On the Closure Under Intersections of Probabilistic Language Recognizers

Victor Ardulov
University of Southern California
`ardulov@usc.edu`

Abstract

Probabilistic Language Recognizers are a popular way to statstically model languages. While varying in the underlying mechanism, each recongizer attempts to represent the set of strings accepted by a language and the probability of that string. Finding the weighted intersection of 2 languages implies finding all (if any) strings that exist in the set of both languages presented. The discussion will examine in particular current work on the closure of 2 different types of Probabilistic Language Recognizers.

1 Introduction

A fundamental part of Natural Language Processing, is the availability of language recongnizers. Typically learned from collecting data, language recongizers in their most basic form are a machine that determines whether an observed sequence of tokens belongs to a language or not. Throughout the study of the domain many different architectures have been discovered, and used for different problems with varying conveniences and properties.

Two of these popular Language Reconizers are Probabilistic Finite State Acceptors and Recurrent Neural Networks (RNNs). While these are popular forms of representing languages, and though they are defined as mathematical formalisms, some of their properties are unknown. In particular it is unknown whether or not these 2 classes are closed under intersection.

In more intuitive terms, it is common place to intersect languages, evaluating the sequences that belong only to *all* of the intersected languages. Closure under the intersection implies that given 2 machines of the same class representing different languages, can we construct a third machine of the same class, that recognizes only the overlapping sequences from the original languages. The discussion will be theoretically guided, and explore some new work into the determining this property.

2 Background

2.1 Definitions

We begin by formally defining the existance of an *alphabet*, $\mathcal{A}$, from which we can sample tokens to construct a *string*, $s = [a_i, \dots, a_n]$, such that $\forall i \in [1, \dots, n]$, $a_i \in \mathcal{A}$. We say that $\mathcal{S}$ is the set of all possible strings that are constructable with a given alphabet. A *language*, L, is a collection of strings from $\mathcal{S}$, ($L \subseteq \mathcal{S}$).

2.2 Finite State Recognizers

A Weighted Finite State Acceptor/Recognizer (WFSA) is a network used to represent a "language" L, such that,

$$W(s) = \begin{cases} 0, & s \notin L \\ \mathbb{R}^+, & s \in L \end{cases}$$

If a string, $s \in L$, then W maps it to a positive real number that represents the "weight" of the string.

WFSAs are often represented as a network of nodes and transitions which accept and reject tokens from parameterized in with the following tuple $\langle Q, T, E, \iota \rangle$:

- Q - a set of states
- $T = [T_{q_1}, \dots T_{q_N}]$ - set of weighted transitions, such that $\forall q \in Q, T_q \in T$ represents the set of all edges from state q. Furthermore

$$\forall (t, w) \in T_q,\, t \times q \mapsto Q, w \in \mathbb{R}^+,\, t \in \mathcal{A}$$

I.e. t represents a token which, when crossed with a state q, maps to the next state, and w represents the weight assigned to the that transtion

- $E \subseteq Q$ - set of states that are terminal, meaning that if a strings tokens follow a path to state $q \in E$, then $s \in L$
- $\iota \in Q$ - initial entry into the WFSA

It is said that there is a *path*,, $\tau_{i,j}$, between states q_i and q_j if there exists some sequece of transitions that can be followed from state q_i to q_j. If the string $s \in L$ then there is a path τ_s through the WFSA that assigned a weight ω to the string by;

$$\omega(\tau) = \prod_{\forall (t,w) \in \tau_s} w$$

A special class of WFSA, called Probabilistic Finite State Acceptors/Recognizers (PFSA), is a WFSA with 2 added constraints:

- All terminal states cannot have any outgoing edges
- The sum of the weights outgoing any state q must strictly be 1

2.3 Neural Network Recognizers

An RNN, R is said to map these strings to a "weight", formally, that is to say that: $\forall s \in \mathcal{S}, R : s \mapsto [0, 1]$. More specifically, a single-layer RNN is paramtereized by, $\langle N, h_0, \Theta, \Phi, B, \Psi, C, \sigma \rangle$, representing:

- N which is the number of states, sometimes called "neurons"
- h_0 the intial output of the activation layer.
- Θ, the "input" weight matrix of size, $N \times |\mathcal{A}|$
- Φ, the transition weight matrix of size, $N \times N$
- B, a bias vector of size N
- Ψ, the output weight matirx of size, $|\mathcal{A}| \times N$
- C, a bias vector of size $|\mathcal{A}|$
- σ, an activation function. (E.g. Rectified Linear Unit (ReLU), Sigmoid, Identity)

