# A Gang of Adversarial Bandits

**Mark Herbster*, Stephen Pasteris***
Department of Computer Science
University College London
London WC1E 6BT
{m.herbster,s.pasteris}@cs.ucl.ac.uk

**Fabio Vitale**
University of Lille
59653 Villeneuve d'Ascq CEDEX
France
fabio.vitale2@univ-lille.fr

**Massimiliano Pontil**
CSML, Istituto Italiano di Tecnologia and
Department of Computer Science
University College London
massimiliano.pontil@iit.it

## Abstract

We consider running multiple instances of multi-armed bandit (MAB) problems in parallel. A main motivation for this study are online recommendation systems, in which each of $N$ users is associated with a MAB problem and the goal is to exploit users' similarity in order to learn users' preferences to $K$ items more efficiently. We consider the adversarial MAB setting, whereby an adversary is free to choose which user and which loss to present to the learner during the learning process. Users are in a social network and the learner is aided by a-priori knowledge of the strengths of the social links between all pairs of users. It is assumed that if the social link between two users is strong then they tend to share the same action. The regret is measured relative to an arbitrary function which maps users to actions. The smoothness of the function is captured by a resistance-based dispersion measure $\Psi$. We present two learning algorithms, GABA-I and GABA-II which exploit the network structure to bias towards functions of low $\Psi$ values. We show that GABA-I has an expected regret bound of $\mathcal{O}(\sqrt{\ln(NK/\Psi)\Psi KT})$ and per-trial time complexity of $\mathcal{O}(K\ln(N))$, whilst GABA-II has a weaker $\mathcal{O}(\sqrt{\ln(N/\Psi)\ln(NK/\Psi)\Psi KT})$ regret, but a better $\mathcal{O}(\ln(K)\ln(N))$ per-trial time complexity. We highlight improvements of both algorithms over running independent standard MABs across users.

## 1 Introduction

During the last decade multi-armed bandits (MAB) have received a great deal of attention in machine learning and related fields, due to their wide practical and theoretical importance. The central problem is to design a decision strategy whereby a learner explores sequentially the environment in order to find the best item (arm) within a prescribed set. At each step in the exploration the learner chooses an arm, after which feedback (typically a loss or reward corresponding to the selected arm) is observed from the environment. Then the next decision is made by the learner based on past interactions, and the process repeats. The goal is to design efficient exploration strategies which incur a small cumulative loss in comparison to the cumulative loss that would have been obtained by always

* Equal contribution.

35th Conference on Neural Information Processing Systems (NeurIPS 2021).

selecting the best arm in hindsight. Applications of MAB are numerous, including recommender systems [1], clinical trials [2], and adaptive routing [3], among others.

In this paper we study the problem in which the learner is facing several MAB problems that are related according to a prescribed interaction graph. A main motivation behind this problem are online recommendation systems, whereby each of several users is associated with a MAB problem (task), where the arms correspond to a finite set of products, and the graph represents a social network among users. The goal is to exploit users' similarity in order to improve the efficiency of learning users' preferences via online exploration of products. In the standard full information setting, a lot of work has been done showing that techniques from multitask learning are effective in reducing the amount of data needed to learn each of the individual tasks, both in the statistical and adversarial settings, see [4, 5, 6, 7, 8, 9, 10, 11] and references therein. Graphs have been used to model task relationships, with different tasks' parameters encouraged to be close according to the graph topology. In contrast, multitask learning in the bandit setting is much less explored.

The algorithms that we present exploit the *network homophily* principle [12] which formulates that users that are connected in the network have similar preferences, that is, they tend to share preferred recommendations. We will show that our algorithms exploit graph structure and enjoy potentially much smaller regret bounds than the cumulative regret of standard MAB run independently on each user. Since the original graph may be dense, we exploit a randomized sparsification technique to develop fast prediction algorithms. Our approach builds upon previous work on online learning over graphs [13, 14] to generate a perfect full oriented binary tree, whose leaves are in one-to-one correspondence with the nodes of the original graph. This construction approximately preserves the relevant graph properties in expectation, and provides the starting point for designing our efficient algorithms. A further ingredient in our algorithm is provided by the method of *specialists* [15, 16]. Our learning strategies combine the above ingredients to devise efficient online algorithms under partial feedback.

**Contributions.** We introduce two Gang of Adversarial BAndit algorithms, GABA-I and GABA-II that learn jointly MAB models for $N$ users over $K$ possible actions. Both algorithms are designed to exploit network structure while being (extremely) computationally efficient. We derive expected (over the algorithms' randomizations) regret bounds. The bounds scale with the *dispersion* measure $\Psi \in [1, N]$ of the best actions over the graph. For GABA-I the bound[1] is of order of $\mathcal{O}(\sqrt{\ln(NK/\Psi)\Psi KT})$, where $T$ is the number of trials, and has a per-trial time complexity of $\mathcal{O}(K\ln(N))$. On the other hand GABA-II has a weaker expected regret bound of $\mathcal{O}(\sqrt{\ln(N/\Psi)\ln(NK/\Psi)\Psi KT})$ but is faster, having a per-trial time complexity of $\mathcal{O}(\ln(K)\ln(N))$. Thus the GABA-I algorithm improves on algorithms that treat each user independently, as in the best case the regret improves from $\mathcal{O}(\sqrt{N})$ to $\mathcal{O}(\sqrt{\ln N})$ and in the worst case the regret degrades by at most a constant factor. GABA-II has slightly weaker regret bounds; however, it is more computationally efficient.

**Outline of Main Results.** The social network graph $\mathcal{G}$ is determined by a set of undirected links between users $\{\omega_{u,v}\}_{u<v}^{N}$ where $\omega_{u,v} \in [0, \infty)$ indicates the magnitude of the link between user $u$ and $v$. For all $t \in [T]$ we have a user $u_t \in [N]$ and a loss vector $\boldsymbol{\ell}_t \in [0,1]^K$ which are selected by *Nature* before learning begins and are unknown to *Learner*; i.e., Nature is a *deterministic oblivious adversary* (see e.g., [17, Section 5.1]). Learning then proceeds in trials $t = 1, 2, \ldots, T$. On trial $t$:

1. Nature reveals user $u_t \in [N]$ to Learner,
2. Learner selects action $a_t \in [K]$,
3. Nature reveals loss $\ell_{t,a_t} \in [0,1]$ to Learner.

Before reflecting on the $N$-user case we review the well-known results for the single user ($N = 1$). The seminal EXP3 algorithm [18] obtains the following *(uniform)* regret bound[2] ,

$$\mathbb{E}\Big[\sum_{t\in[T]} \ell_{t,a_t}\Big] - \min_{a\in[K]} \sum_{t\in[T]} \ell_{t,a} \in \mathcal{O}\left(\sqrt{K\ln(K)T}\right) , \tag{1}$$

[1]The bounds of GABA-I and GABA-II however depend on oracular knowledge of optimal tuning parameters. We discuss this as well as a means of lessening this dependency following Corollary 5.

[2]An algorithm was given in [19] that removed the $\ln K$ term from the regret.

where the expectation is with respect to the internal randomization of the EXP3 algorithm. In the $N$-user setting, by running a copy of EXP3 independently for each user, we may obtain a uniform regret bound of (see e.g., [20])

$$\mathbb{E}\Big[\sum_{t\in[T]} \ell_{t,a_t}\Big] - \min_{y:[N]\to[K]} \sum_{t\in[T]} \ell_{t,y(u_t)} \in \mathcal{O}\left(\sqrt{K\ln(K)NT}\right) , \tag{2}$$

i.e., for each user $u$ the best action is $y(u)$ and we now pay an additional constant factor of $\sqrt{N}$ in our regret. In this work we exploit the social network structure to prove a *non-uniform* regret bound for the GABA-I algorithm (see Corollary 4) of

$$R(y) := \mathbb{E}\Big[\sum_{t\in[T]} \ell_{t,a_t}\Big] - \sum_{t\in[T]} \ell_{t,y(u_t)} \in \mathcal{O}\left(\sqrt{K\ln\left(\frac{KN}{\Psi(y)}\right)\Psi(y)T}\right) , \tag{3}$$

for any mapping of users to actions $y : [N] \to [K]$. The non-uniform regret now depends on $\Psi(y) \in [1, N]$ (see (5)) which measures dispersion of users' 'best' actions across the network. Thus, by taking network structure into account, we may upper bound the scaling in the regret with respect to the number of users by $\mathcal{O}(\sqrt{\ln(\frac{eN}{\Psi(y)})\Psi(y)})$. When the best action across the network is nearly uniform then the dispersion $\Psi(y) \in \mathcal{O}(1)$, in contrast when the dispersion is maximal then $\Psi(y) = N$ thus in the best case the regret with respect to the number of users improves from $\mathcal{O}(\sqrt{N})$ to $\mathcal{O}(\sqrt{\ln N})$ and in the worst case the regret only increases by a constant factor. The first algorithm GABA-I obtains the regret (3) while requiring $\mathcal{O}(K \ln N)$ time to predict and update. The second algorithm GABA-II's regret (see Corollary 5) is larger by a $\mathcal{O}(\sqrt{\ln N/\Psi(y)})$ factor but now prediction is at an even faster $\mathcal{O}(\ln(K)\ln(N))$ time per trial, that is, prediction time improves exponentially with respect to the cardinality of the action set $[K]$. Thus both algorithms support very large user networks and the second algorithm allows efficient prediction with very large action sets.

**Related Work.** We mention here some of the key papers that are more closely related to ours and refer the reader to the technical appendices for an expanded literature review. There has been much work in the heterogenous multi-user setting for linear-stochastic bandits. Out of these works, those closest to us are when the users are in a known social network and it is assumed that neighbouring users respond to context vectors in a similar way [21, 22, 23, 24, 25, 26] but as far as we are aware no works on this model have so far been done in the adversarial setting. Other works on this topic include those in which it is assumed that there is an unknown clustering of the users, where users in the same cluster are homogenous [27, 28, 29, 30, 31, 32, 33, 34, 35]; as well as other models [36, 37, 38, 39, 40, 41, 42, 43]. There are also works on networked, homogenous multi-user bandit models with limited communication between users [44, 45, 46, 47, 48, 49, 50]. Related to the multi-user setting are works on transfer learning and meta-learning with linear-stochastic bandits [51, 52]. Whilst our work assumes a known network structure over the users, there is a wide literature on bandit problems in which the actions are structured in a network and it is assumed that neighbouring actions give similar losses [53, 54, 55, 56, 57, 58, 59, 60], as well as other networked-action models [61, 62, 63]. In addition to the seminal paper on adversarial bandits [18], our work utilises ideas from several different papers [13, 14, 15, 16, 64, 65].

**Notation.** Given a set $X$ we define $2^X$ to be its power-set, that is: the set of all subsets of $X$. For any positive integer $m$, we define $[m] := \{1, 2, \ldots, m\}$. For any predicate PRED, $[\![\text{PRED}]\!] := 1$ if PRED is true and equals 0 otherwise. Given vectors $\boldsymbol{x}, \boldsymbol{x'} \in \mathbb{R}^K$ we define $\boldsymbol{x} \cdot \boldsymbol{x'}$ to be their inner product (i.e., equal to $\sum_{i\in[K]} x_i x_{i'}$) and we define $\boldsymbol{x} \odot \boldsymbol{x'}$ to be their component-wise product (i.e., $(x \odot x')_i := x_i x'_i$ for all $i \in [K]$). We define '$\mathbf{1}$' to be the $K$-dimensional vector in which each component is equal to 1 . Given a full oriented binary tree $\mathcal{B}$ we denote the set of its vertices also by $\mathcal{B}$. Given a non-leaf vertex $n \in \mathcal{B}$ let $\triangleleft(n)$ and $\triangleright(n)$ be its left child and right child respectively. Given a non-root vertex $n \in \mathcal{B}$ let $\uparrow(n)$ be its parent. Given a vertex $n \in \mathcal{B}$ let $\Uparrow(n)$ and $\Downarrow(n)$ be its set of ancestors and leaf-descendants (i.e. descendants which are leaves) respectively. Given a vertex $n \in \mathcal{B}$ we define $\blacktriangleleft(n)$ and $\blacktriangleright(n)$ as the left-most and right-most descendants (which are leaves) of $n$ respectively. Finally, we denote the user graph by $\mathcal{G}$, which is an undirected connected graph with edge weights $\{\omega_{u,v} : 1 \leq u < v \leq N\}$. For convenience we assume $N$ is a power of two.[3]

[3]This assumption does not limit our results because to run our algorithms one can always add dummy vertices without altering input weights, so as to force $N$ to be a power of two.

# 2 Modeling a Social Network as a Resistive Network

In this section we introduce the tools necessary to formalize our complexity measures, as well as the ones to implement our algorithms.

## 2.1 Conceptual Tools

To minimize the incurred loss, Learner can exploit the similarity between any pair of users defined by the weights $\omega_{u,v}$ of user graph edges for all $u, v \in [N]$. The function $y : [N] \to [K]$ is completely unknown to Learner, and can be viewed as labeling each user with its best/favorite action. Within this context, our homophilic bias can be stated as follows: users strongly connected w.r.t. the link weights $\omega$, tend to be associated with the same label.

The complexity measure used for this problem is the *robustified resistance weighted cutsize* $\Psi(y)$, which we now define formally. Within the graph-based learning context, the *cutsize* is defined as the number of edges connecting users with different labels, i.e., $\sum_{u<v} [\![\omega_{u,v} \neq 0]\!] [\![y(u) \neq y(v)]\!]$, and the *weighted cutsize* is defined as the sum of the edge weights $\omega_{u,v}$ over all pairs of users $u$ and $v$ having different labels, i.e., $\sum_{u<v} \omega_{u,v} [\![y(u) \neq y(v)]\!]$ [14]. The *effective resistance* between two given nodes $u$ and $v$ of a graph is a commonly used measure that expresses the degree of the connection strength between $u$ and $v$ (see, e.g., [66]). More precisely, viewing the graph as an electrical circuit, where each edge weight $\omega_{u,v}$ corresponds to a $\frac{1}{\omega_{u,v}}$ *resistor*, the effective resistance between $u$ and $v$ is the power required to hold between them a unit voltage difference for a unit time. Informally, the more there are paths between two nodes $u$ and $v$ that are short, edge-disjoint and formed by edges with large weights, the lower is $r(u, v)$ because the amount of flow between the two considered nodes is larger. A formal definition of effective resistance $r(u, v)$ between users $u$ and $v$ is

$$r(u, v) := \frac{1}{\min_{\boldsymbol{x} \in \mathbb{R}^N} \{\sum_{i<j}^{N} \omega_{i,j} (x_i - x_j)^2 : x_u - x_v = 1\}} .$$

Interestingly enough, for all $u, v \in [N]$, $r(u, v)$ is exactly equal to the probability that the edge $\{u, v\}$ is included in a uniformly generated random spanning tree of the given user graph $\mathcal{G}$ (see, e.g., [66]).

The *resistance weighted cutsize* $\Phi(y)$ [67] is the weighted sum of the effective resistances $r(u, v)$ between any two nodes $u$ and $v$ with different labels. i.e.,

$$\Phi(y) := \sum_{u<v}^{N} \omega_{u,v} r(u, v) [\![y(u) \neq y(v)]\!] \tag{4}$$

and then its robustifcation is defined as

$$\Psi(y) := 1 + \min_{z:[N]\to[K]} \Big( \Phi(z) + \sum_{u\in[N]} [\![z(u) \neq y(u)]\!] \Big) . \tag{5}$$

The first quantity (4) can be viewed as a dispersion measure based on the above mentioned homophilic tendency. It has several advantages compared to the weighted cutsize in measuring the degree of homophily violation [67]. The most significant property is that it is *locally* density-dependent because the contribution to $\Phi(y)$ of each edge $(u, v)$ such that $y(u) \neq y(v)$ is inversely proportional to how strongly $u$ and $v$ are connected in their user graph local area. Indeed, because of the effective resistance, the potential contribution to $\Phi(y)$ of the edges in dense areas is smaller than the ones of the edges in sparse areas. In fact if the graph is well-clustered i.e., it can be partitioned into dense clusters (many intra-cluster edges) and fewer inter-cluster edges and the labeling $y$ respects these clusters then in many cases $\Phi(y) \ll N$. As an archetypical instance consider the following proposition where the clusters are represented by cliques.

**Proposition 1.** *Consider an unweighted graph $\mathcal{G}$ partitioned into $G$ clusters and a labeling function $y(\cdot)$, where each cluster is an $n$-clique and, if $u, v$ are vertices in the cluster, then $y(u) = y(v)$. For any pair of such clusters $C, C' \subset \mathcal{G}$, suppose that there are $\frac{n-1}{G}$ edges connecting the nodes of $C$ with the nodes of $C'$. Then we have $\Phi(y) \in \mathcal{O}(G)$.*

**CONSTRUCTBST-$\mathcal{C}$** (User graph: $\mathcal{G}$)

1. Sample a uniform random spanning tree $\mathcal{T}$ from the user graph.
2. Perform a depth-first visit of $\mathcal{T}$ to provide an order of the users. Without loss of generality assume that, for all $u \in [N]$ , we have that user $u$ is the $u$-th vertex visited.
3. Construct a perfect full oriented binary tree $\mathcal{C}$ of depth $h := \log_2(N)$ whose $u$-th leftmost leaf of its graphical representation is user $u$ .[a].

