Supplementary material: Nondeterminism subject to output commitment in combinatorial filters

Yulin Zhang[1] and Dylan A. Shell[2]

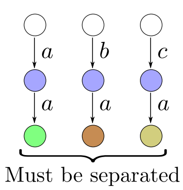
[1] The University of Texas at Austin, Austin TX, USA
[2] Texas A&M University, College Station TX, USA

Lemma 1. *The $\mathcal{F}_{\text{det}}$ of Figure 3 is a minimal tracing-deterministic filter that output simulates $\mathcal{F}_{\text{inp}}$.*

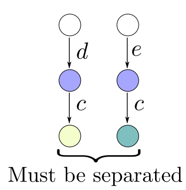
Proof. In $\mathcal{F}_{\text{det}}$, other than gray and royal-blue, every other color is reached by some string that reaches no other color, so at least one vertex must be present representing that color. For gray, any pair of the 7 vertices has a pink/light-blue conflict on an extension. So none of those 7 pairs can be merged. For the 5 royal-blue vertices, extensions under a, b, c force each pair apart. □

Lemma 2. *$\mathcal{F}_{\text{nd}}$ is a minimal tracing-nondeterministic filter that output simulates $\mathcal{F}_{\text{inp}}$.*

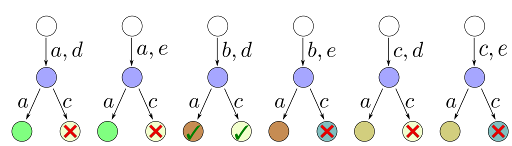
Proof. The same argument as before justifies all vertices that are the sole representative of their color (i.e., those with colors other than royal-blue and gray). Next, we show that $\mathcal{F}_{\text{nd}}$ has the minimal royal-blue and gray states. As shown in Figure A-1a, strings 'aa', 'ba', 'ca' must go through at least 3 different royal-blue vertices. Otherwise, if two of them go through the same royal-blue, then it will produce two different outputs for each of those two strings, which violates output simulation. Similarly in Figure A-1b, 'dc' and 'ec' must go though at least 2 different royal-blue states. Without any limit, we might create 5 different royal-blue for the above five strings to avoid conflicts. But to use only 3 or fewer royal-blue states, strings 'dc','ec' have to be overlaid with 'aa', 'ba', 'ca' such that 'dc', 'ec' go through the same royal-blue states as 'aa', 'ba', 'ca' do. There



(a) Strings with suffix 'a'.



(b) Strings with suffix 'b'.



(c) The 6 possible ways to overlay strings with suffix 'b' on the strings ending with 'a'.

Fig. A-1: (a). Strings 'aa', 'ba', 'ca' need to go through at least 3 different royal-blue states. (b). Strings 'dc', 'ec' need at least 2. (c) When overlaying 'dc', 'ec' with strings 'aa', 'ba', 'ca', 'ec' cannot go through the same royal-blue state as any of 'aa', 'ba', 'ca' without causing a conflict. Hence, at least 4 royal-blue states are required to carry strings 'aa', 'ba', 'ca', 'ec'.

are 6 ways as shown in Figure A-1c: only 'dc' can be overlaid with 'ba'. The others will cause a conflict. For example, if 'ec' visits the same royal-blue state as 'aa' does, then 'ac' outputs lime-green, which is incompatible with the original output, teal for 'ac', in $\mathcal{F}_{\mathrm{inp}}$. Therefore, we need at least 4 royal-blue states in the tracing-nondeterministic minimizer: a total of 3 for 'aa', 'ba', 'ca', and 1 for 'ec'. Using the same argument, we need at least 6 gray states. Therefore, $\mathcal{F}_{\mathrm{nd}}$ has the minimal number of states for every color, and hence is minimal. □

Lemma 3. *$\mathcal{F}_{\mathrm{sso}}$ is a string single-output minimizer of $\mathcal{F}_{\mathrm{inp}}$.*

Proof. The upper-half follows the same argument as that from Lemma 2, and the upper-half of $\mathcal{F}_{\mathrm{sso}}$ is minimal. For the lower-half, when overlaying 'ec' with 'dc', string 'bc' outputs both violet and lime, which is not string single-output. Hence, neither of 'dc, ec' can be overlayed with strings 'aa, ba, ca', so as to create a string single-output filter. Hence, we need at least 5 royal-blue states: 3 for 'aa, ba, ca', 2 for 'dc, ec', and the lower-half of $\mathcal{F}_{\mathrm{sso}}$ is also minimal. □

Theorem 4 (FM(DF$_{\#1}$→SSO) and FM(DF$_{\#1}$→SMO) are in P). *Given a tracing-deterministic input filter $\mathcal{F}$ with $|Y(\mathcal{F})| = 1$ (unitary alphabet $Y = \{y\}$), then it is P to find the minimal tracing-nondeterministic filter that output simulates $\mathcal{F}$.*

Proof. This is proved by showing that there is always a tracing-deterministic minimizer, and we can use the same procedure in Lemma 3. First, the minimizer only needs to have at most one cycle. If there are multiple, remove any edges to break all other cycles and only keep one cycle. Second, we only need to keep one outgoing edge for every state. If there are two outgoing edges for some state, then denote these two child states as v and w. Then there is a (non-strict) ordering between the extensions of v and w as follows. Either one is a subset of the other, or they are equal. This can be decided by checking the longest extension in these two states in polynomial time. If the extension visits some state twice, then the extensions are Y^*. Otherwise, the length of the extension should be within size $|V(\mathcal{F})|$. Remove one outgoing edge, either the state smaller set of extensions, or an arbitrary one if they are sets are identical. The preceding two steps will yield a deterministic minimizer. Compared to the original given minimizer, this new tracing-deterministic minimizer has the same number of states and the same language (since the edge-cutting operations were selected to preserve it). Since no new outputs for the existing strings will be introduced, this new minimizer output simulates the input filter if the original minimizer does. Therefore, the minimizer for both FM(DF$_{\#1}$→SSO) and FM(DF$_{\#1}$→SMO) are tracing-deterministic, and can be constructed following the procedure in Lemma 3. □