 1.8: p.4, proof of Lemma 1, $J_G(x_P)$ should be $J_G(x)$?

Reply. We revised $J_G(x_P)$ to $J_G(x)$.

Review point 1.9: p.5-6, please render figures with font size similar to text, ensure legibility when printed.

Reply. We revised Figs. 1–4.

Review point 1.10: p.5, 1), please correct "stay" $\rightarrow$ "stays"

Reply. We revised it.

Review point 1.11: p.5, mentions condition $b \leq \alpha$ for the energy function, but this does not seem to be satisfied in Fig. 3?

Reply. In the example, $\alpha = 91$, as mentioned in the first paragraph of 2) Results, on the right of p. 5. Here, we have $b \leq \alpha$ in both cases of $b = 20$ and $b = 30$. In Fig. 3, the blue lines, representing the rank along sample trajectories, start from $\xi(x_{P,0}) = 14$ while the red lines, representing the permissiveness level, start from $b \in \{20, 30\}$.

2 Reply to Reviewer 3 (19279)

Review point 2.1: The first question for the authors is the following. Since the supervisory control (both on-line and off-line) for a given specification language is well established, if we first translate a scLTL specification into a language specification, then the problem can be easily solved using existing control synthesis methods. So, what are the advantages of the direct synthesis proposed in this paper? The following are possible reasons that one wants to use the direct synthesis. I would like to ask the authors to show one of them is true. (1) Not all scLTL specifications can be translated into a language specification. (2) The computational complexity of the direct synthesis is lower than that of the indirect synthesis. (3) The closed-loop language generated by the direct synthesis is larger than that of the indirect synthesis.

Reply. Honestly, we do not understand what "indirect synthesis" means. However, our method has the following advantages.

- By considering specifications described by scLTL formulas, we can consider more general control objectives than the conventional language-based supervisory control problems.
- The conversion of the scLTL specification to an equivalent DFA enables us to explicitly capture distances to acceptance by the ranking function. Quantifying such distances makes it easier to consider the tradeoff between the acceptance of the scLTL specification and the permissiveness of the supervisor.

- A control pattern at each step is computed with a permissiveness function, which has an unbounded domain of time step. It is unrealistic, therefore, to keep the list of all returns of the feedback function $\mu_C = \mathsf{online}$. By applying on-line control, we achieve the control specification with a finite-state supervisor while explicitly taking care of the tradeoff.
- Our method is flexible to the change of permissiveness functions. When one wants to replace a permissiveness function to another, it is enough to replace the component for the calculation of the permissiveness function. We do not have to recompute the transition system of the finite-state supervisor, which captures the dynamics of the on-line supervisor.

Review point 2.2: I am not sure what on-line means in the paper. Usually, on-line means that you do not design the entire control for all possible states of the system, but rather you design the control one step at time, as the system progresses, using a limited or variable lookahead window. I did not see this in Algorithm 2.

Reply. In this work, we use the term "on-line" to mean that the supervisor computes its control action at each time step. Although the supervisor incorporates the information computed off-line in its transition system, the feedback function is dynamically computed by the function online, which refers to the time-varying permissiveness level.

Review point 2.3: My conjecture is that they can also be applied to supervisor synthesis using language specification. If that is the case, then I encourage the authors to investigate them.

Reply. As you pointed out, our method can also be applied to supervisor synthesis for language-based specifications. We mention the fact in the revised manuscript as follows.

> (p. 5, left; Remark 1) Our method can be applied to supervisory control problems discussed in a language-based manner. In general, however, the proposed on-line supervisor is not maximally permissive as designed by the conventional method [1]. On the other hand, a maximally permissive supervisor computed by regarding the DFA A_φ as an acceptor for a $*$-language specification does not always satisfy the scLTL specification in the supervised DES due to the existence of livelock situations, which must be avoided in our setting. Despite the difference, the proposed on-line supervisor may determine a maximally permissive control pattern that includes not only all legal transitions but also all neutral ones if the permissiveness level is sufficiently close to α.