Similary to PFSAs, tokens of the string $s = [a_1, \ldots, a_n]$ are iteratively fed as input into a neural network which for which a compounding product value is computed representing the likelihood that that string exists with the language L. Spccifically this is done by following the dynamics as described below:

- The "hidden" state evolutions of the RNN

$$h_i = \sigma(\Theta a_i + \Phi h_{i-1} + B)$$

- The unsmoothed token prediction y_i:

$$y_i = \Psi h_i + \beta$$

- The "softmax"-normalized token prediction:

$$p_i = \text{softmax}(y_i) = \frac{1}{\sum_{j=0}^{|\mathcal{A}|} e^{y_{i,j}}} \begin{bmatrix} e^{y_{i,1}} \\ \vdots \\ e^{y_{i,|\mathcal{A}|}} \end{bmatrix}$$

where $y_{i,j}$ is the j-th element of the y_i element.

- Finally, the RNN output "weight" assigned to a string of length n:

$$P(s) = \prod_{i=0}^{n} p_{i\chi_i}$$

where χ_i returns the index for p_i which is associated with x_i

2.4 Closure Under Intersection

A class of functions, can be thought of as a set of functions (sets) that share certain characteristics (e.g. the set of all PFSAs and the set of all RNNs,). When refering to closure under an operation, it generally refers to the idea that given a set and some arbitrary operation $*$, when the operation is applied to members of the set, the output of the operation is a member of the same set. If we imagine functions as abstract representations of sets, then the intersection of sets (and thus the functions used to represent them) is the collection of overlapping members that belong to all sets being intersected. The *weighted* intersection of members from a class, is a membership intersection between weighted sets (i.e. sets where members also have a weight assigned to them). The resulting set members are also weighted proportionally to the weights of the original operands.

The discussion presented will circulate around the proof of closure under weighted intersection for the 2 different classes of PFSAs and RNNs. In the following sections we will exmine the exact definition of the closure property for the classes, as well as methods of construction and proofs about the property for each.

3 Probabilistic Finite State Acceptors

Using the definition in §2.2 the PFSA, we define the objective is answer the following questions:

- Given arbitrarily many PFSAs, $P_1, \ldots, P_m$ does there exist a PFSA, $P_{\text{intersect}}$, such that:

$$\begin{aligned} P_{\text{intersect}}(s) &= P_1(s) \cap P_2(s) \cap \ldots \cap P_m \\ &= \frac{\prod_{i=1}^{m} P_i(s)}{\sum_{s' \in \mathcal{S}} \prod_{j=1}^{m} P_i(s')} \end{aligned}$$

- Does there exist an algorithm for construction of $P_{\text{intersect}}$?

We begin by demonstrating certain facts about PFSAs.

Lemma 1. *Given a PFSA, P, parameterized by, $\langle Q, T, E, \iota \rangle$:*

$$\sum_{\forall s \in \mathcal{S}} P(s) = 1$$

Upholding this propoerty is the definition of consistent.

Proof. We will begin with the construction of a *singleton* PFSA. We define the M-singleton by taking an M-sized ($M \geq 1$) set of terminal states, E, and then defining $Q = \{\iota\} \cup E$. (i.e. the PFSA has exactly one non-terminal state, therefore all transistion must go to a terminal state).

It is clear to see that through the construction of this M-singleton, and the definitions presented above for consistency, and PFSAs that the M-singleton must be consistent.

Next let us define a *strictly-forward PFSA* (SF-PFSA), in this case the edges from a state q to transition to other states q' such that there exists no paths $\tau_{q',q}$ (i.e. no loops are permitted). From this construction, it logically follows that the network must contain within it at least one node that when isolated from the rest of the network must be a singleton. If there is not one, this would imply that there exists a state that must point "backwards" which therefore breaks our construction of the SF-PFSA.

Furthermore, we observe that given state q with n-many edges to k-many M-singleton states $[q'_1, \ldots q'_M]$, the consistency is preserved since if the edges go to an M-singleton they are "locally" consistent, and paths exiting q must be themselves "locally" consistent as well to follow the construction of a PFSA.