[a] In this context, by *oriented* we mean that the leaves of $\mathcal{C}$ are numbered sequentially from the leftmost to the rightmost one so that, for each internal vertex of $\mathcal{C}$, both its left and right subtree contain subsets of leaves uniquely determined by the depth-first visit of $\mathcal{T}$.

Figure 1: Binary Support Tree Construction Algorithm

Thus in this archetypical case our regret bounds now scale strongly with the number of clusters of users $G$ (see (3)) rather than with the number of users $N$ (comparing to the baseline (2)).

The second quantity (5) is an extension of $\Phi(y)$ to deal with adversarial label perturbation, viz., capturing the regularity of all labelings $y$ such that $\Phi(y)$ can be dramatically reduced by simply changing the labels of a relatively small number of users. To give an insight into the advantages of $\Psi(y)$ w.r.t. $\Phi(y)$ regarding its noise-tolerance property, consider an input star graph with all edge weights equal to 1 and where all vertex labels are equal except for the one of the central node $u$. It is natural to consider this labeling regular w.r.t. our bias, because it is sufficient to change only $y(u)$ to obtain a cutsize equal to 0. This is precisely the labeling property that is captured by $\Psi(y)$, which is equal to the minimum, over all labelings $z$, of the sum of $\Phi(z)$ and the number of vertices for which $y$ and $z$ differ (plus 1). In this case we have therefore $\Psi(y) = 1 + \Phi(y^*) + 1 = 2$, where $y^*$ is the labeling obtained by changing $y(u)$ to make it equal to all other labels, so that $\Phi(y^*) = 0$, whereas $\Phi(y) = N - 1$ .

### 2.2 An Embedding to Enable Fast Computation

A uniformly generated random spanning tree (RST) is defined as a spanning tree selected with a probability proportional to the product of the weights of all its edges (see, e.g., [66]). It represents a fundamental tool in several mathematical fields, e.g., combinatorial geometry, algebraic graph theory, stationary Markov chains [68], and can be viewed as a way to summarize the topological information of the input network. When the input graph is weighted as in our case, it can be generated in time almost linear in the number of edges [69, 70].

In a preliminary phase, our algorithms operate as follows (see Fig. 1). A RST $\mathcal{T}$ of the input social network is drawn (step 1). Thus, an order of the $N$ users is determined through a depth-first visit of $\mathcal{T}$ (step 2). From here on we assume, without loss of generality, that user $u \in [N]$ is the $u$-th vertex visited. This step is necessary to make the algorithms noise-tolerant, and is strictly related to the improvement of the complexity measure $\Psi(y)$ over $\Phi(y)$. Finally, a full perfect binary tree, called the Binary Support Tree (BST), and having the users, ordered from left to right, as leaves (step 3) is constructed. The BST forms the geometry that underlies the data-structures of our algorithms.

We conclude this section by showing a result which will be useful in the analysis of our algorithms, and stems directly *only* from the user order determined by the depth-first visit of $\mathcal{T}$. If we consider the line graph $\mathcal{L}$ connecting the users $u$ with $u + 1$ for all $u \in [N - 1]$, we have that, as stated in the following theorem, the cutsize of $\mathcal{L}$ is at most twice the robustified resistance weighted cutsize of the input user graph. This result can be viewed as the multi-class extension of part 2 of Theorem 6 in [67].

**Lemma 2** ([67, Theorem 6])**.** *For any given input user graph, we have*

$$\mathbb{E}\left[\sum_{u\in[N-1]} [\![y(u) \neq y(u+1)]\!]\right] \leq 2\Psi(y) ,$$

*where the expectation is over the draw of the uniform random spanning tree $\mathcal{T}$ .*

**SPECIALISTEXP (**Learning rate $\eta > 0$; Distribution $w_1 : \mathbb{S} \to [0,1]$ s.t. $\sum_{s\in\mathbb{S}} w_1(s) = 1$.**)**

**For** $t = 1, \ldots, T$ **do**

1. $\forall a \in [K], \quad p_{t,a} \leftarrow \sum_{s\in\mathbb{S}:s(u_t)=a} w_t(s)$;
2. **Predict** $a_t$ by drawing from $[K]$ with probability $\mathbb{P}[a_t = a] := p_{t,a}/\|\boldsymbol{p}_t\|_1$;
3. **Receive** $\ell_{t,a_t}$
4. $\lambda_t \leftarrow \exp(-\eta\ell_{t,a_t}\|\boldsymbol{p}_t\|_1/p_{t,a_t}); \quad z_t \leftarrow \|\boldsymbol{p}_t\|_1/(\|\boldsymbol{p}_t\|_1 - (1-\lambda_t)p_{t,a_t})$;
5. $\forall s \in \mathbb{S}$:

$$w_{t+1}(s) \leftarrow \begin{cases} w_t(s) & s(u_t) = \square \\ w_t(s)z_t & s(u_t) \neq a_t \\ w_t(s)z_t\lambda_t & s(u_t) = a_t \end{cases}$$

Figure 2: SPECIALISTEXP Algorithm

# 3 Predicting with Specialists

We build on the *prediction with expert advice framework* [71, 72, 73, 74], specifically that with bandit feedback: pioneered by the EXP4 algorithm [18]. This type of online algorithm maintains a distribution over a set of predictors ("experts"). After the predictors predict they incur a loss and the distribution is updated accordingly. Although, except in special cases, this procedure does not have a natural Bayesian interpretation, probabilistic methods still may be transferred into the expert advice framework. In particular we will exploit an analogue of message-passing as used in graphical models [75] to predict very efficiently over exponentially-sized sets of predictors. Broadly speaking we would like build a graphical model that is isomorphic to the user graph $\mathcal{G}$. However it is well-known that exact prediction with graphical models that contain cycles is NP-hard [76]. Thus a benefit of the embedding to a BST (see Section 2.2) is that it enables fast and exact computation as the graph is now cycle-free and Lemma 2 ensures that the embedding only modestly increases our regret bounds. Surprisingly, we improve in terms of computation over standard message passing techniques, i.e., if we embedded to a "line" graph we would require $\mathcal{O}(KN)$ time to predict [75] per trial or using the method of [77] $\mathcal{O}(K^3 \log N)$ time. However, we will require only $\mathcal{O}(K \log N)$ and $\mathcal{O}(\log K \log N)$ for the GABA-I and GABA-II algorithms respectively (see Figures 3 and 4). To accomplish this technically we adapt the method of *specialists* [15, 16].

A specialist is a prediction function $s : [N] \to \{1, 2, \ldots, K, \square\}$ from a context space to an extended output space with *abstentions*. For us the context space is just the set of users $[N]$; and the extended output space is $\{1, 2, \ldots, K, \square\}$ where $[K]$ corresponds to predicted actions, but '$\square$' indicates that the specialist abstains from predicting an action. Thus a specialist *specializes* its prediction to part of the context space. We denote the set of all specialists as $\mathbb{S} := \{1, \ldots, K, \square\}^{[N]}$. As a single specialist only predicts over part of the context space, we need a set of specialists $\mathcal{S} \subseteq \mathbb{S}$ if we wish to define a function that predicts an *action* for every context. A specialist set $\mathcal{S} \subseteq \mathbb{S}$ is *well-formed* if for each $u \in [N]$ there exists a unique specialist $s \in \mathcal{S}$ such that $s(u) \in [K]$. For such a user $u$ and specialist $s$ we then define $\mathcal{S}^\dagger(u) := s(u)$ so that $\mathcal{S}^\dagger$ is a function from $[N]$ into $[K]$. Finally a specialist *model* is defined by giving a distribution $w_1 : \mathbb{S} \to [0,1]$ s.t. $\sum_{s\in\mathbb{S}} w_1(s) = 1$. To predict with specialists we adapt [15] to the EXP3/4 [18] setting giving the SPECIALISTEXP algorithm (see Figure 2). We then bound the regret by combining the analysis of [15, 18] into the following theorem.

**Theorem 3.** *The expected regret of* SPECIALISTEXP *with initial specialist distribution* $w_1 : \mathbb{S} \to [0,1]$ *and learning rate* $\eta > 0$ *is bounded above by*

$$\mathbb{E}\left[\sum_{t\in[T]} \ell_{t,a_t} - \ell_{t,\mathcal{S}^\dagger(u_t)}\right] \leq \frac{1}{\eta}\sum_{s\in\mathcal{S}} \ln\left(\frac{1}{w_1(s)|\mathcal{S}|}\right) + \frac{\eta K T}{2} \tag{6}$$

*for all well-formed specialist sets* $\mathcal{S} \subseteq \mathbb{S}$.

In the following we give the two distributions that define the two specialist models corresponding to GABA-I and GABA-II in (7) and (9), and in the supplementary material we detail how these distributions lead to the regret bounds in Corollaries 4 and 5.

We now give the distribution $w_1(\cdot)$ over $\mathbb{S}$ that defines the GABA-I model. The model has a single parameter $\phi \in (0,1)$ and we give the following helper functions to define the distribution,

$$\begin{aligned}
\mathrm{valid1}(s) &:= [\![\forall u, v \in [N] : s(u) = s(v) \text{ or } s(u) = \square \text{ or } s(v) = \square]\!] \\
\mathrm{cut}(s) &:= \sum_{u \in [N-1]} [\![s(u) \neq s(u+1)]\!] \\
\mathrm{startfactor}(s) &:= \frac{K-1}{K}[\![s(1) \neq \square]\!] + \frac{1}{K}[\![s(1) = \square]\!].
\end{aligned}$$

The function $\mathrm{valid1}(\cdot)$ determines the support of $w_1(\cdot)$ which are the specialists that predict a unique action or abstain, hence the cardinality of the support of $w_1(\cdot)$ is $K \times (2^N - 1) + 1$. The remaining two functions quantitatively determine probability mass of a specialist as:

$$w_1(s) := \mathrm{valid1}(s) \times \frac{1}{K} \times \mathrm{startfactor}(s) \times (1-\phi)^{N-1-\mathrm{cut}(s)} \phi^{\mathrm{cut}(s)} \quad (\forall s \in \mathbb{S}). \tag{7}$$

We note that this specialist selection is similar to that of the Markov circadian specialists in [16] except that the nodes of the Markov chain are now users instead of trials.

**Corollary 4.** *The expected regret of* SPECIALISTEXP *with distribution $w_1(\cdot)$ as defined by* (7) *with parameter $\phi = 4\Psi(y)/(K(N-1))$, learning rate $\eta = \sqrt{\frac{10\Psi(y)\ln(KN/\Psi(y))}{KT}}$ and with $\Psi(y) \leq (N-1)/4$ is bounded above by:*

$$\mathbb{E}\left[\sum_{t \in [T]} \ell_{t,a_t} - \ell_{t,y(u_t)}\right] \in \mathcal{O}\left(\sqrt{K \ln\left(\frac{KN}{\Psi(y)}\right) \Psi(y) T}\right) \tag{8}$$

*for any mapping of users to actions $y : [N] \to [K]$.*

We now give the distribution $w_1(\cdot)$ over $\mathbb{S}$ that defines the GABA-II model. Whereas for GABA-I the cardinality of the support was exponential in $N$, for GABA-II the cardinality is just $K(2N-1)$. The supported specialists in GABA-II predict a unique action over a contiguous $l, \ldots, r$ and abstain everywhere else, thus they are of the form:

$$s_a^{l,r}(u) := \begin{cases} a & u \in \{l, \ldots, r\} \\ \square & u \notin \{l, \ldots, r\} \end{cases},$$

but not all contiguous segments are supported. The segments supported are those that correspond to the set of all leaf-descendents of a node in the BST (see Section 2.2). As an example if $N = 4$ the supported $(l, r)$ segments are $\{(1,1), (2,2), (3,3), (4,4), (1,2), (3,4), (1,4)\}$. Expressing this algebraically leads to a relatively complex "validity" function

$$\mathrm{valid2}(s) := [\![\exists a \in [K]; l, r \in [N]; i, j \in [\log_2 N] : 1+r-l = 2^i \text{ and } l = 2^i(j-1)+1 \text{ and } s = s_a^{l,r}]\!],$$

and then the distribution is defined as,

$$w_1(s) := \mathrm{valid2}(s) \times \frac{1}{K(2N-1)} \quad (\forall s \in \mathbb{S}). \tag{9}$$

We note that this selection of specialists is a simple multi-action extension of those defined in [65].

**Corollary 5.** *The expected regret of* SPECIALISTEXP *with distribution $w_1(\cdot)$ as defined by* (9) *with learning rate $\eta = \sqrt{\frac{8\Psi(y)\log_2(eN/\Psi(y))\ln(3KN/2\Psi(y))}{KT}}$ and with $\Psi(y) \leq N/2$ is bounded above by:*

$$\mathbb{E}\left[\sum_{t \in [T]} \ell_{t,a_t} - \ell_{t,y(u_t)}\right] \in \mathcal{O}\left(\sqrt{K \ln\left(\frac{N}{\Psi(y)}\right) \ln\left(\frac{KN}{\Psi(y)}\right) \Psi(y) T}\right) \tag{10}$$

*for any mapping of users to actions $y : [N] \to [K]$.*

## 4 The GABA Algorithms

We now introduce the GABA algorithms. Both algorithms are based on the BST $\mathcal{C}$ (see Section 2.2).

### 4.1 GABA-I

Since we have an exponential number of non-zero weight specialists in GABA-I a direct implementation of SPECIALISTEXP would take per-trial time and space exponential in $N$. We now describe how GABA-I implements SPECIALISTEXP, bringing the per-trial time down to $\mathcal{O}(K\ln(N))$ and the space down to $\mathcal{O}(KN)$. The implementation works by, for each action independently, performing online belief propagation [64] over the tree $\mathcal{C}$. We note that each of these $K$ online belief propagations is over two states $\{0,1\}$ and hence takes a per-trial time of only $\mathcal{O}(\ln(N))$. We now detail this procedure:

GABA-I maintains a vector valued function $\boldsymbol{\alpha}_t : \mathcal{C}\times\{0,1\}\times\{0,1\}\to\mathbb{R}^K$ which, for all $i,j\in\{0,1\}$ and $t\in[T]$, has the following properties:

$$\forall u\in[N]\setminus\{u_t\}\,,\quad \boldsymbol{\alpha}_{t+1}(u,i,j)=\boldsymbol{\alpha}_t(u,i,j) \tag{11}$$

and for all internal vertices $n$ of $\mathcal{C}$ we have:

$$\boldsymbol{\alpha}_t(n,i,j)=\sum_{k\in\{0,1\}}\boldsymbol{\alpha}_t(\triangleleft(n),i,k)\odot\boldsymbol{\alpha}_t(\triangleright(n),k,j) \tag{12}$$

On trial $t$ GABA-I computes $\boldsymbol{p}_t$ by sending vector valued messages down the path in $\mathcal{C}$ from the root to $u_t$. Specifically, we construct the left and right message functions $\boldsymbol{\beta}_t^{\Leftarrow},\boldsymbol{\beta}_t^{\Rightarrow} : \Uparrow(u_t)\times\{0,1\}\to\mathbb{R}^K$ as follows. Each (non-root, proper) ancestor $n$ of $u_t$ receives, for $i\in\{0,1\}$, $K$ dimensional vector messages $\boldsymbol{\beta}_t^{\Leftarrow}(\uparrow(n),i)$ and $\boldsymbol{\beta}_t^{\Rightarrow}(\uparrow(n),i)$ from its parent and then constructs its own messages $\boldsymbol{\beta}_t^{\Leftarrow}(n,i)$ from $\boldsymbol{\beta}_t^{\Leftarrow}(\uparrow(n),j)$ and $\boldsymbol{\alpha}_t(\triangleleft(\uparrow(n)),j,i)$ and messages $\boldsymbol{\beta}_t^{\Rightarrow}(n,i)$ from $\boldsymbol{\beta}_t^{\Rightarrow}(\uparrow(n),j)$ and $\boldsymbol{\alpha}_t(\triangleright(\uparrow(n)),i,j)$, for all $i,j\in\{0,1\}$. It then sends these messages to its child that is next on the path to $u_t$. Once $u_t$ has received the messages from its parent it combines them with $\boldsymbol{\alpha}_t(u_t,1,i)$ (for $i\in\{0,1\}$) to create $\boldsymbol{p}_t$.

On the receipt of $\ell_{t,a_t}$ we update the function $\boldsymbol{\alpha}_t$ to $\boldsymbol{\alpha}_{t+1}$ noting that by (11) and (12) we need only modify the values $\boldsymbol{\alpha}_t(n,i,j)$ when $n$ is an ancestor of $u_t$.

### 4.2 GABA-II

For GABA-II we have $\mathcal{O}(K\ln(N))$ non-zero weight specialists that don't abstain on any given trial so a direct implementation of SPECIALISTEXP would take a per-trial time of $\mathcal{O}(K\ln(N))$. We now show how GABA-II implements SPECIALISTEXP, which takes the per-trial time down to $\mathcal{O}(\ln(K)\ln(N))$ whilst maintaining the space complexity of $\mathcal{O}(KN)$.

We first note that SPECIALISTEXP maintains a weight for each specialist. For any vertex $n$ of $\mathcal{C}$ and any action $a$, the weight, on trial $t$, of the specialist that predicts $a$ whenever $u_t$ is its descendant and abstains otherwise, is kept, by GABA-II in the following factored form:

$$\frac{\mu_t(n)\theta_t(n,a)}{K(2N-1)} \tag{13}$$

where $\mu_{t+1}(n):=\mu_t(n)$ whenever $n\notin\Uparrow(u_t)$, and $\theta_{t+1}(n,a):=\theta_t(n,a)$ whenever $n\notin\Uparrow(u_t)$ or $a\neq a_t$.

