Benchmarking Compositionality with Formal Languages

Josef Valvoda[🦾] **Naomi Saphra**[⚙] **Jonathan Rawski**[🤖]
Adina Williams[🛠] **Ryan Cotterell**[💪]
[🦾]University of Cambridge [⚙]New York University
[🤖]San José State University [🛠]FAIR [💪]ETH Zürich
jv406@cam.ac.uk nsaphra@nyu.edu jon.rawski@sjsu.edu
adinawilliams@fb.com ryan.cotterell@inf.ethz.ch

Abstract

Recombining known primitive concepts into larger novel combinations is a quintessentially human cognitive capability. Whether large neural models in NLP can acquire this ability while learning from data is an open question. In this paper, we investigate this problem from the perspective of formal languages. We use deterministic finite-state transducers to make an unbounded number of datasets with controllable properties governing compositionality. By randomly sampling over many transducers, we explore which of their properties contribute to learnability of a compositional relation by a neural network. We find that the models either learn the relations completely or not at all. The key is transition coverage, setting a soft learnability limit at 400 examples per transition.

https://github.com/valvoda/neuralTransducer

1 Introduction

Compositionality is a hallmark of human language (Montague, 1970; Partee, 1995; Fodor, 1998). It is arguably a requirement for any model to count as a model of language, or to achieve human-like natural language understanding. Compositionality seems to be such a deep property of language that speakers draw conclusions about the overall meaning of sentences even when the meanings of individual words are not known. For instance, English speakers reading the Jabberwocky (Carroll, 1871) comprehend that the noun phrase *slithy toves* is built from the composition of the adjective *slithy* with the plural noun *toves*, despite lacking a clear understanding of what *slithy* or *toves*—let alone their composition—could mean. In cognitive science, whether neural networks can learn to combine a limited number of primitives (in the case of language, word or morphemes) to describe a complex environment has been debated for over 30 years (Fodor and Pylyshyn, 1988; Marcus, 1998).

In recent work, researchers have explored the inherent limitations of neural models to exhibit compositionality by analyzing sequence-to-sequence model performance on small, controlled datasets (Lake and Baroni, 2018; Hewitt et al., 2020; Hupkes et al., 2020a; White and Cotterell, 2021; Dankers et al., 2022; White and Cotterell, 2022). However, the conclusions of these studies are often murky. For instance, Lake and Baroni (2018) cast doubt on neural models' ability to do compositional generalization using their toy SCAN dataset, but shortly thereafter, Bastings et al. (2018) demonstrated that an out-of-the-box sequence-to-sequence model could indeed fully master the task.

Instead of hand-crafting small challenge datasets, we propose to test for compositionality by randomly sampling from a whole class of string-to-string functions. In doing so, we draw on two linguistic traditions. On the one hand, we follow Montague's assertion that no important theoretical difference exists between natural and artificial languages (Montague, 1970). Following this logic, the question of whether neural networks compositionally process human language is fundamentally equivalent to asking whether they compositionally process artificial languages. On the other hand, we draw lessons from the field of grammatical inference (de la Higuera, 2010; Rawski and Heinz, 2019), and evaluate neural sequence-to-sequence models on many automatically generated artificial languages sampled from particular classes of functions—as is standard practice at grammatical inference competitions (Balle et al., 2017).

In this paper, we study the class of string functions encoded by subsequential finite-state transducers (SFSTs), a restricted class of general finite-state transducers (Mohri, 1997). We sample arbitrary SFSTs to generate many different string-to-string datasets and evaluate the behavior of neural sequence-to-sequence models when learning them. By controlling the formal properties of the SFSTs we sample from, we are able to make precise statements about the learnability of systematic phenomena.

Empirically, we find that neural sequence-to-sequence models are, in many cases, capable of perfectly learning SFSTs from finite data. Moreover, we observe an interesting tendency for neural models to either generalize correctly or to fail outright—with little middle ground. Our analysis reveals a possible explanation for this—generalization seems to be possible when the training data has sufficient coverage, i.e., when every transition in a given transducer is crossed in a minimum number of training examples (≈ 400 in our experiments).

We then turn to analyze a popular hand-crafted dataset, SCAN, through the lens of an SFST that encodes it. We find that SCAN is peculiar in that it seems to serve as a counterexample to our transition coverage finding. This suggests that there is a more nuanced story to tell: We predict the learnability of a language based on a notion of complexity native to subregular languages, but it may be that a more consistently predictive complexity metric would come from a higher point on the Chomsky hierarchy. Future work might seek to limit the number of outlier languages like SCAN in order to identify a notion of complexity that is native to the architecture of the model itself. Such a notion of complexity would identify the level of abstraction that best reflects the representations learned by the model. A fruitful avenue towards this goal might lie in exploring more complex formalisms.

2 Finite-State Transducers

This section provides a short technical overview of finite-state transducers and motivates our choice to learn this class of relations.

2.1 Why Learn Finite-State Transductions?

Our study focuses on learning a particular kind of transduction. Specifically, we focus on restricted classes of regular relations, which are those relations computable by finite-state transducers. We believe this is a natural starting point since this class of formal languages is mathematically well-studied, has provable learning guarantees, and has a long use history in linguistics and NLP (Mohri, 1997).

Finite-state transducers also encompass most previous work on compositionality: many datasets, e.g., SCAN (Lake and Baroni, 2018) and gSCAN (Ruis et al., 2020), describe *finite* string relations and are, therefore, finite-state by definition. These handcrafted datasets have many advantages, like easy interpretability and domain specificity since they directly encode particular relevant relationships like movement over a grid or specific linguistic phenomena. However, this realism pays the price of diminished robustness of any findings over such datasets (Rogers and Pullum, 2011), see White and Cotterell (2021) for more discussion of this point. By removing the ability to simply adjust properties of the underlying function class, and the transducers which compute it, one loses the possibility to experiment more robustly over a function type, rather than just one token instantiation of it.

Rather than manually designing individual datasets ourselves, we generate unboundedly many new datasets via randomly sampled SFSTs. This offers a principled view of the problem of learning artificial languages by simply varying properties of the class of transducers that generate them. Furthermore, as we will see in §6, one may view existing compositionality tasks as learning a specific SFST. We contend this view enables a deeper understanding of modeling results. Both specific artificial languages, such as the compositionality datasets mentioned above, and those randomly sampled from a particular function class such as the work presented in this paper, are worth studying. However, our approach has been missing from the compositionality discourse.

2.2 Basic Theory

Now we will overview the basic elements of finite-state theory that will be necessary for the rest of the paper; we start with some definitions.