More rigorously, since for all *pre-terminal* states which are M-singleton, S,:

$$\sum_{\forall s \in \mathcal{S}} S(s) = 1$$

then we know that for nodes that point strictly to terminal states and pre-terminal states are themselves consistent, since

$$\sum_{\forall (t,w) \in T_q} \begin{cases} w, & \text{if } t \times q \mapsto E, \\ wS(s'), & \text{if } t \times q \mapsto \text{pre-terminal state} \end{cases}$$

$$= \sum_{\forall (t,w) \in T_q} w = 1$$

Recursively, we can observe that this relationship allows us to show consistency for arbitrarily long SF-PFSA.

Finally let us imagine an uncountably-infinitely long SF-PFSA (since there is no longer a finite number of states we refer to it as a *infinite strictly forward probabilistic state acceptor* (ISFP-SA)). Above we showed that through recursion it must also be consistent. Now notice that a loop can be deconstructed and represented as an ISFP-SA with a particular pattern representing the states and transitions found in the loop. Notice that loops will also now preserve consistency, and as such we have removed any requirements on the transtions that are in our PFSA.

Therefore all PFSAs are consistent. This concludes the proof. □

Corollary 1.1. *All PFSAs can be represented as a strictly forward structure.*

Now that we know that PFSAs are stricly consistent, and that they can be taken under when treated under as a WFSA a weighted intersection can be taken such that given 2 PFSAs P_A and P_B theres is

$$W'(s) = P_A(s) P_B(s)$$

which can be deterministically constructed. (**?**)

From this fact the following lemma is derived:

Lemma 2 (Weighted Intersection of PFSAs are finitely bounded)**.** *Given WFSA W, constructed as the weighted intersection of 2 PFSAs P_A and P_B, such that:*

$$W(s) = P_A(s)P_B(s)$$

then,

$$\sum_{s \in \mathcal{S}} W(s) \leq 1$$

Proof. Following Lemma 1, that PFSAs are consistent by definition, then it holds that,

$$\sum_{s \in \mathcal{S}} P_A(s) = 1$$

and,

$$\sum_{s \in \mathcal{S}} P_B(s) = 1$$

we also assume that,

$$\forall s \in \mathcal{S}, P_A(s), P_B(s) \geq 0$$

From this it follows that,

$$\forall s_1, s_2 \in \mathcal{S}, P_A(s_1)P_B(s_2) \geq 0$$

Knowing this, the following holds true,

$$\Big(\sum_{s \in \mathcal{S}} P_A(s)\Big)\Big(\sum_{s \in \mathcal{S}} P_B(s)\Big) = 1$$

$$= (P_A(s_1)+P_A(s_2)+\ldots)(P_B(s_1)+P_B(s_2)+\ldots)$$

$$= (P_A(s_1)P_B(s_1) + P_A(s_1)P_B(s_2) + \ldots$$

$$+P_A(s_2)P_B(s_1) + P_A(s_2)P_B(s_2) + \ldots$$

$$= \sum_{s \in \mathcal{S}} P_A(s)P_B(s) + \sum_{s_1 \neq s_2 \in \mathcal{S}} P_A(s_1)P_B(s_2)$$

$$\therefore \sum_{s \in \mathcal{S}} P_A(s)P_B(s) = 1 - \sum_{s_1 \neq s_2 \in \mathcal{S}} P_A(s_1)P_B(s_2)$$

And since,

$$\sum_{s_1 \neq s_2 \in \mathcal{S}} P_A(s_1)P_B(s_2) \geq 0$$

then it holds that,

$$\sum_{s \in \mathcal{S}} P_A(s)P_B(s) \leq 1$$

this implies that the

$$\sum_{s \in \mathcal{S}} W(s) \leq 1$$

and that the sum is finitely bounded. □

At this point by taking the "weighted intersection" of 2 PFSAs, we have shown that we are able to extract a structure for a WFSA, and that the WFSA has a finite sum for all paths recognized by the weighted intersection. From this we can logically extract that each of "sub-graphs" also must have a finite sum.

Next by applying this logical extension, we can demonstrate that PFSAs must be closed under intersection:

Theorem 1 (Closure Under Intersection)**.** *Given 2 PFSAs A and B which assign probabilities P_A and P_B to strings $s \in \mathcal{S}$ it is possible to construct PFSA C such that:*

$$P_C(s) = \frac{P_A(s)P_B(s)}{\sum_{s' \in \mathcal{S}} P_A(s')P_B(s')}$$

Proof. Let there be a WFSA W with finite sum K and let, K_j represent the finite sum of all paths extending from state, q_j. ($K_\iota = K$). By extending the method we found in Lemma 1, we can show that

$$K_j = \sum_{(t_k, w_k) \in T_j} w_k K_{(q_j \times t_k)}$$

where $K_j = 1$ if q_j is terminal.