In addition to the tree $\mathcal{C}$, GABA-II also works with an oriented full binary tree $\mathcal{B}$ whose leaves are the actions (in this overview we assume that the cardinality of the action set is an integer power of two, although this is not required by GABA-II). For any vertex $n$ of $\mathcal{C}$ the function $\theta_t(n,\cdot)$ is extended onto all internal vertices of $\mathcal{B}$ by the following inductive relationship:

$$\theta_t(n,m):=\theta_t(n,\triangleleft(m))+\theta_t(n,\triangleright(m)) \tag{14}$$

To sample the action $a_t$ GABA-II first samples an ancestor $\delta_t$ of $u_t$ with probability $\mathbb{P}[\delta_t=n]\propto\mu_t(n)\theta_t(n,r)$ where $r$ is the root of $\mathcal{B}$. GABA-II then uses the function $\theta_t(\delta_t,\cdot)$ to sample action $a_t$ with probability $\mathbb{P}[a_t=a\mid\delta_t=n]=\theta_t(n,a)/\theta_t(n,r)$ in $\mathcal{O}(\ln(K))$ time. The law of total probability and (13) can then be used to show that $\mathbb{P}[a_t=a]\propto p_{t,a}$ where $p_{t,a}$ is as defined in SPECIALISTEXP.

On the receipt of $\ell_{t,a_t}$ we update the functions $\mu_t$ and $\theta_t$ to $\mu_{t+1}$ and $\theta_{t+1}$ noting that by the equalities between these functions and (14) we need only modify the values $\mu_t(n)$ and $\theta_t(n,m)$ when $n$ is an ancestor of $u_t$ and $m$ is an ancestor of $a_t$.

**GABA-I** (Learning rate : $\eta > 0$; Model parameter: $\phi \in (0,1)$)

0. Construct binary support tree $\mathcal{C}$ via CONSTRUCTBST-$\mathcal{C}$ algorithm (see Figure 1).
1. $\forall$ leaf $n \in \mathcal{C}$, $\forall i, j \in \{0,1\}$, $\quad \boldsymbol{\alpha}_1(n,i,j) \leftarrow [\![i \neq j]\!]\phi\mathbf{1} + [\![i = j]\!](1-\phi)\mathbf{1}$;
2. **For** $d = 1, 2, \ldots, h-1$, $\forall n \in \mathcal{C}$ at depth $h-d$, $\forall i, j \in \{0,1\}$, **do**
   $\boldsymbol{\alpha}_1(n,i,j) \leftarrow \sum_{k\in\{0,1\}} \boldsymbol{\alpha}_1(\triangleleft(n),i,k) \odot \boldsymbol{\alpha}_1(\triangleright(n),k,j)$;

**For** $t = 1, 2 \ldots T$, **do**

3. $\forall d \in [h] \cup \{0\}$ $\quad \nu_{t,d} \leftarrow$ ancestor of $u_t$ at depth $d$ in $\mathcal{C}$;
4. $\forall i \in \{0,1\}$, $\quad \boldsymbol{\beta}_t^{\Leftarrow}(\nu_{t,0},i) \leftarrow (1 + [\![i=0]\!](K-2))\mathbf{1}/K$; $\quad \forall i \in \{0,1\}$, $\quad \boldsymbol{\beta}_t^{\Rightarrow}(\nu_{t,0},i) \leftarrow \mathbf{1}$;
5. **For** $d = 1, 2, \ldots, h$, **do**
   (a) **if** $\nu_{t,d} = \triangleleft(\nu_{t,d-1})$ **then** $\forall i \in \{0,1\}$
      i. $\boldsymbol{\beta}_t^{\Leftarrow}(\nu_{t,d},i) \leftarrow \boldsymbol{\beta}_t^{\Leftarrow}(\nu_{t,d-1},i)$;
      ii. $\boldsymbol{\beta}_t^{\Rightarrow}(\nu_{t,d},i) \leftarrow \sum_{j\in\{0,1\}} \boldsymbol{\alpha}_t(\triangleright(\nu_{t,d-1}),i,j) \odot \boldsymbol{\beta}_t^{\Rightarrow}(\nu_{t,d-1},j)$;
   (b) **if** $\nu_{t,d} = \triangleright(\nu_{t,d-1})$ **then** $\forall i \in \{0,1\}$
      i. $\boldsymbol{\beta}_t^{\Rightarrow}(\nu_{t,d},i) \leftarrow \boldsymbol{\beta}_t^{\Rightarrow}(\nu_{t,d-1},i)$;
      ii. $\boldsymbol{\beta}_t^{\Leftarrow}(\nu_{t,d},i) \leftarrow \sum_{j\in\{0,1\}} \boldsymbol{\beta}_t^{\Leftarrow}(\nu_{t,d-1},j) \odot \boldsymbol{\alpha}_t(\triangleleft(\nu_{t,d-1}),j,i)$;
6. $\bar{\boldsymbol{p}}_t \leftarrow (1/K)\sum_{i\in\{0,1\}} \boldsymbol{\beta}_t^{\Leftarrow}(\nu_{t,h},1) \odot \boldsymbol{\alpha}_t(\nu_{t,h},1,i) \odot \boldsymbol{\beta}_t^{\Rightarrow}(\nu_{t,h},i)$;
7. **Predict** $a_t \in [K]$ with probability $\mathbb{P}[a_t = a] = \bar{p}_{t,a}/\|\bar{\boldsymbol{p}}_t\|_1$;
8. **Receive** $\ell_{t,a_t}$
9. $\forall a \in [K]$, $c_{t,a} \leftarrow \exp(-\eta[\![a = a_t]\!]\ell_{t,a_t}\|\bar{\boldsymbol{p}}_t\|_1/\bar{p}_{t,a})$; $\boldsymbol{\pi}^t \leftarrow (\|\bar{\boldsymbol{p}}_t\|_1\boldsymbol{c}_t)/(\bar{\boldsymbol{p}}_t \cdot \boldsymbol{c}_t)$;
10. $\forall i \in \{0,1\}$, $\quad \boldsymbol{\alpha}_{t+1}(\nu_{t,h},1,i) \leftarrow \boldsymbol{\pi}^t \odot \boldsymbol{\alpha}_t(\nu_{t,h},1,i)$;
11. $\forall i \in \{0,1\}$, $\quad \boldsymbol{\alpha}_{t+1}(\nu_{t,h},0,i) \leftarrow \boldsymbol{\alpha}_t(\nu_{t,h},0,i)$;
12. $\forall n \in \mathcal{C} \setminus \{\nu_{t,d} \mid d \in [h] \cup \{0\}\}$, $\forall i,j \in \{0,1\}$, $\quad \boldsymbol{\alpha}_{t+1}(n,i,j) \leftarrow \boldsymbol{\alpha}_t(n,i,j)$;
13. **For** $d = 1, 2, \ldots, h-1$, **do** $\forall i, j \in \{0,1\}$
    $\boldsymbol{\alpha}_{t+1}(\nu_{t,(h-d)},i,j) \leftarrow \sum_{k\in\{0,1\}} \boldsymbol{\alpha}_{t+1}(\triangleleft(\nu_{t,(h-d)}),i,k) \odot \boldsymbol{\alpha}_{t+1}(\triangleright(\nu_{t,(h-d)}),k,j)$;

Figure 3: GABA-I Algorithm

**GABA-II** (Learning rate: $\eta > 0$)

0. Construct binary support tree $\mathcal{C}$ via CONSTRUCTBST-$\mathcal{C}$ algorithm (see Figure 1).
1. Construct a full perfect oriented binary tree $\mathcal{B}$ with height $g := \lceil \log_2(K) \rceil$, whose first $K$ leaves represent the actions $[K]$; $\quad$ Set $r$ to be the root of $\mathcal{B}$;
2. $\forall$ vertex $n \in \mathcal{C}$:
   (a) $\mu_1(n) \leftarrow 1$; $\quad \forall$ leaf $m \in \mathcal{B}$, $\quad$ **if** $m \in [K]$ **then** $\theta_1(n,m) \leftarrow 1$; **else** $\theta_1(n,m) \leftarrow 0$;
   (b) $\forall d \in \{1,2,\ldots,g\}$, $\forall m \in \mathcal{B}$ at depth $g-d$, $\quad \theta_1(n,m) := \theta_1(n,\triangleleft(m)) + \theta_1(n,\triangleright(m))$;

**For** $t = 1, 2 \ldots T$, **do**

3. Draw $\delta_t$ from $\Uparrow(u_t)$ with prob. $\mathbb{P}[\delta_t = n] \propto \mu_t(n)\theta_t(n,r)$; $\quad \zeta_{t,0} \leftarrow r$;
4. **For** $d = 0, \ldots, g-1$: draw $\zeta_{t,d+1}$ from $\{\triangleleft(\zeta_{t,d}), \triangleright(\zeta_{t,d})\}$ with prob. $\mathbb{P}[\zeta_{t,d+1} = m] \propto \theta_t(\delta_t, m)$;
5. **Predict** $a_t \leftarrow \zeta_{t,g}$;
6. **Receive** $\ell_{t,a_t}$
7. $\psi_t \leftarrow \sum_{n\in\Uparrow(u_t)} \mu_t(n)\theta(n,r)$; $\quad \varrho_t \leftarrow \sum_{n\in\Uparrow(u_t)} \mu_t(n)\theta(n,a_t)$; $\quad \bar{\lambda}_t \leftarrow \exp(-\eta\ell_{t,a_t}\psi_t/\varrho_t)$;
8. $\forall n \in \Uparrow(u_t)$:
   (a) $\mu_{t+1}(n) \leftarrow \mu_t(n)\psi_t/(\psi_t - (1-\bar{\lambda}_t)\varrho_t)$; $\quad \theta_{t+1}(n,a_t) \leftarrow \bar{\lambda}_t\theta_t(n,a_t)$;
   (b) $\forall m \in \mathcal{B} \setminus \Uparrow(a_t)$, $\quad \theta_{t+1}(n,m) := \theta_t(n,m)$;
   (c) **For** $d = 1, 2, \ldots g$: $\quad \theta_{t+1}(n,\zeta_{t,(g-d)}) \leftarrow \theta_{t+1}(n,\triangleleft(\zeta_{t,(g-d)})) + \theta_{t+1}(n,\triangleright(\zeta_{t,(g-d)}))$;
9. $\forall n \in \mathcal{C} \setminus \Uparrow(u_t)$, $\quad \mu_{t+1}(n) := \mu_t(n)$; $\quad \forall m \in \mathcal{B}$, $\theta_{t+1}(n,m) := \theta_t(n,m)$;

Figure 4: GABA-II Algorithm

### 4.3 Parameter Tuning

A limitation of the GABA-I regret bound is that it is dependent on knowing the optimal values of the parameters $\phi$ and $\eta$, and for GABA-II on the parameter $\eta$. In the following, we will 1) sketch how to autotune $\phi$ at little cost and 2) autotune $\eta$, however at essentially the cost of moving $\Psi(y)$ outside of the square root.

We first sketch how to automatically tune the parameter $\phi$ that appears in GABA-I. Assume, without loss of generality, that $N$ is an integer power of 2. The idea of our tuning method is that since $\phi$ is unknown we will "mix" over possible values of $\phi \in [0, 1]$. In fact, at little cost in regret it is sufficient to just mix over the exponentially increasing values of $\phi = 2/N, 4/N, 8/N, \ldots, N/N$ . Thus each specialist is split into $\log_2 N$ specialists, so that the new distribution over specialists is

$$w_1(s_\phi) := \frac{1}{\log_2 N} \times \text{valid1}(s) \times \frac{1}{K} \times \text{startfactor}(s) \times (1-\phi)^{N-1-\text{cut}(s)} \phi^{\text{cut}(s)} ,$$

where $\sum_{s \in \mathbb{S}, \phi \in \{2/N, 4/N, \ldots, 1\}} w_1(s_\phi) = 1$. Implementing this efficiently is similar to the implementation of GABA-I, except that we now have $\log_2 N$ copies of the BST $\mathcal{C}_\phi$, each initialized with a different value of $\phi$. On each trial the computed values from the $\log_2 N$ copies of the BST $\mathcal{C}_\phi$ are summed to find the prediction vector. After receipt of the loss, all copies of the BST are updated as in GABA-I. The regret bound of this autotuning with respect to $\phi$ is equal, up to an $\mathcal{O}(\sqrt{\log(\log(N))})$ factor, to that of GABA-I with the optimal $\phi$, but comes at the cost of an additional $\mathcal{O}(\log(N))$ factor in the computation time.

Now that we have shown how to automatically tune $\phi$ in GABA-I we are left with the learning rate $\eta$ in both algorithms. We first note that, with any $\eta$, the regret of both algorithms is $\Upsilon/\eta + \eta KT/2$, where $\Upsilon$ is the robustified resistance weighted cutsize $\Psi(y)$ multiplied by logarithmic terms (one in GABA-I and two in GABA-II). By setting $\eta = \sqrt{2/KT}$ we get a regret of $(\Upsilon + 1)\sqrt{KT/2}$. In addition, if $T$ is unknown then a doubling trick can be performed with this result to get a regret bound of $\mathcal{O}(\Upsilon\sqrt{KT})$ with no parameters needed. We compare this to the regret bound of $\mathcal{O}(\sqrt{\Upsilon KT})$ that comes from the optimal tuning of $\eta$. It remains an open problem to bring $\Upsilon$ inside the square-root.

Even with the above knowledge-free tuning of $\eta$, our methods improve over the *baseline* comparator of running an *independent* EXP3 algorithm for each of the $N$ users in many natural scenarios. Recall that in this case the induced regret is then $\widetilde{\mathcal{O}}(\sqrt{NKT})$ (see (2)). Consider a very large social network where the bandit problem is to show 1-of-$K$ advertisements (for simplicity assume $K \in \mathcal{O}(1)$) at the nodes (users). Now consider the case that each user is served at most *one* advert, i.e., there is at most a single trial for any given user. Since $N \geq T$ the bound of the baseline is now the vacuous regret $\Theta(T)$. We can intuitively see that this analysis is correct since the baseline algorithm is now just picking a single "uniformly at random" advertisement from $[K]$ for each user independently. However, observe that when $\Psi(y) \in \tilde{o}(\sqrt{T})$ we get $\widetilde{\mathcal{O}}(\Psi(y)\sqrt{T}) \subseteq \tilde{o}(T)$, which is non-vacuous. Intuitively, GABA-I/II may achieve this result since algorithmically they are exploiting the network structure.

## 5 Conclusion

We considered a contextual, non-stochastic bandit problem in which the finite set of contexts (a.k.a users) form a social network and the inductive bias is that if the social link between two users is strong then actions that perform well for one of these users are likely to perform well for the other. We gave two highly efficient algorithms for this problem, both with good regret bounds. Since this work is theoretical in nature we cannot foresee any potential negative societal impacts.

In the future it may be interesting to investigate extensions of our algorithms to the stochastic setting, as well as continuous bandit settings. Finally, it would be valuable to study potential applications of our algorithms, with large scale recommender systems being a natural candidate. On the theory side our bounds are based on an exponential potential function. Improved adversarial regret bounds were proven for an alternate potential function in [19] and it is an open question if our techniques can be extended to that potential.

**Acknowledgements.** Mark Herbster was supported by the U.S. Army Research Laboratory and the U.K. Ministry of Defence under Agreement Number W911NF-16-3-0001. Stephen Pasteris and Massimiliano Pontil were supported in part by EPSRC Grant N. EP/P009069/1 and SAP SE.

## References


[1] Lihong Li, Wei Chu, John Langford, and Robert E. Schapire. A contextual-bandit approach to personalized news article recommendation. In *Proceedings of the 19th International Conference on World Wide Web*, WWW '10, pages 661–670, New York, NY, USA, 2010. Association for Computing Machinery.

[2] Sofía S Villar Villar, Jack Bowden, and James Wason. Multi-armed Bandit Models for the Optimal Design of Clinical Trials: Benefits and Challenges. *Statistical Science*, 30(2):199 – 215, 2015.

[3] Baruch Awerbuch and Robert Kleinberg. Online linear optimization and adaptive routing. *Journal of Computer and System Sciences*, 74(1):97–114, 2008.

[4] Rie Kubota Ando and Tong Zhang. A framework for learning predictive structures from multiple tasks and unlabeled data. *J. Mach. Learn. Res.*, 6:1817–1853, December 2005.

[5] Andreas Maurer and Massimiliano Pontil. Excess risk bounds for multitask learning with trace norm regularization. In *Conference on Learning Theory*, pages 55–76, 2013.

[6] Maria-Florina Balcan, Mikhail Khodak, and Ameet Talwalkar. Provable guarantees for gradient-based meta-learning. In *International Conference on Machine Learning*, pages 424–433, 2019.

[7] Giovanni Cavallanti, Nicolò Cesa-Bianchi, and Claudio Gentile. Linear algorithms for online multitask classification. *J. Mach. Learn. Res.*, 11:2901–2934, December 2010.

[8] Pierre Alquier, The Tien Mai, and Massimiliano Pontil. Regret Bounds for Lifelong Learning. In Aarti Singh and Jerry Zhu, editors, *Proceedings of the 20th International Conference on rtificial Intelligence and Statistics*, volume 54 of *Proceedings of Machine Learning Research*, pages 261–269, Fort Lauderdale, FL, USA, 20–22 Apr 2017. PMLR.