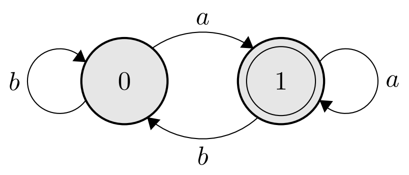
Definition 1. *A* ***finite-state automaton*** *(FSA) is a 5-tuple* $\mathcal{A} = \langle \Sigma, Q, q_0, F, \delta \rangle$ *where*

- Σ *is an input alphabet whose elements are denoted* σ*;*
- Q *is a finite set of states whose elements are denoted* q*;*
- $q_0 \in Q$ *is the unique start state;*
- $F \subseteq Q$ *is the set of final states;*
- $\delta : Q \times \Sigma \cup \{\varepsilon\} \to Q$ *is the transition relation and* ε *is an empty string.*

*We denote transitions, i.e., when $q' \in \delta(q, \sigma)$, with the more suggestive notation $q \xrightarrow{\sigma} q'$. We say that the automaton $\mathcal{A}$ **accepts** a string $\boldsymbol{\sigma} \in \Sigma^*$ iff there exists a path*[1] *through the automaton states $q_0 \xrightarrow{\sigma_1} q_1 \xrightarrow{\sigma_2} q_2 \cdots \xrightarrow{\sigma_N} q_N$ where q_0 is the initial state and $q_N \in F$ and $\sigma_1 \cdots \sigma_N = \boldsymbol{\sigma}$. In our notation σ_n can be the empty string ε. We call $\boldsymbol{\sigma}$ the **yield** of the path $q_0 \xrightarrow{\sigma_1} q_1 \xrightarrow{\sigma_2} q_2 \cdots \xrightarrow{\sigma_N} q_N$.*

Furthermore, we call an automaton **complete** when one may transition from every state to every symbol, i.e., $\delta(q, \sigma)$ is defined for all $q \in Q$ and $\sigma \in \Sigma$. And, we say an automaton is **deterministic**, if, given a state q and an alphabet symbol $\sigma \in \Sigma$, there is at most one transition for σ from q, i.e., $|\delta(q, \sigma)| \leq 1$. We have that $|\delta(q, \sigma)| = 1$ for all $q \in Q$ and $\sigma \in \Sigma$ if the automaton is *both* complete and deterministic. In our examples below we denote the **final state** as a circle with a double border.

Example 1. *Below we exhibit a complete deterministic finite-state automaton.*



The above finite-state automaton accepts the language $\{(b^i a^j)^n \mid i, j, n \in \mathbb{Z}_+\}$.

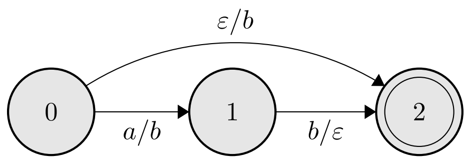
Definition 2. *A **finite-state transducer** (FST) is a 6-tuple $\mathcal{T} = \langle \Sigma, \Gamma, Q, q_0, F, \delta \rangle$:*

- *Σ is an input alphabet whose elements are denoted σ;*
- *Γ is an output alphabet whose elements are denoted γ;*
- *Q is a finite set of states whose elements are denoted q;*
- *$q_0 \in Q$ is the unique start state;*
- *$F \subseteq Q$ is the set of final states;*
- *$\delta : Q \times \Sigma \cup \{\varepsilon\} \to Q \times \Gamma^*$ is the transition relation.*

*We denote transitions, i.e., when $(q', \gamma) \in \delta(q, \sigma)$, with the more suggestive notation $q \xrightarrow{\sigma/\gamma} q'$. We say that the transducer $\mathcal{T}$ **transduces** a string $\boldsymbol{\sigma} \in \Sigma^*$ to a string $\boldsymbol{\gamma} \in \Gamma^*$ iff there exists a path $q_0 \xrightarrow{\sigma_1/\gamma_1} q_1 \xrightarrow{\sigma_2/\gamma_2} q_2 \cdots \xrightarrow{\sigma_N/\gamma_N} q_N$ where q_0 is the initial state and $q_N \in F$, $\sigma_1 \cdots \sigma_N = \boldsymbol{\sigma}$ and $\gamma_1 \cdots \gamma_N = \boldsymbol{\gamma}$. In our notation either σ_n or γ_n can be the empty string ε, from which it follows that the length $|\boldsymbol{\sigma}|$ must not equal $|\boldsymbol{\gamma}|$. We call $\boldsymbol{\sigma}$ the **input yield** and $\boldsymbol{\gamma}$ the **output yield**, respectively, of the path $q_0 \xrightarrow{\sigma_1/\gamma_1} q_1 \xrightarrow{\sigma_2/\gamma_2} q_2 \cdots \xrightarrow{\sigma_N/\gamma_N} q_N$.*

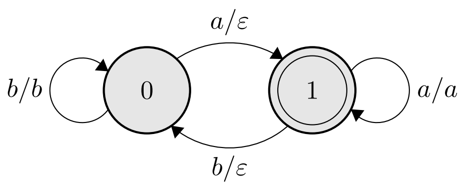
As in the case of an FSA, we say an FST is **complete** if δ is defined for all states and all symbols, i.e., $|\delta(q, \sigma)| > 0$ for $q \in Q$ and $\sigma \in \Sigma$. Furthermore, we say an FST is **subsequential** if it is deterministic with respect to the input, i.e., if $|\delta(q, \sigma)| \leq 1$ for all $q \in Q$ and $\sigma \in \Sigma$ and the FST does not have transitions of the form $q \xrightarrow{\varepsilon/\gamma} q'$. By construction, subsequential transducers (SFSTs) are **functional**, i.e., the string-to-string relations they encode are functions rather than relations. Indeed, it is this functionality that makes them a useful tool for the analysis of neural sequence-to-sequence models. The class of **subsequential functions** are those describable with SFSTs.[2] They are a subclass of the regular relations, but a superclass of the finite relations.

Example 2. *Below is an example of a non-deterministic finite-state transducer:*



The above transducer only has two paths $q_0 \xrightarrow{a/b} q_1 \xrightarrow{b/\varepsilon} q_2$ and $q_0 \xrightarrow{\varepsilon/b} q_2$. The first transduces $ab \mapsto b$ and the second $\varepsilon \mapsto b$.

Example 3. *Below is an example of a complete subsequential transducer over the input alphabet $\Sigma = \{a, b\}$ and output alphabet $\Gamma = \{a, b\}$.*



Note the absence of ε on the input side; however, we do have ε on the output side.

3 Compositionality Formalized

It is widely held that compositionality is a cornerstone of human language. However, language researchers use the term compositionality to refer to a variety of different concepts (Hupkes et al.,

[1]A path is a sequence of transitions.

[2]Note there are other algebraic and logical characterisations (Oncina et al., 1993; Bhaskar et al., 2020).

2020b). Here, we discuss the specific definition we employ throughout the paper and give a theoretical justification for the use of finite-state transducers as an instantiation of that definition.

3.1 Montague's Compositionality[3]

Montague famously gave a mathematical definition of what it means for a language to be compositional (Montague, 1970)—specifically, he proposed that a relation, e.g., the mapping from syntax to semantics, be called compositional if and only if it is a homomorphism, i.e., if it is a structure-preserving map between an input and an output algebra (Andreas, 2019). To make this notion more formal, we have to be specific about what structure we hope our function preserves. In this paper, we will exclusively focus on monoidal structure. We emphasize, however, that the notion of a homomorphism is not restricted to monoids.

