A Basic Compositional Model for Spiking Neural Networks

Nancy Lynch
MIT
lynch@csail.mit.edu

Cameron Musco
Microsoft Research
camusco@microsoft.com

August 14, 2018

1 Introduction

This paper is part of a project on developing an *algorithmic theory of brain networks*, based on *stochastic Spiking Neural Network (SNN)* models. Inspired by tasks that seem to be solved in actual brains, we are defining abstract problems to be solved by these networks. In our work so far, we have developed models and algorithms for the *Winner-Take-All* problem from computational neuroscience [LMP17a, Mus18], and problems of similarity detection and neural coding [LMP17b]. We plan to consider many other problems and networks, including both static networks and networks that learn.

This paper is about basic theory for the stochastic SNN model. In particular, we define a simple version of the model. This version assumes that the neurons' only state is a Boolean, indicating whether the neuron is firing or not. In later work, we plan to develop variants of the model with more elaborate state; we expect that our results should extend to these variants as well, but this remains to be worked out. We also define an *external behavior notion* for SNNs, which can be used for stating requirements to be satisfied by the networks.

We then define a *composition operator* for SNNs. We prove that our external behavior notion is "compositional", in the sense that the external behavior of a composed network depends only on the external behaviors of the component networks. We also define a *hiding operator* that reclassifies some output behavior of an SNN as internal. We give basic results for hiding.

Finally, we give a formal definition of a *problem* to be solved by an SNN, and give basic results showing how composition and hiding of networks affect the problems that they solve. We illustrate our definitions with three examples: building a circuit out of gates, building an "Attention" network out of a "Winner-Take-All" network and a "Filter" network, and a toy example involving combining two networks in a cyclic fashion.

2 The Model

For our model definitions, we first specify the structure of our networks. Then we describe how the networks execute; this involves defining individual (non-probabilistic) executions and then defining probabilistic behavior. Next we define the external behavior of a network. Finally, we give two examples: a Boolean circuit and a Winner-Take-All network.

2.1 Network structure

Assume a universal set U of neuron names. A *firing pattern* for a set $V \subseteq U$ of neuron names is a mapping from V to $\{0, 1\}$. Here, 1 represents "firing" and 0 represents "not firing".

A neural network $\mathcal{N}$ consists of:

- N, a subset of U, partitioned into input neurons N_{in}, output neurons N_{out}, and internal (auxiliary) neurons N_{int}. We sometimes write N_{ext} as shorthand for $N_{in} \cup N_{out}$, and N_{lc} as shorthand for $N_{out} \cup N_{int}$. (Here, lc stands for "locally controlled")..

  Each neuron $u \in N_{lc}$ has an associated *bias*, $bias(u) \in \mathbb{R}$; this can be any real number, positive, negative, or 0.

- E, a set of directed edges between neurons. We permit self-loops.

  Each edge e has a *weight*, $weight(e)$, which is a nonzero real number.

- F_0, an initial firing pattern for the set N_{lc} of non-input neurons.

We assume that input neurons have no incoming edges, not even self-loops. Output neurons may have incoming or outgoing edges, or both.

2.2 Executions and probabilistic executions

2.2.1 Executions and traces

A *configuration* of a neural network $\mathcal{N}$ is a firing pattern for N, the set of all the neurons in the network. We consider several related definitions:

- An *input configuration* is a firing pattern for the input neurons, N_{in}. An *output configuration* is a firing pattern for the output neurons, N_{out}. An *internal configuration* is a firing pattern for the internal neurons, N_{int}.

- A *non-input configuration* is a firing pattern for the internal and output neurons, N_{lc}.

- An *external configuration* is a firing pattern for the input and output neurons, N_{ext}.

We define projections of configurations onto subsets of N. Thus, if C is a configuration and $M \subseteq N$, then $C\lceil M$ is the firing pattern for M obtained by projecting C onto the neurons in M. In particular, we have $C\lceil N_{in}$ for the projection of C on the input neurons, $C\lceil N_{out}$ for the output neurons, $C\lceil N_{int}$ for the internal neurons, $C\lceil N_{ext}$ for the external neurons, and $C\lceil N_{lc}$ for the non-input neurons.

An *initial configuration* is a configuration C such that $C\lceil N_{lc} = F_0$. The values for the input neurons are arbitrary.

An *execution* α of $\mathcal{N}$ is a (finite or infinite) sequence of configurations, $C_0, C_1, \ldots$, where C_0 is an initial configuration. The *length* of a finite execution $\alpha = C_0, C_1, ..., C_t$, $length(\alpha)$, is defined to be t. The *length* of an infinite execution is defined to be ∞.

We define projections of executions: If $\alpha = C_0, C_1, \ldots$ is an execution of $\mathcal{N}$ and $M \subseteq N$, then $\alpha\lceil M$ is the sequence $C_0\lceil M, C_1\lceil M, \ldots$. We define an M-execution of $\mathcal{N}$ to be $\alpha\lceil M$ for any execution α of $\mathcal{N}$. Note that an M-execution restricts the initial firing states of only the non-input neurons that are in M, that is, the neurons in $M \cap N_{lc}$. We define an *input execution* to be an M-execution where $M = N_{in}$, and similarly for an *output execution*, an *internal execution*, an *external execution*, and a *locally-controlled execution (or lc-execution)* .

For an execution α, we sometimes write $trace(\alpha)$ to denote $\alpha\lceil N_{ext}$, the projection of α on the external neurons. We define a *trace* of $\mathcal{N}$ to the the trace of any execution of α.

If γ is any finite M-execution, for $M \subseteq N$, then we define $A(\gamma)$ to be the set of executions α of $\mathcal{N}$ such that γ is a prefix of $\alpha\lceil M$. This means that α can have any firing states for the neurons that are not in M, except for the initial states of neurons in N_{lc}, which are determined by F_0. We will often consider the special case where $M = N_{ext}$, i.e., where γ is a trace of $\mathcal{N}$.

Lemma 1 *Let α_1 and α_2 be finite executions of $\mathcal{N}$.*

1. *If neither α_1 nor α_2 is an extension of the other, then $A(\alpha_1)$ and $A(\alpha_2)$ are disjoint.*
2. *If α_1 is an extension of α_2, then $A(\alpha_1) \subseteq A(\alpha_2)$.*

2.2.2 Probabilistic executions

We define a unique "probabilistic execution" for any particular infinite input execution β_{in}. Formally, such a probabilistic execution is a probability distribution P on the sample space of infinite executions α of the network such that $\alpha \lceil N_{in} = \beta_{in}$; we say that such executions are *consistent with* β_{in}. Note that all of these executions have the same initial configuration, call it C_0. This is constructed from the 0 element of β_{in} and the initial non-input firing pattern for the network, F_0.

The σ-algebra of measurable sets is generated from the "cones", each of which is the set of infinite executions that extend a particular finite execution. Formally, if α is a finite execution such that $\alpha \lceil N_{in}$ is a prefix of β_{in}, then the "cone" of α is simply $A(\alpha)$, as defined earlier. The other measurable sets in the σ-algebra are obtained by starting from these cones and closing under countable union, countable intersection, and complement.

Now we define the probabilities for the measurable sets. We start by explicitly defining the probabilities for the cones, $P(A(\alpha))$. Based on these, we can derive the probabilities of the other measurable sets in a unique way, using general measure extension theorems. Segala presents a similar construction for probabilistic executions in his PhD thesis, Chapter 4 [Seg95].

We compute the probabilities $P(A(\alpha))$ recursively based on the length of α (which is here always assumed to be consistent with β_{in}):

1. α is of length 0.
   Then α consists of just the initial configuration C_0; define $P(A(\alpha)) = 1$.

2. α is of length t, $t > 0$.
   Let α' be the length-$(t-1)$ prefix of α. We determine the probability q of extending α' to α. Then the probability $P(A(\alpha))$ is simply $P(A(\alpha')) \times q$.

   Let C be the final configuration of α and C' the final configuration of α'. Then for each neuron $u \in N_{lc}$ separately, use C' and the weights of u's incoming edges to compute the potential and then the firing probability for neuron u. In more detail: For each u, we first calculate a *potential*, pot_u, defined as

   $$\sum_{(v,u)\in E} C'(v) weight(v,u) - bias(u).$$

   We then convert pot_u to a firing probability p_u using the standard sigmoid function:

   $$p_u = \frac{1}{1 + e^{-pot_u/\lambda}},$$

   where λ is a positive real number "temperature" parameter.[1] [2] We combine all those probabilities to compute the probability of generating C from C': for each $u \in N_{lc}$ such that

[1] We assume a standard sigmoid function. However, the results of this paper don't appear to depend much on the precise function definition. Different functions could be used, subject to some basic constraints.

[2] We will try to generalize our model to include other state besides just firing status. For example, we might remember history of firing, or history of incoming potentials.

$C(u) = 1$, use the calculated probability p_u, and for each $u \in N_{lc}$ for which $C(u) = 0$, use $1 - p_u$. The product

$$\prod_{u \in N_{lc}: C(u)=1} p_u \times \prod_{u \in N_{lc}: C(u)=0} (1 - p_u)$$

is the probability of generating C from C', which is the needed probability q of extending α' to α.

We will often consider conditional probabilities of the form $P(A(\alpha_1)|A(\alpha_2))$. Because we use a sigmoid function, we know that $P(A(\alpha_2))$ cannot be 0, and so this conditional probability is actually defined.[3]

From now on in this subsection, we assume a particular β_{in} and P. The following lemma follows immediately from Lemma 1.

Lemma 2 *Let α_1 and α_2 be finite executions of $\mathcal{N}$ that are consistent with β_{in}.*

1. *If neither α_1 nor α_2 is an extension of the other, then $P(A(\alpha_1)|A(\alpha_2)) = 0$.*
2. *If α_1 is an extension of α_2, then $P(A(\alpha_1)|A(\alpha_2)) = \frac{P(A(\alpha_1))}{P(A(\alpha_2))}$.*

So we can easily compute the conditional probabilities from the absolute probabilities. Conversely, we can easily compute the absolute probabilities from the conditional ones, by unwinding the recursive definition above:

Lemma 3 *Let α be a length-t execution of $\mathcal{N}$, $t > 0$. Let α_i, $0 \leq i \leq t$ be the successive prefixes of α (so that $\alpha_t = \alpha$). Then*

$$P(A(\alpha)) = P(A(\alpha_1)|A(\alpha_0)) \times P(A(\alpha_2)|A(\alpha_1)) \cdots \times P(A(\alpha_t)|A(\alpha_{t-1})).$$

Notice in the above expression, we did not start with a term for $P(\alpha_0)$. This is not needed because we are considering only traces in which α_0 is obtained from β_{in} and the initial assignment F_0. So α_0 is determined, and $P(\alpha_0) = 1$.

Since we can compute the conditional and absolute probabilities from each other, either can be used to characterize the probabilistic execution.

Tree representation: The probabilistic execution for β_{in} can be visualized as an infinite tree of configurations, where the tree nodes at level t represent the configurations that might occur at time t (with the given input execution). The configuration at the root of the tree is the initial configuration C_0. Each infinite branch of the tree represents an infinite execution of the network, and finite initial portions of branches represent finite executions. If α is a finite branch in the tree, then we can associate the probability $P(A(\alpha))$ with the node at the end of the branch; this is simply the probability of reaching the node during probabilistic operation of the network, using the inputs from β_{in}.

[3] One useful property of our sigmoid functions is that the probabilities are never exactly 0 or 1, which makes it unnecessary to worry about 0-probability sets when conditioning. We have to be careful to retain this property if we consider different functions.

2.2.3 Probabilistic traces

Now we define a unique "probabilistic trace" for any particular infinite input execution β_{in}. Formally, such a probabilistic trace is a probability distribution Q on the sample space of infinite traces β of the network such that $\beta \lceil N_{in} = \beta_{in}$. All of these traces have the same initial configuration, constructed from the 0 element of β_{in} and the initial output firing pattern for the network, $F_0 \lceil N_{out}$.

The basic measurable sets are the sets of traces that extend a particular finite trace. For a particular finite trace β, we define

$$B(\beta) = \{trace(\alpha) : \beta \text{ is a prefix of } trace(\alpha)\}$$

To define probabilities for the sets $B(\beta)$, we rely on the probabilistic execution for β_{in}. If β is a finite trace of $\mathcal{N}$, then $A(\beta)$ has already been defined. Then define the probability of $B(\beta)$ to be simply $P(A(\beta))$.

The following lemma expands the probability $P(A(\beta))$ in terms of probabilities for the relevant executions.

Lemma 4 *If β is a finite trace of $\mathcal{N}$, then*

$$A(\beta) = \bigcup_{\alpha: trace(\alpha)=\beta} A(\alpha), \text{ and } P(A(\beta)) = \sum_{\alpha: trace(\alpha)=\beta} P(A(\alpha)).$$

The next lemma describes conditional probabilities for one-step extensions:

Lemma 5 *Let α be a finite execution of $\mathcal{N}$ of length > 0 that is consistent with β_{in}. Suppose α' is its one-step prefix. Let $\beta = trace(\alpha) = \alpha \lceil N_{ext}$, and $\beta' = trace(\alpha') = \alpha' \lceil N_{ext}$. Then α', β, and β' are also consistent with β_{in},[4] and*

1. *$A(\alpha) \subseteq A(\alpha')$, and $P(A(\alpha)|A(\alpha')) = \frac{P(A(\alpha))}{P(A(\alpha'))}$.*
2. *$A(\alpha) \subseteq A(\beta)$, and $P(A(\alpha)|A(\beta)) = \frac{P(A(\alpha))}{P(A(\beta))}$.*
3. *$A(\alpha) \subseteq A(\beta')$, and $P(A(\alpha)|A(\beta')) = \frac{P(A(\alpha))}{P(A(\beta'))}$.*
4. *$A(\alpha') \subseteq A(\beta')$, and $P(A(\alpha')|A(\beta')) = \frac{P(A(\alpha'))}{P(A\beta'))}$.*
5. *$A(\beta) \subseteq A(\beta')$, and $P(A(\beta)|A(\beta')) = \frac{P(A(\beta))}{P(A(\beta'))}$.*

Lemma 6 *Let α, α', β, and β' be as in Lemma 5. Then*

$$P(A(\alpha)|A(\beta)) = P(A(\alpha)|A(\beta')) \times \frac{P(A(\beta'))}{P(A(\beta))} = \frac{P(A(\alpha)|A(\beta'))}{P(A(\beta)|A(\beta'))}.$$

Proof. By Lemma 5. □

The next lemma gives some simple equivalent formulations of a one-step extension of traces, by unwinding definitions in terms of executions.

Lemma 7 *Suppose that β is a finite trace of length $t > 0$ that is consistent with β_{in}. Suppose that β' is the length-$(t-1)$ prefix of β. Then $P(A(\beta)|A(\beta'))$ is equal to all of the following:*

1. *$\sum_{\alpha': trace(\alpha')=\beta'} (P(A(\alpha')|A(\beta')) \times P(A(\beta)|A(\alpha')))$.*

[4] As before, this means that their projections on N_{in} are prefixes of β_{in}.