3 Reply to Reviewer 5 (19325)

Abstract (1): The meaning of sentence "we define a ranking function that decreases its value if an accepting state is being approached" is not so clear. According to this reviewer's understanding, a "ranking function" is similar to a "priority function". If so, the meaning of "decreases its value" is vague since this could either increase or decrease the priority/ranking. Moreover, by reading this sentence several times, it is not so clear what does the "it" referring to. It could be either "the specification" or "a state". Accorrding to the structure of the whole sentence, this reviewere prefers to believe that it is referring to "the specification". However, I guess, according to this paper, it makes no sense.

Reply. A ranking function returns the minimum number of steps required to reach an accepting state from the current state and is not similar to a priority function. We revised the sentence you pointed out as follows:

> (p. 1, left; Abstract) ... we define a ranking function that returns the minimum number of steps required to reach an accepting state from each state.

Abstract (2): "In addition, we introduce an energy function that indicates a time-variant permissive level." Personally, this reviewer does not believe that "energy function" is a good presentation because it indicates the "time- variant". Maybe the following paper and its references could inspire you to assign a better name:
Brandin B A, Wonham W M. Supervisory control of timed discrete-event systems[J]. IEEE Transactions on Automatic Control, 1994, 39(2): 329-342.

Reply. We agreed with your concern and thus renamed the energy function as a "permissiveness function" to explicitly represent its role. Accordingly, the sentence you pointed out was revised as follows.

> (p. 1, left; Abstract) In addition, we introduce a permissiveness function that indicates a time-varying permissive level.

We do not believe that the terminologies in the work by Brandin and Wonham are suitable for our framework because we just consider logical time step.

Introduction (1): Paragraph 2: why do you say "in this letter"?

Reply. We eliminated the phrase.

Introduction (2): Paragraph 3: "The ranking function decreases its value if an accepting state of the product automaton is being approached". Personally, this reviewer thinks that increases/decreases the "ranking" will be much better than "decreases its value". Again, a higher ranking could be either a larger or a smaller value, which does not really matter.

Introduction (3): Paragraph 4: "The ranking function decreases its value if an accepting state of the product automaton is being approached, i.e., the scLTL specification is more likely to be satisfied than the previous time step." Again, it is still not clear. Is it the ranking function of a state or a specification?

Reply. It seems that you mentioned the same sentence in these review comments Introduction (2) and Introduction (3). Hence, we reply to both of the comments here. As we explained in the reply to Abstract (1), a ranking function captures the minimum number of steps required to reach an accepting state from each state. Thus, the return of the ranking function decreases along a transition (x, σ, x') if from x' a less number of steps is required to reach an accepting state than from the current state x. In this sense, we say "the scLTL specification is more likely to be satisfied than the previous time step". We revised the sentence as follows.

> (p. 1, right) The rank decreases if an accepting state of the product automaton is being approached, i.e., the scLTL specification is more likely to be satisfied than the previous time step.

Section II (1): (j+1)-th or (j+1)-st?

Reply. We revised $(j+1)$-st to $(j+1)$-th.

Subsection II.A.1 (1) Is the "Transition Systems" so necessary? It seems like a state-based structure, which is inconsistant with the basic setting given in the first paragraph in the Introduction.

Reply. We solely define a transition system because it is a common component for a DES, a DFA, a finite-state supervisor, and a product automaton. Although we mention the traditional DES literature in Introduction, we do not mean to apply such an event-based setting in this work.

Subsection II.A.1 (2) Why do you define the enabled event set at a state x as $\Sigma(x)$? It is a little bit confusion. At least, a proper reference (if any) to support your choice is missing.

Reply. We follow the notation used in [13] and thus refer to it in the revised manuscript.

Subsection II.A.2 (1): Why do you define a DES as $G = ((X, \Sigma, \delta, x_0), AP, L)$? What is the benefit of defining it without following the traditional 5-tuple style? Again, at least, a proper reference (if any) to support your choice is missing.

Reply. By defining a DES with the labeling function with respect to atomic propositions, we can consider more general characterization of system properties than the traditional marking-based modeling. We follow the definition of DESs in [7] as well as that of Kripke structures in [3], like many other LTL-based studies. In the revised manuscript, we refer to [7].