Definition 3. *A* ***monoid*** *is a set A equipped with a binary operation $\bullet$ such that*

- *For $f, g \in A$, $f \bullet g \in A$ (**closure**);*
- *For $f, g, h \in A$, $f \bullet (g \bullet h) = (f \bullet g) \bullet h$ (**associativity**);*
- *There exists a unique element e such that for every $g \in A$, we have that $g \bullet e = e \bullet g = g$ (**identity**).*

Definition 4. *A* ***free monoid*** *over strings is the structure $(\Sigma^*, \circ)$ where Σ^* is the Kleene closure of the alphabet Σ and $\circ$ is string concatenation.*[4] *The empty string ε is the identity element as $\varepsilon \circ \boldsymbol{\sigma} = \boldsymbol{\sigma} \circ \varepsilon = \boldsymbol{\sigma}$ for any $\boldsymbol{\sigma} \in \Sigma^*$.*

Definition 5. *Let $(A, \bullet_A)$ and $(B, \bullet_B)$ be two monoids with unique identity elements e_A and e_B. We call a function $f : A \to B$ a* ***homomorphism*** *if it obeys the following two properties:*

- *$f(e_A) = e_B$;*
- *$f(x \bullet_A y) = f(x) \bullet_B f(y), \quad \forall x, y \in A$*

We call a homomorphism between two free monoids a **string homomorphism**. As it turns out, there is a precise connection between string homomorphisms and finite-state theory.

Proposition 1. *Let Σ and Γ be two alphabets. The function f is a string homomorphism between $(\Sigma^*, \circ)$ and $(\Gamma^*, \circ)$ iff it is a minimal non-empty complete subsequential finite-state transducer with one state.*

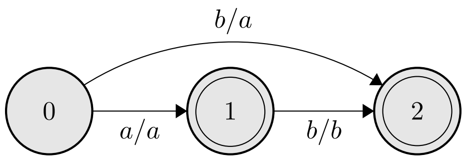
Proof. See App. A. ■

Proposition 1 starts to shed light on the connection between Montague's notion of compositionality and finite-state transducers. However, this connection is quite weak because multi-state transducers are not covered. We remedy this disparity in the subsequent section.

3.2 Transducers as Homomorphisms

Now we offer a more formal treatment of the exact sense in which SFSTs may be considered homomorphism and thus fall under Montague's definition of compositionality. As shown by Proposition 1, in general, SFSTs do *not* encode string homomorphisms. Indeed, it is straightforward to find a counterexample that hammers this point home.

Example 4. *Below we exhibit a two-state subsequential finite-state transducer that is* not *a string homomorphism.*



In the above example, we have $ab \mapsto ab$, but also $a \mapsto a$ and $b \mapsto a$. Thus, it is not a homomorphism.

Example 4 is dissatisfying; it contradicts the intuition that an SFST encodes some notion of Montague-esque compositionality. Luckily, as it turns out, we can find a precise sense in which an SFST is indeed a homomorphism. The idea is to lift the free monoid into a matrix. Given a complete SFSA $\mathcal{A} = \langle \Sigma, Q, q_0, F, \delta \rangle$ over $|Q| = N$ states, for every symbol $\sigma \in \Sigma$ define an $N \times N$ **symbol transition matrix M^σ** whose entries are

$$M^\sigma_{ik} \overset{\text{def}}{=} \begin{cases} \sigma, & \textbf{if } q_i \xrightarrow{\sigma} q_k \in \delta \\ \mathbf{0}, & \textbf{otherwise} \end{cases} \tag{1}$$

where $\mathbf{0}$ is a distinguished symbol, which is not in Σ, called the **zero string**. The zero string is an **anhilitator**, i.e., it has the property that $\mathbf{0} \circ \sigma = \sigma \circ \mathbf{0} = \mathbf{0}$ for any $\sigma \in \Sigma$.

[3]**A clarifying note**: Many readers, post publication, have reported that they interpreted our paper as asserting that the natural language syntax–semantics interface can be thought of as a regular string-to-string transduction. This was *not* our intent. Instead, our goal is to exhibit a simple benchmark, derived from randomly sampled SFSTs, that are formally compatible with Montague's definition of compositionality and allow for extreme experimentation.

[4]When clear from context, we write $\sigma \circ \sigma'$ as $\sigma\sigma'$.

Importantly, because $\mathcal{A}$ is complete and deterministic, there is exactly one non-$\mathbf{0}$ entry in every row of $\boldsymbol{M}^{\sigma}$. Now, we define an operation $\otimes$ between two such matrices. Since the automaton is complete and deterministic, we know there exists a unique j' such that $M^{\sigma}_{ij'} \neq \mathbf{0}$ and, by the same argument, there exists a unique k' such that $M^{\sigma}_{j'k'} \neq \mathbf{0}$. Then, in terms of j' and k', we have

$$(\boldsymbol{M}^{\sigma} \otimes \boldsymbol{M}^{\sigma'})_{ik} = \begin{cases} M^{\sigma}_{ij'} \circ M^{\sigma}_{j'k'}, & \textbf{if } k = k' \\ \mathbf{0}, & \textbf{otherwise} \end{cases} \tag{2}$$

Clearly, $\boldsymbol{M}^{\sigma} \otimes \boldsymbol{M}^{\sigma'}$, for any $\sigma, \sigma' \in \Sigma$, enforces that the resulting product still has the property that there is exactly one element of every row that is not equal to $\mathbf{0}$.[5] With $\boldsymbol{M}^{\boldsymbol{\sigma}}$, where $\boldsymbol{\sigma} = \sigma_1 \cdots \sigma_K \in \Sigma^*$ is a string of length K, we denote the product of matrices $\boldsymbol{M}^{\sigma_1} \otimes \cdots \otimes \boldsymbol{M}^{\sigma_K}$.

Proposition 2. *Let $\mathcal{A} = \langle \Sigma, Q, q_0, F, \delta \rangle$ be a complete deterministic finite-state automaton. Let $\mathcal{M} \overset{\text{def}}{=} \left\{ \boldsymbol{M}^{\sigma} \mid \sigma \in \Sigma \right\}$ be the set of $\mathcal{A}$'s symbol transition matrices. Then $(\mathcal{M}^*, \otimes)$ with $\otimes$ defined in Eq. (2) is a **transition monoid** with $\boldsymbol{E}$ as a distinguished identity element.*[6]

Proof. See App. A. ∎

Now we are in a position to discuss the precise sense in which subsequential finite-state transducers are homomorphisms. In the case of a finite-state transducer $\mathcal{T} = \langle \Sigma, \Gamma, Q, q_0, F, \delta \rangle$, there are two symbol transition matrices. The *input* symbol transition matrix is defined analogously to that of a finite-state acceptor. However, the *output* symbol transition matrix is defined slightly differently:

$$M^{\sigma}_{ik} \overset{\text{def}}{=} \begin{cases} \gamma, & \textbf{if } q_i \xrightarrow{\sigma/\gamma} q_k \in \delta \\ \mathbf{0}, & \textbf{otherwise} \end{cases} \tag{3}$$

Indeed, for any product of K output symbol transition matrices $\boldsymbol{M}^{\boldsymbol{\sigma}} = \boldsymbol{M}^{\sigma_1} \otimes \cdots \otimes \boldsymbol{M}^{\sigma_K}$, the entries of $\boldsymbol{\sigma}$ have a clear interpretation. Let $\boldsymbol{\sigma} = \sigma_1 \cdots \sigma_K$ and suppose $\boldsymbol{M}^{\boldsymbol{\sigma}}_{ik} = \boldsymbol{\gamma} \neq \mathbf{0}$. Then, we know that if we start in state q_i and read in input string $\boldsymbol{\sigma}$ we end up in state q_k and output string $\boldsymbol{\gamma}$. There are exactly $|Q|$ non-zero entries in $\boldsymbol{M}^{\boldsymbol{\sigma}}$. Thus, for any given finite-state transducer there is both an input and output transition monoid, which we denote as $(\mathcal{M}^*_{\Sigma}, \otimes)$ and $(\mathcal{M}^*_{\Gamma}, \otimes)$, respectively. Then, the intuition is that SFSTs constitute a homomorphism over the closure of the symbol transition matrices. We state the more formal result below.