2. $\frac{1}{P(A(\beta'))} \sum_{\alpha': trace(\alpha')=\beta'} (P(A(\alpha')) \times P(A(\beta)|A(\alpha')))$.

3. $\frac{1}{P(A(\beta'))} \sum_{\alpha': trace(\alpha')=\beta'} P(A(\alpha')) \sum_{\alpha: trace(\alpha)=\beta \text{ and } \alpha \text{ extends } \alpha'} P(A(\alpha)|A(\alpha'))$.

4. $\frac{1}{P(A(\beta'))} \sum_{\alpha,\alpha': trace(\alpha)=\beta, \alpha' \text{ is the length } t-1 \text{prefix of } \alpha} (P(A(\alpha')) \times P(A(\alpha)|A(\alpha')))$.

5. $\frac{1}{P(A(\beta'))} \sum_{\alpha: trace(\alpha)=\beta} P(A(\alpha))$.

6. $\frac{P(A(\beta))}{P(A(\beta'))}$.

We can also give a lemma about repeated conditioning, as for probabilistic executions:

Lemma 8 *Let β be a length-t trace of $\mathcal{N}$, $t > 0$. Let β_i, $0 \leq i \leq t$, be the successive prefixes of β (so that $\beta_t = \beta$). Then*

$$P(A(\beta)) = P(A(\beta_1)|A(\beta_0)) \times P(A(\beta_2)|A(\beta_1) \cdots \times P(A(\beta_t)|A(\beta_{t-1})).$$

As before, in the above expression, we did not use a separate term for $P(\beta_0)$. This is not needed because we are considering only traces in which β_0 is obtained from β_{in} and the initial assignment F_0. So β_0 is determined, and $P(\beta_0) = 1$.

We will need some other easy facts about executions and traces, for example:

Lemma 9 *Let α be a finite execution of $\mathcal{N}$ of length > 0, that is consistent with β_{in}. Let α' be the one-step prefix of α and $\beta' = trace(\alpha')$. Then $P(A(\alpha)|A(\beta')) = P(A(\alpha)|A(\alpha')) \times P(A(\alpha')|A(\beta'))$.*

Proof. By Lemma 5, we see that

$$P(A(\alpha)|A(\beta')) = \frac{P(A(\alpha))}{P(A(\beta'))} = \frac{P(A(\alpha))}{P(A(\alpha'))} \times \frac{P(A(\alpha'))}{P(A(\beta'))} = P(A(\alpha)|A(\alpha')) \times P(A(\alpha')|A(\beta')).$$

□

Lemma 10 1. *Suppose that α is a finite execution of $\mathcal{N}$ that is consistent with β_{in}. Then $P(A(\alpha)) = P(A(\alpha\lceil N_{lc}))$.*

2. *Suppose that β is a finite trace of $\mathcal{N}$ that is consistent with β_{in}. Then $P(A(\beta)) = P(A(\beta\lceil N_{out}))$.*

Proof. Since the input execution is already fixed at β_{in}, the probability for α is just the probability for the projection of α on the non-input neurons. Similarly for β. □

2.3 External behavior of a network

So far we have talked about individual probabilistic traces, which depend on a fixed input execution. Now we define the *external behavior* of a network, to capture its visible behavior for all possible inputs. Later in the paper, in Section 5, we will show that our notion of external behavior is *compositional*, which implies that the external behavior of $\mathcal{N}^1 \times \mathcal{N}^2$ is unabiguously determined by the external behavior of $\mathcal{N}^1$ and the external behavior of $\mathcal{N}^2$.

Behavior Definition: Our definition of external behavior is based on the entire collection of probabilities for the cones of all finite traces. Namely, the external behavior $Beh(\mathcal{N})$ is the mapping f that maps each infinite input execution β_{in} of $\mathcal{N}$ to the collection of probabilities $\{P(A(\beta))\}$, where β is a finite trace of $\mathcal{N}$ that is consistent with β_{in}.[5]

Other definitions of external behavior might be possible. Any such definition would have to assign some "behavior object" to each network. In general, we define two external behavior notions B_1 and B_2 to be *equivalent* provided that the following holds. Suppose that $\mathcal{N}$ and $\mathcal{N}'$ are two networks with the same input neurons and the same output neurons. Then $B_1(\mathcal{N}) = B_1(\mathcal{N}')$ if and only if $B_2(\mathcal{N}) = B_2(\mathcal{N}')$.

In this paper, we find it useful to define a second, "auxiliary" external behavior notion, based on one-step conditional probabilities. This will be useful in our proofs for compositionality.

Auxiliary Behavior Definition: $Beh_2(\mathcal{N})$ is the mapping f_2 that maps each infinite input execution β_{in} of $\mathcal{N}$ to the collection of conditional probabilities $\{P(A(\beta)|A(\beta'))\}$, where β is a finite trace of $\mathcal{N}$ with length > 0 that is consistent with β_{in}, and β' is the one-step prefix of β.

Lemma 11 *The two behavior notions Beh and Beh_2 are equivalent.*

Proof. Suppose that $\mathcal{N}$ and $\mathcal{N}'$ are two networks with the same input neurons and the same output neurons. We show that Beh and Beh_2 are equivalent by arguing two directions separately:

1. If $Beh(\mathcal{N}) = Beh(\mathcal{N}')$ then $Beh_2(\mathcal{N}) = Beh_2(\mathcal{N}')$. This follows because the conditional probability $P(A(\beta)|A(\beta'))$ is determined as a function of the unconditional probabilities $P(A(\beta))$ and $P(A(\beta'))$; see Lemma 5, Part 5.

2. If $Beh_2(\mathcal{N}) = Beh_2(\mathcal{N}')$ then $Beh(\mathcal{N}) = Beh(\mathcal{N}')$. This follows because the unconditional probability $P(A(\beta))$ is determined as a function of the conditional probabilities, see Lemma 8.

□

2.4 Examples

In this subsection we give two fundamental examples to illustrate our definitions so far.

2.4.1 Simple Boolean gate networks

Figure 1 depicts the structure of simple SNNs that represent and-gates, or-gates, and not-gates. For completeness, we also include an SNN representing the identity computation.

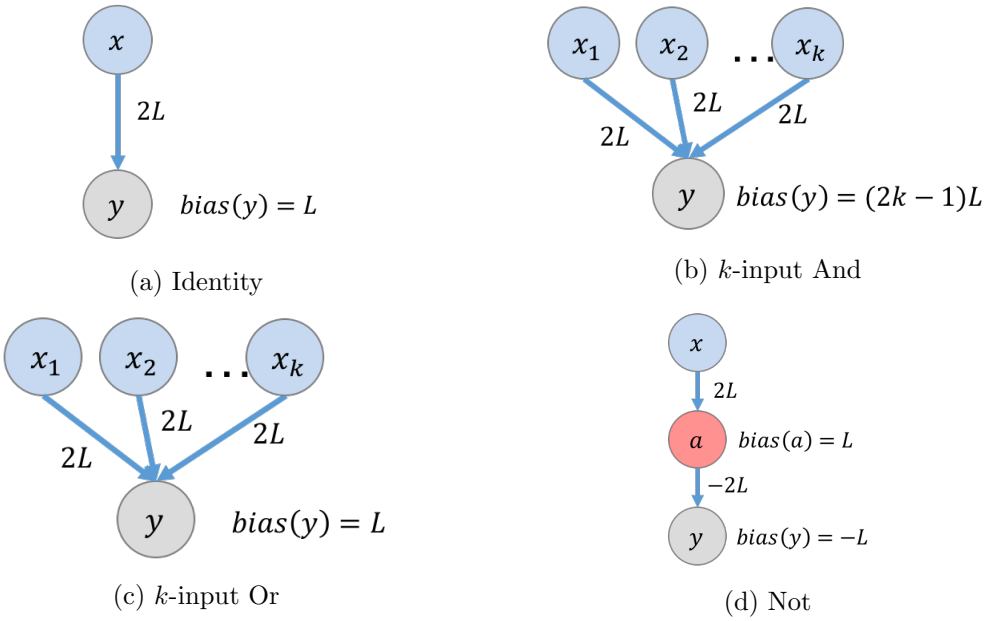
We describe the operation of each of these types of networks, in turn. Fix a value λ for the temperature parameter of the sigmoid function. Fix an error probability δ, $0 < \delta < 1$. Assume for each case below that the initial firing status for the non-input neurons is 0.

Throughout this section, we use the abbreviation L for the quantity $\lambda \ln(\frac{1-\delta}{\delta})$.

Identity network: This has one input neuron and one output neuron, connected by an edge with weight w. The output neuron has bias b. We define $b = L$ and $w = 2L$. Then we have

$$e^{b/\lambda} = \frac{1-\delta}{\delta} = \frac{1}{\delta} - 1.$$

[5] Formally, this collection is a mapping from finite traces β to probabilities $P(A(\beta))$, but the use of two mappings here may look slightly confusing.



(a) Identity

(b) k-input And

(c) k-input Or

(d) Not

Figure 1: Networks representing simple Boolean gates

With these settings, we get potential $-b$ and firing probability δ when the input firing state is 0, and potential $w - b = b$ and firing probability $1 - \delta$ when the input firing state is 1. More precisely, consider just the input firing state at time 0. Whether it is 0 or 1, the probability that the output firing state at time 1 is the same is exactly $1 - \delta$.

Our model also describes what happens with an arbitrary infinite input firing sequence, not just the initial inputs. Let β_{in} be an arbitrary infinite firing sequence for the input neuron.

Let β be a trace of length $t \geq 1$ that is consistent with β_{in}. Suppose further that, for every t', $1 \leq t' \leq t$, the output status at time t' in β is equal to the input status at time $t' - 1$. Then by repeated use of the argument above, we get that $P(\beta) = (1 - \delta)^{t-1}$.

Now suppose that β is a length t trace as above. Suppose that the output firing status at time t in β is equal to the input status at time $t' - 1$, but the output status values for all earlier times is arbitrary. Suppose that β' is the one-step prefix of β. Then we can show that $P(\beta|\beta') = 1 - \delta$. It follows that, for every time $t \geq 1$, the probability that the output at time t is equal to the input at time $t - 1$ is $1 - \delta$. This uses the law of Total Probability, over all the possible length $t - 1$ output firing sequences.

k-input And network: This has k input neurons and one output neuron. Each input neuron is connected to the output neuron by an edge with weight w. The output neuron has bias b. The Identity network is a special case of this network, where $k = 1$.

The idea here is to treat this as a threshold problem, and set b and w so that being over or under the threshold gives value 1 or 0, respectively, in each case with probability at least $1 - \delta$. For a k-input And network, the output neuron should fire with probability at least $1 - \delta$ if all k input neurons fire, and with probability at most δ if at most $k - 1$ input neurons fire.

The settings for b and w generalize those for the Identity network. Namely, define $b = (2k - 1)L$ and $w = \frac{2b}{2k-1} = 2L$. When all k input neurons fire, the potential is $kw - b = L$, and (expanding L and plugging into the sigmoid function), the firing probability is just $1 - \delta$. When $k - 1$ input neurons fire, the potential is $(k - 1)w - b = -L$, and the firing probability is just δ. If fewer than $k - 1$ fire, the potential and the firing probability are smaller. Similar claims about multi-round

computations to what we argued for the Identity network also hold for the And network.

k-input Or network: This has the same structure as the k-input And network. The k-input Or network also generalizes the Identity network, which is the same as the 1-input Or network. Now the output neuron should fire with probability at least $1-\delta$ if one or more of the input neurons fire, and with probability at most δ if no input neurons fire. This time we set $b = L$ and $w = 2L$. When one input neuron fires, the potential is $w - b = L$ and the firing probability is $1-\delta$. If more than one fire, then the firing probability is even greater. When no input neurons fire, the potential is $-b = -L$, and the firing probability is δ. Again, similar claims about multi-round computations hold for the Or network.

Not network: This network has one input, one output, and one internal neuron, which acts as an inhibitor for the output neuron.[6] The network contains two edges, one from the input neuron to the internal neuron with weight w, and one from the internal neuron to the output neuron with weight w'. The internal neuron has bias b and the output neuron has bias b'.

The assembly consisting of the input and internal neurons acts like the identity gate, with settings of b and w as before: $b = L$ and $w = 2L$. So, for example, if we consider just the input firing state at time 0. the probability that the internal neuron's firing state at time 1 is the same is exactly $1-\delta$.

Let b', the bias of the output neuron, be $-L$, and let w', the weight of the outgoing edge of the inhibitor, be $-2L$. Then if the inhibitor fires at time 1, the output fires at time 2 with probability δ, and if the inhibitor does not fire at time 1, the output fires at time 2 with probability $1-\delta$. This yields probability $1-\delta$ of correct inhibition, which then yields probabiity at least $(1-\delta)^2$ that the output at time 2 gives the correct answer for the Not-network. Similar claims about multi-round computations also hold for the Not network, except that the Not network has a delay of 2 instead of 1.

2.4.2 Winner-Take-All circuits

This example is a simple Winner-Take-All network for n inputs and n corresponding outputs. It is based on a network presented in [LMP17a] and Chapter 5 of [Mus18]. Assume that some nonempty subset of the input neurons fire, in a stable manner. The output firing behavior is supposed to converge to a configuration in which exactly one of the outputs corresponding to the firing inputs fires. We would like this convergence to occur quickly, in some fairly short time t_c. And we would like the resulting configuration to remain stable for a fairly long time t_s. Figure 2 depicts the structure of the network.

So fix β_{in} to be an infinite input firing sequence, in which all the input configurations are the same, and at least one input neuron is firing. Let P be the resulting probabilistic execution. In [LMP17a, Mus18] we prove that, for certain values of t_c and t_s, the probability of convergence within time t_c to an output configuration that remains stable for time t_s is at least $1-\delta$.

The formal theorem statement is as follows. Here, γ is the weighting factor used in the biases and edge weights in the network, δ is a bound on the failure probability, and c_1 and c_2 are particular small constants.

Theorem 12 *Assume $\gamma \geq c_1 \log(\frac{nt_s}{\delta})$. Then starting from any configuration, with probability $\geq 1-\delta$, the network converges, within time $t_c \leq c_2 \log n \log(\frac{1}{\delta})$, to a single firing output corresponding*

[6] We generally classify neurons into two categories: *excitatory neurons*, all of whose outgoing edges have positive weights, and *inhibitory neurons*, whose outgoing edges have negative weights. However, this classification is not needed for the results in this paper.

