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 φ^{e}, System safety/transition rules ρ^{a}.
- Specification ξ representing the set of states to be separated ($[\![\xi]\!]$).
- BDD ρ^{reach} representing the set of reachable states for the system.
- Set of propositions $\mathscr{X} \subseteq \mathrm{AP}$ over which the states must be partitioned and the map f_{param}.

Output:
- Equicontrollable classes $\alpha_1, \alpha_2, \alpha_3, \ldots, \alpha_k$ s.t. $\alpha_i \cap \alpha_j = \emptyset$ for $i \neq j$, $\bigcup_{l=1}^{k} \alpha_i = [\![\xi]\!]$.

Define $\varphi_{\xi}^{\mathrm{param}} := \varphi^{\mathrm{e}} \rightarrow \Box\rho^{\mathrm{a}} \wedge \Diamond \left(\xi \wedge \bigwedge_{t \in \mathscr{X}} \left(t \leftrightarrow f_{\mathrm{param}}(t)\right)\right)$
Compute winning states ($W_{\varphi_{\xi}^{\mathrm{param}}}$) for $\varphi_{\xi}^{\mathrm{param}}$
Equicontrollable Classes $= \emptyset$
for $x \subseteq \mathscr{X}$ **do**
  $t_1 = f_{\mathrm{param}}^{-1}(x)$; $EquivFlag = 0$
  for $p \in$ Equicontrollable Classes **do**
    $t_2 = f_{\mathrm{param}}^{-1}(p)$
    if $\left(\exists s.s|_{\mathscr{X}} = x \wedge (s, t_2) \in W_{\varphi_{\xi}^{\mathrm{param}}} \wedge \exists s.s|_{\mathscr{X}} = p \wedge (p, t_1) \in W_{\varphi_{\xi}^{\mathrm{param}}}\right)$ **then**
      $EquivFlag = 1$
    end if
  end for
  if $EquivFlag = 0$ and $(\exists s \in \Sigma . s \models \rho^{\mathrm{reach}} \wedge s|_{\mathscr{X}} = x)$ **then**
    Equicontrollable Classes $=$ Equicontrollable Classes $\cup \{s | s \in \Sigma, s|_{\mathscr{X}} = x\}$
  end if
end for
return Equicontrollable Classes

---

2 Appendix

Example to Demonstrate the Incompleteness of the Approach

Let the set of atomic propositions be $\mathrm{AP} = \{b, c, d\}$ with $\mathrm{AP}_{\mathrm{e}} = \{c\}$ and $\mathrm{AP}_{\mathrm{a}} = \{b, d\}$.

Define the transition rule for the environment:

$$(d \rightarrow \bigcirc c). \tag{1}$$

Define the transition rule for the controlled agent:

$$\rho^{\mathrm{e}} = \Big(\big(\neg c \wedge (b \vee \neg d)\big) \rightarrow \bigcirc \neg (b \vee d) \Big). \tag{2}$$

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

$$\theta \wedge \Box \rho^{\mathrm{e}} \rightarrow \Box \rho^{\mathrm{a}} \wedge \Box \Diamond b \wedge \Box \Diamond d.$$

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 $\Diamond b$ 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 $\Diamond d$ from all states that satisfy b. To see this consider the state (b). $\neg(b \vee d)$ has to hold at the next step and if the environment decides to set $\neg c$, $\neg(b \vee d)$ 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 $[\![b]\!]$ was parameterized over both b and c, the hierarchical approach would have had a cycle.