[9] Giulia Denevi, Carlo Ciliberto, Dimitris Stamos, and Massimiliano Pontil. Incremental learning-to-learn with statistical guarantees. *arXiv preprint arXiv:1803.08089*, 2018.

[10] Giulia Denevi, Carlo Ciliberto, Riccardo Grazzi, and Massimiliano Pontil. Learning-to-learn stochastic gradient descent with biased regularization. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 1566–1575, Long Beach, California, USA, 09–15 Jun 2019. PMLR.

[11] Anastasia Pentina and Ruth Urner. Lifelong learning with weighted majority votes. In D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems 29*, pages 3612–3620. Curran Associates, Inc., 2016.

[12] Miller McPherson, Lynn Smith-Lovin, and James M Cook. Birds of a feather: Homophily in social networks. *Annual Review of Sociology*, 27(1):415–444, 2001.

[13] M. Herbster, G. Lever, and M. Pontil. Online prediction on large diameter graphs. In *NIPS*, 2008.

[14] Nicolò Cesa-Bianchi, C. Gentile, Fabio Vitale, and Giovanni Zappella. Random spanning trees and the prediction of weighted graphs. In *ICML*, 2010.

[15] Y. Freund, R. Schapire, Y. Singer, and Manfred K. Warmuth. Using and combining predictors that specialize. In *STOC '97*, 1997.

[16] Wouter M. Koolen, Dmitry Adamskiy, and Manfred K. Warmuth. Putting bayes to sleep. In *NIPS*, 2012.

[17] Aleksandrs Slivkins. Introduction to multi-armed bandits. *Foundations and Trends in Machine Learning*, 12(1-2):1–286, 2019.

[18] Peter Auer, Nicolò Cesa-Bianchi, Yoav Freund, and Robert E. Schapire. The nonstochastic multiarmed bandit problem. *SIAM J. Comput.*, 32(1):48–77, January 2003.

[19] Jean-Yves Audibert and Sébastien Bubeck. Regret bounds and minimax policies under partial monitoring. *Journal of Machine Learning Research*, 11(94):2785–2836, 2010.

[20] Sébastien Bubeck and Nicolo Cesa-Bianchi. Regret analysis of stochastic and nonstochastic multi-armed bandit problems. *Foundations and Trends in Machine Learning*, 5(1):1–122, 2012.

[21] N. Cesa-Bianchi, C. Gentile, and Giovanni Zappella. A gang of bandits. In *NIPS*, 2013.

[22] Kaige Yang, X. Dong, and L. Toni. Laplacian-regularized graph bandits: Algorithms and theoretical analysis. In *AISTATS*, 2020.

[23] Sharan Vaswani, M. Schmidt, and L. Lakshmanan. Horde of bandits using gaussian markov random fields. In *AISTATS*, 2017.

[24] Qingyun Wu, Huazheng Wang, Quanquan Gu, and Hongning Wang. Contextual bandits in a collaborative environment. *Proceedings of the 39th International ACM SIGIR conference on Research and Development in Information Retrieval*, 2016.

[25] Xin Wang, S. Hoi, Chenghao Liu, and M. Ester. Interactive social recommendation. *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*, 2017.

[26] Huazheng Wang, Qingyun Wu, and Hongning Wang. Factorization bandits for interactive recommendation. In *AAAI*, 2017.

[27] C. Gentile, S. Li, and Giovanni Zappella. Online clustering of bandits. In *ICML*, 2014.

[28] S. Li, C. Gentile, and Alexandros Karatzoglou. Graph clustering bandits for recommendation. *ArXiv*, abs/1605.00596, 2016.

[29] C. Gentile, S. Li, Purushottam Kar, Alexandros Karatzoglou, Giovanni Zappella, and Evans Etrue. On context-dependent clustering of bandits. In *ICML*, 2017.

[30] S. Li, C. Gentile, Alexandros Karatzoglou, and Giovanni Zappella. Data-dependent clustering in exploration-exploitation algorithms. *ArXiv*, abs/1502.03473, 2015.

[31] S. Li and Purushottam Kar. Context-aware bandits. *ArXiv*, abs/1510.03164, 2015.

[32] S. Li, C. Gentile, Alexandros Karatzoglou, and Giovanni Zappella. Online context-dependent clustering in recommendations based on exploration-exploitation algorithms. *ArXiv*, abs/1608.03544, 2016.

[33] S. Li, Alexandros Karatzoglou, and C. Gentile. Collaborative filtering bandits. *Proceedings of the 39th International ACM SIGIR conference on Research and Development in Information Retrieval*, 2016.

[34] Kaige Yang and L. Toni. Graph-based recommendation system. *2018 IEEE Global Conference on Signal and Information Processing (GlobalSIP)*, pages 798–802, 2018.

[35] Trong T. Nguyen and H. W. Lauw. Dynamic clustering of contextual multi-armed bandits. *Proceedings of the 23rd ACM International Conference on Conference on Information and Knowledge Management*, 2014.

[36] Zongguo Wang, Chicheng Zhang, M. Singh, L. Riek, and K. Chaudhuri. Multitask bandit learning through heterogeneous feedback aggregation. In *AISTATS*, 2021.

[37] Zongguo Wang, M. Singh, Chicheng Zhang, L. Riek, and K. Chaudhuri. Stochastic multi-player bandit learning from player-dependent feedback. 2020.

[38] Hassan Saber, Pierre Ménard, and Odalric-Ambrym Maillard. Optimal strategies for graph-structured bandits. *ArXiv*, abs/2007.03224, 2020.

[39] Sabina Tomkins, Peng Liao, P. Klasnja, and S. Murphy. Intelligentpooling: Practical thompson sampling for mhealth. *ArXiv*, abs/2008.01571, 2020.

[40] Silviu Maniu, Stratis Ioannidis, and B. Cautis. Bandits under the influence. *2020 IEEE International Conference on Data Mining (ICDM)*, pages 1172–1177, 2020.

[41] L. Yang, Bo Liu, L. Lin, Feng Xia, Kai Chen, and Qiang Yang. Exploring clustering of bandits for online recommendation system. *Fourteenth ACM Conference on Recommender Systems*, 2020.

[42] Huazheng Wang, Qingyun Wu, and Hongning Wang. Learning hidden features for contextual bandits. *Proceedings of the 25th ACM International on Conference on Information and Knowledge Management*, 2016.

[43] Swapna Buccapatnam, A. Eryilmaz, and N. Shroff. Multi-armed bandits in the presence of side observations in social networks. *52nd IEEE Conference on Decision and Control*, pages 7309–7314, 2013.

[44] S. Kar, H. Poor, and S. Cui. Bandit problems in networks: Asymptotically efficient distributed allocation rules. *IEEE Conference on Decision and Control and European Control Conference*, pages 1771–1778, 2011.

[45] Balázs Szörényi, R. Busa-Fekete, I. Hegedüs, Róbert Ormándi, M. Jelasity, and B. Kégl. Gossip-based distributed stochastic bandit algorithms. In *ICML*, 2013.

[46] N. Korda, Balázs Szörényi, and S. Li. Distributed clustering of linear bandits in peer to peer networks. *ArXiv*, abs/1604.07706, 2016.

[47] Abishek Sankararaman, A. Ganesh, and S. Shakkottai. Social learning in multi agent multi armed bandits. *Proceedings of the ACM on Measurement and Analysis of Computing Systems*, 3:1 – 35, 2019.

[48] L. E. Celis and F. Salehi. Lean from thy neighbor: Stochastic and adversarial bandits in a network. *ArXiv*, abs/1704.04470, 2017.

[49] Yogev Bar-On and Y. Mansour. Individual regret in cooperative nonstochastic multi-armed bandits. In *NeurIPS*, 2019.

[50] N. Cesa-Bianchi, C. Gentile, Y. Mansour, and Alberto Minora. Delay and cooperation in nonstochastic bandits. *ArXiv*, abs/1602.04741, 2016.

[51] Bo Liu, Y. Wei, Y. Zhang, Z. Yan, and Qiang Yang. Transferable contextual bandit for cross-domain recommendation. In *AAAI*, 2018.

[52] L. Cella, A. Lazaric, and M. Pontil. Meta-learning with stochastic linear bandits. *ArXiv*, abs/2005.08531, 2020.

[53] Michal Valko, R. Munos, B. Kveton, and Tomás Kocák. Spectral bandits for smooth graph functions. In *ICML*, 2014.

[54] M. K. Hanawal, Venkatesh Saligrama, Michal Valko, and R. Munos. Cheap bandits. *ArXiv*, abs/1506.04782, 2015.

[55] A. Deshmukh, Ü. Dogan, and C. Scott. Multi-task learning for contextual bandits. In *NIPS*, 2017.

[56] Michal Valko. Bandits on graphs and structures. Diss., École normale supérieure de Cachan-ENS Cachanz, 2016.

[57] M. K. Hanawal and Venkatesh Saligrama. Efficient detection and localization on graph structured data. *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5590–5594, 2015.

[58] M. K. Hanawal and Venkatesh Saligrama. Cost effective algorithms for spectral bandits. *2015 53rd Annual Allerton Conference on Communication, Control, and Computing (Allerton)*, pages 1323–1329, 2015.

[59] Sylvain Lamprier, Thibault Gisselbrecht, and P. Gallinari. Variational thompson sampling for relational recurrent bandits. In *ECML/PKDD*, 2017.

[60] Aadirupa Saha, Shreyas Sheshadri, and C. Bhattacharyya. Be greedy: How chromatic number meets regret minimization in graph bandits. In *UAI*, 2019.

[61] N. Alon, N. Cesa-Bianchi, C. Gentile, Shie Mannor, Y. Mansour, and O. Shamir. Nonstochastic multi-armed bandits with graph-structured feedback. *SIAM J. Comput.*, 46:1785–1826, 2017.

[62] N. Alon, N. Cesa-Bianchi, O. Dekel, and T. Koren. Online learning with feedback graphs: Beyond bandits. In *COLT*, 2015.

[63] Meng Fang and D. Tao. Networked bandits with disjoint linear payoffs. *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, 2014.

[64] M. Herbster, Stephen Pasteris, and Fabio Vitale. Online sum-product computation over trees. In *NIPS*, 2012.

[65] M. Herbster and J. Robinson. Online prediction of switching graph labelings with cluster specialists. In *NeurIPS*, 2019.

[66] Lyons, Russell, and Yuval Peres. *Probability on trees and networks*. Cambridge University Press, 2017.

[67] Nicolò Cesa-Bianchi, Claudio Gentile, Fabio Vitale, and Giovanni Zappella. Random spanning trees and the prediction of weighted graphs. *Journal of Machine Learning Research*, 14(2):1251–1284, 2013.

[68] R. Pemantle. Uniform random spanning trees. *arXiv: Probability*, 2004.

[69] Aaron Schild. An almost-linear time algorithm for uniform random spanning tree generation. *Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing*, 2018.

[70] D. Durfee, Rasmus Kyng, J. Peebles, A. Rao, and Sushant Sachdeva. Sampling random spanning trees faster than matrix multiplication. *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing*, 2017.

[71] Volodimir G. Vovk. Aggregating strategies. *Proc. of Computational Learning Theory*, 1990.

[72] David Haussler, Jyrki Kivinen, and Manfred K Warmuth. Sequential prediction of individual sequences under general loss functions. *IEEE Transactions on Information Theory*, 44(5):1906–1925, 1998.

[73] Nick Littlestone and Manfred K. Warmuth. The weighted majority algorithm. *Information and computation*, 108(2):212–261, 1994.

[74] Yoav Freund and Robert E Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of computer and system sciences*, 55(1):119–139, 1997.

[75] David Barber. *Bayesian Reasoning and Machine Learning*. Cambridge University Press, 2012.

[76] Leslie Ann Goldberg and Mark Jerrum. The complexity of ferromagnetic ising with local fields. *Combinatorics, Probability & Computing*, 16(1):43–61, 2007.

[77] Arthur L. Delcher, Adam J. Grove, Simon Kasif, and Judea Pearl. Logarithmic-time updates and queries in probabilistic networks. *CoRR*, abs/1408.1479, 2014.

[78] Guy Bresler, D. Shah, and L. F. Voloch. Collaborative filtering with low regret. *Proceedings of the 2016 ACM SIGMETRICS International Conference on Measurement and Modeling of Computer Science*, 2016.

[79] Guy Bresler, D. Shah, and L. F. Voloch. Regret guarantees for item-item collaborative filtering. *arXiv: Learning*, 2015.

[80] Nicolò Cesa-Bianchi and G. Lsugosi. Combinatorial bandits. *J. Comput. Syst. Sci.*, 78:1404–1422, 2009.

[81] Alon Cohen, Tamir Hazan, and T. Koren. Tight bounds for bandit combinatorial optimization. In *COLT*, 2017.

[82] S. Ito, Daisuke Hatano, Hanna Sumita, Kei Takemura, Takuro Fukunaga, Naonori Kakimura, and K. Kawarabayashi. Improved regret bounds for bandit combinatorial optimization. In *NeurIPS*, 2019.

[83] Kareem Amin, M. Kearns, and U. Syed. Graphical models for bandit problems. In *UAI*, 2011.

[84] Aleksandrs Slivkins. Contextual bandits with similarity information. *J. Mach. Learn. Res.*, 15:2533–2568, 2011.

[85] Qingyun Wu, N. Iyer, and Hongning Wang. Learning contextual bandits in a non-stationary environment. *The 41st International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2018.

[86] Qingyun Wu, Huazheng Wang, Yanen Li, and Hongning Wang. Dynamic ensemble of contextual bandits to satisfy users' changing interests. *The World Wide Web Conference*, 2019.

[87] Xiao Xu, Fang Dong, Yanghua Li, Shao jian He, and X. Li. Contextual-bandit based personalized recommendation with time-varying user interests. *ArXiv*, abs/2003.00359, 2020.

[88] Chuanhao Li, Qingyun Wu, and Hongning Wang. Unifying clustered and non-stationary bandits. In *AISTATS*, 2021.

[89] Mark Herbster, Stephen Pasteris, and Lisa Tse. Online multitask learning with long-term memory. In H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 17779–17791. Curran Associates, Inc., 2020.

# Technical Appendices

In the following appendices we provide proofs and additional background for all results in the body as well as an extended literature review. In the following appendix we provide a synopsis of our results and analyses as well as a brief guide to the remaining appendices.

## A Synopsis

In this work we provided two novel algorithms GABA-I and GABA-II (see Figures 3 and 4, respectively). We gave their regret bounds in Corollaries 4 and 5. We lightly discussed their time complexities in the main body of the paper and we elaborated on that discussion in the lead-in to Propositions 24 and 38. In this section we provide a very brief overview of the supporting analysis.

Step 1: Embed user graph $\mathcal{G}$ to a random spanning tree $\mathcal{T}$ (see Figure 1).

Step 2: Embed random spanning tree $\mathcal{T}$ to a linear order $\mathcal{L}$ (see Figure 1).

Step 3: Choose a prior $w_1(\cdot)$ for GABA-I (7) or GABA-II (9) aligning with the ordering $\mathcal{L}$.

*Observation:* Corollaries 4 and 5 rely only on the above steps.

Step 4: Construct from linear order $\mathcal{L}$ the binary support tree $\mathcal{C}$ (see Figure 1).

*Observation:* The tree $\mathcal{C}$ is the "skeleton" of the data structures that underpins the algorithms GABA-I and GABA-II.

Figure 5: A Schematic Overview

There are two technical tools that underly the regret analysis and the fast algorithms. The first tool is a sequence of embedding steps that produce a "binary support tree" (BST) $\mathcal{C}$ and a linear embedding $\mathcal{L}$ from the user graph $\mathcal{G}$. The second tool is the SPECIALISTEXP algorithm (see Figure 2) and its regret bound in Theorem 3. We give a four step schema in Figure 5 which shows how these tools interact. The linear embedding provided by Steps 1 and 2 reduces (with approximation guarantees) the user graph to a linear ordering. This construction is used at Step 3 to map the priors in Equations (7) and (9) to the ordering. Then, using these priors with Theorem 3, the proofs of Corollaries 4 and 5 follow in conjunction with the approximation guarantees given in Lemmas 2 and 36, the latter lemma is needed for the analysis of GABA-II only. By using the priors "directly" we would have per trial prediction and update times of $\Theta(K^N)$ and $\Theta(KN)$ for the GABA-I and GABA-II models, respectively. This motivates Step 4 which gives the basic data structure, a BST, upon which more elaborate data structures are then built that enable the algorithms GABA-I/II to speed-up prediction[4] to $\mathcal{O}(K \ln N)$ and $\mathcal{O}(\ln(K) \ln(N))$ time, respectively.

For GABA-I, from a bird's eye perspective we may understand the computational motivation behind the BST as that since we have a linear ordering over $N$ users then to perform belief propagation directly it will require $\Theta(N)$ time per action per trial. However, on the BST no two users are more than $2 \log N$ vertices apart. Thus by caching partial computations "in" the BST we may "online belief propagate" in $\Theta(\log N)$ time per action per trial. For GABA-II, however, the algorithm does not resemble belief propagation, rather it may be interpreted as a simpler "message passing" algorithm where there is a natural mapping between the vertices of the BST and the specialists themselves. The proofs of our results are contained in the remaining appendices whose structure we outline below.