Proposition 3. *Let $\mathcal{T} = \langle \Sigma, \Gamma, Q, q_0, F, \delta \rangle$ be a complete subsequential finite-state transducer. Let $(\mathcal{M}^*_{\Sigma}, \otimes)$ be the transition monoid associated with the input alphabet Σ and let $(\mathcal{M}^*_{\Gamma}, \otimes)$ be the transition monoid associated with the output alphabet Γ. Then there exists a homomorphism $f : \mathcal{M}^*_{\Sigma} \to \mathcal{M}^*_{\Gamma}$.*

Proof. See App. A. ∎

At a higher level, Proposition 3 simply fixes the bug present in Example 4 by incorporating the state into the values. Moreover, Proposition 3 is a clear generalization of Proposition 1 in that if we have a single-state SFST, we have 1×1 matrices that may be viewed as single symbols and the operation $\otimes$ defined in Eq. (2) reduces to string concatenation.

Briefly back to Montague. To put the above results in context, we showed by Proposition 3 that arbitrary complete SFSTs are compositional in the sense of Montague. Specifically, SFSTs encode a homomorphism between the free monoid $(\Sigma^*, \cdot)$, which is isomorphic to $(\mathcal{M}^*_{\Sigma}, \otimes)$, and the transition monoid $(\mathcal{M}^*_{\Gamma}, \otimes)$. This instantiates compositionality by directly linking it to the robust class-based characteristics of subsequential functions. There are of course other ways, but subsequentiality gives us a theoretical underpinning to use learning SFSTs from finite data as a benchmark for compositionality, and motivates our experiments and analysis in the coming sections.

4 Experimental Methods

Next we introduce the methodology for generating our datasets as well as the neural and symbolic models employed in the empirical portion of our paper.

4.1 Generating Random SFSTs

We generate random SFSTs using the following stochastic process. We first generate random unlabeled directed graphs that correspond to the symbol-specific transition matrices. Given a set of states Q (let $N = |Q|$) and an input alphabet Σ, we sample a matrix $\boldsymbol{B}^{\sigma} \in \mathbb{B}^{N \times N}$ for every $\sigma \in \Sigma$ where $\mathbb{B} = \{0, 1\}$. During sampling, we enforce the constraint that there be *at most* one non-zero entry in every row vector $\mathbf{b}^{\sigma}_i$. This constraint

[5] This property is reminiscent of the "tails" of a subsequential function (Oncina et al., 1993)

[6] The element $\boldsymbol{E}$ can be thought of as an $N \times N$ matrix where every element is ε.

ensures that the resulting SFST is subsequential by construction. To achieve this, we sample from $\{1, \ldots, N\}$ uniformly at random for each of the N rows to determine the location of the non-zero entry in each of the N rows. In terms of the SFST, if the entry $b_{ik}^{\sigma} = 1$, then our generated SFST has a transition from state $q_i \xrightarrow{\sigma} q_k$ with input symbol σ. Then, for every transition $q_i \xrightarrow{\sigma} q_k$ in our generated SFST, we sample its output symbol γ from a uniform distribution over the output alphabet Γ. This results in a transition $q_i \xrightarrow{\sigma/\gamma} q_k$. Finally, to get a canonical representation for particular SFSTs, we perform finite-state minimization (Choffrut, 2003). Minimization also ensures that our sampled SFSTs are canonicalized SFSTs which allows us to check duplicates and ensure the same SFST is not sampled more than once.

4.2 Generating the Datasets

Next we discuss the creation of the datasets we use to investigate the learnability of our sampled SFSTs. We start by sampling 1000 unique SFSTs using the process described in §4.1. All SFSTs have an input alphabet Σ of 10 symbols and output alphabet of 30 symbols. We consider SFSTs with states numbering from 10 to 100 in increments of 10, and we sample 100 unique SFSTs for each number. Additionally, we sample extra SFSTs with 21 to 39 states because it was at this point that we empirically observed performance drop-off during a preliminary investigation. In total, our experiments use 2800 unique SFSTs.

Sampling Input–output Pairs. To generate the input–output pairs to train and evaluate a neural model, we perform a random walk through the SFST where we select a transition (including the option to halt if we are in a final state) uniformly at random. Following Lake and Baroni (2018), the maximum length of an input string is capped at 50, i.e., we reject walks with more than 50 steps during sampling. Using this process, we collect $20,000$ unique input–output pairs from each SFST. We considered larger dataset sizes $(40,000)$ for the SFSTs with 29 to 39 states since we observe an accuracy drop-off in this region. All datasets are randomly split 80–20 into training and test sets.

4.3 Neural Sequence-to-Sequence Models

Our experiments make use of Wu and Cotterell's (2019) open-source neural transduction library.[7] Our experiments consider an LSTM encoder–decoder with attention in the style of Bahdanau et al. (2015). We use the following hyperparameters: 200-dimensional hidden states in the encoder and decoder, each of which have 2 layers, maximum gradient clipping normalization of 5, dropout set to 0.5, and a batch size of 64. Additionally, our alphabet tokens are embedded as 100-dimensional vectors. We train the model for $100,000$ epochs using the Adam optimizer (Kingma and Ba, 2015) with the default learning rate of 0.001. To determine the effect of model capacity, we also consider a neural sequence-to-sequence model with 300-dimensional hidden states and all other hyperparameters kept the same.

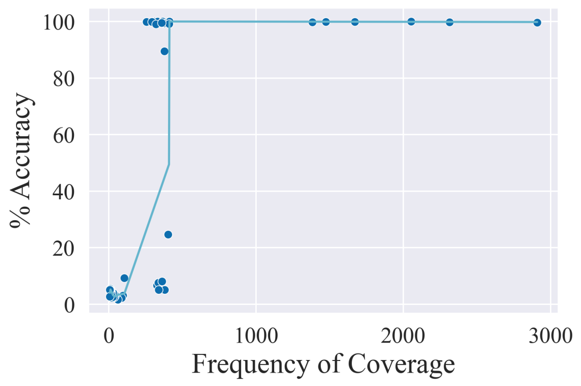


Figure 1: Accuracy versus frequency of transition coverage, with an inflection point at 400.

