Creating Shuffle Automata using Composition and Intersection

Georg Jähnig*
University of Potsdam, Department of Linguistics

January 28, 2011

Introduction

Creating an automaton that accepts the shuffle language of two given automata is usually done by the shuffle operation defined on the states of the automata. This short instruction shows how a shuffle automaton can be created using standard algebra operations like composition and intersection. It might be useful when working with a FSM framework that has no shuffle operation implemented.

Shuffle operation

Given any fixed alphabet Σ with the empty word ϵ, the shuffle operation $\bullet$ is defined inductively as follows [1, page 1]:

$$u \bullet \epsilon = \epsilon \bullet u = u \qquad \forall u \in \Sigma^* \qquad (1)$$

$$au \bullet bv = a(u \bullet bv) \cup b(au \bullet v) \qquad \forall u, v \in \Sigma^*, \forall a, b \in \Sigma \qquad (2)$$

*georg@jaehnig.org

Thus the shuffle is the set of all interleavings of the given words – the relative order of the symbols within the words is kept although. For instance, given the symbols a, b, c, x, y we obtain the following shuffle languages:

$$ab \bullet xy = \{abxy, axby, axyb, xaby, xayb, xyab\} \tag{3}$$
$$ab \bullet bc = \{abbc, abcb, babc, bacb, bcab\} \tag{4}$$

For any languages $L_1, L_2 \subseteq \Sigma^*$, the shuffle is defined as:

$$L_1 \bullet L_2 = \bigcup_{u \in L_1, v \in L_2} u \bullet v \tag{5}$$

The shuffle operation applied on two automata accepting the languages L_1 and L_2 creates an automaton accepting $L_1 \bullet L_2$. Below follows a description how to obtain such a shuffle automaton using only intersection and composition with 6 auxiliary transducers.

Preparation

$$m_1 = (\Sigma \to \Sigma_1) \cup \Sigma_2 \tag{6}$$
$$m_2 = (\Sigma \to \Sigma_2) \cup \Sigma_1 \tag{7}$$
$$r_1 = (\Sigma_1 \to \Sigma) \cup \Sigma_2 \cup \Sigma \tag{8}$$
$$r_2 = (\Sigma_2 \to \Sigma) \cup \Sigma_1 \cup \Sigma \tag{9}$$
$$i_1 = (\{\epsilon\} \times \Sigma_2) \cup \Sigma_1 \tag{10}$$
$$i_2 = (\{\epsilon\} \times \Sigma_1) \cup \Sigma_2 \tag{11}$$

Before application, we priorly create 6 auxiliary transducers. m_1 and m_2 are mapping from Σ to respectively Σ_1 and Σ_2, with $\Sigma_1 \cap \Sigma_2 = \emptyset$[1]. We do also create appropriate remap transducers r_1 and r_2 mapping back to Σ.

Additionally, we construct i_1 and i_2 as transducers reading and preserving words containing only Σ_1 (resp. Σ_2) symbols and inserting any number of Σ_2 (resp. Σ_1) symbols at any point of the input words.

[1]If we don't want to use 2 other alphabets, we can as well keep Σ and use markers following every symbol, e.g. $x \to x1, x \to x2$. i_1, i_2, r_1, r_2 need be adjusted then.

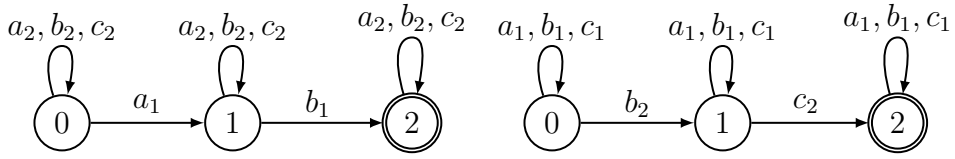

Figure 1: Auxiliary transducers applied on $u = \{a, b\}$ and $v = \{b, c\}$

Application

Given the 6 auxiliary transducers, we can create the shuffle automaton of two input automata u and v using only intersection and composition.

$$u \bullet v = P_2((P_2(u \circ m_1 \circ i_1) \cap P_2(v \circ m_2 \circ i_2)) \circ r_1 \circ r_2) \tag{12}$$

First, we compose u with m_1 and i_1 (v resp. with m_2 and i_2) and keep only the second projection of the result transducers (i.e. their output symbols). For example, when applying the auxiliary transducers on $u = \{ab\}$ and $v = \{bc\}$, the result automata will look as depicted in Figure 1.

With intersecting these automata we nearly obtain the final shuffle automaton. What remains is just remapping Σ_1 and Σ_2 back to Σ using the remap transducers r_1 and r_2.

Acknowledgements

I was inspired to the mapping into two disjuct alphabets by the degree computing algorithm in [2, page 11].

References

[1] Joanna Jędrzejowicz. Structural properties of shuffle automata. *Grammars*, 2(1):35–51, 1999.

[2] François Yvon. Finite-state transducers solving analogies on words. Technical report, 2003.