By making a copy of W and P we want to find $\hat{T}$ such that the resulting graph is a PFSA. For transitions all $T_j \in T$ we construct $\hat{T}_j$ with all of the same transitions and compute the weights in the following manner:

$$\forall (t_k, w_k) \in T_j, \hat{w}_k = \frac{w_k K_{(q_j \times t_k)}}{K_j}$$

We can see that by construction that the new edges' weight must sum to 1, and this will by construction meet that requirement that we therefore have built a PFSA

Now given string $s \in L_W$, let us to follow an arbitrary path $\hat{\tau}_{\iota,k}$ such that $q_k \in E$. We can see that following a path

$$P(s) = \omega(\hat{\tau_{\iota,k}}) = \prod_{(t_j, \hat{w}_j) \in \hat{\tau}_{\iota,k}} \hat{w}_j$$

$$= (\frac{w_1 K_{(\iota \times t_1)}}{K})(\frac{w_2 K_{(q_1 \times t_2)}}{K_{(\iota \times t_1)}}) \ldots (\frac{w_k}{1})$$

$$= \frac{\prod_{(t_j, w_j) \in \tau_{\iota,k}} w_j}{K}$$

$$= \frac{W(s)}{K}$$

This implies that given 2 PFSAs A and B for which we can take the weighted intersection of, we can construct a WFSA $W(s) = P_A(s)P_B(s)$ and applying by applying Lemma 2 and the above weight assignment we can show that we can construct PFSA P_C such that

$$P_C(s) = \frac{P_A(s)P_B(s)}{\sum_{s' \in \mathcal{S}} P_A(s')P_B(\sigma)}$$

$\square$

In the Supplemental Materials we provide the method and algorithm for computing the weights of the PFSA in more detail.

4 Recurrent Neural Networks

Similarly to objective presented in §3, we with to find now identify the existance and find the intersection of Probabilistic Language Models that are represented by RNNs:

- Given RNN A and B, can the existance of RNN C be determined such that,

$$R_C(s) = R_A(s) \cap R_B(s) = \frac{R_A(s)R_B(s))}{\sum_{s' \in \mathcal{S}} R_A(s')R_B(s')}$$

  ?

- Does there exist a method to deterministically constructing this RNN if it is determined to exist?

These questions have proven to be highly nontrivial, so to approach them we will begin with a relaxed definition of the problem.

4.1 Nonconsistent Weighted Intersection

One of the difficulties on showing the intersection of RNNs is the final layer of the system. This is because softmax is not an invertible function. More specifically given a vector, $y = \text{softmax}(x)$, there is no way to effectively reconstruct x., because softmax is invariant to translational constant factors. Explicitly,

$$\text{softmax}(\begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ {}_m \end{bmatrix}) = \text{softmax}(\begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_m \end{bmatrix} + \begin{bmatrix} k \\ k \\ \vdots \\ k \end{bmatrix})$$

So we begin with a modified question:
Does there exist RNN $\hat{R}_C$ such that, given RNN A and B:

$$\hat{R}_C(s) = \frac{R_A(s)R_B(s))}{\kappa}$$

where κ is some scalar value?

We try to tackle this problem by first showing that the desired output is even constructable by some softmax function

Lemma 3 (Composition of Softmax Functions)**.** *Given vectors r_A = softmax($\vec{a}$) and r_B = softmax($\vec{b}$) it is possible to construct r_C = softmax($\vec{c}$) such that*

$$r_{C,i} = \frac{p_{A,i}p_{B,i}}{\sum_{\forall j} r_{A,j}r_{B,j}}$$

and,

$$\vec{c} = \vec{a} + \vec{b}$$

Proof.