As background we expand on our literature review in Appendix B. In Appendix C we provide a proof to Proposition 1. We then give notational conventions that hold in the remaining appendices in Appendix D. In Appendix E we give the pseudocode and overview the analysis of the PROTOGABA algorithm. This algorithm is essentially a special case of SPECIALISTEXP. However, the notational conventions of the PROTOGABA algorithm will prove to be more natural for the data structures

[4]Note that the GABA-I/II algorithms do not take as explicit input the prior distributions. Rather they are prediction-equivalent to SPECIALISTEXP with the correct prior.

used in the GABA-I/II algorithms. The pseudocode and more detailed overviews of the GABA-I/II algorithms are given in Appendices F and G respectively. We then provide all (remaining) proofs in the appendices H, I and J.

## B Expanded Literature Review

In order to give more background we expand on literature review in the the main body.

There has been much work in the multi-user setting for linear-stochastic bandits. In this setting (the pure stochastic setting being a special case) each user has a weight vector and on each trial we need to choose an action from a given finite set of context vectors which varies over trials: the loss on a trial is equal to the inner product of the selected context vector and the weight vector of the current user, plus some zero-mean random noise. Out of these works, those closest to us are when the users are in a known social network and it is assumed that neighbouring users in this network are likely to have similar weight vectors [21, 22, 23, 24, 25, 26] but as far as we are aware no works on this model have so far been done in the adversarial setting. Other works on this topic include those in which no social network is given but it is assumed that there is an unknown clustering of the users, sometimes dependent on the context, and all users in a cluster have the same weight vector or respond to the context in the same way [27, 28, 29, 30, 31, 32, 33, 34, 35]; those in which the norm of the difference of the weight vectors between any two users is bounded [36, 37], sometimes by the weight of the corresponding edge in a social network [38]; those in which the user weight vectors are equal to an unknown global vector plus a user-specific random vector [39]; those in which the user weight vectors evolve over time by linearly incorporating weights of neighbouring users [40]; those in which each user's weight vector is a linear combination of a set of unknown vectors [41]; those in which the users are unrelated but the actions have, in addition to the observed context vector, hidden features/components [42]; and those in which, when an action is selected for a user, the loss is observed for all its neighbours also [43]. Another multi-user topic is in the works [78, 79] in which users either like or dislike items (the suggestion of an item being the action) but we can't suggest the same item to a user twice: when a new item is selected its like-dislike vector (over users) is drawn uniformly at random from some probability distribution which has constraints on it. There is also a multi-user adversarial bandit problem which is a special case of the combinatorial bandit problem [80, 81, 82], where on each trial actions are selected for all users but only the total loss is observed. The above works all assume that the bandit problem for different users is different (i.e. not all users have the same weight vectors or best actions) and the algorithm is centralised: another line of research is multi-user bandit problems (both stochastic and adversarial) in which the users are collaboratively trying to solve the same problem but the algorithm is distributed in that there is limited communication between users over the social network [44, 45, 46, 47, 48, 49, 50]. Related to the multi-user setting are works on transfer learning and meta-learning with linear-stochastic bandits [51, 52].

Whilst our work assumes a known network structure over the users, there is a wide literature on bandit problems in which the actions are structured in a network and it is assumed that neighbouring actions give similar losses [53, 54, 55, 56, 57, 58, 59, 60] or that when an action is selected, the losses of its neighbouring actions are revealed [61, 62] sometimes contributing to the incurred loss [63]. Related to this is the work of [83] in which contexts and actions are vectors, with each component being of a finite set of values. In this work, each component of the context and action vector corresponds to a node of a given graph and the expected reward of a context/action pair is defined via a graphical model on the graph. There is also the work [84] in the stochastic setting where we have users and actions and there is a known metric space over all feasible user/action pairs such that the difference between the mean rewards of two user/action pairs is bounded by the distance between them.

Since, in our work, the best action varies over the users and is assumed to be similar for neighbouring users, our work is related to non-stationary linear-stochastic bandits where the user's (often a single user) weight vector varies over the trials but is assumed to be similar for neighbouring trials [85, 86, 87, 88]

We now cite the works that haven given us the mathematical tools to formulate our algorithms. Both our main algorithms GABA-I and GABA-II are efficient implementations of instances of our underlying algorithm SPECIALISTEXP which is dependent on a weighted set of subsets of users and is a combination of ideas from [18, 15, 16, 89]. Also, in order to define the weighted set of subsets that is input to PROTOGABA , as well as performing the implementations themselves, both GABA-I and GABA-II linearize the social network as was introduced, in the machine learning context, in [13, 14]. GABA-I then utilises ideas from [16, 64] whilst GABA-II utilises ideas from [65].

# C Proof of Proposition 1

First we recall Proposition 1 and then prove it.

**Proposition 1.** *Consider an unweighted graph $\mathcal{G}$ partitioned into $G$ clusters and a labeling function $y(\cdot)$, where each cluster is an $n$-clique and, if $u, v$ are vertices in the cluster, then $y(u) = y(v)$. For any pair of such clusters $C, C' \subset \mathcal{G}$, suppose that there are $\frac{n-1}{G}$ edges connecting the nodes of $C$ with the nodes of $C'$. Then we have $\Phi(y) \in \mathcal{O}(G)$.*

*Proof.* Let $\mathcal{C} := \{C_1, C_2, \ldots, C_G\}$ be the set of all clusters. For each inter-cluster cut-edge $\{v_i, v_j\}$, where $v_i \in C_i$ and $v_j \in C_j$, we show how to find $\Theta(n)$ edge-disjoint paths of constant length in $\mathcal{G}$ connecting $v_i$ with $v_j$. Such paths can be partitioned into $G-2$ equally sized sets, where each of such sets contains $\frac{n-1}{G}$ paths passing through one of the clusters in $\mathcal{C} \setminus \{C_i, C_j\}$. This way, by applying the rule of resistors in parallel combined with Rayleigh's monotonicity, we have that the effective resistance between $u$ and $v$ is upper bounded by $\mathcal{O}\left(\frac{1}{n}\right)$. On the other hand, we have $\frac{n-1}{G}\binom{G}{2}$-many inter-cluster edges in $\mathcal{G}$, which gives us a total effective resistance $\Phi(y)$ bounded by $\mathcal{O}(G)$.

For the sake of simplicity, assume that there are at least 3 clusters[5]. Consider any cluster $C_k \in \mathcal{C} \setminus \{C_i, C_j\}$. Let $V_{i,k}$ and $V_{j,k}$ be the subset of nodes of $C_k$ that are incident to edges connecting $C_k$ with $C_i$ and $C_k$ with $C_j$ respectively. For each node $u \in V_{i,k} \cup V_{j,k}$, let $d_u^i$ and $d_u^j$ be the number of edges connecting $u$ with the nodes of $C_i$ and the nodes of $C_j$ respectively.

We first show that we can connect the nodes of $V_{i,k}$ with the nodes of $V_{j,k}$ through $\frac{n-1}{G}$ edge disjoint-paths of length at most 2 lying all within $C_k$ [6]. Each node $u \in V_{i,k}$ is connected with the nodes $V_{j,k}$ through $d_u^i$-many edge-disjoint paths of length *at most* 2 within $C_k$. Symmetrically, these edge-disjoint paths connect the nodes of $V_{i,k}$ with the nodes of $V_{j,k}$ in such a way that each node $v \in V_{j,k}$ is a terminal node of $d_u^j$-many of such edge-disjoint paths.

In the simplest case scenario, if $|V_{i,k}| = \frac{n-1}{G}$, $|V_{i,j}| = \frac{n-1}{G}$ and $V_{i,k} \cap V_{j,k} = \emptyset$, then we can immediately find $\frac{n-1}{G}$ edges (edge disjoint paths of length 1) lying within $C_k$. If $V_{i,k} \cap V_{j,k} \neq \emptyset$, we can create $\min(d_u^i, d_u^j)$-many paths of length 0 for each node $u \in V_{i,k} \cap V_{j,k}$. If we have instead that $|V_{i,k}| < \frac{n-1}{G}$ or $|V_{i,j}| < \frac{n-1}{G}$, then we can always exploit the clique structure of the cluster to create new paths of length 2 lying within $C_k$. To see how, consider for instance the extreme case where $|V_{i,k}| = 1$, $|V_{j,k}| = 1$ and $V_{i,k} \cap V_{j,k} = \emptyset$. The clique structure ensures that we can connect the nodes in $V_{i,k}$ and $V_{j,k}$ through 1 edge plus $\frac{n-1}{G} - 1$ edge-disjoint paths of length 2 all within $C_k$. For the paths of length 2 we use $\frac{n-1}{G} - 1$ *other* nodes of $C_k$ that are not terminal nodes of such paths (i.e., they are the nodes in the middle of these paths). This possibility is guaranteed by the fact $n \geq 1 + \frac{n-1}{G} = 2 + \frac{n-1}{G} - 1$. Finally, if $|V_{i,k}| < \frac{n-1}{G}$ or $|V_{i,j}| < \frac{n-1}{G}$ and $V_{i,k} \cap V_{j,k} \neq \emptyset$, then, as we mentioned above, we can exploit each node $u \in V_{i,k} \cap V_{j,k}$ to create $\min(d_u^i, d_u^j)$-many paths of length 0, which makes the problem easier. Notice that, whenever $V_{i,k} \cap V_{j,k} \neq \emptyset$, the problem can be reduced to finding just $\left(\frac{n-1}{G} - \sum_{u \in V_{i,k} \cap V_{j,k}} \min(d_u^i, d_u^j)\right)$-many edge-disjoint paths within $C_k$, connecting the nodes in $V_{i,k}$ and $V_{j,k}$

Now we show how to connect $v_i$ (resp. $v_j$) with the nodes in $V_{i,k}$ (resp. $V_{j,k}$) by $\frac{n-1}{G}$ edge-disjoint paths of length at most three, for each $k$ in turn (so that all paths are edge-disjoint). Without loss of generality, we will focus on $C_i$ and $v_i$. We proceed incrementally by connecting one by one each node $u \in V_{i,k}$ (for all $k \in [G] \setminus \{i, j\}$ in turn) to $v_i$ through $d_u^i$ paths via the following algorithm. For all edges $\{w, u\}$ where $w \in C_i$ we create a path as follows:

[5] If there are only 2 clusters, then it can be shown that $\Phi(y) \in \mathcal{O}(1)$ by using a very similar argument.

[6] In the special case a node of $C_k$ is incident to an edge connecting $C_i$ with $C_k$, and one connecting $C_k$ with $C_j$, then we view it by convenience as a node of a path of length 0 belonging to $C_k$. Clearly such special path is edge-disjoint from any other path lying in $C_k$ because it does not have any edge.

- If $w = v_i$ then our path is $(v_i, u)$.
- If $w \neq v_i$ choose a node $v \in C_i \setminus \{v_i\}$ which hasn't been a "middle node" of any path so far. We have two cases:
  - If $w = v$ then our path is $(v_i, v, u)$
  - If $w \neq v$ then our path is $(v_i, v, w, u)$

  The node $v$ is called the "middle node" of our path, so can't be selected as the middle node of any future path.

It is easy to verify that all the paths formed exist and are edge disjoint, and that we have $n-1$ possibilities for middle nodes. Note that we need to use at most $\frac{n-1}{G}$ middle nodes for each cluster $C_k \in \mathcal{C} \setminus \{C_i, C_j\}$. Since the total number of clusters is $G$, we then need to have $(G-2)\frac{n-1}{G} \leq n-1$, which is always true.

Thus we can ensure that, for all inter-cluster edges $\{v_i, v_j\}$ where $v_i \in V_i$ and $v_j \in V_j$, there exist $(G-2)\frac{n-1}{G} = \Theta(n)$ edge-disjoint paths formed by, for all $k$, concatenating the edge-disjoint paths from node $v_i$ into $V_{i,k}$ , the edge-disjoint paths lying within $C_k$ , and finally the edge-disjoint paths from node $v_j$ into $V_{j,k}$ . The total length of each of such paths cannot therefore exceed $3+2+3=8$, which concludes the proof. $\square$

## D Conventions

In the following we present some conventions which lighten the notation of the main body as well as introduce the subroutine structure shared by the three algorithms ProtoGaba and Gaba-I/II.

In order to make our pseudocode clearer, the learning algorithms we present each have three subroutines: **Initialization**, **Prediction**$(\cdot)$, which returns an element of $[K]$ , and **Update**$(\cdot)$. The learning protocol (i.e. the sequence of subroutine calls) is given in Figure 6.

**Learning Protocol**

1. **Initialization**
2. For all $t \in [T]$ in order:
   (a) $a_t \leftarrow \mathbf{Prediction}_t(u_t)$
   (b) $\mathbf{Update}_t(\ell_{t,a_t})$

where $\{u_t \mid t \in [T]\} \subseteq [N]$ and $\{\boldsymbol{\ell}_t \mid t \in [T]\} \subseteq [0,1]^K$ are arbitrary, fixed a-priori, and unknown to the algorithm.

Figure 6: Learning Algorithm Protocol

We recall that $y$ is an arbitrary function from the users $[N]$ to the actions $[K]$. We abbreviate the Robustified Resistance Weighted Cutsize (see (5)) $\Psi(y)$ to $\Psi$ and we recall the expected regret notation (see (3)),

$$R(y) := \mathbb{E}\left[\sum_{t\in[T]} (\ell_{t,a_t} - \ell_{t,y(u_t)})\right] ,$$

which we abbreviate to $R$.

## E SpecialistExp and ProtoGaba

We first introduce the algorithm ProtoGaba of which the Gaba-I/II algorithms implement instances of. ProtoGaba takes as input a set $\mathcal{E}$, of subsets of $[N]$, and a probability distribution $\sigma$ over $\mathcal{E}$, and runs SpecialistExp with initial weights defined as follows:

- Given $U \in \mathcal{E}$ and $a \in [K]$, the specialist that predicts $a$ whenever $u_t \in U$ and abstains otherwise is given initial weight $\sigma(U)/K$ .

**ProtoGaba Subroutines**

The ProtoGaba algorithm takes a parameter $\eta \in \mathbb{R}^+$ and a set $\mathcal{E} \subseteq 2^{[N]}$ along with a function $\sigma : \mathcal{E} \to \mathbb{R}^+$ satisfying:

$$\sum_{U \in \mathcal{E}} \sigma(U) = 1$$

**Initialization** :

1. For all $U \in \mathcal{E}$ and all $a \in [K]$ :
   (a) $\kappa_1(a, U) \leftarrow \sigma(U)/K$ .

$\mathbf{Prediction}_t(u_t)$ :

1. For all $a \in [K]$ set $p_{t,a} \leftarrow \sum_{U \in \mathcal{E}: u_t \in U} \kappa_t(a, U)$ .
2. Draw $a_t$ from $[K]$ with probability $\mathbb{P}[a_t = a] \propto p_{t,a}$ .
3. Return $a_t$ .

$\mathbf{Update}_t(\ell_{t,a_t})$ :

1. $\lambda_t \leftarrow \exp(-\eta \ell_{t,a_t} \|\boldsymbol{p}_t\|_1 / p_{t,a_t})$ .
2. $z_t \leftarrow \|\boldsymbol{p}_t\|_1 / (\|\boldsymbol{p}_t\|_1 - (1 - \lambda_t) p_{t,a_t})$ .
3. For all $U \in \mathcal{E}$ and all $a \in [K]$ :
   (a) If $u_t \notin U$ then $\kappa_{t+1}(a, U) := \kappa_t(a, U)$ .
   (b) If $u_t \in U$ :
      i. If $a \neq a_t$ then $\kappa_{t+1}(a, U) \leftarrow z_t \kappa_t(a, U)$ .
      ii. If $a = a_t$ then $\kappa_{t+1}(a, U) \leftarrow z_t \lambda_t \kappa_t(a, U)$ .

Figure 7: ProtoGaba Subroutines

- If there does not exist such a $U$ and $a$ then the specialist has initial weight $0$ .

The subroutines of ProtoGaba are given in Figure 7. We shall now bound the regret of ProtoGaba, noting that all results are proved (in order) in Appendix H. Note that Theorem 3 is proved in exactly the same way but replacing $\mathcal{A}$ by $\mathcal{S}$ and replacing the set $[K] \times \mathcal{E}$ by the non-zero weight specialists.

The regret bound of ProtoGaba depends on any set $\mathcal{A} \subseteq [K] \times \mathcal{E}$ satisfying the following conditions:

**Definition 6.** *Take an arbitrary set $\mathcal{A} \subseteq [K] \times \mathcal{E}$ such that:*

- *For all $u \in [N]$ there exists a unique pair $(a, U) \in \mathcal{A}$ with $u \in U$ .*
- *For all $u \in [N]$ and $(a, U) \in \mathcal{A}$ with $u \in U$ , we have that $a = y(u)$ .*

We begin our analysis by bounding the expected "progress" on each trial:

**Lemma 7.** *For all $t \in [T]$ we have:*

$$\mathbb{E}\left[\sum_{(a,U) \in \mathcal{A}} \ln\left(\frac{\kappa_{t+1}(a, U)}{\kappa_t(a, U)}\right)\right] \geq \eta\left(\mathbb{E}\left[\ell_{t,a_t}\right] - \ell_{t,y(u_t)}\right) - \frac{K\eta^2}{2}.$$

We will utilise the following inequality:

**Lemma 8.** *We have:*

$$\sum_{(a,U) \in \mathcal{A}} \ln\left(|\mathcal{A}| \kappa_{T+1}(a, U)\right) \leq 0.$$

Lemmas 7 and 8 lead to the following bound on the expected regret:

**Theorem 9.** *We have:*

$$R \leq \frac{1}{\eta} \sum_{(a,U)\in\mathcal{A}} \ln\left(\frac{K}{|\mathcal{A}|\sigma(U)}\right) + \frac{\eta K T}{2}.$$

## F GABA-I

We now introduce and analyse our algorithm GABA-I which has a per trial time-complexity of $\mathcal{O}(K\ln(N))$, a space complexity of $\mathcal{O}(KN)$, and a regret bound of $\mathcal{O}\left(\sqrt{\ln\left(\frac{KN}{\Psi}\right)\Psi KT}\right)$. The subroutines of GABA-I are given in Figure 8. We shall now outline the proof of the bound the on regret of GABA-I, noting that all results are proved (in order) in Appendix I. For simplicity assume, without loss of generality, that for all $u \in [N]$ we have that $u$ is the $u$-th leftmost leaf of $\mathcal{C}$.

