ATTEMPT AT MEMBERSHIP

EVAN BERGERON

Given an automaton A and an automorphism f, we decide if $f \in \mathcal{S}(A)$.

We're concerned only with self-similar automorphisms, as it makes little computational sense otherwise. (Regardless, all members of $\mathcal{S}(A)$ are self-similar). We assume we are given f in its wreath product form

$$f = (f_0, f_1)\sigma$$

where the set of transductions reachable by residuation is finite. Call the set of such transductions F and let $n = |F|$.

If h and h' are transductions, we write $h \equiv h'$ if h and h' are equivalent when restricted to $\mathbf{2}^n$.

Claim: Equivalent transductions on $\mathbf{2}^*$ form a regular language

Obviously, $G = \{g \mid g : \mathbf{2}^n \to \mathbf{2}^n\}$ is finite. Define a DFA with state set G, start state $\mathtt{id}$ and transitions

$$g_i \xrightarrow{a} g_i a$$

for $a \in F$. As usual, we have function application on the right (that is, $xg_i a = a(g_i(x))$).

Then for $f \in F$, we have that the following language is regular:

$$L_f := \{l \mid l \equiv f\}.$$

we can represent this as a DFA. Further, residuation is rational, so compose those two DFAs. That's how we residuate.

Simple Cycles

Define $\mathtt{fix}_f$ to be the set of strings $s \in \mathbf{2}^*$ for which $f = \partial_s(f)$. To ensure $\mathtt{fix}_f$ is finite, we impose the additional restriction that for all $s \in \mathtt{fix}_f$, no nontrivial proper prefix of s is also in $\mathtt{fix}_f$. So $\mathtt{fix}_f$ denotes the "shortest" such strings. TODO How to compute $\mathtt{fix}_f$? Should just be simple cycles?

(We can just make f the start and end state, and chop off all transitions out of the end. (Really just copy the start state)).

Then define

$$M_f = L_f \cap \bigcap_{s \in \mathtt{fix}_f} (\partial_s(L_f))$$

Note that every member of M_f is fixed. True? Certainly the **set** is fixed under residuation. But this doesn't implies point-wise fixed. when residuating around simple cycles. Further, for all m in M_f, $m \equiv f$ (as $M_f \subseteq L_f$).

General Paths

In the case where f is a single strongly connected component, we may simply DFS through the machine, accumulating the intersection of all $M_{\partial_w f}$'s and residuating as we go along. Once we have the intersection of all the states, we residuate back to the root. Then return `True` iff the accumulated language is nonempty.

Any member of the accumulated language works. Take an arbitarily long path in the machine. Since f is 1SCC, this path may be represented as a sequence of paths around cycles. Invoke splitting lemma.

If f is not 1SCC, can we just "induct" over the connected components? We have a rooted tree of SCCs. Start at the "leaf" SCCs and work up?