$$\begin{aligned}\frac{r_{A,i}p_{B,i}}{\sum_{\forall j} r_{A,j}r_{B,j}} &= \frac{(\frac{e^{a_i}}{\sum_{\forall p} e^{a_p}})(\frac{e^{b_i}}{\sum_{\forall q} e^{b_q}})}{\sum_{\forall j}(\frac{e^{a_j}}{\sum_{\forall r} e^{a_r}})(\frac{e^{b_j}}{\sum_{\forall s} e^{b_s}})} \\ &= \frac{e^{a_i}e^{b_i}}{\sum_{\forall j} e^{a_j}e^{b_j}} \\ &= \frac{e^{a_i+b_i}}{\sum_{\forall j} e^{a_j+b_j}}\end{aligned}$$

$$\therefore r_C = \text{softmax}(\vec{a} + \vec{b})$$

$\square$

Applying Lemma 3 implies that given RNN R_a and R_b parameterized by

$$\langle N_a, h_{a,0}, \Theta_a, \Phi_a, B_a, \Psi_a, C_a, \sigma \rangle$$

and,

$$\langle N_b, h_{b,0}, \Theta_b, \Phi_b, B_b, \Psi_b, C_b, \sigma \rangle$$

respectively, where everything except the activation functions can be unique, an RNN R_C,

$$\langle N_c, h_{c,0}, \Theta_c, \Phi_c, B_c, \Psi_c, C_c, \sigma \rangle$$

is constructable with the following properties:

$$N_c = N_a + N_b$$

$$h_{c,0} \leftarrow \begin{bmatrix} h_{a,0} \\ h_{b,0} \end{bmatrix} \Theta_c \leftarrow \begin{bmatrix} \Theta_a \\ \Theta_b \end{bmatrix}$$

$$\Phi_c \leftarrow \begin{bmatrix} \Phi_a & \emptyset \\ \emptyset & \Phi_b \end{bmatrix} B_c = \begin{bmatrix} B_a \\ B_b \end{bmatrix}$$

$$\Psi_c = \begin{bmatrix} \Psi_a \Psi_b \end{bmatrix} C_c = C_a + C_b$$

This is because we have effectively encoded both of our original neural networks into the parameters of R_c, and now its compounded product is

$$R_c(s) = \prod_{\forall a_i \in s} \frac{e^{(y_{A,i,\chi_i})} e^{(y_{B,i,\chi_i})}}{\sum_{y_{i,j} \in y_i} e^{(y_{A,i,j})} e^{(y_{B,i,j})}}$$

This is close to our desired consistent vector, in that the output of the softmax is a vector that points to in the correct direction as our desired vector, however it is off by some linear scale. Unfortunately we cannot really control this linear scale and therefore this is not a construction that seems to help us further. The evidence points to the complexity arising from the non-linear and non-invertible nature of the softmax.

4.2 Consistent Weighted Intersection

A corollary of Lemma 3.2 in Chen et al. (2017), is that RNNs can be constructed to have unbounded sums over the set of strings $s \in \mathcal{S}$. Logically it follows that this would imply that simply constructing an RNN R' such that given some R:

$$R'(s) = \frac{R(s)}{\sum_{s \in \mathcal{S}} R(s)}$$

is not always possible.

In contrast, a corollary of Lemma 2 implies that the intersection of 2 consistent RNNs would also have a finitely bounded sum. This implies that there needs to be an RNN that can be used to represent a linear scaling of the weighted intersection. However applying this linear scale is not trivial due to he compounding product overtime and the non-linearity of the softmax and σ activation functions. In generanl thought the results of Kilian and Siegelmann (1993) and Seigelmann and Sontag (1995) on the general power of RNNs leads us the following conjecture.

Conjecture 1. *Given 2 consistent RNNs, there exists a third RNN which represents the consistent weighted intersection. This would imply that consistent RNNs are closed under intersection*

While we believe that the consistent RNN is closed under intersection, the result of Theorem 3.4 Chen et al. (2017) would imply that finding the scaling factor is not possible, otherwise implying a contradiction on the decidability of consistency. This leads us to propose the following:

Conjecture 2. *There does not exist a deterministic method to construct the weighted intersection of 2 consistent RNNs.*

5 Conclusions

We have shown that PFSAs are closed under intersection, their structure and behavior allow for the definitive manipulation and exploration. The decidability and boundedness over many of the preliminary properties of PFSAs allow us to observe, comment, and deduct more complex functionality of the structure. In comparison,RNNs sacrifice simplicity and boundedness in return for being computationally more powerful. As a result we have demonstrated that the the very notion of intersection is highly dependant on the definition we present, and the liberties in the architecture we allow. Furthermore, there seem to exist clear boundaries to the determinism of closure of RNNs under intersection.

It is believed that the result of Theorem 1 on PFSA closure under intersection is novel, and that the intermediate Lemmas can prove to be functionally useful in the continued research and use of WFSAs and PFSAs are probabilistic language recognizers.