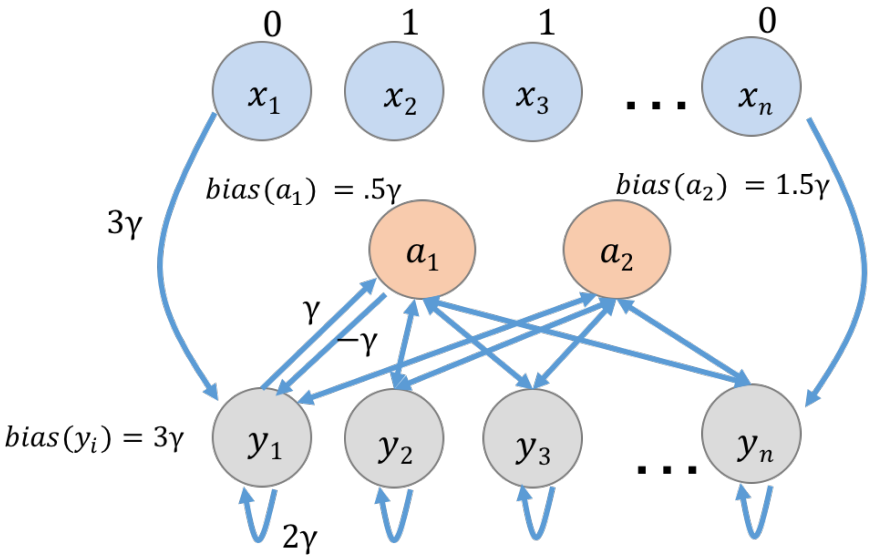


Figure 2: A basic Winner-Take-All network

to a firing input, and remains stable for time t_s. c_1 and c_2 are universla constants, independent of n,t_s, and δ.

The proof appears in [Mus18], based on work in [LMP17a]. The basic idea is that, when more than one output is firing, both inhibitors are triggered to fire. When they both fire, they cause each firing output to continue firing with probability $\frac{1}{2}$. This serves to reduce the number of firing outputs at a predictable rate. Once only a single output fires, only one inhibitor continues to fire; its effect is sufficient to prevent other non-firing outputs from beginning to fire, but not sufficient to stop the firing output from firing. All this, of course, is probabilistic.

Noting that the network is symmetric with respect to the n outputs. Therefore, we can refine the theorem above to assert that, for any particular output neuron y_i that corresponds to a firing input neuron x_i, the probability that y_i is the eventual firing output neuron is at least $\frac{1-\delta}{n}$.

3 Composition

In this section, we define composition of networks. We focus on composing two networks, but the ideas should extend easily to any finite number of networks.

3.1 Composition of two networks

Networks that are composed must satisfy some basic compatibility requirements. These are analogous to those used for I/O automata and similar models. Namely, two networks $\mathcal{N}^1$ and $\mathcal{N}^2$ are said to be *compatible* provided that:

1. No internal neuron of $\mathcal{N}^1$ is a neuron of $\mathcal{N}^2$.
2. No internal neuron of $\mathcal{N}^2$ is a neuron of $\mathcal{N}^1$.
3. No neuron is an output neuron of both $\mathcal{N}^1$ and $\mathcal{N}^2$.

On the other hand, we are allowed to have common input neurons, and also output neurons of one of the networks that are also input neurons of the other.[7] [8] Assuming $\mathcal{N}^1$ and $\mathcal{N}^2$ are compatible, we define their composition $\mathcal{N} = \mathcal{N}^1 \times \mathcal{N}^2$ as follows:

- N, the set of neurons of $\mathcal{N}$, is the union of N^1 and N^2, which are the sets of neurons of the two respective sub-networks. Note that common neurons are inserted only once. Each neuron inherits its bias from its original sub-network. This definition of bias is unambiguous: If a neuron belongs to both sub-networks, it must be an input of at least one of them, and input neurons do not have biases.

  Thus, when an input of one sub-network is combined with an output of the other sub-network, the resulting neuron acquires the bias from its output "precursor".

- E, the set of edges, is defined as follows. If e is an edge from neuron u to neuron v in either $\mathcal{N}^1$ or $\mathcal{N}^2$, then we include e also in $\mathcal{N}$. Each edge inherits its weight from its original sub-network. This definition of weight is unambiguous, since, as noted earlier, e cannot be an edge of both sub-networks.

  Thus, if the source neuron u is an input of both sub-networks, then it has edges in $\mathcal{N}$ to all the nodes to which its "precursors" have edges in the two sub-networks. If u is an output of $\mathcal{N}^1$ and an input of $\mathcal{N}^2$, then in $\mathcal{N}$, it has all the incoming and outgoing edges it has in $\mathcal{N}^1$ as well as the outgoing edges it has in $\mathcal{N}^2$.

  On the other hand, the target neuron v cannot be an input of both networks since it has an incoming edge in one of them. So v must be an output of one, say $\mathcal{N}^1$, and an input of the other, say $\mathcal{N}^2$. Then in $\mathcal{N}$, v has all the incoming and outgoing edges it had in $\mathcal{N}^1$ as well as the outgoing edges it has in $\mathcal{N}^2$.

- F_0, the initial non-input firing pattern of $\mathcal{N}$, gets inherited directly from the two sub-networks' initial non-input firing patterns. Since the two sub-networks have no non-input neurons in common, this is well-defined.

In the composed network, the neurons retain their classification as input/output/internal, except that a neuron that is an input of one sub-network and output of the other gets classified as an output neuron of $\mathcal{N}$.

The probabilistic executions and probabilistic traces of the new network $\mathcal{N}$ are defined as usual. In Sections 4 and 5, we show how to relate these to the probabilistic executions and probabilistic traces of $\mathcal{N}^1$ and $\mathcal{N}^2$.

Here are some basic lemmas analogous to those for general probabilistic executions and traces: For these lemmas, fix $\mathcal{N} = \mathcal{N}^1 \times \mathcal{N}^2$ and a particular input execution β_{in} of $\mathcal{N}$, which yields a particular probabilistic execution P.

Lemma 13 *Let α be a finite execution of $\mathcal{N}$ of length > 0 that is consistent with β_{in}. Suppose that α' is its one-step prefix. Let $\beta = trace(\alpha) = \alpha\lceil N_{ext}$ and $\beta' = trace(\alpha') = \alpha'\lceil N_{ext}$. Let $j \in \{1, 2\}$. Let $\alpha^j = \alpha\lceil N^j$, $\alpha'^j = \alpha'\lceil N^j$, $\beta^j = \beta\lceil N^j$, and $\beta'^j = \beta'\lceil N^j$. Then*

1. *$A(\alpha^j) \subseteq A(\alpha'^j)$, and $P(A(\alpha^j)|A(\alpha'^j)) = \frac{P(A(\alpha^j))}{P(A(\alpha'^j))}$.*

[7] In the brain setting, common input neurons for two different networks seem to make sense: the same neuron might have two separate sets of outgoing edges (synapses), leading to different neurons in the two different networks.

[8] We can prove from these requirements that $\mathcal{N}^1$ and $\mathcal{N}^2$ cannot have any edge in common. For if they had a common edge, then it would have to have the same source neuron and the same target neuron in both sub-networks. Since the target neuron is shared, it would have to be an input neuron of at least one of the networks. But then that network would then have an edge leading to one of its input neurons, which is forbidden by our network definition.

2. $A(\alpha^j) \subseteq A(\beta^j)$, *and* $P(A(\alpha^j)|A(\beta^j)) = \frac{P(A(\alpha^j))}{P(A(\beta^j))}$.

3. $A(\alpha^j) \subseteq A(\beta'^j)$, *and* $P(A(\alpha^j)|A(\beta'^j)) = \frac{P(A(\alpha^j))}{P(A(\beta'^j))}$.

4. $A(\alpha'^j) \subseteq A(\beta'^j)$, *and* $P(A(\alpha'^j)|A(\beta'^j)) = \frac{P(A(\alpha'^j))}{P(A(\beta'^j))}$.

5. $A(\beta^j) \subseteq A(\beta'^j)$, *and* $P(A(\beta^j)|A(\beta'^j)) = \frac{P(A(\beta^j))}{P(A(\beta'^j))}$.

Lemma 14 *Let α^j, α'^j, β^j, and β'^j be as in Lemma 13. Then*

$$P(A(\alpha^j)|A(\beta^j)) = P(A(\alpha^j)|A(\beta'^j)) \times \frac{P(A(\beta'^j))}{P(A(\beta^j))} = \frac{P(A(\alpha^j)|A(\beta'^j))}{P(A(\beta^j)|A(\beta'^j))}.$$

Lemma 15 *Let α be a finite execution of $\mathcal{N}$ of length > 0, that is consistent with β_{in}. Let α' be the one-step prefix of α and $\beta' = trace(\alpha')$. Let $j \in \{1, 2\}$. Then $P(A(\alpha\lceil N^j_{lc})|A(\beta'\lceil N^j)) = P(A(\alpha\lceil N^j_{lc})|A(\alpha'\lceil N^j)) \times P(A(\alpha'\lceil N^j)|A(\beta'\lceil N^j))$.*

Proof. We have that

$$P(A(\alpha\lceil N^j_{lc})|A(\beta'\lceil N^j)) = P((A(\alpha\lceil N^j_{lc}) \cap A(\beta'\lceil N^j))|A(\beta'\lceil N^j)),$$

by basic conditional probability, which is equal to

$$P((A(\alpha\lceil N^j_{lc}) \cap A(\alpha'\lceil N^j))|A(\beta'\lceil N^j)),$$

because $\alpha\lceil N^j_{lc}$ already fixes all the firing patterns for neurons in N^j_{lc}. This last expression is equal to

$$\frac{P((A(\alpha\lceil N^j_{lc}) \cap A(\alpha'\lceil N^j)))}{P(A(\beta'\lceil N^j))},$$

which is equal to

$$\frac{P((A(\alpha\lceil N^j_{lc}) \cap A(\alpha'\lceil N^j)))}{P(A(\alpha'\lceil N^j))} \times \frac{P(A(\alpha\lceil N^j))}{P(A(\beta'\lceil N^j))}.$$

This last expression is equal to $P(A(\alpha\lceil N^j_{lc})|A(\alpha'\lceil N^j)) \times P(A(\alpha'\lceil N^j)|A(\beta'\lceil N^j))$, as needed. □

3.2 A special case: acyclic composition

An important special case of composition is acyclic composition, in which outputs of $\mathcal{N}^2$ are not inputs to $\mathcal{N}^1$. That is, $\mathcal{N}^1$ may have inputs only from the outside world, and its outputs can go to $\mathcal{N}^1$, $\mathcal{N}^2$, and the outside world. $\mathcal{N}^2$ may have inputs from the outside world and from $\mathcal{N}^1$, and its outputs go just to $\mathcal{N}^2$ and the outside world. Formally, the definition of acyclic composition is the same as the general definition of composition, except for the additional restriction that $N^1_{in} \cap N^2_{out} = \emptyset$.

3.3 Examples

Here we give three examples. The first two represent acyclic composition, and the third is a toy example that involves cycles.

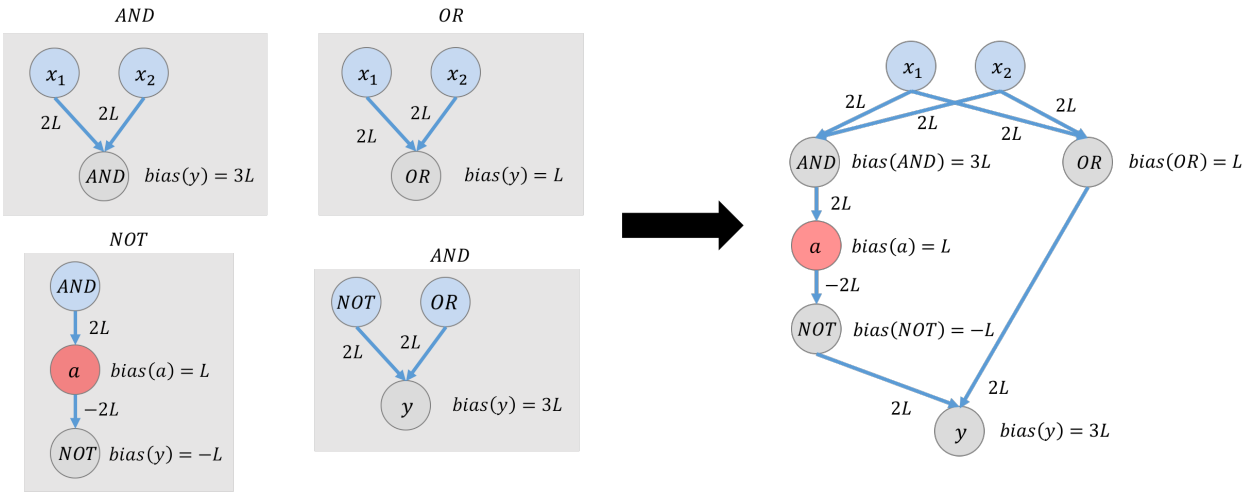


Figure 3: Composing four Boolean gate circuits into an Xor network

3.3.1 Boolean circuits

Figure 3 contains a circuit which is a composition of four Boolean gate circuits of the types described in Section 2.4.1: two And networks, one Or network, and a Not network. We compose these networks into a larger network that is intended to compute an Xor function.

In terms of binary composition operator, we can compose the four networks in three steps, as follows:

1. Compose one of the And networks and the Not network to get a network with 2 inputs, 2 outputs, and 1 internal neuron, by identifying the output neuron of the And network with the input neuron of the Not network. Note that the composed network has two outputs because the And gate remains an output—the composition operator does not reclassify it as an internal neuron. The composed network is intended to compute the Nand of the two inputs (as well as the And).

2. Compose the network produced in Step 1 with the Or network to get a 2-input 3-output, 1-internal network, by identifying the the corresponding inputs in the two networks. The resulting network has outputs corresponding to the Nand and the Or of the two inputs (as well as the And).

3. Finally, compose the Nand network and the Or network with the second And network, by identifying the Nand output neuron and the Or output neuron with the two input neurons for the And network. The resulting network has an output corresponding to the Xor of the two original inputs (as well as outputs for the first And, the Nand, and the Or).

To state a simple guarantee for this composed circuit, let's assume that the inputs fire consistently, in an unchanged firing pattern. Then, working from the previously-shown guarantees of the individual networks, we can say that the probability that the final output neuron produces its required Xor value at time 4 is at least $(1 - \delta)^5$. We will say more about this later, in Section 4.2.

3.3.2 Attention using WTA

Figure 4 depicts the composition of our WTA network from Section 2.4.2 with a $2n$-input n output Filter network. The Filter network is, in turn, a composition of n disjoint And gates. The composition is acyclic since information can flow from WTA to Filter but not vice versa.