Subsection II.A.2 (2): Please clarify what is "the set of atomic propositions"? Again, at least, a proper reference (if any) to support your choice is missing.

Reply. Atomic propositions are basic properties to characterize the system [3]. In the revised manuscript, we added an explanation as follows.

> (p. 2, left) ... AP is the set of atomic propositions, namely, the set of simple known statements that are either true or false [3], ...

Subsection II.A.2 (3): Since the "atomic proposition" is not defined, this reviewer is confused while reading the sentence "Each run in Runs(G) generates a sequence of letters, namely a word over 2^{AP}". Personally, this reviewer believes that it should be "a word over AP" instead. Just as in traditional DES, we say that a language is a set of string over Sigma.

Reply. As mentioned in the reply to Subsection II.A.2(2), we added an explanation of atomic propositions. The labeling function assigns to each state x a subset of atomic propositions that holds at x. In general, several properties may hold at each state and thus we assign a subset of atomic propositions. As for LTL, then, the input alphabet is given by $2^{AP} (= \Sigma_A)$. In addition, a sequence in $(2^{AP})^\omega$ is referred to as "a word over 2^{AP}" in [3]. Actually, we avoid referring to an event string $s \in \Sigma^*$ of the DES or the product automaton as a "word" in the manuscript. Whenever we use the term *words*, we mean sequences associated with atomic propositions.

Subsection II.B (1): "In this letter"?

Reply. We eliminated the phrase.

Subsection II.B (2): Formula φ is defined without explainations. In particular, how should this reviewer understand "$\dagger\varphi$" and "$\varphi_1 \mathbf{U} \varphi_2$"?

Reply. An scLTL formula φ on an infinite word $w \in (2^{AP})^\omega$ are interpreted inductively as follows:

$$
\begin{array}{lcll}
w & \models & \text{true} & \\
w & \models & a & \iff a \in w[0] \\
w & \models & \neg a & \iff a \notin w[0] \\
w & \models & \varphi_1 \wedge \varphi_2 & \iff w \models \varphi_1 \text{ and } w \models \varphi_2 \\
w & \models & \varphi_1 \vee \varphi_2 & \iff w \models \varphi_1 \text{ or } w \models \varphi_2 \\
w & \models & \bigcirc \varphi & \iff w[1]w[2] \ldots \models \varphi \\
w & \models & \varphi_1 \mathbf{U} \varphi_2 & \iff \exists k \in \mathbb{N} : w[k]w[k+1] \ldots \models \varphi_2 \text{ and } 0 \leq \forall j < k : w[j]w[j+1] \ldots \models \varphi_1
\end{array}
$$

Due to lack of space in the manuscript, we gave up putting the whole semantics. Instead, we referred to [3] and added intuitive explanations for the temporal operators $\bigcirc$ and $\mathbf{U}$ to the revised manuscript as follows.

(p. 2, right) Intuitively, $\bigcirc\varphi$ is true if φ holds on the word from the *next* step and $\varphi_1 \mathbf{U} \varphi_2$ if φ_1 keeps to be satisfied *until* φ_2 turns true at some step.

Section III (1): S(s) is used without a clear definition.

Reply. We revised $S(s)$ to $\mathcal{S}(s)$.

Section III (2): Could you please provide an example for the memory?

Reply. The supervisor determines its control action based on the observation from the initial time step to the current. Then, memory is used to capture such information and characterizes the dynamics of the supervisor. We used the term memory because it is commonly used in game theory [19]. However, we renamed a finite-memory supervisor to a finite-state supervisor to avoid confusing readers. Accordingly, we do not use the term "memory" in the revised manuscript.

Section III (3): Problem 1. In SCDES, a deadlock-free DES plant is not so necessary. Considering Σ^* as the specification, a DES with deadlocks could be converted into a DES with optimal nonblocking behaviors. Thus, this reviewer is interested to see how the contribution of this work to handle the DES with or without deadlock-free states.