The contributions on the intersection of RNNs, has spoken to the continued complexity in understanding fundamental properties. However with the conjectures, and continued contributions being made in the field we believe a more complete proof on the decidability of these properties is available.

6 Future Work

In general there is a strong desire to better describe the computability class of RNNs or categorize RNNs into relevant sub-categories that would make fundamental properties more approachable to discovery. In particular recent work by Weiss et al. (2017) and Frosst and Hinton (2017) have demonstrated the ability to distill information from neural networks into finite state graphs and transitions. It seems likely that there are fundamental properties of the neural structures that makes it possible to distill these graphs from them. An evaluation of these properties may yield a deeper understanding on the neural networks that we use to represent languages.

Acknowledgments

I would like to personally thank Timofey Peshin for his continuous delibiration and review of the work presented. I would also like to thank Kevin Knight for his input and guidance throughout the course of work. Without this help I would have been sorely lost. FInally a plethora of gratitude to Olga White, who helped make me English good.

References

Yining Chen, Sorcha Gilroy, Kevin Knight, and Jonathan May. 2017. Recurrent Neural Networks as Weighted Language Recognizers http://arxiv.org/abs/1711.05408.

Nicholas Frosst and Geoffrey Hinton. 2017. Distilling a Neural Network Into a Soft Decision Tree http://arxiv.org/abs/1711.09784.

J Kilian and H Siegelmann. 1993. On the power of sigmoid neural networks. *Proc. 6th Annu. Conf. on Comput. Learning Theory* pages 137–143.

Hava T. Seigelmann and Eduardo D. Sontag. 1995. On the Computational Power of Neural Nets. https://doi.org/10.1162/089976600300014827.

Gail Weiss, Yoav Goldberg, and Eran Yahav. 2017. Extracting Automata from Recurrent Neural Networks Using Queries and Counterexamples http://arxiv.org/abs/1711.09576.

Supplemental Material

Recall that in §3 we provided a proof that there must exist a PFSA that represents the desired property. We noticed then that the sum through particular sub-graph at a node can be represented as a weighted sum of the sum of through the subgraphs of all nodes reachable.

With this in mind if we allow x_i to represent the sum of the subgraph starting at $q_i \notin E$ and

$$x = \begin{bmatrix} x_1 \\ \vdots \\ x_m \end{bmatrix}$$

represent a column vector of the sums of non-terminal states.

We can show that,

$$x_i = \sum_{\forall (t,w) \in T_i | q_i \times t \notin E} w x_{(q_i \times t)} + \sum_{\forall (t,w) \in T_i | q_i \times t \in E} w$$

$$\therefore x_i - \sum_{\forall (t,w) \in T_i | q_i \times t \notin E} w x_{(q_i \times t)} = \sum_{\forall (t,w) \in T_i | q_i \times t \in E} w$$

This relationship in conjunction with Theorem 1 implies that there at least one solution and that there is a linear system of equations which is solvable.

Algorithm 1 shows an algorithm that constructs and solves this system of linear equations.

Algorithm 1 Normalization of Edge Weights for Finitely-Bounded WFSAs

Let $\hat{Q}$ represent the non-terminal state and $|\hat{Q}|$ represents its size

```
function NORMALIZE_GRAPH_WEIGHTS($\hat{Q}, E, T$)
    $\Omega \leftarrow \mathbf{0}^{|\hat{Q}| \times |\hat{Q}|}$
    $\mu \leftarrow \mathbf{0}^{|\hat{Q}|}$
    for $q_i \in Q$ do
        for $(t, w) \in T_q$ do
            $q_j \leftarrow (q_i \times t)$
            if $q_j \in E$ then
                $\mu_i \leftarrow \mu_i + w$
            else
                $\Omega_{i,j} \leftarrow \Omega_{i,j} + w$
            end if
        end for
    end for
    $K \leftarrow (I - \Omega)^{-1} \mu$    ▷ This solves the set of linear equations
    $\hat{T} \leftarrow \{\emptyset\}$
    for $q_i \in Q$ do
        $\hat{T}_i \leftarrow \{\emptyset\}$
        for $(t, w) \in T_q$ do
            $q_j \leftarrow (q_i \times t)$
            $\hat{w} = \frac{wK_j}{K_i}$
            $\hat{T}_i \leftarrow \hat{T}_i \cup \{(t, \hat{w})\}$
        end for
        $\hat{T} \leftarrow \hat{T} \cup \hat{T}_i$
    end for
    return $\hat{T}$
end function
```