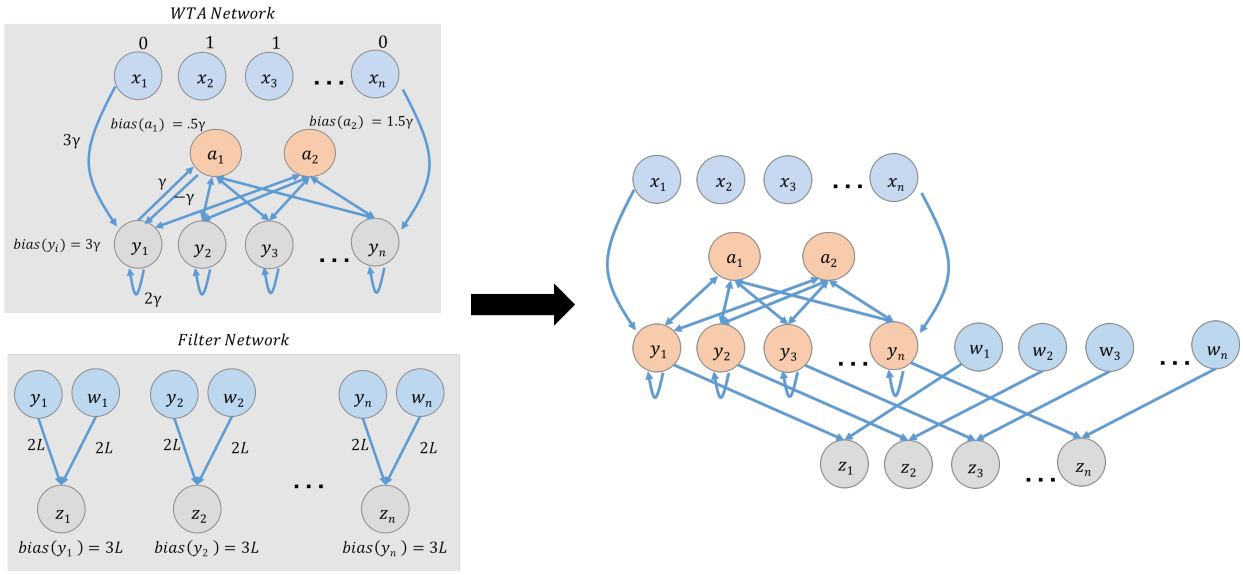


Figure 4: An Attention network built from a WTA network and a Filter network

The Filter network is designed to fire any of its outputs z_i right after the corresponding w_i input fires, provided that its y_i input (which is an output of the WTA) also fires. In this way, the WTA network is used to select particular outputs for the Filter network to fire—those that are "reinforced" by the inputs from the WTA.

When the WTA and Filter networks are composed, and the WTA inputs fire stably, with at least one input firing, the WTA network should soon stabilize as we described in Section 2.4.2, to a configuration with a single firing output y_i, which is equally likely to be any of the n outputs. That configuration should persist for a fairly long time. The detailed bounds are given in Theorem 12. After the WTA stabilizes, it reinforces only a particular input w_i for the Filter. From that point on, the Filter's z_i outputs should mirror its w_i inputs, and no other z outputs should fire. The probability of such mirroring should be at least $(1-\delta')^{nt_s}$, if δ' denotes the failure probability for an And gate. (Recall the definition of t_s from Example 2.4.2.) In this way, the composition can be viewed as an "Attention" circuit, which pays attention to just a single input stream.

Note that the composed network behaves on two different time scales: the WTA takes some time to converge, but after that, the responses to the selected intput stream will be essentially immediate.

3.3.3 Cyclic composition

In this section we give a toy example, consisting of two networks that affect each other's behavior in a simple way. Throughout this section, we use the abbreviation L for the quantity $\lambda \ln(\frac{1-\delta}{\delta})$, just as we did in Section 2.4.1.

Figure 5 shows a network $\mathcal{N}^1$ with one input neuron x_1, one output neuron x_2, and one internal neuron a_1. It has edges from x_1 to a_1, from a_1 to x_2, and from x_2 to itself (a self-loop). The biases of a_1 and x_2 are L and the weights on all edges are $2L$.

Network $\mathcal{N}^1$ behaves so that, at any time $t \geq 1$, the firing probability for the internal neuron a_1 is exactly $1-\delta$ if a_1 fires at time $t-1$, and is exactly δ if a_1 does not fire at time $t-1$. This is as for the Identity network in Section 2.4.1. The firing probability of the output neuron x_2 is:

- δ, if neither a_1 nor x_2 fires at time $t-1$.

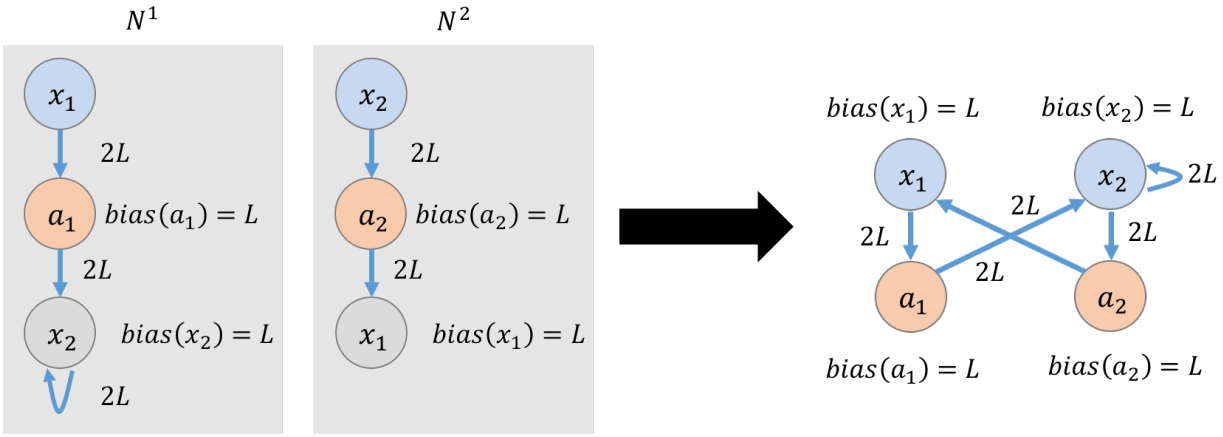


Figure 5: A cyclic composition

- $1 - \delta$, if exactly one of a_1 and x_2 fires at time $t - 1$.
- $1 - \frac{\delta^3}{(1-\delta)^3+\delta^3}$ if both a_1 and x_2 fire at time $t - 1$.

Thus, if input x_1 fires, output x_2 will be likely to fire 2 times later (with probability at least $(1 - \delta)^2$). Without any additional input firing, the firing of x_2 is sustained only by the self-loop, which means that the firing probability decreases steadily over time, by a factor of $(1 - \delta)$ at each time. Eventually, the firing should "die out".

Network $\mathcal{N}^2$ is similar, replacing x_1, a_1, and x_2 by x_2, a_2, and x_1, respectively. However, we omit the self-loop edge on x_1. The biases are L and the weights on the two edges are $2L$.

Network $\mathcal{N}^2$ behaves so that, at any time $t \geq 1$, the firing probability for the internal neuron a_2 is exactly $1 - \delta$ if x_2 fires at time $t - 1$, and is exactly δ if x_2 does not fire at time $t - 1$. Likewise, the firing probability for the output neuron x_1 is exactly $1 - \delta$ if a_2 fires at time $t - 1$ and δ if a_2 does not fire. Thus, if input x_2 fires, then output x_1 will be likely to fire 2 times later (with probability at least $(1 - \delta)^2$). However, the firing of x_1 is not sustained.

Now consider the composition of $\mathcal{N}^1$ and $\mathcal{N}^2$, identifying the output x_2 of $\mathcal{N}^1$ with the input x_2 of $\mathcal{N}^2$, and the output x_1 of $\mathcal{N}^2$ with the input x_1 of $\mathcal{N}^1$. The behavior of the composition depends on the starting firing pattern. Let us suppose that both a_1 and a_2 do not fire initially; we consider the behavior for the various starting firing patterns for x_1 and x_2. We assume that δ is "sufficiently small".

First, if neither x_1 nor x_2 fires at time 0, then with "high probability", none of the four neurons will fire for a long time. If one or both of x_1 and x_2 fire at time 0, then with "high probability", they will trigger all the neurons to fire and continue to fire for a long time. We give some details in Section 5.4.1.

3.4 Compositionality definitions

We have defined a specific external behavior notion *Beh* for our networks. We have also allowed the possibility of other external behavior notions. Here we define compositionality for general behavior notions. Later in the paper, in Section 5.3, we will show that our particular behavior notion *Beh* is compositional.

In general, we define an external behavior notion B to be *compositional* provided that the following holds: Consider any four networks $\mathcal{N}^1$, $\mathcal{N}^2$, $\mathcal{N}'^1$, and $\mathcal{N}'^2$, where $\mathcal{N}^1$ and $\mathcal{N}'^1$ have the same sets of input and output neurons, $\mathcal{N}^2$ and $\mathcal{N}'^2$ have the same sets of input and output neurons,

$\mathcal{N}^1$ and $\mathcal{N}^2$ are compatible, and $\mathcal{N}'^1$ and $\mathcal{N}'^2$ are compatible. Suppose that $B(\mathcal{N}^1) = B(\mathcal{N}'^1)$ and $B(\mathcal{N}^2) = B(\mathcal{N}'^2)$. Then $B(\mathcal{N}^1 \times \mathcal{N}^2) = B(\mathcal{N}'^1 \times \mathcal{N}'^2)$.

We show that, in general, if two external behavior definitions are equivalent and one is compositional, then so is the other. This will provide us with a method that will be helpful in Section 5.3 for showing compositionality.

Theorem 16 *If B and B' are two equivalent external behavior notions for stochastic SNNs and B is compositional, then also B' is compositional.*

Proof. Suppose that B and B' are two external behavior notions and B is compositional. We show that B' is compositional. For this, consider any four networks $\mathcal{N}^1$, $\mathcal{N}^2$, $\mathcal{N}'^1$, and $\mathcal{N}'^2$, where $\mathcal{N}^1$ and $\mathcal{N}'^1$ have the same sets of input and output neurons, $\mathcal{N}^2$ and $\mathcal{N}'^2$ have the same sets of input and output neurons, $\mathcal{N}^1$ and $\mathcal{N}^2$ are compatible, and $\mathcal{N}'^1$ and $\mathcal{N}'^2$ are compatible. Suppose that $B'(\mathcal{N}^1) = B'(\mathcal{N}'^1)$ and $B'(\mathcal{N}^1) = B'(\mathcal{N}'^1)$. We must show that $B'(\mathcal{N}^1 \times \mathcal{N}^2) = B'(\mathcal{N}'^1 \times \mathcal{N}'^2)$.

Since B and B' are equivalent and $B'(\mathcal{N}^1) = B'(\mathcal{N}'^1)$, we have that $B(\mathcal{N}^1) = B(\mathcal{N}'^1)$. Likewise, since $B'(\mathcal{N}^2) = B'(\mathcal{N}'^2)$, we have that $B(\mathcal{N}^2) = B(\mathcal{N}'^2)$. Since B is assumed to be compositional, this implies that $B(\mathcal{N}^1 \times \mathcal{N}^2) = B(\mathcal{N}'^1 \times \mathcal{N}^2)$. Then since B and B' are equivalent, we get that $B'(\mathcal{N}^1 \times \mathcal{N}^2) = B'(\mathcal{N}'^1 \times \mathcal{N}^2)$, as needed. □

Lemma 17 *An external behavior notion B is compositional if and only if, for all compatible pairs of networks $\mathcal{N}^1$ and $\mathcal{N}^2$, $B(\mathcal{N}^1 \times \mathcal{N}^2)$ is determined by $B(\mathcal{N}^1)$ and $B(\mathcal{N}^2)$.*

Proof. Straightforward. □

4 Theorems for Acyclic Composition

Our general composition results appear in Section 5. Those are a bit complicated, mainly because of the possibility of connections in both directions between the sub-networks. Acyclic composition is a very important special case of general composition, in fact, most interesting examples seem to satisfy the acyclic restriction. Since the results for this case are much simpler, we present those first.

For this section, fix the notation $\mathcal{N} = \mathcal{N}^1 \times \mathcal{N}^2$, and assume that we have no edges from $\mathcal{N}^2$ to $\mathcal{N}^1$, that is, that $N^1_{in} \cap N^2_{out} = \emptyset$.

In this section, and from now on in the paper, we will mostly avoid writing the cone notation $A()$. Thus, instead of $P(A(\beta))$, we will write just $P(\beta)$. We hope this does not cause much confusion.

4.1 Compositionality

We have not formally defined "compositionality" for the special case of acyclic composition. So instead of proving "compositionality" here, we will simply show how to express $Beh(\mathcal{N})$ in terms of $Beh(\mathcal{N}^1)$ and $Beh(\mathcal{N}^2)$.[9]

Specifically, we fix any particular input execution β_{in} of $\mathcal{N}$, which generates a particular probabilistic execution P of $\mathcal{N}$. Then we consider an arbitrary finite trace β of $\mathcal{N}$ that is consistent with β_{in}. We show how to express $P(\beta)$ in terms of probability distributions P^1 and P^2 that are generated by $\mathcal{N}^1$ and $\mathcal{N}^2$, respectively, from certain input executions.

We begin by deriving a simple expression for $P(\beta)$, for an arbitrary finite trace β of $\mathcal{N}$ that is consistent with β_{in}.

[9] We could, presumably, define compositionality for this special case as before but based on a modified definition of compatibility—one that includes the extra acyclic condition. Then we could give a characterization similar to Lemma 17 and use our result to show this version of compositionality. We leave this for later.

Lemma 18 *Let β be a finite trace of $\mathcal{N}$ that is consistent with β_{in}. Then*

$$P(\beta) = P(\beta \lceil N^1_{out}) \times P((\beta \lceil N^2_{out}) | (\beta \lceil N^2_{in})).$$

Proof. Since $\beta \lceil N_{in}$ is fixed, we have that

$$P(\beta) = P(\beta \lceil N_{out}) = P((\beta \lceil (N^1_{out} \cup N^2_{out})).$$

This last expression is equal to

$$P(\beta \lceil N^1_{out}) \times P((\beta \lceil N^2_{out}) | (\beta \lceil N^1_{out}))$$

by basic conditional probability reasoning.

We have that

$$P((\beta \lceil N^2_{out}) | (\beta \lceil N^1_{out})) = P((\beta \lceil N^2_{out}) | (\beta \lceil (N^1_{out} \cap N^2_{in}))),$$

because the behavior of $\mathcal{N}^2$ does not depend on neurons in $N^1_{out} - N^2_{in}$. And this is equal to

$$P((\beta \lceil N^2_{out}) | (\beta \lceil (N^2_{in}))),$$

because N^2_{in} consists of $N^1_{out} \cap N^2_{in}$ plus some neurons in N_{in}, which are fixed in β_{in}. Substituting yields

$$P(\beta) = P(\beta \lceil N^1_{out}) \times P((\beta \lceil N^2_{out}) | (\beta \lceil N^2_{in})),$$

as needed. □

Thus, Lemma 18 assumes an arbitrary input execution β_{in} of $\mathcal{N}$, which generates a probability distribution P. The Lemma expresses $P(\beta)$, for an arbitrary β, in terms of the P-probabilities of other finite traces. However, what we really need to do is to express $P(\beta)$ in terms of probability distributions P^1 and P^2 that are generated by $\mathcal{N}^1$ and $\mathcal{N}^2$, respectively, from certain infinite input executions for those respective sub-networks. We define these input executions and distributions as follows.

- Input execution β^1_{in} and distribution P^1 for $\mathcal{N}^1$:

  Define the infinite input execution β^1_{in} of $\mathcal{N}^1$ to be $\beta_{in} \lceil N^1_{in}$, that is, the projection of the given input execution on the inputs of $\mathcal{N}^1$.