4.4 OSTIA

The onward subsequential transducer inference algorithm (OSTIA; Oncina et al., 1993) learns the class of subsequential functions from positive presentations of input–output strings. OSTIA works by first building a prefix-tree transducer of the training data, which is then transformed through a series of state-merging operations into the SFST encoding the function the data is drawn from. If a characteristic sample is contained in the learning data, OSTIA finds a correct transducer in polynomial (cubic) time. Since OSTIA is designed specifically to learn subsequential relations, it provides a useful baseline. Unfortunately, due to its cubic runtime, OSTIA is too slow to use on larger datasets. To give a practical speed-up, we limit the number of samples we provide to OSTIA to 1000, which is only 5% to 10% of what the neural transducers train on. This keeps OSTIA's run time

[7]The code is available at `https://github.com/shijie-wu/neural-transducer`.

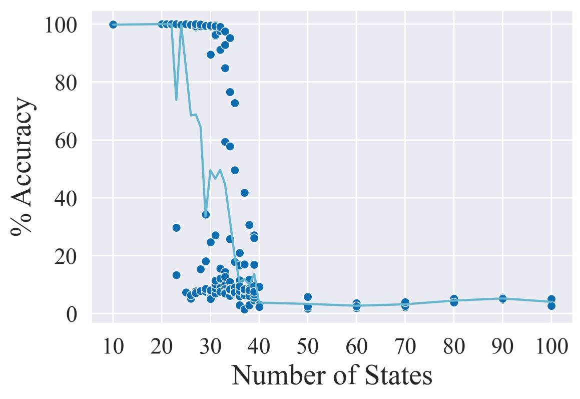


Figure 2: As the number of states in SFST increases, accuracy drops to nearly zero.

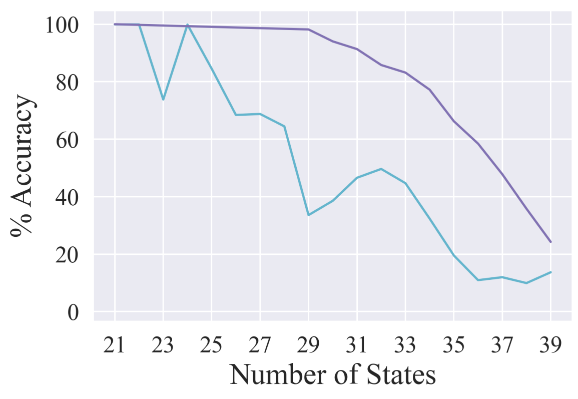


Figure 3: As SFST state size increases, average accuracy (blue line) decreases. Increasing the neural model's size improves accuracy (purple line).

roughly equivalent to its neural counterparts, but unlevels the playing field, as it were.

5 Results and Discussion

We train a neural network, as described in §4.3, on the datasets taken from our sampled SFSTs. We discuss and analyze the results below.

Minimum Transition Coverage. We define transition coverage of a given transition as the percentage of samples in the training dataset that cross that transition. The results reported in Fig. 1 reveal a threshold on the number of samples per transition required to comfortably learn the transducer: 400 samples. This may seem unsurprising given neural networks' "notorious thirst" for data (Lake and Baroni, 2018). In the vast majority of the cases when we train on a dataset that does not meet our transition coverage threshold, the neural network does not generalize to held-out data. And, indeed, in the few cases where they manage to have non-zero accuracy, we observe that early transitions in such SFSTs have attained sufficient coverage, and are responsible for the above zero performance. These findings give credence to the idea that there is a relatively simple complexity metric on the SFSTs, i.e., transition coverage, that determines whether or not the neural model will generalize.

Bigger Models Generalize Better. Additionally, we find that our discovered transition coverage threshold is *not* constant across all network sizes. For instance, when we increase the size of the hidden layers in the encoder and decoder from 200 to 300 dimensions, the sequence-to-sequence models are able to generalize on datasets where the transition coverage is lower; see Fig. 3, where the purple line is the average accuracy of the larger model. With the exception of the higher transition coverage threshold, these models follow the same trend as their lower dimensional counterparts.

OSTIA is Slow. In terms of wallclock time, we find that an open-source implementation of OSTIA[8] does not scale to dataset sizes above 1000. This makes it impossible to perform an apples-to-apples comparison of our neural sequence-to-sequence models against OSTIA. On the one hand, reducing the size of the training dataset disadvantages OSTIA. On the other hand, providing OSTIA with the full $20,000$ samples did not terminate after 3 days on a single dataset. OSTIA provably halts after a finite number of steps, but given the above, a proper comparison with neural models is not possible.

6 What about SCAN?

We now turn to the SCAN dataset and examine it in light of our findings above. First, we encode SCAN as an SFST. In so doing, we find it to be an outlier in terms of the number of states it requires, which far exceeds the 100 states of our largest SFST. In fact, we calculate that the full SFST encoding SCAN has $7,728$ states; see a small example in Fig. 4. With a finite dataset size of $20,000$ input–output pairs, it should not be possible to learn SCAN with high accuracy. However, unlike other datasets of a similar size, SCAN turns out to be nearly perfectly learnable in our (random-split) experiments. This result stands in contrast to our randomly generated SFSTs, which exhibit a consistent relationship between the complexity of a formal language and its learnability.

[8]The code is available at `github.com/alenaks/OSTIA`.

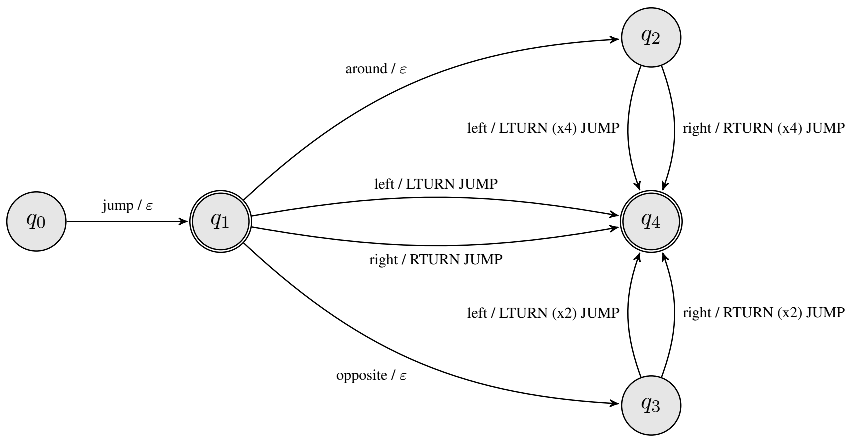


Figure 4: A part of the minimal SFST encoding SCAN; in its entirety, it has > 7000 states.

We offer a possible theoretical behavior for SCAN's surprising learnability. The class of subregular relations has a native complexity metric built into the formalism: The size of the SFST itself, as measured by the number of transitions. However, the results on the SCAN dataset indicate that the number of transitions in an SFST is *not* the native complexity metric of neural networks under consideration. This should not come as a surprise because neural networks can learn context-free transductions, which are not even representable by an SFST. While it is interesting that we empirically identify a scaling law that consistently applies to our randomly sampled SFSTs, it is not the whole story.