We will show that GABA-I implements PROTOGABA with the following choice of $\mathcal{E}$ and $\sigma$:

**Definition 10.** *For all $i, j \in \{0,1\}$ we define:*

$$\iota_j := \frac{1}{K}[\![j=1]\!] + \left(1 - \frac{1}{K}\right)[\![j=0]\!]$$

*and:*

$$\tau_{i,j} := \phi[\![i \neq j]\!] + (1-\phi)[\![i=j]\!].$$

*For* GABA-I *we set:*

$$\mathcal{E} := 2^{[N]}$$

*and for all $U \in \mathcal{E}$ we set:*

$$\sigma(U) := \iota_{[\![1\in U]\!]} \prod_{u\in[N-1]} \tau_{[\![u\in U]\!],[\![u+1\in U]\!]}.$$

The following lemma states that, as required, $\sigma$ is a probability distribution over $\mathcal{E}$:

**Lemma 11.** *We have that $\sigma$ is a probability distribution, in that:*

$$\sum_{U\in\mathcal{E}} \sigma(U) = 1.$$

We will show that the function $\kappa_t$, constructed in PROTOGABA, is equal to the function $\bar{\kappa}_t$ that is defined as follows:

**Definition 12.** *For all $t \in [T+1]$ we define a function $\bar{\kappa}_t : [K]\times\mathcal{E} \to \mathbb{R}^+$ inductively as follows. For all $(a,U) \in [K]\times\mathcal{E}$:*

- $\bar{\kappa}_1(a,U) := \sigma(U)/K$ ‘
- For all $t \in [T]$ $\bar{\kappa}_{t+1}(a,U) = \begin{cases} \bar{\kappa}_t(a,U) & \text{if } u_t \notin U \\ \pi_{t,a}\bar{\kappa}_t(a,U) & \text{otherwise.} \end{cases}$

To quantify the vector valued functions $\boldsymbol{\alpha}_t$, $\boldsymbol{\beta}_t^{\Leftarrow}$ and $\boldsymbol{\beta}_t^{\Rightarrow}$ we need the following definitions:

**Definition 13.** *For all $t \in [T]$, $u \in [N]$ and $a \in [K]$ define:*

$$\xi_{t,u,a} := \prod_{s\in[t-1]:u_s=u} \pi_{s,a}.$$

**Definition 14.** *Given $u, v \in [N+1]$ with $u \leq v$ let $I(u,v)$ be the set of functions that map the set $\{w \in \mathbb{N} \mid u \leq w \leq v\}$ to $\{0,1\}$.*

**Definition 15.** *Given $f \in I(u,v)$ for some $u, v \in [N+1]$ with $u \leq v$, and some $a \in [K]$, let:*

$$\Omega_{t,a}(f) := \prod_{w\in\mathbb{N}:u\leq w<v} \tau_{f(w),f(w+1)}([\![f(w)=0]\!] + [\![f(w)=1]\!]\xi_{t,w,a}).$$

**GABA-I Subroutines**

GABA-I takes parameters $\eta \in \mathbb{R}^+$ and $\phi \in [0,1]$.

**Initialization** :

1. Construct BST $\mathcal{C}$ as in Section 2.2 .
2. For all leaves $n \in \mathcal{C}$ :
   (a) $\forall i,j \in \{0,1\}$, $\boldsymbol{\alpha}_1(n,i,j) \leftarrow [\![i \neq j]\!]\phi\mathbf{1} + [\![i=j]\!](1-\phi)\mathbf{1}$ .
3. For all $d \in [h-1]$ in order:
   (a) For all vertices $n \in \mathcal{C}$ at depth $h-d$ :
      i. $\forall i,j \in \{0,1\}$, $\boldsymbol{\alpha}_1(n,i,j) \leftarrow \sum_{k\in\{0,1\}} \boldsymbol{\alpha}_1(\triangleleft(n),i,k) \odot \boldsymbol{\alpha}_1(\triangleright(n),k,j)$ .

**Prediction**$_t(u_t)$ :

1. For all $d \in [h] \cup \{0\}$ let $\nu_{t,d}$ be the ancestor (in $\mathcal{C}$) of $u_t$ at depth $d$.
2. $\forall i \in \{0,1\}$, $\boldsymbol{\beta}_t^{\Leftarrow}(\nu_{t,0},i) \leftarrow (1+[\![i=0]\!](K-2))\mathbf{1}/K$ .
3. $\forall i \in \{0,1\}$, $\boldsymbol{\beta}_t^{\Rightarrow}(\nu_{t,0},i) \leftarrow \mathbf{1}$ .
4. For all $d \in [h]$ in order:
   (a) If $\nu_{t,d} = \triangleleft(\nu_{t,d-1})$ then:
      i. $\forall i \in \{0,1\}$, $\boldsymbol{\beta}_t^{\Leftarrow}(\nu_{t,d},i) \leftarrow \boldsymbol{\beta}_t^{\Leftarrow}(\nu_{t,d-1},i)$ .
      ii. $\forall i \in \{0,1\}$, $\boldsymbol{\beta}_t^{\Rightarrow}(\nu_{t,d},i) \leftarrow \sum_{j\in\{0,1\}} \boldsymbol{\alpha}_t(\triangleright(\nu_{t,d-1}),i,j) \odot \boldsymbol{\beta}_t^{\Rightarrow}(\nu_{t,d-1},j)$ .
   (b) If $\nu_{t,d} = \triangleright(\nu_{t,d-1})$ then:
      i. $\forall i \in \{0,1\}$, $\boldsymbol{\beta}_t^{\Rightarrow}(\nu_{t,d},i) \leftarrow \boldsymbol{\beta}_t^{\Rightarrow}(\nu_{t,d-1},i)$ .
      ii. $\forall i \in \{0,1\}$, $\boldsymbol{\beta}_t^{\Leftarrow}(\nu_{t,d},i) \leftarrow \sum_{j\in\{0,1\}} \boldsymbol{\beta}_t^{\Leftarrow}(\nu_{t,d-1},j) \odot \boldsymbol{\alpha}_t(\triangleleft(\nu_{t,d-1}),j,i)$ .
5. $\bar{\boldsymbol{p}}_t \leftarrow (1/K)\sum_{i\in\{0,1\}} \boldsymbol{\beta}_t^{\Leftarrow}(\nu_{t,h},1) \odot \boldsymbol{\alpha}_t(\nu_{t,h},1,i) \odot \boldsymbol{\beta}_t^{\Rightarrow}(\nu_{t,h},i)$ .
6. Draw $a_t$ from $[K]$ with probability $\mathbb{P}[a_t = a] \propto \bar{p}_{t,a}$ .
7. Return $a_t$ .

**Update**$_t(\ell_{t,a_t})$ :

1. $\forall a \in [K]$, $c_{t,a} \leftarrow \exp(-\eta[\![a=a_t]\!]\ell_{t,a_t}\|\bar{\boldsymbol{p}}_t\|_1/\bar{p}_{t,a})$ .
2. $\boldsymbol{\pi}^t \leftarrow (\|\bar{\boldsymbol{p}}_t\|_1\boldsymbol{c}_t)/(\bar{\boldsymbol{p}}_t \cdot \boldsymbol{c}_t)$ .
3. $\forall i \in \{0,1\}$, $\boldsymbol{\alpha}_{t+1}(\nu_{t,h},1,i) \leftarrow \boldsymbol{\pi}^t \odot \boldsymbol{\alpha}_t(\nu_{t,h},1,i)$ .
4. $\forall i \in \{0,1\}$, $\boldsymbol{\alpha}_{t+1}(\nu_{t,h},0,i) := \boldsymbol{\alpha}_t(\nu_{t,h},0,i)$ .
5. For all $n \in \mathcal{C} \setminus \{\nu_{t,d} \mid d \in [h] \cup \{0\}\}$ :
   (a) $\forall i,j \in \{0,1\}$, $\boldsymbol{\alpha}_{t+1}(n,i,j) := \boldsymbol{\alpha}_t(n,i,j)$ .
6. For all $d \in [h-1]$ in order:
   (a) $\forall i,j \in \{0,1\}$ set $\boldsymbol{\alpha}_{t+1}(\nu_{t,(h-d)},i,j) \leftarrow \sum_{k\in\{0,1\}} \boldsymbol{\alpha}_{t+1}(\triangleleft(\nu_{t,(h-d)}),i,k) \odot \boldsymbol{\alpha}_{t+1}(\triangleright(\nu_{t,(h-d)}),k,j)$ .

Figure 8: GABA-I Subroutines

To prove lemmas 17 and 18 we will need the following lemma:

**Lemma 16.** *Given $i, j, k \in \{0,1\}$, $u, v, w \in [N]$ with $u < v < w$, and $a \in [K]$ we have:*

$$\left(\sum_{f \in F} \Omega_{t,a}(f)\right)\left(\sum_{f \in G} \Omega_{t,a}(f)\right) = \sum_{f \in H} \Omega_{t,a}(f)$$

*where:*

- $F := \{f \in I(u,v) \mid f(u) = i\,, f(v) = j\}$
- $G := \{f \in I(v,w) \mid f(v) = j\,, f(w) = k\}$
- $H := \{f \in I(u,w) \mid f(u) = i\,, f(v) = j\,, f(w) = k\}$.

We now quantify the vector valued function $\boldsymbol{\alpha}_t$ :

**Lemma 17.** *For all $t \in [T] \cup \{0\}$, all non-root vertices $n$ of $\mathcal{C}$, all $i, j \in \{0,1\}$ and all $a \in [K]$ we have:*

$$\alpha_t(n,i,j)_a = \sum_{f \in A_t(n,i,j)} \Omega_{t,a}(f)$$

*where:*

$$A_t(n,i,j) := \{f \in I(\blacktriangleleft(n), \blacktriangleright(n)+1) \mid f(\blacktriangleleft(n)) = i\,, f(\blacktriangleright(n)+1) = j\}.$$

Lemma 17 now allows us to quantify the vector valued functions $\boldsymbol{\beta}_t^{\Leftarrow}$ and $\boldsymbol{\beta}_t^{\Rightarrow}$. We note that the following lemma is proved via induction over $d$.

**Lemma 18.** *For all $t \in [T]$, $d \in [h] \cup \{0\}$, $i \in \{0,1\}$ and $a \in [K]$ we have:*

$$\beta_t^{\Rightarrow}(\nu_{t,d}, i)_a = \sum_{f \in B_t^{\Rightarrow}(\nu_{t,d}, i)} \Omega_{t,a}(f)$$

*where:*

$$B_t^{\Rightarrow}(\nu_{t,d}, i) := \{f \in I(\blacktriangleright(\nu_{t,d})+1, N+1) \mid f(\blacktriangleright(\nu_{t,d})+1) = i\}$$

*and:*

$$\beta_t^{\Leftarrow}(\nu_{t,d}, i)_a = \sum_{f \in B_t^{\Leftarrow}(\nu_{t,d}, i)} \iota_{f(1)} \Omega_{t,a}(f)$$

*where:*

$$B_t^{\Leftarrow}(\nu_{t,d}, i) := \{f \in I(1, \blacktriangleleft(\nu_{t,d})) \mid f(\blacktriangleleft(\nu_{t,d})) = i\}.$$

Lemmas 17 and 18 allow us to write the vector $\bar{\boldsymbol{p}}_t$ in terms of the function $\bar{\kappa}_t$ :

**Lemma 19.** *For all $t \in [T]$ and $a \in [K]$ we have:*

$$\bar{p}_{t,a} = \sum_{U \in \mathcal{E} : u_t \in U} \bar{\kappa}_t(a, U)$$

Lemma 19 implies that GABA-I does indeed implement PROTOGABA :

**Lemma 20.** GABA-I *implements* PROTOGABA *with $\mathcal{E}$ and $\sigma$ defined as in Definition 10.*

We choose, as required in the analysis of PROTOGABA, the set $\mathcal{A}$ to be as follows:

**Definition 21.** *We define:*

$$\mathcal{A} := \{(a, \{u \in [N] \mid y(u) = a\}) \mid a \in [K]\}.$$

The following lemma states that $\mathcal{A}$ is valid:

**Lemma 22.** *We have that:*

- *For all $u \in [N]$ there exists a unique pair $(a, U) \in \mathcal{A}$ with $u \in U$.*
- *For all $u \in [N]$ and $(a, U) \in \mathcal{A}$ with $u \in U$, we have that $a = y(u)$.*

Lemmas 20 and 22 allow us to invoke Theorem 9. Noting that $|\mathcal{A}| = K$, Lemma 2 allows us to bound the expectation of the summation of $\ln(1/\sigma(a, U))$ appearing in Theorem 9, leading to our main result:

**Theorem 23.** *Given $\Psi \leq (N-1)/4$ and the parameters are tuned as:*

$$\phi := 4\Psi/(K(N-1))$$

*and:*

$$\eta := \sqrt{\frac{10\Psi \ln(KN/\Psi)}{KT}}$$

*we have:*

$$R \in \mathcal{O}\left(\sqrt{\ln\left(\frac{KN}{\Psi}\right)\Psi KT}\right).$$

We now argue the per-trial time complexity of GABA-I. In the prediction algorithm each of the $\mathcal{O}(\ln(N))$ proper ancestors $n$ of $u_t$ constructs four vector valued messages $\boldsymbol{\beta}_t^{\Leftarrow}(n,0)\,, \boldsymbol{\beta}_t^{\Leftarrow}(n,1)\,, \boldsymbol{\beta}_t^{\Rightarrow}(n,0)\,, \boldsymbol{\beta}_t^{\Rightarrow}(n,1)$, each taking a time of $\mathcal{O}(K)$ to construct. In the update algorithm the values $\boldsymbol{\alpha}_t(n,i,j)$ are updated to $\boldsymbol{\alpha}_{t+1}(n,i,j)$, only being modified when $n$ is one of the $\mathcal{O}(\ln(N))$ ancestors of $u_t$. Since each such modification takes a time of $\mathcal{O}(K)$ we then have the following proposition:

**Proposition 24.** GABA-I *takes a per-trial time of* $\mathcal{O}(K\ln(N))$.

## G GABA-II

We now introduce and analyse our algorithm GABA-II which has a per trial time-complexity of $\mathcal{O}(\ln(K)\ln(N))$, a space complexity of $\mathcal{O}(KN)$, and a regret bound of $\mathcal{O}\left(\sqrt{\ln\left(\frac{N}{\Psi}\right)\ln\left(\frac{KN}{\Psi}\right)\Psi KT}\right)$. The subroutines of GABA-II are given in Figure 9. We shall now outline the proof of the bound on the regret of GABA-II, noting that all results are proved (in order) in Appendix J. For simplicity assume, without loss of generality, that for all $u \in [N]$ we have that $u$ is the $u$-th leftmost leaf of $\mathcal{C}$.

We will show that GABA-II implements PROTOGABA with the following choice of $\mathcal{E}$ and $\sigma$:

**Definition 25.** *For* GABA-II *we set:*

$$\mathcal{E} := \{\Downarrow(n) \mid n \in \mathcal{C}\}$$

*and for all $U \in \mathcal{E}$ we set:*

$$\sigma(U) := \frac{1}{2N-1}.$$

Note that as $|\mathcal{E}| = |\mathcal{C}| = 2N-1$ it is clear that $\sigma$ is a probability distribution over $\mathcal{E}$, as required.

We will show that, for all $n \in \mathcal{C}$ and $a \in [K]$, the value $\kappa_t(a, \Downarrow(n))$, constructed in PROTOGABA, is equal to the value $\bar{\kappa}_t(a,n)$ that is defined as follows:

**Definition 26.** *For all $t \in [T]$ we define:*

$$\bar{z}_t := \psi_t/(\psi_t - (1-\bar{\lambda}_t)\varrho_t).$$

**Definition 27.** *We define the functions $\bar{\kappa}_t : [K] \times \mathcal{C} \rightarrow [0,1]$ inductively as follows. For all $(a,n) \in [K] \times \mathcal{C}$:*

- $\bar{\kappa}_1(a,n) := 1/K(2N-1)$
- *For all $t \in [T]$:*
  - *If $n \notin \Uparrow(u_t)$ then $\bar{\kappa}_{t+1}(a,n) = \bar{\kappa}_t(a,n)$*
  - *If $n \in \Uparrow(u_t)$ and $a \neq a_t$ then $\bar{\kappa}_{t+1}(a,n) := \bar{z}_t\bar{\kappa}_t(a,n)$*
  - *If $n \in \Uparrow(u_t)$ and $a = a_t$ then $\bar{\kappa}_{t+1}(a,n) := \bar{\lambda}_t\bar{z}_t\bar{\kappa}_t(a,n)$.*

We will define a vector $\bar{\boldsymbol{p}}_t$ from $\bar{\kappa}_t$ in the same way that $\boldsymbol{p}_t$ is defined from $\kappa_t$ in PROTOGABA:

**GABA-II Subroutines**

GABA-II takes a parameter $\eta \in \mathbb{R}^+$ .