  Then define P^1 to be the probability distribution that is generated by $\mathcal{N}^1$ from input execution β^1_{in}.

- Input execution β^2_{in} and distribution P^2 for $\mathcal{N}^2$:

  This is a bit more complicated, since the inputs to $\mathcal{N}^2$ depend not only on the external input β_{in}, but on the outputs produced by $\mathcal{N}^1$.

  Define the infinite input execution β^2_{in} of $\mathcal{N}^2$ as follows. First, note that $N^2_{in} \subseteq N_{in} \cup N^1_{out}$, that is, every input of $\mathcal{N}^2$ is either an input of $\mathcal{N}$ or an output of $\mathcal{N}^1$. Define the firing patterns of the neurons in $N^2_{in} \cap N_{in}$ using β_{in}, that is, define $\beta^2_{in} \lceil (N^2_{in} \cap N_{in}) = \beta_{in} \lceil N^2_{in}$. And for the firing patterns of the neurons in $N^2_{in} \cap N^1_{out}$, use β, that is, define $\beta^2_{in} \lceil (N^2_{in} \cap N^1_{out}) = \beta \lceil (N^2_{in} \cap N^1_{out})$ for times $0, \ldots, length(\beta)$ and the default 0 for all later times.

  Then define P^2 to be the probability distribution that is generated by $\mathcal{N}^2$ from input execution β^2_{in}.

Note that, in the second case above, the choice of the input execution β_{in}^2 depends on the particular trace β for which we are trying to express the P-probability. This is allowed because our external behavior notion Beh for any network includes a probability distribution for every infinite input execution of the network.

The next lemma restates the result of Lemma 18 in terms of the new probability distributions P^1 and P^2.

Lemma 19 *Let β be a finite trace of $\mathcal{N}$ that is consistent with β_{in}. Then*

$$P(\beta) = P^1(\beta\lceil N_{out}^1) \times P^2(\beta\lceil N_{out}^2).$$

Proof. Fix β, a finite trace of of $\mathcal{N}$ that is consistent with β_{in}. By Lemma 18, we know that:

$$P(\beta) = P(\beta\lceil N_{out}^1) \times P((\beta\lceil N_{out}^2)|(\beta\lceil N_{in}^2)).$$

It suffices to show that these two terms are equal to the corresponding terms in this lemma, that is, that

$$P(\beta\lceil N_{out}^1) = P^1(\beta\lceil N_{out}^1)$$

and

$$P((\beta\lceil N_{out}^2)|(\beta\lceil N_{in}^2)) = P^2(\beta\lceil N_{out}^2).$$

These two statements follow directly by unwinding the definitions of P^1 and P^2, respectively. □

The next lemma has a slightly simpler statement than Lemma 19.

Lemma 20 *Let β be a finite trace of $\mathcal{N}$ that is consistent with β_{in}. Then*

$$P(\beta) = P^1(\beta\lceil N^1) \times P^2(\beta\lceil N^2).$$

Proof. This follows from Lemma 19 because in each term on the right-hand-side of the equation in this lemma, the probability depends on the output traces only—the input traces are fixed. Formally, this uses Lemma 10. □

Finally, Lemma 20 yields a kind of compositionality theorem for acyclic composition:

Theorem 21 *$Beh(\mathcal{N})$ is determined by $Beh(\mathcal{N}^1)$ and $Beh(\mathcal{N}^2)$.*

We prove a more general compositionality result in Section 5.

4.2 Examples

4.2.1 Boolean circuits

Let $\mathcal{N}$ be the seven-gate Boolean circuit from Section 3.3.1. Express $\mathcal{N}$ as the composition $\mathcal{N}^1 \times \mathcal{N}^2$, where $\mathcal{N}^1$ denotes the (Nand,Or) network and $\mathcal{N}^2$ denotes the second And network. That is, we are considering the final composition in the order of compositions described in Section 3.3.1.

Fix β_{in} to be any infinite input execution with stable inputs, and let P be the probabilistic execution of $\mathcal{N}$ generated from β_{in}. In P, we should expect to have stable, correct outputs for a long while starting from time 4, because the depth of the entire network is 4. Here we consider just the situation at precisely time 4, that is, we consider the probabilities $P(\beta)$ for finite traces β of length exactly 4. Specifically, we would like to use Lemma 19 to help us show that the probability of a correct Xor output at time 4 is at least $(1-\delta)^5$.

We work compositionally. In particular, we assume that, in the probabilistic execution of $\mathcal{N}^1$ on β_{in}, or any other stable input sequence, the probability of correct (Nand,Or) outputs at time 3 is at least $(1-\delta)^4$. We also assume that, in the probabilistic execution of $\mathcal{N}^2$ on any input sequence, the probability that the output at time 4 is the And of its two inputs at time 3 is at least $1-\delta$. These assumptions could be proved for these two networks, but we simply assume them here and use them to get our result about the composed network $\mathcal{N}$.

So define event B to be the set of traces β of $\mathcal{N}$ of length 4 such that β gives a correct Xor output at time 4, as well as correct (Nand, Or) outputs at time 3. We will argue that $P(B) \geq (1-\delta)^5$, which implies our desired result.

We have that $P(B) = \sum_{\beta \in B} P(\beta)$. By Lemma 19, this is equal to

$$\sum_{\beta \in B} P^1(\beta \lceil N^1_{out}) \times P^2(\beta \lceil N^2_{out}).$$

Here, P^1 and P^2 are defined as in Section 4.1, based on $\beta^1_{in} = \beta_{in}$, and for each particular β, based on β^2_{in} equal to $\beta \lceil N^2_{in}$, extended to an infinite sequence by adding 0's. Note that the choice of input sequence β^2_{in} for $\mathcal{N}^2$ is uniquely determined by $\beta \lceil N^1_{out}$.

We break this expression up into:

$$\sum_{\beta^1} (\sum_{\beta^2} P^1(\beta^1 \lceil N^1_{out}) \times P^2(\beta^2 \lceil N^2_{out})).$$

Here, β^1 ranges over traces of $\mathcal{N}^1$ that are consistent with β_{in} and yield correct (Nand, Or) outputs at time 3. And for each particular β^1, β^2 ranges over traces of $\mathcal{N}^2$ that are consistent with the input sequence β^2_{in} determined from $\beta^1 \lceil N^1_{out} = \beta \lceil N^1_{out}$, and whose output at time 4 is the Xor of its inputs at time 3. This is equal to (collecting terms for each β^1):

$$\sum_{\beta^1} P^1(\beta^1 \lceil N^1_{out}) \sum_{\beta^2} P^2(\beta^2 \lceil N^2_{out}).$$

Now, for any particular β^1, we know that:

$$\sum_{\beta^2} P^2(\beta^2 \lceil N^2_{out}) \geq (1-\delta),$$

by our assumptions about the behavior of $\mathcal{N}^2$. So the overall expression is at least

$$\sum_{\beta^1} P^1(\beta^1 \lceil N^1_{out})(1-\delta) = (1-\delta) \sum_{\beta^1} P^1(\beta^1 \lceil N^1_{out}).$$

We also know that

$$\sum_{\beta^1} P^1(\beta^1 \lceil N^1_{out}) \geq (1-\delta)^4,$$

by our assumption about the behavior of $\mathcal{N}^1$. So the overall expression is at least

$$(1-\delta)(1-\delta)^4 = (1-\delta)^5,$$

as needed.

4.2.2 Attention using WTA

We consider the composition of the WTA network and the Filter network as described in Section 3.3.2. Now call the composition $\mathcal{N}$, the WTA network $\mathcal{N}^1$, and the Filter network $\mathcal{N}^2$. We assume that the WTA network satisfies Theorem 12, with particular values of δ, t_c, t_s, γ, c_1 and c_2. We assume that each And network within Filter is correct at each time with probability at least $1-\delta'$.

Fix β_{in} to be any infinite input execution of $\mathcal{N}$ with stable x_i inputs, such that at least one x_i is firing. The w_i inputs are unconstrained. Let P be the probabilistic execution of $\mathcal{N}$ generated from β_{in}. We want to prove that, according to P, with probability at least $(1-\delta)(1-\delta')^{nt_s}$, there is some $t \leq t_c$ such that: (a) the y outputs stabilize by time t to one steadily-firing output y_i, which persists through time $t+t_s-1$, and (b) for this particular i, starting from time $t+1$ and continuing for a total of t_s times, the z_i outputs correctly mirror the w_i inputs at the previous time, and all the other z neurons do not fire.

We work compositionally. We assume that, in the probabilistic execution of the WTA network $\mathcal{N}^1$ on $\beta_{in}\lceil N_{in}$, the probability of correct, stable outputs as in Theorem 12 is at least $1-\delta$. We also assume that, in the probabilistic execution of $\mathcal{N}^2$ on any input sequence, conditioned on any finite execution prefix, the probability of correct mirroring of inputs for the next t times is at least $(1-\delta')^{nt_s}$. These assumptions could be proved for the two networks, but we simply assume them here.

Now define B to be the set of traces β of $\mathcal{N}$ of length t_c+t_s-1 such that all the desired conditions hold in β, that is, there is some $t \leq t_c$ such that in β, (a) the y outputs stabilize by time t to one steadily-firing output y_i, which persists through time $t+t_s-1$, and (b) for this particular i, starting from time $t+1$ and continuing for a total of t_s times, the z_i outputs correctly mirror the w_i inputs at the previous time, and all the other z neurons do not fire. We will argue that $P(B) \geq (1-\delta)(1-\delta')^{nt_s}$. We follow the same pattern as in the Boolean circuit network example in Section 4.2.

We have that $P(B) = \sum_{\beta \in B} P(\beta)$. By Lemma 19, this is equal to

$$\sum_{\beta \in B} P^1(\beta\lceil N^1_{out}) \times P^2(\beta\lceil N^2_{out}).$$

Here, P^1 and P^2 are defined as in Section 4.1, based on $\beta^1_{in} = \beta_{in}\lceil N^1_{in}$ and for each particular β, based on β^2_{in} equal to $\beta\lceil N^2_{in}$ and extended to an infinite sequence by adding 0's. Note that β^2_{in} is uniquely determined by $\beta\lceil(N_{in} \cup N^1_{out})$.

This expression is greater than or equal to:

$$\sum_{\beta^1}(\sum_{\beta^2} P^1(\beta^1\lceil N^1_{out}) \times P^2(\beta^2\lceil N^2_{out})).$$

Here, β^1 ranges over traces of $\mathcal{N}^1$ that are consistent with β_{in} and for which there is some $t \leq t_c$ such that in β^1, the y outputs stabilize by time t to one steadily-firing output y_i, which persists through time $t+t_s-1$. And for each particular β^1, β^2 ranges over traces of $\mathcal{N}^2$ that are consistent with the input sequence β^2_{in} determined from β_{in} and $\beta^1\lceil N^1_{out} = \beta\lceil N^1_{out}$, and that satisfy the following correctness condition for $\mathcal{N}^2$: for the first t and associated i that witness the correctness condition for β^1, at times $t+1, \ldots, t_{t_s-1}$, the z_i outputs correctly mirror the w_i inputs at the previous time, and all the other z neurons do not fire.[10]

[10] The technical reason why this is an inequality rather than an equation is that it is possible for β^2 to satisfy the correctness condition for $\mathcal{N}^2$ starting from some other time than the initial t satisfying the condition for $\mathcal{N}^1$.

This is equal to (collecting terms for each β^1):

$$\sum_{\beta^1} P^1(\beta^1 \lceil N^1_{out}) \sum_{\beta^2} P^2(\beta^2 \lceil N^2_{out}).$$

Now, for any particular β^1, we know that:

$$\sum_{\beta^2} P^2(\beta^2 \lceil N^2_{out}) \geq (1-\delta')^{nt_s},$$

by our assumptions about the behavior of $\mathcal{N}^2$. So the overall expression is at least

$$\sum_{\beta^1} P^1(\beta^1 \lceil N^1_{out})(1-\delta')^{nt_s} = (1-\delta')^{nt_s} \sum_{\beta^1} P^1(\beta^1 \lceil N^1_{out}).$$

We also know that

$$\sum_{\beta^1} P^1(\beta^1 \lceil N^1_{out}) \geq (1-\delta),$$

by our assumption about the behavior of $\mathcal{N}^1$. So the overall expression is at least

$$(1-\delta)(1-\delta')^{nt_s},$$

as needed.

5 Theorems for General Composition

For general composition, a simple approach like the one in Section 4 does not work. Apparent circularities in dependencies get in the way. In the general case, we can break these circularities using time.

For this entire section, fix $\mathcal{N} = \mathcal{N}^1 \times \mathcal{N}^2$. We will continue to mostly avoid writing the cone notation $A()$.

5.1 Composition results for executions and traces

For this subsection and the following, fix a particular input execution β_{in} of $\mathcal{N}$, which yields a particular probabilistic execution P. The main result of this subsection is Lemma 23. It says that the probability of a certain execution α of the entire network $\mathcal{N}$, conditioned on its trace β, is simply the product of the probabilities of the two projections of α on the two sub-networks, each conditioned on its projected trace. In other words, once we fix all the external behavior of the network, including the part of the behavior involved in interaction between the two sub-networks, the internal states of the neurons within the two sub-networks are determined independently. We begin with a straightforward lemma that treats the two sub-networks asymmetrically.

Lemma 22 *Let α be a finite execution of $\mathcal{N}$ with trace β, such that α is consistent with β_{in}. Then*

$$P(\alpha|\beta) = P((\alpha \lceil N^1_{int})|\beta) \times P((\alpha \lceil N^2_{int})|(\alpha \lceil N^1_{int}), \beta).$$

Proof. Standard conditional probability. □

And now we remove the asymmetry:

Lemma 23 *Let α be a finite execution of $\mathcal{N}$ with trace β, such that α is consistent with β_{in}. Then*

$$P(\alpha|\beta) = P((\alpha\lceil N^1)|(\beta\lceil N^1)) \times P((\alpha\lceil N^2)|(\beta\lceil N^2)).$$

Proof. Lemma 22 says that

$$P(\alpha|\beta) = P((\alpha\lceil N^1_{int})|\beta) \times P((\alpha\lceil N^2_{int})|(\alpha\lceil N^1_{int}), \beta).$$

It suffices to show:

1. $P((\alpha\lceil N^1_{int})|\beta) = P((\alpha\lceil N^1)|(\beta\lceil N^1))$.

   For this, note that

   $$P((\alpha\lceil N^1_{int})|\beta) = P((\alpha\lceil N^1|\beta),$$

   because β already includes the firing patterns for all the neurons in $N^1 - N^1_{int} = N^1_{ext}$. And

   $$P((\alpha\lceil N^1|\beta) = P((\alpha\lceil N^1)|(\beta\lceil N^1)),$$

   because of locality—the neurons in N^1 are the only ones that $\alpha\lceil N^1$ depends on. Putting these two facts together yields the needed equation.