Indeed, SCAN is not a randomly sampled dataset from the class of SFSTs: It is generated by a handcrafted synchronous context-free grammar (Lake and Baroni, 2018, Figure 6). While SCAN is a complex SFST, requiring thousands of transitions, it can be encoded by a very small synchronous context-free grammar. Casting SCAN as an SFST therefore misrepresents its native complexity. Thus, it is left for future work to identify a class of automata whose native complexity metric can consistently predict the learnability of arbitrary language tasks; such a class will surely be higher than SFSTs on the Chomsky hierarchy given our reported results.

7 Related Work

Our paper builds on two common strains of research: The construction of datasets to benchmark compositional behavior in neural networks and research in grammatical inference.

7.1 Compositionality Datasets

There is a growing number of artificial language datasets focused on compositionality. Lake and Baroni (2018) introduced a SCAN dataset, made up of simple navigational text commands. The task is to translate the command in the simple natural language into sequences of actions. One successor to SCAN is the NACS dataset (Bastings et al., 2018), which is comparable to SCAN, but instead of mapping multiple input signals to a single duplicated output symbol (e.g., *walk twice* $\rightarrow$ `WALK WALK`), NACS does the opposite (`WALK WALK` $\rightarrow$ *walk twice*). Since SCAN is a finite language, its inverse NACS is also a finite language, and it can, thus, also be encoded as an SFST. However, this does not hold true for general SFSTs. Inverting an SFST often results in non-subsequential transducer because the output tape of SFSTs is, in general, not deterministic. Another successor is gSCAN (Ruis et al., 2020) focuses on grounding SCAN-like commands in states of a grid world. This makes gSCAN closer to Mikolov et al.'s (2016) grid world grounding for their agents. In contrast to SCAN, gSCAN requires the agent to learn differences between sizes and colours of different geometric shapes and interact with them, by moving them around the grid world. Executing a gSCAN command is therefore much more difficult than to execute its SCAN counterpart. As Ruis et al. (2020) assert, the gSCAN dataset removes artefacts in SCAN which are not central to the compositional generalization. They find that models perform worse on gSCAN than on SCAN. More recently, Bogin et al. (2022) identify that unobserved "local structures" in compositionality datasets are harder to learn if no similar structures are observed during training.

7.2 Grammatical Inference

Grammatical inference studies the ability to learning classes of formal language from data. Our work focuses on the learning of a restricted class of functions generated by a correspondingly restricted class of finite-state transducers. This allows us to synthesize our study of compositionality in neural models as rule-based inference by neural models, which we can restrict in principled ways. Finite-state machines generalise many techniques in NLP: probabilistic finite-state automata, hidden Markov models, Markov chains, n-grams, probabilistic suffix trees, deterministic stochastic probabilistic automata, weighted automata, and other syntactic objects which generate distributions over sets of possible infinite cardinality of strings, sequences, words, trees, and graphs (Vidal et al., 2005). Many grammatical inference studies of neural networks test them on samples drawn from some deterministic finite-state acceptors (Cleeremans et al., 1989). See Jacobsson (2005) for a review.

Others experiment with neural networks to see if they can learn languages higher up the Chomsky hierarchy. LSTMs (Hochreiter and Schmidhuber, 1997) can perform dynamic counting and variably learn simple counter languages such as some k-Dyck languages and $a^n b^n$ patterns (Weiss et al., 2018; Suzgun et al., 2019; Bhattamishra et al., 2020; Hewitt et al., 2020), which are generated by a finite-state machine with a counter on top (Schützenberger, 1962). In contrast, Avcu et al. (2017) show that LSTM and other RNN architectures often fail to learn long-distance dependencies drawn from simpler subregular language classes, even on large benchmarks (Mahalunkar and Kelleher, 2019). Nelson et al. (2020) study the inference of sequence-to-sequence networks, showing that RNN, LSTM, and GRU (Cho et al., 2014) systematically fail to learn a wide range of regular string copying functions generated from a family of two-way transducers, which characterize regular string-to-string functions. When augmented with attention, they reliably learn every function, and the attention history mirrors the derivations of the corresponding two-way transducers. These independently productive strands of work in compositionality and inference suggest that our work is a reasonable starting point for future interactions.

8 Conclusion

We study whether neural sequence-to-sequence models are capable of learning string-to-string transductions with Montague-style compositionality, i.e., where compositional behavior is defined to be homomorphic. To execute our study, we first provide a theoretical justification of why SFSTs meet Montague's definition. In the empirical portion of the paper, we randomly sample 2800 SFSTs using the process described in §4.1, and, then, sample input–output pairs from each SFST to create our unique string-to-string transduction datasets. We find that neural networks tend to either generalize completely or fail miserably—with little middle ground. Moreover, we identify a simple complexity metric, transition coverage, that seems to reliably allow us to predict when an SFST in our randomly sampled dataset is learnable from the dataset sampled from it.

Finally, our paper discusses how analyzing SCAN as an SFST provides a counterexample to our contention that transtion coverages reliably predicts the learnability of an SFST from a given dataset. It seems that while transition coverage is a good metric for subregular languages, there are datasets generated by synchronous context-free grammars that are learnable *despite* requiring a large number of transitions when encoded as an SFST. For example, SCAN's learnability is likely due to the fact that the synchronous context-free grammar used to generate it is relatively small and, thus, under a metric such as production coverage it would be considered simple. In conclusion, to get a more complete view of the factors underlying learnability, it may be fruitful to consider not only SFSTs, but other formalisms that describe more complex formal relations. We hypothesize that there might yet be a class of automata whose native complexity will more consistently predict the learnability of a language task.

Acknowledgements

We would like to thank the following people for their valuable insights that made this work possible: Tim Vieira, Hagen Blix, Dieuwke Hupkes, Douwe Kiela, Brenden Lake, Koustuv Sinha, Anej Svete, Alexandra Butoi, Eleanor Chodroff, Darcey Riley, and William Merrill. We also acknowledge Jennifer C. White, who, post publication, spotted some infelicities in the technical exposition which we subsequently corrected.

References

Jacob Andreas. 2019. Measuring compositionality in representation learning. In *7th International Confer-*

ence on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019. OpenReview.net.

Enes Avcu, Chihiro Shibata, and Jeffrey Heinz. 2017. Subregular complexity and deep learning. In *CLASP Papers in Computational Linguistics: Proceedings of the Conference on Logic and Machine Learning in Natural Language (LaML 2017), Gothenburg, 12 –13 June*, pages 20–33.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

Borja Balle, Rémi Eyraud, Franco M. Luque, Ariadna Quattoni, and Sicco Verwer. 2017. Results of the sequence prediction challenge (SPiCe): a competition on learning the next symbol in a sequence. In *Proceedings of The 13th International Conference on Grammatical Inference*, volume 57 of *Proceedings of Machine Learning Research*, pages 132–136, Delft, The Netherlands. PMLR.

Jasmijn Bastings, Marco Baroni, Jason Weston, Kyunghyun Cho, and Douwe Kiela. 2018. Jump to better conclusions: SCAN both left and right. *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 47–55.

Siddharth Bhaskar, Jane Chandlee, Adam Jardine, and Christopher Oakden. 2020. Boolean monadic recursive schemes as a logical characterization of the subsequential functions. In *Language and Automata Theory and Applications - LATA 2020*, Lecture Notes in Computer Science, pages 157–169. Springer.