Reply. We revised the definition of the ranking function so that deadlock states are ranked as the upper bound α. In the revised manuscript, then, we do not assume that the controlled DES is deadlock-free.

(p. 2, right; Problem 1) Given a finite DES $G = ((X, \Sigma, \delta, x_0),\ AP, L)$ and an scLTL formula φ over AP, synthesize a supervisor $\mathcal{S}$ such that $\mathcal{S}/G \models \varphi$.
(p. 3, left; Definition 4) Let $P = ((X_P, \Sigma_P, \delta_P, x_{P,0}), F_P)$ be a product automaton and $\alpha = |X_P| - |F_P| + 1$. A function $\xi : X_P \to \mathbb{N}$ is a *ranking function* for P if both of the following conditions hold for any $x \in X_P$.

1) $\xi(x) = 0 \iff x \in F_P$;

2) $$\xi(x) = \begin{cases} \alpha & \text{if } \Sigma_P(x) = \emptyset, \\ \min\left\{ \min\limits_{\sigma \in \Sigma_{P,c}(x)} \xi(\delta_P(x,\sigma)) + I_{X_P \setminus F_P}(x), \alpha \right\} & \text{if } \Sigma_{P,u}(x) = \emptyset, \\ \min\left\{ \max\limits_{\sigma \in \Sigma_{P,u}(x)} \xi(\delta_P(x,\sigma)) + I_{X_P \setminus F_P}(x), \alpha \right\} & \text{otherwise.} \end{cases}$$

Subsection IV

Paragraph 2:

1. "The specification scLTL formula is translated into the equivalent DFA." By reading this sentence, it is not very easy to find the contribution of this paper. Because this is what exactly the traditional SCDES do! Please emphasis your contribution/benefits.

Reply: The benefits of our method can be explained as follows.

- In our control problem we consider scLTL specifications, which describe system properties associated with atomic propositions. As mentioned in the reply to Section II.A.2 (2), then, we can consider more general characterization of system properties than marking-based representations.
- Technically, the conversion of the scLTL specification to an equivalent DFA enables us to explicitly capture distances to acceptance by the ranking function. Quantifying such distances makes it easier to consider the tradeoff between the acceptance of the scLTL specification and the permissiveness of the supervisor.

In the revised manuscript, we mention these points in Introduction.

> (p. 1, left) We consider a supervisory control problem of a DES under an scLTL constraint, which describes more general characterization of system properties than the conventional marking-based modeling.
>
> (p. 2, right) By referring to the ranking function and the permissiveness function, which quantify distances to acceptance and the permissiveness level, respectively, the supervisor explicitly considers the tradeoff between acceptance of the scLTL specification and its permissiveness.

2. Is it possible to avoid the approach mentioned above? This will be very interesting!

Reply. Applying techniques in bounded model checking [19] may be another approach. However, we doubt that these techniques provide a better solution than ours to the tradeoff between the acceptance of the scLTL specification and the permissiveness of the supervisor because the encoded problem does not have flexibility in the length of controlled trajectories.

Definition 4.

1. Please clearly explain what is a ranking function in plain text as well as in the formula. By only reading the formula, which is very difficult to understand what do you want to say.

2. Could you please assign a name for "alpha" and explain it clearly? It seems to be a very important symbol. It also appears in Proposition 5, Definition 5, and Section VI.2) without any explaination. Hence,this paper is very difficult to follow and understand. This reviewer suggests to formally define it globally instead of locally.

Reply: 1. We added an explanation of a ranking function as follows (please see the quotation below the second item).
2. We define α as the upper bound of the ranking function ξ before Definition 4. We additionally explain the role of α as follows.