2. $P((\alpha\lceil N^2_{int})|(\alpha\lceil N^1_{int}), \beta) = P((\alpha\lceil N^2)|(\beta\lceil N^2))$.

   For this, note that

   $$P((\alpha\lceil N^2_{int})|(\alpha\lceil N^1_{int}), \beta) = P((\alpha\lceil N^2)|(\alpha\lceil N^1_{int}), \beta),$$

   because β already includes the firing patterns for all the neurons in $N^2 - N^2_{int} = N^2_{ext}$. And

   $$P((\alpha\lceil N^2)|(\alpha\lceil N^1_{int}), \beta) = P((\alpha\lceil N^2)|\beta),$$

   because $\alpha\lceil N^2$ does not depend on $(\alpha\lceil N^1_{int})$, except indirectly through β. Finally,

   $$P((\alpha\lceil N^2)|\beta) = P((\alpha\lceil N^2)|(\beta\lceil N^2)),$$

   because of locality—the neurons in N^2 are the only ones that $\alpha\lceil N^2$ depends on. Putting these three facts together yields the needed equation.

□

5.2 Composition results for one-step extensions

In this subsection, we describe how to break circularities in dependencies using time, as a major step toward our general compositionality result. Specifically, we consider one-step extensions of executions and traces of $\mathcal{N}$, and show how we can express them in terms of one-step extensions of executions and traces of $\mathcal{N}^1$ and $\mathcal{N}^2$.

Our first lemma is about extending an execution, either to a particular longer execution, or just to any execution with a particular longer trace.

Lemma 24 *1. Let α be a finite execution of $\mathcal{N}$ of length > 0 that is consistent with β_{in}. Let α' be the one-step prefix of α. Then:*

$$P(\alpha|\alpha') = P((\alpha\lceil N^1_{lc})|(\alpha'\lceil N^1)) \times P((\alpha\lceil N^2_{lc})|(\alpha'\lceil N^2)).$$

2. *Let β be a finite trace of $\mathcal{N}$ of length > 0 that is consistent with β_{in}. Let α' be a finite execution of $\mathcal{N}$ such that trace(α') is the one-step prefix of β. Then:*

$$P(\beta|\alpha') = P((\beta\lceil N^1_{out})|(\alpha'\lceil N^1)) \times P((\beta\lceil N^2_{out})|(\alpha'\lceil N^2)).$$

Proof. 1. The non-input neurons of $\mathcal{N}$ are those in $N_{lc} = N^1_{lc} \cup N^2_{lc}$. The firing states of all of these neurons in the final configuration of α are determined independently. Thus, we have

$$P(\alpha|\alpha') = P((\alpha\lceil N^1_{lc})|\alpha') \times P((\alpha\lceil N^2_{lc})|\alpha').$$

Furthermore, the final firing states for the neurons in N^1_{lc} depend only on the immediately previous states of the neurons in N^1, and similarly for N^2_{lc} and N^2, so this last expression is equal to

$$P((\alpha\lceil N^1_{lc})|(\alpha'\lceil N^1)) \times P((\alpha\lceil N^2_{lc})|(\alpha'\lceil N^2)),$$

as needed.

2. The output neurons of $\mathcal{N}$ are those in $N_{out} = N^1_{out} \cup N^2_{out}$. The firing states of all of these neurons in the final configuration of β are determined independently. Thus, we have

$$P(\beta|\alpha') = P((\beta\lceil N^1_{out})|\alpha') \times P((\beta\lceil N^2_{out})|\alpha').$$

Furthermore, the final firing states for the neurons in N^1_{out} depend only on the immediately previous states of the neurons in N^1, and similarly for N^2_{out} and N^2, so this last expression is equal to

$$P((\beta\lceil N^1_{out})|(\alpha'\lceil N^1)) \times P((\beta\lceil N^2_{out})|(\alpha'\lceil N^2)),$$

as needed.

□

The second lemma is about extending a trace, either to an execution or to a longer trace. This is a bit more difficult because we are conditioning only on traces, which do not include the internal behavior of the two sub-networks.

Lemma 25 *1. Let α be a finite execution of $\mathcal{N}$ of length > 0 that is consistent with β_{in}. Let β' be the one-step prefix of trace(α). Then:*

$$P(\alpha|\beta') = P((\alpha\lceil N^1_{lc})|(\beta'\lceil N^1)) \times P((\alpha\lceil N^2_{lc})|(\beta'\lceil N^2)).$$

2. Let β be a finite trace of $\mathcal{N}$ of length > 0 that is consistent with β_{in}. Let β' be the one-step prefix of trace(α). Then:

$$P(\beta|\beta') = P((\beta\lceil N^1_{out})|(\beta'\lceil N^1)) \times P((\beta\lceil N^2_{out})|(\beta'\lceil N^2)).$$

Proof. 1. Fix α and β' as described. Let α' be the one-step prefix of α. By Lemma 9, we have:

$$P(\alpha|\beta') = P(\alpha|\alpha') \times P(\alpha'|\beta').$$

Lemma 24 implies that

$$P(\alpha|\alpha') = P((\alpha\lceil N^1_{lc})|(\alpha'\lceil N^1)) \times P((\alpha\lceil N^2_{lc})|(\alpha'\lceil N^2)).$$

Lemma 23 implies that

$$P(\alpha'|\beta') = P((\alpha'\lceil N^1)|(\beta'\lceil N^1)) \times P((\alpha'\lceil N^2)|(\beta'\lceil N^2)).$$

Substituting, we get that:

$P(\alpha|\beta') = P((\alpha\lceil N^1_{lc})|(\alpha'\lceil N^1)) \times P((\alpha\lceil N^2_{lc})|(\alpha'\lceil N^2)) \times P((\alpha'\lceil N^1)|(\beta'\lceil N^1)) \times P((\alpha'\lceil N^2)|(\beta'\lceil N^2))$.

Rearranging terms and using Lemma 15, we see that the right-hand side is equal to

$$P((\alpha\lceil N^1_{lc})|(\beta'\lceil N^1)) \times P((\alpha\lceil N^2_{lc})|(\beta'\lceil N^2)),$$

as needed.

2. Fix β and β' as described. Let B denote the set of executions α of $\mathcal{N}$ such that $trace(\alpha) = \beta$, i.e., such that $\alpha\lceil N_{ext} = \beta$. Note that what varies among the different executions in B is just the firing patterns of the neurons in $N_{int} = N^1_{int} \cup N^2_{int}$. Then $P(\beta|\beta')$ can be expanded as

$$\sum_{\alpha \in B} P(\alpha|\beta').$$

By Part 1, this is equal to

$$\sum_{\alpha \in B} (P((\alpha\lceil N^1_{lc})|(\beta'\lceil N^1)) \times P((\alpha\lceil N^2_{lc})|(\beta'\lceil N^2)).$$

Now define B^1 to be the set of executions α^1 of $\mathcal{N}^1$ such that $trace(\alpha^1) = \beta\lceil N^1$. Note that all that varies among these α^1 is the firing patterns of the neurons in N^1_{int}. Analogously, define B^2 to be the set of executions α^2 of $\mathcal{N}^2$ such that $trace(\alpha^2) = \beta\lceil N^2$. All that varies among these α^2 is the firing patterns of the neurons in N^2_{int}.

Now we project the α executions onto N^1 and N^2, and we get that the above is equal to:

$$\sum_{\alpha^1 \in B^1, \alpha^2 \in B^2} (P(\alpha^1|(\beta'\lceil N^1)) \times P(\alpha^2|(\beta'\lceil N^2))).$$

This sum can be split into the product of sums:

$$\sum_{\alpha^1 \in B^1} P(\alpha^1|(\beta'\lceil N^1)) \times \sum_{\alpha^2 \in B^2} P(\alpha^2|(\beta'\lceil N^2)).$$

This is, in turn, equal to

$$P((\beta\lceil N^1_{out})|(\beta'\lceil N^1)) \times P((\beta\lceil N^2_{out})|(\beta'\lceil N^2)),$$

as needed.

□

5.3 Compositionality

Finally we are ready to prove that our behavior notion Beh is compositional. In view of Theorem 16, it suffices to show that our auxiliary behavior notion Beh_2 is compositional. And in view of Lemma 17, it suffices to show that $Beh_2(\mathcal{N})$ is determined by $Beh_2(\mathcal{N}^1)$ and $Beh_2(\mathcal{N}^2)$.

So here we show how to express $Beh_2(\mathcal{N})$ in terms of $Beh_2(\mathcal{N}^1)$ and $Beh_2(\mathcal{N}^2)$. Recall that the definition of $Beh_2(\mathcal{N})$ specifies, for each infinite input execution β_{in} of $\mathcal{N}$, a collection of conditional probabilities, one for each finite trace β of $\mathcal{N}$ of length > 0 that is consistent with β_{in}. Fix any such input execution, β_{in}, which generates a particular probabilistic execution P of $\mathcal{N}$. Then consider an arbitrary finite trace β of $\mathcal{N}$ of length $t > 0$ that is consistent with β_{in}. Let β' be the length $t-1$ prefix of β. We show how to express $P(\beta|\beta')$ in terms of the conditional probabilities that arise from probability distributions P^1 and P^2, which are generated by $\mathcal{N}^1$ and $\mathcal{N}^2$, respectively, from certain input executions. We define these input executions and distributions as follows.

- Input execution β^1_{in} and distribution P^1 for $\mathcal{N}^1$:

  Then define the infinite input execution β^1_{in} of $\mathcal{N}^1$ as follows. First, note that $N^1_{in} \subseteq N_{in} \cup N^2_{out}$, that is, every input of $\mathcal{N}^1$ is either an input of $\mathcal{N}$ or an output of $\mathcal{N}^2$. Define the firing patterns of the neurons in $N^1_{in} \cap N_{in}$ using β_{in}, that is, define $\beta^1_{in}\lceil(N^1_{in} \cap N_{in}) = \beta_{in}\lceil N^1_{in}$. And for the firing patterns of the input neurons in $N^1_{in} \cap N^2_{out}$, use $\beta'\lceil(N^1_{in} \cap N^2_{out})$ for times $0, \ldots, t-1$, and the default 0 for times $\geq t$. Define P^1 to be the probability distribution that is generated by $\mathcal{N}^1$ from input execution β^1_{in}.

- Input execution β^2_{in} and distribution P^2 for $\mathcal{N}^1$:

  Analogous, interchanging 1 and 2.

Lemma 26

$$P(\beta|\beta') = P^1((\beta\lceil N^1_{out})|(\beta'\lceil N^1)) \times P^2((\beta\lceil N^2_{out})|(\beta'\lceil N^2)).$$

Proof. Lemma 25, Part 2, tells us that:

$$P(\beta|\beta') = P((\beta\lceil N^1_{out})|(\beta'\lceil N^1)) \times P((\beta\lceil N^2_{out})|(\beta'\lceil N^2)).$$

So it suffices to show that

$$P((\beta\lceil N^1_{out})|(\beta'\lceil N^1)) = P^1((\beta\lceil N^1_{out})|(\beta'\lceil N^1)),$$

and similarly for $\mathcal{N}^2$.

There are two differences between the two expressions: First, in the first expression, we fix only the *external inputs* of $\mathcal{N}^1$, and consider the probabilistic execution of the entire network. We then consider the conditional probability $P((\beta\lceil N^1_{out})|(\beta'\lceil N^1))$, which means that we fix all the inputs and outputs of $\mathcal{N}^1$ through time $t-1$ to be as in β', and consider the probability that the firing pattern for the outputs of $\mathcal{N}^1$ at time t coincides with what is given in β. In the second expression, we fix *all the inputs* of $\mathcal{N}^1$, and consider the probabilistic execution of just $\mathcal{N}^1$. We then consider the conditional probability $P^1((\beta\lceil N^1_{out})|(\beta'\lceil N^1))$, which means that we again fix all inputs and outputs of $\mathcal{N}^1$ through time $t-1$ to be as in β', and consider the probability that the firing pattern for the outputs of $\mathcal{N}^1$ at time t coincides with what is given in β.

The second difference is that the first expression involves different input sequences to $\mathcal{N}^1$ starting from time t. The second expression fixes those inputs to 0. This should not matter, because we are concerned only with what happens up to time t, and the probabilities for times up to t depend only on inputs through time $t-1$.

The equivalence of these two expressions follows by unwinding the definitions. □

Lemma 26 seems to be a nice statement of how the probabilities decompose, and we generalize this in Lemma 31. However, it is not in exactly the right form to actually prove the compositionality of Beh_2. For this, we need a technical modification of the lemma.

Namely, define γ^1 to be the length-t trace of $\mathcal{N}^1$ such that $\gamma^1\lceil N^1_{out} = \beta\lceil N^1_{out}$ and $\gamma^1\lceil N^1_{in}$ is a prefix of β^1_{in}. That is, γ^1 pastes together the output from $\beta\lceil N^1_{out}$ with the input used in the definition of P^1. Note that $\beta'\lceil N^1$ is the one-step prefix of γ^1. Define γ^2 analogously. Now we can state a lemma that expresses conditional probabilities for $\mathcal{N}$ with β_{in} in terms of condition probabilities for $\mathcal{N}^1$ with β^1_{in} and $\mathcal{N}^2$ with β^2_{in}.

Lemma 27

$$P(\beta|\beta') = P^1(\gamma^1|(\beta'\lceil N^1)) \times P^2(\gamma^2|(\beta'\lceil N^2)).$$

Proof. By Lemma 26, we have that

$$P(\beta|\beta') = P^1((\beta\lceil N^1_{out})|(\beta'\lceil N^1)) \times P^2((\beta\lceil N^2_{out})|(\beta'\lceil N^2)).$$

So it suffices to show that the corresponding terms are the same, that is:

$$P^1((\beta\lceil N^1_{out})|(\beta'\lceil N^1)) = P^1(\gamma^1|(\beta'\lceil N^1)),$$

and similarly for $\mathcal{N}^2$. The first case follows because the definition of P^1 fixes the firing patterns for the neurons in N^1_{in} through time t, in a way that is consistent with γ^1, and the γ^1 and β agree on the neurons in N^1_{out}. Similarly for the second case. □

Now we can argue compositionality:

Lemma 28 *For all compatible pairs of networks $\mathcal{N}^1$ and $\mathcal{N}^2$, $Beh_2(\mathcal{N})$ is determined by $Beh_2(\mathcal{N}^1)$ and $Beh_2(\mathcal{N}^2)$.*

Proof. Follows directly from Lemma 27. □

Theorem 29 *Beh_2 is compositional.*

Proof. By Lemmas 28 and 17. □

Theorem 30 *Beh is compositional.*

Proof. By Theorems 29 and 16. □

We end this section with a generalization of Lemma 26 that applies to all four combinations of executions and traces. The proof should be similar to that for Lemma 26, based on earlier Lemmas 24 and 25. We will use this later, in Section 5.4.1.