Satwik Bhattamishra, Kabir Ahuja, and Navin Goyal. 2020. On the practical ability of recurrent neural networks to recognize hierarchical languages. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 1481–1494, Barcelona, Spain (Online). International Committee on Computational Linguistics.

Ben Bogin, Shivanshu Gupta, and Jonathan Berant. 2022. Unobserved local structures make compositional generalization hard. arXiv.

Lewis Carroll. 1871. *Through the Looking Glass*. Macmillan London.

Kyunghyun Cho, Bart van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using RNN encoder–decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1724–1734, Doha, Qatar. Association for Computational Linguistics.

Christian Choffrut. 2003. Minimizing subsequential transducers: a survey. *Theoretical Computer Science*, 292(1):131 – 143. Selected Papers in honor of Jean Berstel.

Axel Cleeremans, David Servan-Schreiber, and James L. McClelland. 1989. Finite state automata and simple recurrent networks. *Neural Computation*, 1(3):372–381.

Verna Dankers, Elia Bruni, and Dieuwke Hupkes. 2022. The paradox of the compositionality of natural language: A neural machine translation case study. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 4154–4175, Dublin, Ireland. Association for Computational Linguistics.

Jerry Fodor. 1998. There are no recognitional concepts; not even red. *Philosophical Issues*, 9:1–14.

Jerry A. Fodor and Zenon W. Pylyshyn. 1988. Connectionism and cognitive architecture: A critical analysis. *Cognition*, 28(1-2):3–71.

John Hewitt, Michael Hahn, Surya Ganguli, Percy Liang, and Christopher D. Manning. 2020. RNNs can generate bounded hierarchical languages with optimal memory. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1978–2010, Online. Association for Computational Linguistics.

Colin de la Higuera. 2010. *Grammatical Inference: Learning Automata and Grammars*. Cambridge University Press.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural Computation*, 9(8):1735–1780.

Dieuwke Hupkes, Verna Dankers, Mathijs Mul, and Elia Bruni. 2020a. Compositionality decomposed: How do neural networks generalise? (extended abstract). In *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, IJCAI-20*, pages 5065–5069. International Joint Conferences on Artificial Intelligence Organization. Journal track.

Dieuwke Hupkes, Verna Dankers, Mathijs Mul, and Elia Bruni. 2020b. Compositionality decomposed: How do neural networks generalise? (extended abstract). In *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, IJCAI-20*, pages 5065–5069. International Joint Conferences on Artificial Intelligence Organization. Journal track.

Henrik Jacobsson. 2005. Rule extraction from recurrent neural networks: A taxonomy and review. *Neural Computation*, 17(6):1223–1263.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *3rd International Conference on Learning Representations,* *ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

Brenden Lake and Marco Baroni. 2018. Generalization without systematicity: On the compositional skills of sequence-to-sequence recurrent networks. In *International Conference on Machine Learning*, pages 2873–2882. PMLR.

Abhijit Mahalunkar and John Kelleher. 2019. Multi-element long distance dependencies: Using SPk languages to explore the characteristics of long-distance dependencies. In *Proceedings of the Workshop on Deep Learning and Formal Languages: Building Bridges*, pages 34–43, Florence. Association for Computational Linguistics.

Gary F. Marcus. 1998. Rethinking eliminative connectionism. *Cognitive Psychology*, 37(3):243–282.

Tomas Mikolov, Armand Joulin, and Marco Baroni. 2016. A roadmap towards machine intelligence. In *International Conference on Intelligent Text Processing and Computational Linguistics*, pages 29–61. Springer.

Mehryar Mohri. 1997. Finite-state transducers in language and speech processing. *Computational Linguistics*, 23(2):269–311.

Richard Montague. 1970. Universal grammar. *Theoria*.

Max Nelson, Hossep Dolatian, Jonathan Rawski, and Brandon Prickett. 2020. Probing RNN encoder-decoder generalization of subregular functions using reduplication. In *Proceedings of the Society for Computation in Linguistics 2020*, pages 167–178, New York, New York. Association for Computational Linguistics.

José Oncina, Pedro García, and Enrique Vidal. 1993. Learning subsequential transducers for pattern recognition interpretation tasks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 15(5):448–458.

Barbara Partee. 1995. Lexical semantics and compositionality. *An Invitation to Cognitive Science: Language*, 1:311–360.

Jonathan Rawski and Jeffrey Heinz. 2019. No free lunch in linguistics or machine learning: Response to pater. *Language*, 95(1):e125–e135.

James Rogers and Geoffrey K. Pullum. 2011. Aural pattern recognition experiments and the subregular hierarchy. *Journal of Logic, Language and Information*, 20(3):329–342.

Laura Ruis, Jacob Andreas, Marco Baroni, Diane Bouchacourt, and Brenden M. Lake. 2020. A benchmark for systematic generalization in grounded language understanding. In *Advances in Neural Information Processing Systems 34*.

Marcel P. Schützenberger. 1962. Finite counting automata. *Information and Control*, 5:91–107.

Mirac Suzgun, Yonatan Belinkov, Stuart Shieber, and Sebastian Gehrmann. 2019. LSTM networks can perform dynamic counting. In *Proceedings of the Workshop on Deep Learning and Formal Languages: Building Bridges*, pages 44–54, Florence. Association for Computational Linguistics.

E. Vidal, F. Thollard, C. de la Higuera, F. Casacuberta, and R. C. Carrasco. 2005. Probabilistic finite-state machines - part I. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(7):1013–1025.

Gail Weiss, Yoav Goldberg, and Eran Yahav. 2018. On the practical computational power of finite precision RNNs for language recognition. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 740–745, Melbourne, Australia. Association for Computational Linguistics.

Jennifer C. White and Ryan Cotterell. 2021. Examining the inductive bias of neural language models with artificial languages. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 454–463, Online. Association for Computational Linguistics.

Jennifer C. White and Ryan Cotterell. 2022. Equivariant transduction through invariant alignment. In *Proceedings of the 29th International Conference on Computational Linguistics*, Gyeongju, Republic of Korea. International Committee on Computational Linguistics.

Shijie Wu and Ryan Cotterell. 2019. Exact hard monotonic attention for character-level transduction. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 1530–1537, Florence, Italy. Association for Computational Linguistics.