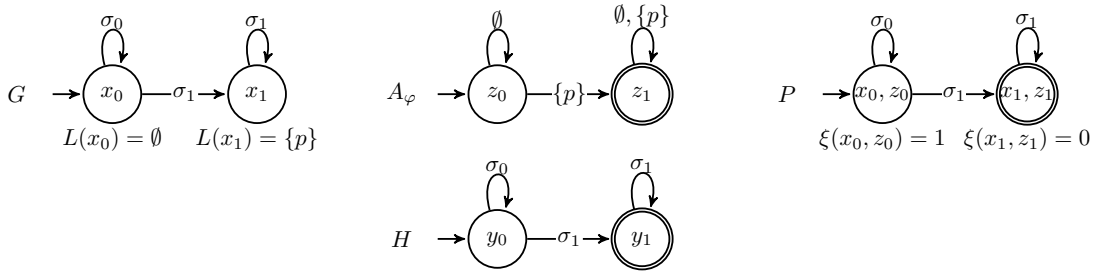
> (p. 3, left) Intuitively, the rank of a state x represents the minimum number of steps required to reach an accepting state from x under some control. We define $\alpha = |X_P| - |F_P| + 1$ as the upper bound of the ranking for the product automaton P. If x is ranked as α, then it is impossible to force the product automaton to reach an accepting state from x, i.e., either 1) x is among a strongly connected component from which no accepting state is reachable, or 2) it is inevitable to go to such a nonaccepting sink from x because of uncontrollable events.

The paragraph below Proposition 5.

1: The authors say "Indeed, we are likely to obtain more permissive supervisors if we allow not only legal transitions but also neutral ones to be enabled." This reviewer's question is: with a given scLTL specification, can the proposed approach obtain the optimal supervisor of a DES/TDES? If not, why?

Reply. First of all, we emphasize that our control problem is based on ω-languages, not $*$-languages, because we consider (a subclass of) LTL specifications. In general, an optimal finite-state supervisor does not exist for LTL specifications [20]. Therefore, an "optimal" supervisor as defined in the conventional framework based on the $*$-language does not always work in our setting. We explain this fact with the following example.

Let G be a DES as shown in the left of the figure, where $X = \{x_0, x_1\}$ with x_0 being the initial state; $\Sigma = \Sigma_c = \{c_0, c_1\}$; the transition function is such that $\delta(x_0, c_0) = x_0$, $\delta(x_0, c_1) = x_1$, and $\delta(x_1, c_1) = x_1$; $AP = \{p\}$; $L(x_0) = \emptyset$ and $L(x_1) = \{p\}$.



Let $\varphi = \Diamond p$ be an scLTL specification, requiring that the atomic proposition p holds in the future. The converted DFA A_φ is shown in the upper-middle of the figure (transition labels are explicitly represented as elements in 2^{AP} unlike the manuscript). Any infinite sequence whose prefix belongs to a $*$-language $K = \{c_0\}^*\{c_1\}^+$ satisfies the scLTL specification and an acceptor of K is given by a DFA H in the lower-middle of the figure. Then, an optimal supervisor $\mathcal{S}_{opt}$ for the language-based specification K is such that $\mathcal{S}_{opt}(s) = \{c_0, c_1\}$ whenever $\delta(x_0, s) = x_0$, which results in $\mathcal{L}_m(\mathcal{S}_{opt}/G) = K$. Note that, on the other hand, we have $c_0^* \in \mathcal{L}(\mathcal{S}_{opt}/G)$, by which the product automaton of G and A_φ never goes out from the nonaccepting initial state. That is, $\mathcal{S}_{opt}/G \not\models \varphi$ because we cannot avoid livelock in (x_0, z_0) under the control by $\mathcal{S}_{opt}$.

In our control scheme, we introduce a permissiveness function to surely avoid such livelock situations. We briefly mentioned the fact explained above in the revised manuscript.

> (p. 5, left; Remark 1) Our method can be applied to supervisory control problems discussed in a language-based manner. In general, however, the proposed on-line supervisor is not maximally permissive as designed by the conventional method [1]. On the other hand, a maximally permissive supervisor computed by regarding the DFA A_φ as an acceptor for a $*$-language specification does not always satisfy the scLTL specification in the supervised DES due to the existence of livelock situations, which must be avoided in our setting. Despite the difference, the proposed on-line supervisor may determine a maximally permissive control pattern that includes not only all legal transitions but also all neutral ones if the permissiveness level is sufficiently close to α.