Lemma 31 *Let α, α', β, and β' be as usual, P_1 and P_2 as defined earlier in this section. Then*

1. $P(\alpha|\alpha') = P^1((\alpha\lceil N^1_{lc})|(\alpha'\lceil N^1)) \times P^2((\alpha\lceil N^2_{lc})|(\alpha'\lceil N^2))$.
2. $P(\beta|\alpha') = P^1((\beta\lceil N^1_{out})|(\alpha'\lceil N^1)) \times P^2((\beta\lceil N^2_{out})|(\alpha'\lceil N^2))$.
3. $P(\alpha|\beta') = P^1((\alpha\lceil N^1_{lc})|(\beta'\lceil N^1)) \times P^2((\alpha\lceil N^2_{lc})|(\beta'\lceil N^2))$.
4. $P(\beta|\beta') = P^1((\beta\lceil N^1_{out})|(\beta'\lceil N^1)) \times P^2((\beta\lceil N^2_{out})|(\beta'\lceil N^2))$.

5.4 Examples

5.4.1 Cyclic composition

We consider the cyclic composition example from Section 3.3.3. We analyze just one case in detail, namely, where x_1 fires initially and x_2 does not. We prove here just that, with probability at least $(1-\delta)^7$, both x_1 and x_2 fire at time 4.

The input firing sequence β_{in} is trivial here, since we have no input neurons for the entire network $\mathcal{N}$. For this example, we assume that, in the initial configuration, x_1 fires and the other three neurons do not fire. So with all these restrictions, we have just a single probability distribution P for executions of $\mathcal{N}$.

We argue the result compositionally, in terms of executions. Probably we could carry out a similar analysis in terms of traces, but coping with the hidden neurons would make things more complicated.

So let A be the set of executions of length 4 in which both x_1 and x_2 fire at time 4. We aim to show that $P(A) \geq (1-\delta)^7$. For this, we define several other successively-included sets of executions:

- A_0, the set of executions of length 0 consisting of just the initial configuration, in which x_1 is firing and the others are not firing.
- A_1, the set of executions of length 1 whose one-step prefix is in A_0 and in which, in the last configuration, a_1 is firing.
- A_2, the set of executions of length 2 whose one-step prefix is in A_1 and in which, in the last configuration, x_2 is firing.
- A_3, the set of executions of length 3 whose one-step prefix is in A_2 and in which, in the last configuration, x_2 and a_2 are both firing.
- A_4, the set of executions of length 4 whose one-step prefix is in A_3 and in which, in the last configuration, x_1, x_2 and a_2 are all firing.

Then we can see that

$$P(A) \geq P(A_4) = P(A_4|A_3)P(A_3|A_2)P(A_2|A_1)P(A_1|A_0).$$

We need lower bounds for the four conditional probabilities. For example, consider $P(A_4|A_3)$. Let α' be any execution in A_3; we will argue that $P(A_4|\alpha') \geq (1-\delta)^3$, and use Total Probability to conclude that $P(A_4|A_3) \geq (1-\delta)^3$. We have:

$$P(A_4|\alpha') = \sum_{\alpha} P(\alpha|\alpha'),$$

where α ranges over the length-4 executions in A_4 that extend α'. By Lemma 31, we may write:

$$P(\alpha|\alpha') = P^1((\alpha\lceil N^1_{lc})|(\alpha'\lceil N^1)) \times P^2((\alpha\lceil N^2_{lc})|(\alpha'\lceil N^2)),$$

where P^1 and P^2 are defined from $\alpha'\lceil N_{out}$ as in Section 5.3.

So we can rewrite $\sum_\alpha P(\alpha|\alpha')$ as

$$\sum_{\alpha^1}\sum_{\alpha^2} P^1((\alpha^1\lceil N^1_{lc})|(\alpha'\lceil N^1)) \times P^2((\alpha^2\lceil N^2_{lc})|(\alpha'\lceil N^2)),$$

where α^1 ranges over all one-step extensions of $\alpha'\lceil N^1$ such that x_2 fires in the final configuration, and α^2 ranges over all one-step extensions of $\alpha'\lceil N^2$ in which x_1 and a_2 both fire in the final configuration. This summation is equal to

$$\sum_{\alpha^1} P^1((\alpha^1\lceil N^1_{lc})|(\alpha'\lceil N^1)) \times \sum_{\alpha^2} P^2((\alpha^2\lceil N^2_{lc})|(\alpha'\lceil N^2)).$$

The first term is $\geq (1-\delta)$ because we care only that x_2 fires in the final configuration, and we have assumed that it fires in the previous configuration. The second term is $\geq (1-\delta)^2$, because we care that both x_1 and a_2 fire in the final configuration, and we have assumed that a_2 and x_2 fire in the previous configuration. So we have:

$$P(A_4|\alpha') \geq \sum_{\alpha^1} P^1((\alpha^1\lceil N^1_{lc})|(\alpha'\lceil N^1)) \times \sum_{\alpha^2} P^2((\alpha^2\lceil N^2_{lc})|(\alpha'\lceil N^2)) \geq (1-\delta)(1-\delta)^2 = (1-\delta)^3.$$

So we have shown that $P(A_4|A_3) \geq (1-\delta)^3$, Similar arguments can be used to show that $P(A_3|A_2) \geq (1-\delta)^2$, $P(A_2|A_1) \geq (1-\delta)$, and $P(A_1|A_0) \geq (1-\delta)$. Combining all the terms we get that $P(A_4) \geq (1-\delta)^7$, as needed.

6 Hiding

Next, we define a hiding operation for networks, which simply reclassifies some output neurons as internal. Such an operation can be used in conjunction with a composition operation, for example, we often want to compose two networks and then hide the neurons that were used to communicate between them.

6.1 Hiding definition

Given a network $\mathcal{N}$ and a subset V of the output neurons N_{out} of $\mathcal{N}$, we define a new network $\mathcal{N}' = hide(\mathcal{N}, V)$ to be the same as $\mathcal{N}$ except that all the outputs in V are now reclassified as internal neurons. That is, all parts of the definition of $\mathcal{N}'$ and $\mathcal{N}$ are identical except that:

- $N'_{out} = N_{out} - V$, and
- $N'_{int} = N_{int} \cup V$.

The important effect of the hiding operation is to make the hidden neurons ineligible for combining with other neurons in further composition operations.

We give a result in the style of Lemma 28, here saying that the external behavior of $hide(\mathcal{N}, V)$ is determined by the external behavior of $\mathcal{N}$ and V.

Theorem 32 *For all networks $\mathcal{N}$ and subsets $V \subseteq N_{out}$, $Beh(hide(\mathcal{N}, V))$ is determined by $Beh(N)$ and V.*

Proof. Let $\mathcal{N}'$ denote $hide(\mathcal{N}, V)$. Fix any infinite input execution β_{in} for $\mathcal{N}'$, and let P' denote the generated probabilistic execution of $\mathcal{N}'$. Consider any finite trace β of $\mathcal{N}'$ that is consistent with β_{in}. We must express $P'(\beta)$ in terms of the probability distribution of traces generated by $\mathcal{N}$ on some input execution.

To do this, note that the executions of $\mathcal{N}$ are identical to those of $\mathcal{N}'$—only the classification of neurons in V is different. The input execution β_{in} is also an input execution of $\mathcal{N}$, and the probabilistic execution P' generated from β_{in} in $\mathcal{N}$ is identical to P. So we can write $P'(\beta) = P(\beta)$.

This is not quite what we need, because β is not actually a trace of $\mathcal{N}$—it excludes firing patterns for V. But we can define B to be the set of traces γ of $\mathcal{N}$ such that $\gamma \lceil N'_{ext} = \beta$, that is, the traces of $\mathcal{N}$ that project to yield β but allow any firing behavior for the neurons in V. Then we have

$$P'(\beta) = \sum_{\gamma \in B} P(\gamma).$$

This shows the needed dependency. □

6.2 Examples

6.2.1 Boolean circuits

Let $\mathcal{N}$ be the 5-gate Nand circuit from Section 3.3.1. Let V be the singleton set consisting of just the And gate within the circuit. We consider the network $\mathcal{N}' = hide(\mathcal{N}, V)$, which is the same as the Nand circuit except that the And gate is now regarded as internal. Thus, $\mathcal{N}'$ has two internal neurons, the And neuron and a.

Fix β_{in} to be any infinite input execution (of both $\mathcal{N}$ and $\mathcal{N}'$) with stable inputs, and let P and P' be the probabilistic executions of $\mathcal{N}$ and $\mathcal{N}'$, respectively, generated from β_{in}.

In P', we should expect to have stable correct Nand outputs for a long time starting from time 3. Here we consider just the situation at precisely time 3, that is, we consider the probabilities $P'(\beta)$

for finite traces β of length exactly 3. Specifically, we would like to use the connection between P and P' to help us show that the probability of a correct Nand output at time 3 is at least $(1-\delta)^3$.

We work in terms of the hiding operation. In particular, we assume that, in P, the probability of both a correct And output at time 1 and a correct Nand output at time 3 is at least $(1-\delta)^3$. This could be proved for the Nand circuit separately, but we simply assume it here.

Now define event B to be the set of traces β of $\mathcal{N}'$ of length 3 such that β gives a correct Nand output at time 3. We argue that $P'(B) \geq (1-\delta)^3$, which implies our desired result.

We have that $P'(B) = \sum_{\beta \in B} P'(\beta)$. We know that $P'(\beta) = P(\beta)$ for each such β. Since $P'(\beta) = P(\beta)$ for every trace of $\mathcal{N}'$, we have that $P'(B) = \sum_{\beta \in B} P(\beta) = P(B)$. But by assumption, $P(B) \geq (1-\delta)^3$. Therefore, $P'(B) \geq (1-\delta)^3$, as needed.

7 Problems

In this section, we define a formal notion of a *problem* to be solved by a stochastic Spiking Neural Network. We say what it means for an SNN to *solve* a problem. We prove that the solves notion respects composition and hiding operations.

7.1 Problems and solving problems

A *problem* $\mathcal{R}$ for a pair (N_{in}, N_{out}) of disjoint sets of neurons is a nonempty set of *possibilities*. Each possibility $Poss$ is a mapping that takes each infinite sequence β_{in} of firing patterns for N_{in} to a *result*. A result $R = Poss(\beta_{in})$ is a mapping that specifies, for every finite sequence β of firing patterns for $N_{in} \cup N_{out}$ of length ≥ 0 that is consistent with β_{in}, a probability $R(\beta)$.

The probabilites assigned for a particular result R must satisfy certain constraints. They must be such that the probabilities generated from them for all the "cones" form an actual probability distribution, for example, the sum of the probabilities of all one-step extensions of a frace β must be equal to the probabiity of β.[11]

Now suppose that $\mathcal{N}$ is a network with input and output neurons N_{in} and N_{out}, and $\mathcal{R}$ is a problem for (N_{in}, N_{out}). Then we say that that $\mathcal{N}$ *solves* $\mathcal{R}$ provided that, for some $Poss \in \mathcal{R}$ the following holds: Let β_{in} be any infinite input execution of $\mathcal{N}$, and let P be the probabilistic execution of $\mathcal{N}$ generated from β_{in}. Then for every finite trace β of $\mathcal{N}$, $P(\beta) = Poss(\beta_{in})(\beta)$. In other words, $R = Poss(\beta_{in})$ is exactly the trace distribution derived from the probabilistic execution of $\mathcal{N}$ on input β_{in}.

7.2 Composition

We would like a theorem of the form: If $\mathcal{N}^1$ solves problem $\mathcal{R}^1$ and $\mathcal{N}^2$ solves problem $\mathcal{R}^2$ then the composition $\mathcal{N} = \mathcal{N}^1 \times \mathcal{N}^2$ solves the composition $\mathcal{R} = \mathcal{R}^1 \times \mathcal{R}^2$. For this, we must first define the composition of two problems, $\mathcal{R} = \mathcal{R}^1 \times \mathcal{R}^2$.

Definition of composition of problems: Let $\mathcal{R}^1$ be a problem for the pair (N^1_{in}, N^1_{out}) and $\mathcal{R}^2$ a problem for the pair (N^2_{in}, N^2_{out}). Assume that Let $\mathcal{R}^1$ and $\mathcal{R}^2$ are *compatible*, in the sense that $N^1_{out} \cap N^2_{out} = \emptyset$. The composition $\mathcal{R}$ is a problem for the pair (N_{in}, N_{out}), where

- $N_{out} = N^1_{out} \cup N^2_{out}$, and
- $N_{in} = N^1_{in} \cup N^2_{in} - N_{out}$.

[11] Also, since we are assuming that the initial states of networks are fixed rather than determined probabilistically, we might want to insist that R assign probability 1 to one particular trace of length 0. We don't need to do this, so let's avoid this for now.

The composition is a nomempty set of *possibilities*, each of which is a mapping that takes each infinite sequence β_{in} of firing patterns for N_{in} to a *result*. Each result R specifies a probability distribution on the set of infinite sequences of firing patterns for $N_{in} \cup N_{out}$ that are consistent with β_{in}. To define the set of possibilities for $\mathcal{R}$, we simply fix (in an arbitrary way) possibilities $Poss^1$ and $Poss^2$ for $\mathcal{R}^1$ and $\mathcal{R}^2$, respectively, and then construct a possibility $Poss$ from $Poss^1$ and $Poss^2$. Since there may be many ways to fix the combination of choices of $Poss^1$ and $Poss^2$, $\mathcal{R}$ may wind up containing many different possibilities.

To construct $Poss$ from a given $Poss^1$ and $Poss^2$, we fix any infinite sequence β_{in} of firing patterns for N_{in}, and define the result $R = Poss(\beta_{in})$. This requires us to define $R(\beta)$ for every finite sequence β of firing patterns of $N_{in} \cup N_{out}$ that is consistent with β_{in}. We do this using a recursive approach, inspired by the conditional construction used for Lemma 26.

For the base, consider β of length 0. Then we define $R(\beta) = 1$ if

$$\beta \lceil N^1_{out} = F^1_0 \lceil N^1_{out} \text{ and } \beta \lceil N^2_{out} = F^2_0 \lceil N^2_{out},$$

and 0 otherwise. That is, we assign probability 1 to the length-0 sequence β in which the initial output firing states are as specified for networks $\mathcal{N}^1$ and $\mathcal{N}^2$.

For the recursive step, consider β of length ≥ 1. Let β' be the one-step prefix of β. We will define $R(\beta)$ to be $R(\beta') \times T^1 \times T^2$, where T^1 and T^2 correspond to conditional probabilities for the output neurons generated by $Poss^1$ and $Poss^2$, respectively.

Let β^1_{in} be the infinite sequence of firing patterns for N^1_{in} that are constructed from (a) $\beta_{in} \lceil N^1_{in}$, for neurons in $N^1_{in} \cap N_{in}$, and (b) $\beta \lceil (N^1_{in} \cap N^2_{out})$ for times $0, \ldots, t-1$, and the default 0 for times $\geq t$. Define $R^1 = Poss^1(\beta^1_{in})$, that is, the result generated by $\mathcal{R}^1$ on input β^1_{in}. Define $T^1 = R^1((\beta \lceil N^1_{out})|(\beta' \lceil N^1))$.