**Intialization** :

1. Construct BST $\mathcal{C}$ as in Section 2.2 .
2. Construct a full, balanced, oriented binary tree $\mathcal{B}$ of depth $g := \lceil \log_2(K) \rceil$ whose first $K$ leaves are the actions $[K]$.
3. Set $r$ to be the root of $\mathcal{B}$ .
4. For all vertices $n \in \mathcal{C}$ :
   (a) $\mu_1(n) \leftarrow 1$ .
   (b) For all leaves $m \in \mathcal{B}$:
      i. If $m \in [K]$ then $\theta_1(n, m) \leftarrow 1$ .
      ii. If $m \notin [K]$ then $\theta_1(n, m) \leftarrow 0$ .
   (c) For all $d \in [g]$ in order:
      i. For all vertices $m \in \mathcal{B}$ at depth $g-d$ set $\theta_1(n, m) \leftarrow \theta_1(n, \triangleleft(m)) + \theta_1(n, \triangleright(m))$ .

$\mathbf{Prediction}_t(u_t)$ :

1. Draw $\delta_t$ from $\Uparrow(u_t)$ with probability $\mathbb{P}[\delta_t = n] \propto \mu_t(n)\theta_t(n, r)$ .
2. $\zeta_{t,0} \leftarrow r$ .
3. For all $d \in [g-1] \cup \{0\}$ in order:
   (a) Draw $\zeta_{t,d+1}$ from $\{\triangleleft(\zeta_{t,d}), \triangleright(\zeta_{t,d})\}$ with probability $\mathbb{P}[\zeta_{t,d+1} = m] \propto \theta_t(\delta_t, m)$ .
4. $a_t \leftarrow \zeta_{t,g}$ .
5. Return $a_t$ .

$\mathbf{Update}_t(\ell_{t,a_t})$ :

1. $\psi_t \leftarrow \sum_{n \in \Uparrow(u_t)} \mu_t(n)\theta(n, r)$ .
2. $\varrho_t \leftarrow \sum_{n \in \Uparrow(u_t)} \mu_t(n)\theta(n, a_t)$ .
3. $\bar{\lambda}_t \leftarrow \exp(-\eta \ell_{t,a_t} \psi_t / \varrho_t)$ .
4. For all $n \in \Uparrow(u_t)$:
   (a) $\mu_{t+1}(n) \leftarrow \mu_t(n)\psi_t / (\psi_t - (1 - \bar{\lambda}_t)\varrho_t)$ .
   (b) $\theta_{t+1}(n, a_t) \leftarrow \bar{\lambda}_t \theta_t(n, a_t)$ .
   (c) For all $m \in \mathcal{B} \setminus \Uparrow(a_t)$ :
      i. $\theta_{t+1}(n, m) := \theta_t(n, m)$ .
   (d) For all $d \in [g]$ in order:
      i. $\theta_{t+1}(n, \zeta_{t,(g-d)}) \leftarrow \theta_{t+1}(n, \triangleleft(\zeta_{t,(g-d)})) + \theta_{t+1}(n, \triangleright(\zeta_{t,(g-d)}))$ .
5. For all $n \in \mathcal{C} \setminus \Uparrow(u_t)$ :
   (a) $\mu_{t+1}(n) := \mu_t(n)$ .
   (b) $\forall m \in \mathcal{B}, \theta_{t+1}(n, m) := \theta_t(n, m)$ .

Figure 9: GABA-II Subroutines

**Definition 28.** *For all $t \in [T]$ and $a \in [K]$ we define:*

$$\bar{p}_{t,a} := \sum_{n \in \mathcal{C}: n \in \Uparrow(u_t)} \bar{\kappa}_t(a, n).$$

The following lemma states that GABA-II maintains a factorisation of the values $\bar{\kappa}_t(a, n)$ :

**Lemma 29.** *Given $t \in [T]$, $n \in \mathcal{C}$ and $a \in [K]$ we have:*

$$\bar{\kappa}_t(a, n) = \frac{\mu_t(n)\theta_t(n, a)}{(2N - 1)K}.$$

The following lemma comes from how we update the values $\theta(n, m)$ :

**Lemma 30.** *Given $t \in [T]$, $n \in \mathcal{C}$ and $m \in \mathcal{B}$ we have:*

$$\theta_t(n, m) = \sum_{a \in \Downarrow(m) \cap [K]} \theta_t(n, a).$$

Lemmas 29 and 30 then allow us to quantify the probability distribution that $a_t$ is selected from:

**Lemma 31.** *For all $t \in [T]$ and $a \in [K]$ we have:*

$$\mathbb{P}[a_t = a] = \frac{\bar{p}_{t,a}}{\|\bar{\boldsymbol{p}}_t\|_1}.$$

Lemmas 29 and 30 also allow us to quantify $\bar{\lambda}_t$ and $\bar{z}_t$ in terms of $\bar{\boldsymbol{p}}_t$ :

**Lemma 32.** *For all $t \in [T]$ we have:*

$$\bar{\lambda}_t = \exp\left(\frac{-\eta \ell_{t,a_t} \|\bar{\boldsymbol{p}}_t\|_1}{\bar{p}_{t,a_t}}\right)$$

*and:*

$$\bar{z}_t = \frac{\|\bar{\boldsymbol{p}}_t\|_1}{\|\bar{\boldsymbol{p}}_t\|_1 - (1 - \bar{\lambda}_t)\bar{p}_{t,a_t}}.$$

Lemmas 31 and 32 imply that GABA-II does indeed implement PROTOGABA:

**Lemma 33.** GABA-II *implements* PROTOGABA *with $\mathcal{E}$ and $\sigma$ defined as in Definition 25*

We choose, as required in the analysis of PROTOGABA, the set $\mathcal{A}$ to be as follows:

**Definition 34.** *Let $\mathcal{A}^\dagger$ be the set of all $(a, n) \in [K] \times \mathcal{C}$ such that:*

- *For all $u \in \Downarrow(n)$ we have $y(u) = a$.*
- *$n$ is the root of $\mathcal{C}$ or there exists $v \in \Downarrow(\uparrow(n))$ with $y(v) \neq a$.*

*Then define:*

$$\mathcal{A} = \{(a, \Downarrow(n)) \mid (a, n) \in \mathcal{A}^\dagger\}.$$

The following lemma states that $\mathcal{A}$ is valid:

**Lemma 35.** *We have that:*

- *For all $u \in [N]$ there exists a unique pair $(a, U) \in \mathcal{A}$ with $u \in U$.*
- *For all $u \in [N]$ and $(a, U) \in \mathcal{A}$ with $u \in U$, we have that $a = y(u)$.*

Lemma 2 allows us to bound the expected cardinality of $\mathcal{A}$ :

**Lemma 36.** *We have:*

$$\mathbb{E}[|\mathcal{A}|] \leq 4\Psi \log_2\left(\frac{eN}{\Psi}\right).$$

Lemmas 33 and 35 allow us to invoke Theorem 9 which, combined with Lemma 36 gives us our final result:

**Theorem 37.** *Given $\Psi \leq N/2$ and setting:*

$$\eta := \sqrt{\frac{8\Psi \log_2 (eN/\Psi) \ln (3KN/2\Psi)}{KT}}$$

*we have:*

$$R \in \mathcal{O}\left(\sqrt{\ln\left(\frac{N}{\Psi}\right) \ln\left(\frac{KN}{\Psi}\right) \Psi KT}\right).$$

We now argue the per trial time complexity of GABA-II. In the prediction algorithm $\delta_t$ is first sampled from the ancestors of $u_t$, taking a time of $\mathcal{O}(\ln(N))$. Then, in selecting $a_t$ , a path of length $\mathcal{O}(\ln(K))$ in $\mathcal{B}$ is sampled, with each vertex taking a time of $\mathcal{O}(1)$ to sample. In the update algorithm $\psi_t$ and $\varrho_t$ take a time of $\mathcal{O}(\ln(N))$ to compute and then the values $\mu_t(n)$ and $\theta_t(n, m)$ are only modified if $n$ is one of the $\mathcal{O}(\ln(N))$ ancestors of $u_t$ and $m$ is one of the $\mathcal{O}(\ln(K))$ ancestors of $a_t$ , with each update taking $\mathcal{O}(1)$ time. This gives us the following proposition:

**Proposition 38.** GABA-II *takes a per-trial time of* $\mathcal{O}(\ln(K)\ln(N))$.

# H PROTOGABA Proofs

## H.1 Proof of Lemma 7

**Result**. For all $t \in [T]$ we have:

$$\mathbb{E}\left[\sum_{(a,U)\in\mathcal{A}} \ln\left(\frac{\kappa_{t+1}(a, U)}{\kappa_t(a, U)}\right)\right] \geq \eta\left(\mathbb{E}\left[\ell_{t,a_t}\right] - \ell_{t,y(u_t)}\right) - \frac{K\eta^2}{2}.$$

**Proof:**

**Definition 39.** *Let $\tilde{\boldsymbol{p}}_t := \boldsymbol{p}_t/\|\boldsymbol{p}_t\|_1$ and for all $a \in [K]$ let $c_{t,a} := [\![a_t = a]\!]\lambda_t + [\![a_t \neq a]\!]$*

**Lemma 40.** *We have:*

$$\sum_{(a,U)\in\mathcal{A}} \ln\left(\frac{\kappa_{t+1}(a, U)}{\kappa_t(a, U)}\right) = \ln\left(\frac{c_{t,y(u_t)}}{\boldsymbol{c}_t \cdot \tilde{\boldsymbol{p}}_t}\right).$$

*Proof.* Let $(y(u_t), U')$ be the unique pair $(a, U) \in \mathcal{A}$ with $u_t \in U$. For all $(a, U) \in \mathcal{A}\backslash\{(y(u_t), U')\}$ we have $u_t \notin U$ so $\kappa_{t+1}(a, U) = \kappa_t(a, U)$ and hence $\ln(\kappa_{t+1}(a, U)/\kappa_t(a, U)) = 0$ so:

$$\begin{aligned}
\sum_{(a,U)\in\mathcal{A}} \ln\left(\frac{\kappa_{t+1}(a, U)}{\kappa_t(a, U)}\right) &= \ln\left(\frac{\kappa_{t+1}(y(u_t), U')}{\kappa_t(y(u_t), U')}\right) \\
&= \ln\left(\frac{\kappa_t(y(u_t), U')([\![a_t = y(u_t)]\!]\lambda_t + [\![a_t \neq y(u_t)]\!])z_t}{\kappa_t(y(u_t), U')}\right) \\
&= \ln(([\![a_t = y(u_t)]\!]\lambda_t + [\![a_t \neq y(u_t)]\!])z_t) \\
&= \ln(c_{t,y(u_t)}z_t) \\
&= \ln\left(c_{t,y(u_t)}\frac{\|\boldsymbol{p}_t\|_1}{\|\boldsymbol{p}_t\|_1 - (1-\lambda_t)p_{t,a_t}}\right) \\
&= \ln\left(c_{t,y(u_t)}\frac{\|\boldsymbol{p}_t\|_1}{\lambda_t p_{t,a_t} + \sum_{a\in[K]\backslash\{a_t\}} p_{t,a}}\right) \\
&= \ln\left(c_{t,y(u_t)}\frac{\|\boldsymbol{p}_t\|_1}{c_{t,a_t}p_{t,a_t} + \sum_{a\in[K]\backslash\{a_t\}} c_{t,a}p_{t,a}}\right) \\
&= \ln\left(c_{t,y(u_t)}\frac{\|\boldsymbol{p}_t\|_1}{\boldsymbol{c}_t \cdot \boldsymbol{p}_t}\right)
\end{aligned}$$

$$= \ln\left(\frac{c_{t,y(u_t)}}{\boldsymbol{c}_t \cdot \tilde{\boldsymbol{p}}_t}\right)$$

as required. $\square$

**Lemma 41.** *We have:*

$$\mathbb{E}\left[\ln\left(\frac{c_{t,y(u_t)}}{\boldsymbol{c}_t \cdot \tilde{\boldsymbol{p}}_t}\right)\right] \geq \eta\left(\mathbb{E}\left[\ell_{t,a_t}\right] - \ell_{t,y(u_t)}\right) - \frac{K\eta^2}{2}\,.$$

*Proof.* Noting that, for all $a \in [K]$:

$$c_{t,a} = \exp\left(-\frac{\eta[\![a = a_t]\!]\ell_{t,a_t}}{\tilde{p}_{t,a}}\right)$$

we have:

$$\begin{aligned}
&\ln\left(\frac{c_{t,y(u_t)}}{\tilde{\boldsymbol{p}}_t \cdot \boldsymbol{c}_t}\right)\\
&= \ln(c_{t,y(u_t)}) - \ln(\tilde{\boldsymbol{p}}_t \cdot \boldsymbol{c}_t)\\
&= \ln(c_{t,y(u_t)}) - \ln\left(\sum_{a\in[K]} \tilde{p}_{t,a} c_{t,a}\right)\\
&= -\frac{\eta[\![y(u_t) = a_t]\!]\ell_{t,a_t}}{\tilde{p}_{t,y(u_t)}} - \ln\left(\sum_{a\in[K]} \tilde{p}_{t,a} \exp\left(\frac{-\eta[\![a = a_t]\!]\ell_{t,a_t}}{\tilde{p}_{t,a}}\right)\right)\\
&\geq -\frac{\eta[\![y(u_t) = a_t]\!]\ell_{t,a_t}}{\tilde{p}_{t,y(u_t)}} - \ln\left(\sum_{a\in[K]} \tilde{p}_{t,a}\left(1 - \frac{\eta[\![a = a_t]\!]\ell_{t,a_t}}{\tilde{p}_{t,a}} + \frac{\eta^2[\![a = a_t]\!](\ell_{t,a_t})^2}{2(\tilde{p}_{t,a})^2}\right)\right) \qquad (15)\\
&= -\frac{\eta[\![y(u_t) = a_t]\!]\ell_{t,a_t}}{\tilde{p}_{t,y(u_t)}} - \ln\left(\sum_{a\in[K]} \tilde{p}_{t,a} - \sum_{a\in[K]} \eta[\![a = a_t]\!]\ell_{t,a_t} + \sum_{a\in[K]} \frac{\eta^2[\![a = a_t]\!](\ell_{t,a_t})^2}{2\tilde{p}_{t,a}}\right)\\
&= -\frac{\eta[\![y(u_t) = a_t]\!]\ell_{t,a_t}}{\tilde{p}_{t,y(u_t)}} - \ln\left(1 - \sum_{a\in[K]} \eta[\![a = a_t]\!]\ell_{t,a_t} + \sum_{a\in[K]} \frac{\eta^2[\![a = a_t]\!](\ell_{t,a_t})^2}{2\tilde{p}_{t,a}}\right)\\
&= -\frac{\eta[\![y(u_t) = a_t]\!]\ell_{t,a_t}}{\tilde{p}_{t,y(u_t)}} - \ln\left(1 - \eta\ell_{t,a_t} + \frac{\eta^2(\ell_{t,a_t})^2}{2\tilde{p}_{t,a_t}}\right)\\
&\geq -\frac{\eta[\![y(u_t) = a_t]\!]\ell_{t,a_t}}{\tilde{p}_{t,y(u_t)}} + \eta\ell_{t,a_t} - \frac{\eta^2(\ell_{t,a_t})^2}{2\tilde{p}_{t,a_t}} \qquad (16)\\
&= \eta\ell_{t,a_t} - \frac{\eta[\![y(u_t) = a_t]\!]\ell_{t,a_t}}{\tilde{p}_{t,y(u_t)}} - \frac{\eta^2(\ell_{t,a_t})^2}{2\tilde{p}_{t,a_t}}
\end{aligned}$$

where inequalities (15) and (16) are since $\exp(-x) \leq 1 - x + x^2/2$ for $x \geq 0$ and $\ln(1 + x) \leq x$ respectively. This implies:

$$\begin{aligned}
\mathbb{E}\left[\ln\left(\frac{c_{t,y(u_t)}}{\tilde{\boldsymbol{p}}_t \cdot \boldsymbol{c}_t}\right)\right] &\geq \sum_{a\in[K]} \mathbb{P}\left[a_t = a\right]\left(\eta\ell_{t,a} - \frac{\eta[\![y(u_t) = a]\!]\ell_{t,a}}{\tilde{p}_{t,y(u_t)}} - \frac{\eta^2(\ell_{t,a})^2}{2\tilde{p}_{t,a}}\right)\\
&= \sum_{a\in[K]} \tilde{p}_{t,a}\left(\eta\ell_{t,a} - \frac{\eta[\![y(u_t) = a]\!]\ell_{t,a}}{\tilde{p}_{t,y(u_t)}} - \frac{\eta^2(\ell_{t,a})^2}{2\tilde{p}_{t,a}}\right)\\
&= \eta\sum_{a\in[K]} \tilde{p}_{t,a}\ell_{t,a} - \sum_{a\in[K]} \frac{\eta\tilde{p}_{t,a}[\![y(u_t) = a]\!]\ell_{t,a}}{\tilde{p}_{t,y(u_t)}} - \sum_{a\in[K]} \frac{\eta^2(\ell_{t,a})^2}{2}\\
&= \eta\sum_{a\in[K]} \tilde{p}_{t,a}\ell_{t,a} - \frac{\eta\tilde{p}_{t,y(u_t)}\ell_{t,y(u_t)}}{\tilde{p}_{t,y(u_t)}} - \sum_{a\in[K]} \frac{\eta^2(\ell_{t,a})^2}{2}
\end{aligned}$$

$$
\begin{aligned}
&= \eta \sum_{a\in[K]} \tilde{p}_{t,a}\ell_{t,a} - \eta\ell_{t,y(u_t)} - \sum_{a\in[K]} \frac{\eta^2(\ell_{t,a})^2}{2} \\
&= \eta \sum_{a\in[K]} \mathbb{P}\left[a_t = a\right]\ell_{t,a_t} - \eta\ell_{t,y(u_t)} - \sum_{a\in[K]} \frac{\eta^2(\ell_{t,a})^2}{2} \\
&= \eta\mathbb{E}\left[\ell_{t,a_t}\right] - \eta\ell_{t,y(u_t)} - \sum_{a\in[K]} \frac{\eta^2(\ell_{t,a})^2}{2} \\
&\geq \eta\mathbb{E}\left[\ell_{t,a_t}\right] - \eta\ell_{t,y(u_t)} - \sum_{a\in[K]} \frac{\eta^2}{2} \\
&= \eta\mathbb{E}\left[\ell_{t,a_t}\right] - \eta\ell_{t,y(u_t)} - \frac{K\eta^2}{2}
\end{aligned}
$$

as required. □

Lemmas 40 and 41 imply the result. ■

### H.2 Proof of Lemma 8

**Result**. We have:

$$\sum_{(a,U)\in\mathcal{A}} \ln\left(|\mathcal{A}|\kappa_{T+1}(a,U)\right) \leq 0\,.$$

