This document describes the following:

- A correction to Algorithm 1
- An example to demonstrate the incompleteness of the approach outlined in the manuscript

## 1 Erratum

The corrected version of Algorithm 1 in the manuscript is as below:

### Algorithm 1 Separating into equicontrollable Classes

**Input:**
- Environmental Behavior $\phi^e$, System safety/transition rules $\rho^a$.
- Specification $\xi$ representing the set of states to be separated ($\llbracket \xi \rrbracket$).
- BDD $\rho^{reach}$ representing the set of reachable states for the system.
- Set of propositions $\mathcal{X} \subseteq AP$ over which the states must be partitioned and the map $f_{\text{param}}$.

**Output:**
- Equicontrollable classes $\alpha_1, \alpha_2, \alpha_3, \ldots, \alpha_k$ s.t. $\alpha_i \cap \alpha_j = \emptyset$ for $i \neq j$, $\bigcup_i \alpha_i = \llbracket \xi \rrbracket$.

1. Define $\phi_{\text{param}}^\xi := \phi^e \rightarrow \Box \rho^a \land \bigwedge_{t \in \mathcal{X}} \left( \xi \land \bigwedge_{t \in \mathcal{X}} (t \leftrightarrow f_{\text{param}}(t)) \right)$
2. Compute winning states ($W_{\phi_{\text{param}}^\xi}$) for $\phi_{\text{param}}^\xi$
3. Equicontrollable Classes $= \emptyset$
4. for $x \subseteq \mathcal{X}$ do
5.  $t_1 = f_{\text{param}}^{-1}(x)$; $\text{EquivFlag} = 0$
6.  for $p \in \text{Equicontrollable Classes}$ do
7.   $t_2 = f_{\text{param}}(p)$
8.   if $\exists s, s \in \mathcal{X} \land (s, t_2) \in W_{\phi_{\text{param}}^\xi} \land \exists x, s \in \mathcal{X} \land (p, t_1) \in W_{\phi_{\text{param}}^\xi}$ then
9.     $\text{EquivFlag} = 1$
10.   end if
11. end for
12. if $\text{EquivFlag} = 0$ and $\exists s \in \mathcal{X} \land \rho^{reach} \land s_x = x$ then
13.   Equicontrollable Classes $= \text{Equicontrollable Classes} \cup \{s \mid s \in \mathcal{X} \land s_x = x\}$
14. end if
15. end for
16. return Equicontrollable Classes

## 2 Appendix

**Example to Demonstrate the Incompleteness of the Approach**

Let the set of atomic propositions be $AP = \{b, c, d\}$ with $AP_e = \{c\}$ and $AP_a = \{b, d\}$. 
Define the transition rule for the environment:

\[ (d \rightarrow \bigcirc c). \]  

(1)

Define the transition rule for the controlled agent:

\[ \rho^e = \left( (\neg c \land (b \lor \neg d)) \rightarrow \bigcirc (b \lor d) \right). \]

(2)

Let the initial condition be \( \theta = (c \land b) \). Consider the following GR(1) synthesis problem.

\[ \theta \land \Box \rho^e \rightarrow \Box \rho^e \land \Box b \land \Box \bigcirc d. \]

The winning states for this problem are \( \{(b, c, d), (b, c)\} \). From both of these states the agent can pick \( d \) and \( \neg b \) to hold at the next state, forcing \( c \) to hold two instants into the future. When \( c \) holds, the agent can pick \( b \) satisfying the \( \bigcirc b \) and then, it is allowed to pick \( d \) and \( \neg b \) at the next instance and so on, the cycle can continue.

However, when we use the hierarchical approach, we do not obtain a cycle between the liveness guarantees. The controlled agent cannot force the execution to satisfy \( \bigcirc d \) from all states that satisfy \( b \). To see this consider the state \( (b) \). \( \neg (b \lor d) \) has to hold at the next step and if the environment decides to set \( \neg c \), \( \neg (b \lor d) \) has to again hold at the next instant and this goes on. Hence, though we have a winning strategy, we are not able to find it in the abstracted system, demonstrating the incompleteness of the approach. However, if the partitioning of \( \llbracket b \rrbracket \) was parameterized over both \( b \) and \( c \), the hierarchical approach would have had a cycle.