Define β^2_{in}, R^2, and T^2 analogously. Thus, $T^2 = R^2((\beta \lceil N^2_{out})|(\beta' \lceil N^2))$. Finally, define $R(\beta) = R(\beta') \times T^1 \times T^2$.

Theorem 33 *If $\mathcal{N}^1$ solves problem $\mathcal{R}^1$ and $\mathcal{N}^2$ solves problem $\mathcal{R}^2$ then the composition $\mathcal{N} = \mathcal{N}^1 \times \mathcal{N}^2$ solves the composed problem $\mathcal{R} = \mathcal{R}^1 \times \mathcal{R}^2$.*

Proof. Since $\mathcal{N}^1$ solves $\mathcal{R}^1$, we know that there is a possibility $Poss^1 \in \mathcal{R}^1$ such that, for any infinite input execution β^1_{in} of $\mathcal{N}^1$, $Poss^1(\beta^1_{in})$ is identical to the trace distribution derived from $\mathcal{N}^1$ on input β^1_{in}. In other words, $Poss^1 = Beh(\mathcal{N}^1)$. Likewise, since $\mathcal{N}^2$ solves $\mathcal{R}^2$, there is a possibility $Poss^2 \in \mathcal{R}^2$ such that, for any infinite input execution β^2_{in} of $\mathcal{N}^2$, $Poss(\beta^2_{in})$ is identical to the trace distribution derived from $\mathcal{N}^2$ on input β^2_{in}. In other words, $Poss^2 = Beh(\mathcal{N}^2)$. We define a possibility $Poss \in \mathcal{R}$ from $Poss^1$ and $Poss^2$, following the approach just above, in the definition of composition of problems. We claim that $Poss = Beh(\mathcal{N})$.

In order to show that $Poss = Beh(\mathcal{N})$, we must show that, for any infinite sequence β_{in} of firing patterns for N_{in}, $Poss(\beta_{in})$ is the same trace distribution that is generated by $\mathcal{N}$ on input β_{in}. So fix β_{in}, let P be the trace distribution generated by $\mathcal{N}$ on input β_{in}, and let $R = Poss(\beta_{in})$. We must show that, for any finite trace β of $\mathcal{N}$ that is consistent with β_{in}, $P(\beta) = R(\beta)$. We do this by induction on the length of β.

For the base, consider β of length 0. If $\beta \lceil N^1_{out} = F^1_0 \lceil N^1_{out}$ and $\beta \lceil N^2_{out} = F^2_0 \lceil N^2_{out}$, then $R(\beta)$ is defined to be 1, and otherwise it is defined to be 0. The definition of P yields 1 for the starting output configuration of $\mathcal{N}$ and 0 for other output configurations. The starting output configuration is the unique configuration C for which $C \lceil N^1_{out} = F^1_0 \lceil N^1_{out}$ and $C \lceil N^2_{out} = F^2_0 \lceil N^2_{out}$. Since C is the same as the unique configuration in $\beta \lceil N_{out}$, this implies that $P(\beta) = R(\beta)$.

For the inductive step, consider β of length ≥ 1. Let β' be the one-step prefix of β. By the inductive hypothesis, we know that $P(\beta') = R(\beta')$. We must show that $P(\beta) = R(\beta)$.

Fix β^1_{in}, R^1, β^2_{in}, and R^2 as in the recursive definition of $R(\beta)$. Then by the definition of $R(\beta)$, we have

$$R(\beta) = R(\beta') \times R^1((\beta\lceil N^1_{out})|(\beta'\lceil N^1)) \times R^2((\beta\lceil N^2_{out})|(\beta'\lceil N^2)).$$

Similarly, for the same β^1_{in} and β^2_{in}, fix P^1 and P^2, probabilistic traces for $\mathcal{N}^1$ and $\mathcal{N}^2$ respectively. Then by Lemma 26, we have

$$P(\beta) = P(\beta') \times P^1((\beta\lceil N^1_{out})|(\beta'\lceil N^1)) \times P^2((\beta\lceil N^2_{out})|(\beta'\lceil N^2)).$$

By the assumption that $\mathcal{N}^1$ solves $\mathcal{R}^1$ with the particular possibility $Poss^1$, we have that $P^1 = R^1$, so $P^1((\beta\lceil N^1_{out})|(\beta'\lceil N^1)) = R^1((\beta\lceil N^1_{out})|(\beta'\lceil N^1))$. Similarly, $P^2 = R^2$, so $P^2((\beta\lceil N^2_{out})|(\beta'\lceil N^2)) = R^2((\beta\lceil N^2_{out})|(\beta'\lceil N^2))$.

Since the three corresponding terms in the two equations are all equal, we have that their products are equal, that is, $P(\beta) = R(\beta)$, as needed. □

7.3 Hiding

Now we define a hiding operator on problems, analogous to the hiding operation on networks. Namely, given a problem $\mathcal{R}$ for (N_{in}, N_{out}), and a subset V of the output neurons N_{out} of $\mathcal{R}$, we define a new problem $\mathcal{R}' = hide(\mathcal{R}, V)$ for (N'_{in}, N'_{out}), where

- $N'_{out} = N_{out} - V$, and
- $N'_{in} = N_{in}$.

For each possibility $Poss$ of $\mathcal{R}$, we define a possibility $Poss'$ for $\mathcal{R}'$. Namely, if β_{in} is any infinite sequence of firing patterns for $N_{in} = N'_{in}$, and β is any finite sequence of firing patterns for $N'_{in} \cup N'_{out}$, then define $Poss'(\beta_{in})(\beta)$ as follows. First, let B denote the set of finite sequences γ of firing patterns for $N_{in} \cup N_{out}$ such that $\gamma\lceil(N'_{in} \cup N'_{out}) = \beta$. Then define

$$Poss'(\beta_{in})(\beta) = \sum_{\gamma \in B} Poss(\beta_{in})(\gamma).$$

Theorem 34 *If network $\mathcal{N}$ solves problem $\mathcal{R}$, and $V \in N_{out}$, then $hide(\mathcal{N}, V)$ solves $hide(\mathcal{R}, V)$.*

Proof. Since $\mathcal{N}$ solves $\mathcal{R}$, we know that there is a possibility $Poss \in \mathcal{R}$ such that for every infinite input execution β_{in} of $\mathcal{N}$, with P the generated probabilistic execution, the following holds. For every finite trace β of $\mathcal{N}$, $P(\beta) = Poss(\beta_{in})(\beta)$.

Let $\mathcal{N}'$ denote $hide(\mathcal{N}, V)$ and let $\mathcal{R}'$ denote $hide(\mathcal{R}, V)$. We show that $\mathcal{N}'$ solves $\mathcal{R}'$.

Define $Poss'$ for $\mathcal{R}'$ from $Poss$ as in the construction just before this theorem. Fix an infinite sequence β_{in} of firing patterns for $\mathcal{N}'$ and let P' be the generated probabilistic execution of $\mathcal{N}'$. Let β be a finite trace of $\mathcal{N}'$. We must show that $P'(\beta) = Poss'(\beta_{in})(\beta)$.

Let B denote the set of finite sequences γ of firing patterns for $N_{in} \cup N_{out}$ such that $\gamma\lceil(N'_{in} \cup N'_{out}) = \beta$. Then $P'(\beta) = \sum_{\gamma \in B} P(\gamma)$ and $Poss'(\beta_{in})(\beta) = \sum_{\gamma \in B} Poss(\beta_{in})(\gamma)$. Since for each such γ, $P(\gamma) = Poss(\beta_{in})(\gamma)$, the two expressions are equal. □

7.4 Examples

In this section, we define three problems, all satisfying our formal definition of problems. They are Winner-Take-All, Filter, and a problem we call Attention which can be solved by combining Winner-Take-All and Filter.

The Winner-Take-All problem: We define the Winner-Take-All problem formally using notation that corresponds to the statement of Theorem 12: we write it as $WTA(n, \delta, t_c, t_s)$, using four parameters that appear in the theorem statement. The problem statement allows considerable nondeterminism, in the choice of which output ends up firing, in the time when the stable interval begins, and in what happens outside the stable interval.

The set N_{in} is $\{x_1, \ldots, x_n\}$, and N_{out} is $\{y_1, \ldots, y_n\}$. Each possibility is a mapping that takes each infinite sequence β_{in} of firing patterns for N_{in} to a probability distribution on infinite sequences of firing patterns for $N_{in} \cup N_{out}$. In this case (and for the other problems in this section), we simply define allowable results for each β_{in} independently, and combine them in all combinations to get the possibility mappings.

So consider any β_{in}. If the firing pattern for N_{in} in β_{in} is not stable or does not have at least one firing neuron, then we allow all possible distributions that are consistent with β_{in}. Now consider the case where β_{in} is stable with at least one firing neuron. Then the allowable results for β_{in} are exactly the distributions that satisfy the following condition: With probability $\geq 1 - \delta$, there is some $t \leq t_c$ such that the y outputs stabilize by time t to one steadily-firing output y_i, and this firing pattern persists through time $t + t_s - 1$. Notice that these distributions may differ in many ways, for example, they may give equal probabilities to choosing each output, or may favor some over others. They may exhibit different times, or distributions of times, for when the stable interval begins. They may exhibit different types of behavior before and after the stable interval.

We argue that our WTA network from Section 2.4.2 solves the formal WTA problem $WTA(n, \delta, t_c, t_s)$. More specifically, we consider our WTA network with the weighting factor γ satisfying the inequality $\gamma \geq c \log(\frac{nt_s}{\delta})$, and with $t_c \approx c \log n \log(\frac{1}{\delta})$. And we allow initial firing patterns for the internal and output neurons to be arbitrary; so technically, we are talking about a class of networks, not a single network. Then Theorem 12 implies that each of these networks solves the problem.

The Filter problem: We define the Filter problem as $Filter(n, \delta)$. The set N_{in} is $\{w_i, y_i | 1 \leq i \leq n\}$ and the set N_{out} is $\{z_i | 1 \leq i \leq n\}$. The Filter problem is intended to say that, for every i, $1 \leq i \leq n$, the output neuron z_i should fire at any time $t \geq 1$ exactly if both the corresponding inputs w_i and x_i fired at time $t - 1$. Thus, it acts like n And networks.

Formally, each possibility is a mapping that takes each infinite sequence β_{in} of firing patterns for N_{in} to a probability distribution on sequences of firing patterns for $N_{in} \cup N_{out}$. We define the allowable results for each β_{in} independently. Here we express the requirements in terms of conditional probabilities.

So consider any particular β_{in}. Then the allowable results for β_{in} are exactly the distributions P on infinite firing sequences over $N_{in} \cup N_{out}$ that are consistent with β_{in} and satisfy the following condition. Let β be any finite sequence over $N_{in} \cup N_{out}$ of length $t \geq 1$ that is consistent with β_{in}, and let C_t be the final configuration of β. Let β' be the the one-step prefix of β, and C_{t-1} be the final configuration of β'. Suppose that, for every i, $1 \leq i \leq n$, $C_t(z_i) = C_{t-1}(w_i) \wedge C_{t-1}(z_i)$. That is, β extends β' with correct outputs at the final time. Then $P(\beta | \beta') \geq (1 - \delta)^n$. The differences among these distributions may involve different conditional probabilities, as long as they satisfy the inequality.

Our simple Filter network of Section 3.3.2 solves the formal Filter problem, with $\delta = 1 - (1 - \delta')^n$, where δ' is the failure probability for a single And gate at a single time, according to notation used in Section 3.3.2.

The Attention problem: We define the Attention problem formally as

$$Attention(n, \delta, t_c, t_s) = WTA(n, \delta^1, t_c, t_s) \times Filter(n, \delta^2).$$

Here δ, δ^1, and δ^2 are related so that $(1-\delta) = (1-\delta^1)(1-\delta^2)^{t_s}$. The set N_{in} is $\{x_i, w_i | 1 \leq i \leq n\}$, and N_{out} is $\{y_i, z_i | 1 \leq i \leq n\}$.

$Attention(n, \delta, t_c, t_s)$ guarantees that, with probability at least $(1-\delta^1)$, the y_i outputs converge to a single firing output corresponding to a firing input within time t_c, and this configuration persists for time t_s. Furthermore, it guarantees that with probability at least $1-\delta^2$, after any prefix, the z_i outputs exhibit correct And behavior with respect to the previous time's y_i and w_i firing behavior. It follows that, with probability at least $(1-\delta^2)^{t_s}$, the z_i outputs exhibit correct And behavior throughout the stable y_i firing interval. In other words, with probability at least $(1-\delta) = (1-\delta^1)(1-\delta^2)^{t_s}$, the *Attention* network correctly mirrors the inputs corresponding to the chosen y_i output throughout the stable interval.

By Theorem 33, we see that any compatible solutions to $WTA(n, \delta^1, t_c, t_s)$ and $Filter(n, \delta^2)$ can be composed to yield a solution to $Attention(n, \delta, t_c, t_s)$. In particular, the solutions to these problems that we presented in Sections 2.4.2 and 3.3.2 can be composed in this way.

We can also define a version of the Attention problem in which we hide the y_i outputs, $hide(Attention(n, \delta, t_c, t_s),$ The guarantees of this problem are similar to those of the $Attention(n, \delta, t_c, t_s)$ problem, except that the behavior of the y_i neurons is not mentioned explicitly. Essentially, this problem says that, with probability at least $(1-\delta) = (1-\delta^1)(1-\delta^2)^{t_s}$, the network correctly mirrors the inputs corresponding to some y_i output, throughout the stable interval. The same composition as above, with hiding of the y_i outputs, solves this problem.

8 Conclusions

We have described in detail a model for Spiking Neural Networks, including composition and hiding operations. We have proved fundamental theorems about these operators.

The model used in ths paper is very basic; in particular, each neuron in this paper has simple state, just a Boolean representing whether it is firing or not. In future work, we will try to extend the definitions and results to allow a neuron to have more elaborate state, including history of its recent past firing or accumulated potential. Such extensions are realistic for a brain network model.

References

[LMP17a] Nancy Lynch, Cameron Musco, and Merav Parter. Computational tradeoffs in biological neural networks: Self-stabilizing winner-take-all networks. In *Proceedings of the 8th Conference on Innovations in Theoretical Computer Science (ITCS)*, 2017. Full version available at https://arxiv.org/abs/1610.02084.

[LMP17b] Nancy Lynch, Cameron Musco, and Merav Parter. Neuro-RAM unit with applications to similarity testing and compression in spiking neural networks. In *Proceedings of the 2017 Internal Symposium on Distributed Computing (DISC)*, 2017. Full version available at https://arxiv.org/abs/1706.01382.

[Mus18] Cameron Musco. *The Power of Randomized Algorithms: From Numerical Linear Algebra to Biological Systems*. PhD thesis, Electrical Engineering and Computer Science, Massachusetts Institute of Technology, Cambridge, MA 02139, June 2018. Neural algorithms work covered in Chapter 5.

[Seg95] Roberto Segala. *Modeling and Verification of Randomized Distributed Real-Time Systems*. PhD thesis, Laboratory for Computer Science, Massachusetts Institute of Technology, Cambridge, MA 02139, June 1995.