Subsection VI.2 (1): Should it be Section VI.B instead?

Reply. Section IV has two parts discussing 1) Scenario and 2) Results. We avoided dividing the section into subsections indexed by A and B due to lack of space.

Subsection VI.2 (2) : The transitions in Fig. 2 is very difficult to follow. Could the authors please clearly explain at least one of them?

Reply. Here, we explain the transitions defined from z_0 and z_1 of the DFA A_φ in Fig. 2. Like done in the manuscript, we only consider transitions triggered by $\nu \in \{\nu' \in 2^{AP} : |\nu' \cap AP_{pos}| = 1\}$.

- From z_0, the unique outgoing edge is labeled by *true*. This means that a transition from z_0 to z_1 is triggered by any input letter. This rule can be written directly in accordance with the transition function $\delta_A : X_A \times \Sigma_A \to X_A$, where $\Sigma_A = 2^{AP}$, as follows.

$$\forall \nu \in 2^{AP},\ \delta_A(z_0, \nu) = z_1.$$

- From z_1, we have three outgoing edges. For example, a letter ν that corresponds to the label $\overline{p_0}\ \overline{q_s} + \overline{p_0}\ \overline{p_3}\ \overline{p_4}$ satisfies $\big((p_0 \notin \nu) \wedge (q_s \notin \nu)\big) \vee \big((p_0 \notin \nu) \wedge (p_3 \notin \nu) \wedge (p_4 \notin \nu)\big)$, which means either $\nu \in \big\{\{p_i\} : i = 1, 2, 3, 4, 5\big\}$ or $\nu \in \bigcup_{i=1,2,5} \big\{\{p_i\}, \{p_i, q_s\}\big\}$. Then, we have

$\delta_A(z_1, \nu) = z_1$ if $\nu \in \big\{\{p_i\} : i = 1, 2, 3, 4, 5\big\} \cup \big\{\{p_1, q_s\}, \{p_2, q_s\}, \{p_2, q_s\}\big\}$. Similarly, the transition function can be written as follows.

$$\delta_A(z_1, \nu) = \begin{cases} z_1 & \text{if } \nu \in \big\{\{p_i\} : i = 1, 2, 3, 4, 5\big\} \cup \big\{\{p_1, q_s\}, \{p_2, q_s\}, \{p_5, q_s\}\big\}, \\ z_2 & \text{if } \nu = \{p_4, q_s\}, \\ z_4 & \text{if } \nu \in \big\{\{p_0\}, \{p_0, q_s\}, \{p_3, q_s\}\big\}. \end{cases}$$

References

*References in the reply letter are numbered in accordance with the revised manuscript.

[1] Christos G. Cassandras and S. Lafortune, *Introduction to Discrete Event Systems*, 2nd ed. Springer US, 2008.

[3] C. Baier and J.-P. Katoen, *Principles of Model Checking.* MIT Press, 2008.

[7] S. Jiang and R. Kumar, "Supervisory control of discrete event systems with CTL* temporal logic specifications," *SIAM J. Control Optim.*, vol. 44, no. 6, pp. 2079–2103, 2006.

[13] N. B. Hadj-Alouane, S. Lafortune, and F. Lin, "Centralized and distributed algorithms for on-line synthesis of maximal control policies under partial observation," *Discret. Event Dyn. Syst. Theory Appl.*, vol. 6, no. 4, pp. 379–427, 1996.

[18] A. Bhatia, M. R. Maly, L. E. Kavraki, and M. Y. Vardi, "Motion planning with complex goals," *IEEE Robot. Autom. Mag.*, vol. 18, no. 3, pp. 55–64, 2011.

[19] E. M. Clarke, T. A. Henzinger, H. Veith, and R. Bloem, *Handbook of Model Checking.* Springer International Publishing, 2018.

[20] R. Ehlers, S. Lafortune, S. Tripakis, and M. Y. Vardi, "Supervisory control and reactive synthesis: a comparative introduction," *Discret. Event Dyn. Syst.*, vol. 27, no. 2, pp. 209–260, 2017.