A Proofs

Proposition 1. *Let Σ and Γ be two alphabets. The function f is a string homomorphism between $(\Sigma^*, \circ)$ and $(\Gamma^*, \circ)$ iff it is a minimal non-empty complete subsequential finite-state transducer with one state.*

Proof. $\Rightarrow$ Assume a homomorphism f and an alphabet Σ are given. Construct a finite-state transducer with $Q = F = \{q_0\}$. Let $\Gamma = \{f(\sigma) \mid \sigma \in \Sigma\}$. Create a transition $q_0 \xrightarrow{\sigma/f(\sigma)} q_0$ for every $\sigma \in \Sigma$. This finite-state transducer is minimal (i.e. non-empty and already has one state), therefore it is the minimal representation of f; see Choffrut (2003) for more details.
$\Leftarrow$ Since $\mathcal{T} = \langle \Sigma, \Gamma, Q, q_0, F, \delta \rangle$ is non-empty, we know that its only state q_0 is also a final state. Thus, $\mathcal{T}$ maps $\varepsilon \mapsto \varepsilon$, i.e., the identity element to the identity element. Let $\boldsymbol{\sigma} \in \Sigma^*$ and suppose that under $\mathcal{T}$ we have $\boldsymbol{\sigma} \mapsto \boldsymbol{\gamma}$. Suppose $\boldsymbol{\sigma} = \boldsymbol{\sigma}'\boldsymbol{\sigma}''$. Since $\mathcal{T}$ is subsequential, there is a unique path in $\mathcal{T}$ that transduces $\boldsymbol{\sigma} \mapsto \boldsymbol{\gamma}$ with exactly $|\boldsymbol{\sigma}|$ transitions. Let $\boldsymbol{\gamma}'$ be the output yield of the first $|\boldsymbol{\sigma}'|$ transitions and let $\boldsymbol{\gamma}''$ be the output yield of the next $|\boldsymbol{\sigma}''|$ transitions. Thus, $\boldsymbol{\sigma}' \mapsto \boldsymbol{\gamma}'$ and $\boldsymbol{\sigma}'' \mapsto \boldsymbol{\gamma}''$ which shows that $\mathcal{T}$ is a homomorphism since $\boldsymbol{\gamma} = \boldsymbol{\gamma}'\boldsymbol{\gamma}''$ and $\mathcal{T}$ a single-state transducer and we remain in that state. Finally, we assumed completeness so that f is a total function over Σ^*. ■

Proposition 2. *Let $\mathcal{A} = \langle \Sigma, Q, q_0, F, \delta \rangle$ be a complete deterministic finite-state automaton. Let $\mathcal{M} \overset{\text{def}}{=} \left\{ \boldsymbol{M}^\sigma \mid \sigma \in \Sigma \right\}$ be the set of $\mathcal{A}$'s symbol transition matrices. Then $(\mathcal{M}^*, \otimes)$ with $\otimes$ defined in Eq. (2) is a* ***transition monoid*** *with $\boldsymbol{E}$ as a distinguished identity element.*[9]

Proof. We define $\boldsymbol{E}$ to be the identity element, i.e., for any $\boldsymbol{A} \in \mathcal{M}^*$, we define $\boldsymbol{E} \otimes \boldsymbol{A} \overset{\text{def}}{=} \boldsymbol{A} \overset{\text{def}}{=} \boldsymbol{A} \otimes \boldsymbol{E} = \boldsymbol{A}$. Closure follows from the fact for $\boldsymbol{A}, \boldsymbol{B} \in \mathcal{M}^+$, we have that $\boldsymbol{A} \otimes \boldsymbol{B}$ has by construction exactly one non-zero entry in every row, as elaborated upon in the main text. To check associativity, consider

$$\left(\boldsymbol{M}^{\boldsymbol{\sigma}} \otimes \boldsymbol{M}^{\boldsymbol{\sigma}'} \otimes \boldsymbol{M}^{\boldsymbol{\sigma}''} \right)_{ik} = M^{\boldsymbol{\sigma}}_{ij} M^{\boldsymbol{\sigma}'}_{jj'} M^{\boldsymbol{\sigma}''}_{j'k} \tag{4}$$

for some i, j, j', k. By the associativity of string concatenation (including when it is augmented to include the zero string), we have that

$$(M^{\boldsymbol{\sigma}}_{ij} M^{\boldsymbol{\sigma}'}_{jj'}) M^{\boldsymbol{\sigma}''}_{j'k} = M^{\boldsymbol{\sigma}}_{ij} (M^{\boldsymbol{\sigma}'}_{jj'} M^{\boldsymbol{\sigma}''}_{j'k}) \tag{5}$$

which in turn implies that

$$\left((\boldsymbol{M}^{\boldsymbol{\sigma}} \otimes \boldsymbol{M}^{\boldsymbol{\sigma}'}) \otimes \boldsymbol{M}^{\boldsymbol{\sigma}''} \right)_{ik} = \left(\boldsymbol{M}^{\boldsymbol{\sigma}} \otimes (\boldsymbol{M}^{\boldsymbol{\sigma}'} \otimes \boldsymbol{M}^{\boldsymbol{\sigma}''}) \right)_{ik} \tag{6}$$

Thus, we have that $(\mathcal{M}^*, \otimes)$ is a monoid. ■

Proposition 3. *Let $\mathcal{T} = \langle \Sigma, \Gamma, Q, q_0, F, \delta \rangle$ be a complete subsequential finite-state transducer. Let $(\mathcal{M}^*_\Sigma, \otimes)$ be the transition monoid associated with the input alphabet Σ and let $(\mathcal{M}^*_\Gamma, \otimes)$ be the transition monoid associated with the output alphabet Γ. Then there exists a homomorphism $f : \mathcal{M}^*_\Sigma \to \mathcal{M}^*_\Gamma$.*

Proof. We define f as follows. First, we define $f(\boldsymbol{E}) \overset{\text{def}}{=} \boldsymbol{E}$. Next, consider an arbitrary element $\mathcal{M}^*_\Sigma$. Because $\mathcal{M}^*_\Sigma$ is a Kleene closure, we may write this arbitrary element as $\boldsymbol{M}^{\sigma_1}_\Sigma \otimes \cdots \otimes \boldsymbol{M}^{\sigma_K}_\Sigma$. Now we define $f(\boldsymbol{M}^{\sigma_1}_\Sigma \otimes \cdots \otimes \boldsymbol{M}^{\sigma_K}_\Sigma) \overset{\text{def}}{=} \boldsymbol{M}^{\sigma_1}_\Gamma \otimes \cdots \otimes \boldsymbol{M}^{\sigma_K}_\Gamma$. Thus,

$$\begin{aligned}
&f(\boldsymbol{M}^{\sigma_1}_\Sigma \otimes \cdots \otimes \boldsymbol{M}^{\sigma_K}_\Sigma) \\
&= f\left((\boldsymbol{M}^{\sigma_1}_\Sigma \otimes \cdots \otimes \boldsymbol{M}^{\sigma_k}_\Sigma) \otimes (\boldsymbol{M}^{\sigma_{k+1}}_\Sigma \otimes \cdots \otimes \boldsymbol{M}^{\sigma_K}_\Sigma) \right) \\
&= \left(\boldsymbol{M}^{\sigma_1}_\Gamma \otimes \cdots \otimes \boldsymbol{M}^{\sigma_k}_\Gamma \right) \otimes \left(\boldsymbol{M}^{\sigma_{k+1}}_\Gamma \otimes \cdots \otimes \boldsymbol{M}^{\sigma_K}_\Gamma \right) \\
&= \boldsymbol{M}^{\sigma_1}_\Gamma \otimes \cdots \otimes \boldsymbol{M}^{\sigma_K}_\Gamma
\end{aligned}$$

This proves f is a homomorphism. ■

[9]The element $\boldsymbol{E}$ can be thought of as an $N \times N$ matrix where every element is ε.