**Proof:**

**Definition 42.** *Given a finite set $X$ let $\Delta_X$ be the set of functions $f$ from $X$ into $\mathbb{R}^+$ such that $\sum_{x\in X} f(x) = 1$.*

**Lemma 43.** *For all $t \in [T+1]$ we have $\kappa_t \in \Delta_{[K]\times\mathcal{E}}$.*

*Proof.* We prove via induction over $t$. For $t = 1$ the result holds as:

$$
\begin{aligned}
\sum_{(a,U)\in[K]\times\mathcal{E}} \kappa_1(a,U) &= \sum_{(a,U)\in[K]\times\mathcal{E}} \frac{\sigma(U)}{K} \\
&= \frac{1}{K}\sum_{a\in[K]}\sum_{U\in\mathcal{E}}\sigma(U) \\
&= \frac{1}{K}\sum_{a\in[K]} 1 \\
&= 1\,.
\end{aligned}
$$

Now suppose it holds for $t = s$ (for some $s \in [T]$). We now show it holds for $t = s+1$, completing the proof. We have:

$$
\begin{aligned}
&\sum_{(a,U)\in[K]\times\mathcal{E}:u_s\in U} \kappa_{s+1}(a,U) \\
&= \sum_{U\in\mathcal{E}:u_s\in U}\left(\kappa_{s+1}(a_s,U) + \sum_{a\in[K]\setminus\{a_s\}} \kappa_{s+1}(a,U)\right) \\
&= \sum_{U\in\mathcal{E}:u_s\in U}\left(\lambda_s z_s\kappa_s(a_s,U) + \sum_{a\in[K]\setminus\{a_s\}} z_s\kappa_s(a,U)\right) \\
&= z_s\left(\lambda_s\sum_{U\in\mathcal{E}:u_s\in U}\kappa_s(a_s,U) + \sum_{a\in[K]\setminus\{a_s\}}\sum_{U\in\mathcal{E}:u_s\in U}\kappa_s(a,U)\right)
\end{aligned}
$$

$$
\begin{aligned}
&= z_s \left( \lambda_s p_{s,a_s} + \sum_{a \in [K] \setminus \{a_s\}} p_{s,a} \right) \\
&= z_s \left( \lambda_s p_{s,a_s} + (\|\boldsymbol{p}_s\|_1 - p_{s,a_s}) \right) \\
&= z_s \left( \|\boldsymbol{p}_s\|_1 - (1-\lambda_s) p_{s,a_s} \right) \\
&= z_s \frac{\|\boldsymbol{p}_s\|_1}{z_s} \\
&= \|\boldsymbol{p}_s\|_1 \,,
\end{aligned}
$$

so:

$$
\begin{aligned}
&\sum_{(a,U) \in [K] \times \mathcal{E}} \kappa_{s+1}(a,U) \\
&= \sum_{(a,U) \in [K] \times \mathcal{E} : u_s \in U} \kappa_{s+1}(a,U) + \sum_{(a,U) \in [K] \times \mathcal{E} : u_s \notin U} \kappa_{s+1}(a,U) \\
&= \sum_{(a,U) \in [K] \times \mathcal{E} : u_s \in U} \kappa_{s+1}(a,U) + \sum_{(a,U) \in [K] \times \mathcal{E} : u_s \notin U} \kappa_{s}(a,U) \\
&= \|\boldsymbol{p}_s\|_1 + \sum_{(a,U) \in [K] \times \mathcal{E} : u_s \notin U} \kappa_{s}(a,U) \\
&= \sum_{a \in [K]} p_{s,a} + \sum_{(a,U) \in [K] \times \mathcal{E} : u_s \notin U} \kappa_{s}(a,U) \\
&= \sum_{a \in [K]} \sum_{U \in \mathcal{E} : u_s \in U} \kappa_s(a,U) + \sum_{(a,U) \in [K] \times \mathcal{E} : u_s \notin U} \kappa_{s}(a,U) \\
&= \sum_{(a,U) \in [K] \times \mathcal{E}} \kappa_s(a,U) \\
&= 1
\end{aligned}
$$

as required. □

**Definition 44.** *We define the function $\kappa^* \in \Delta_{[K] \times \mathcal{E}}$ by:*

- *For all $(a,U) \in \mathcal{A}$ we have $\kappa^*(a,U) := 1/|\mathcal{A}|$.*
- *For all $(a,U) \in ([K] \times \mathcal{E}) \setminus \mathcal{A}$ we have $\kappa^*(a,U) := 0$.*

**Lemma 45.** *We have:*

$$
\sum_{(a,U) \in [K] \times \mathcal{E}} \kappa^*(a,U) \ln \left( \frac{\kappa^*(a,U)}{\kappa_{T+1}(a,U)} \right) \geq 0 \,.
$$

*Proof.* Given a finite set $X$ and functions $f, f' \in \Delta_X$, the value $\sum_{x \in X} f(x) \ln \left( \frac{f(x)}{f'(x)} \right)$ is their relative entropy and is hence positive. The result then follows by Lemma 43 (i.e. that $\kappa_{T+1} \in \Delta_{[K] \times \mathcal{E}}$) and the fact that $\kappa^* \in \Delta_{[K] \times \mathcal{E}}$ □

By taking limits we have $\kappa^*(a,U) \ln (\kappa^*(a,U)/\kappa_{T+1}(a,U)) = 0$ whenever $(a,U) \notin \mathcal{A}$. By definition of $\kappa^*$ we then have:

$$
\begin{aligned}
-\sum_{(a,U) \in \mathcal{A}} \ln (|\mathcal{A}| \kappa_{T+1}(a,U)) &= |\mathcal{A}| \sum_{(a,U) \in \mathcal{A}} \frac{1}{|\mathcal{A}|} \ln \left( \frac{1/|\mathcal{A}|}{\kappa_{T+1}(a,U)} \right) \\
&= \frac{1}{|\mathcal{A}|} \sum_{(a,U) \in [K] \times \mathcal{E}} \kappa^*(a,U) \ln \left( \frac{\kappa^*(a,U)}{\kappa_{T+1}(a,U)} \right)
\end{aligned}
$$

which, by Lemma 45 is bounded below by zero. This implies the result. ■

### H.3 Proof of Theorem 9

**Result**. We have:

$$R \leq \frac{1}{\eta} \sum_{(a,U)\in\mathcal{A}} \ln\left(\frac{K}{|\mathcal{A}|\sigma(U)}\right) + \frac{\eta K T}{2}\,.$$

**Proof:**

For all $(a, U) \in \mathcal{A}$ we have:

$$\begin{aligned}
\ln(|\mathcal{A}|\kappa_{T+1}(a,U)) - \ln(|\mathcal{A}|\kappa_1(a,U)) &= \sum_{t\in[T]} \left(\ln(|\mathcal{A}|\kappa_{t+1}(a,U)) - \ln(|\mathcal{A}|\kappa_t(a,U))\right) \\
&= \sum_{t\in[T]} \ln\left(\frac{|\mathcal{A}|\kappa_{t+1}(a,U)}{|\mathcal{A}|\kappa_t(a,U)}\right) \\
&= \sum_{t\in[T]} \ln\left(\frac{\kappa_{t+1}(a,U)}{\kappa_t(a,U)}\right)
\end{aligned}$$

so:

$$\sum_{(a,U)\in\mathcal{A}} \left(\ln(|\mathcal{A}|\kappa_{T+1}(a,U)) - \ln(|\mathcal{A}|\kappa_1(a,U))\right) = \sum_{t\in[T]} \sum_{(a,U)\in\mathcal{A}} \ln\left(\frac{\kappa_{t+1}(a,U)}{\kappa_t(a,U)}\right)\,.$$

Applying Lemma 7 then gives us:

$$\begin{aligned}
\mathbb{E}\left[\sum_{(a,U)\in\mathcal{A}} \left(\ln(|\mathcal{A}|\kappa_{T+1}(a,U)) - \ln(|\mathcal{A}|\kappa_1(a,U))\right)\right] &\geq \sum_{t\in[T]} \left(\eta\left(\mathbb{E}\left[\ell_{t,a_t}\right] - \ell_{t,y(u_t)}\right) - \frac{K\eta^2}{2}\right) \\
&= \sum_{t\in[T]} \left(\eta\mathbb{E}\left[\ell_{t,a_t} - \ell_{t,y(u_t)}\right] - \frac{K\eta^2}{2}\right) \\
&= \eta\sum_{t\in[T]} \mathbb{E}\left[\ell_{t,a_t} - \ell_{t,y(u_t)}\right] - \frac{TK\eta^2}{2} \\
&= \eta\mathbb{E}\left[\sum_{t\in[T]} (\ell_{t,a_t} - \ell_{t,y(u_t)})\right] - \frac{TK\eta^2}{2}\,,
\end{aligned}$$

so:

$$\mathbb{E}\left[\sum_{t\in[T]} (\ell_{t,a_t} - \ell_{t,y(u_t)})\right] \leq \frac{1}{\eta}\mathbb{E}\left[\sum_{(a,U)\in\mathcal{A}} \ln(|\mathcal{A}|\kappa_{T+1}(a,U))\right] - \frac{1}{\eta}\mathbb{E}\left[\sum_{(a,U)\in\mathcal{A}} \ln(|\mathcal{A}|\kappa_1(a,U))\right] + \frac{TK\eta}{2}\,.$$

Applying Lemma 8 then gives us:

$$\mathbb{E}\left[\sum_{t\in[T]} (\ell_{t,a_t} - \ell_{t,y(u_t)})\right] \leq -\frac{1}{\eta}\mathbb{E}\left[\sum_{(a,U)\in\mathcal{A}} \ln(|\mathcal{A}|\kappa_1(a,U))\right] + \frac{TK\eta}{2}\,,$$

and the definition of $\kappa_1$ then gives us:

$$\mathbb{E}\left[\sum_{t\in[T]} \left(\ell_{t,a_t} - \ell_{t,y(u_t)}\right)\right] \leq \frac{1}{\eta}\mathbb{E}\left[\sum_{(a,U)\in\mathcal{A}} \ln\left(\frac{K}{|\mathcal{A}|\sigma(U)}\right)\right] + \frac{\eta K T}{2}$$

as required. ■

# I GABA-I Proofs

## I.1 Proof of Lemma 11

**Result**. We have that $\sigma$ is a probability distribution, in that:

$$\sum_{U \in \mathcal{E}} \sigma(U) = 1 \,.$$

**Proof:**

**Definition 46.** *For $f : [N] \rightarrow \{0,1\}$ define:*

$$\hat{\sigma}(f) := \iota_{f(1)} \prod_{u \in [N-1]} \tau_{f(u),f(u+1)} \,.$$

**Definition 47.** *For $U \in \mathcal{E}$ let $\gamma_U : [N] \rightarrow \{0,1\}$ be such that for all $u \in [N]$ we have $\gamma_U(u) := [\![u \in U]\!]$.*

Since $\tau_{0,0} + \tau_{0,1} = 1$ , $\tau_{1,0} + \tau_{1,1} = 1$ and $\tau_{i,j} \geq 0$ for all $i, j \in \{0,1\}$ we have that $\boldsymbol{\tau}$ is the transistion matrix of a Markov chain. Since also $\iota_0 + \iota_1 = 1$ and $\iota_i \geq 0$ for all $i \in \{0,1\}$, we then have, for all $f : [N] \rightarrow \{0,1\}$ , that $\hat{\sigma}(f)$ is the probability of $f$ in a Markov chain. This implies that $\sum_{f:[N]\rightarrow\{0,1\}} \hat{\sigma}(f) = 1$ so:

$$\begin{aligned}
\sum_{U \in \mathcal{E}} \sigma(U) &= \sum_{U \in \mathcal{E}} \iota_{[\![1 \in U]\!]} \prod_{u \in [N-1]} \tau_{[\![u \in U]\!],[\![u+1 \in U]\!]} \\
&= \sum_{U \in \mathcal{E}} \iota_{\gamma_U(1)} \prod_{u \in [N-1]} \tau_{\gamma_U(u),\gamma_U(u+1)} \\
&= \sum_{U \in 2^{[N]}} \iota_{\gamma_U(1)} \prod_{u \in [N-1]} \tau_{\gamma_U(u),\gamma_U(u+1)} \\
&= \sum_{f:[N]\rightarrow\{0,1\}} \iota_{f(1)} \prod_{u \in [N-1]} \tau_{f(u),f(u+1)} \\
&= \sum_{f:[N]\rightarrow\{0,1\}} \hat{\sigma}(f) \\
&= 1
\end{aligned}$$

as required. ■

## I.2 Proof of Lemma 16

**Result**. Given $i, j, k \in \{0,1\}$ , $u, v, w \in [N]$ with $u < v < w$ , and $a \in [K]$ we have:

$$\left(\sum_{f \in F} \Omega_{t,a}(f)\right)\left(\sum_{f \in G} \Omega_{t,a}(f)\right) = \sum_{f \in H} \Omega_{t,a}(f) \,,$$

where:

- $F = \{f \in I(u,v) \mid f(u) = i \,, f(v) = j\}$
- $G = \{f \in I(v,w) \mid f(v) = j \,, f(w) = k\}$
- $H = \{f \in I(u,w) \mid f(u) = i \,, f(v) = j \,, f(w) = k\}$ .

**Proof:**

Given $(f, f') \in F \times G$ let $\xi_{f,f'}$ be the function in $H$ defined by $\xi_{f,f'}(u') := f(u')$ for $u \leq u' \leq v$ and $\xi_{f,f'}(u') := f'(u')$ for $v \leq u' \leq w$. Note first that by definition of $\Omega_{t,a}$ we have, for all

$(f, f') \in F \times G$, that $\Omega_{t,j}(f)\Omega_{t,a}(f') = \Omega_{t,j}(\xi_{f,f'})$. Note also that the function $(f, f') \to \xi_{f,f'}$ is a bijection from $F \times G$ into $H$. Hence, we have:

$$\begin{aligned}
\left(\sum_{f\in F}\Omega_{t,a}(f)\right)\left(\sum_{f\in G}\Omega_{t,a}(f)\right) &= \left(\sum_{f\in F}\Omega_{t,a}(f)\right)\left(\sum_{f'\in G}\Omega_{t,a}(f')\right)\\
&= \sum_{f\in F}\sum_{f'\in G}\Omega_{t,a}(f)\Omega_{t,a}(f')\\
&= \sum_{(f,f')\in F\times G}\Omega_{t,a}(f)\Omega_{t,a}(f')\\
&= \sum_{(f,f')\in F\times G}\Omega_{t,a}(\xi_{f,f'})\\
&= \sum_{f^\dagger\in H}\Omega_{t,a}(f^\dagger)
\end{aligned}$$

as required. ■

### I.3 Proof of Lemma 17

**Result**. For all $t \in [T] \cup \{0\}$, all non-root vertices $n$ of $\mathcal{C}$, all $i, j \in \{0, 1\}$ and all $a \in [K]$ we have:

$$\alpha_t(n, i, j)_a = \sum_{f\in A_t(n,i,j)}\Omega_{t,a}(f)\,,$$

where:

$$A_t(n, i, j) := \{f \in I(\blacktriangleleft(n), \blacktriangleright(n) + 1) \mid f(\blacktriangleleft(n)) = i\,, f(\blacktriangleright(n) + 1) = j\}\,.$$

**Proof:**

**Lemma 48.** *For all $t \in [T] \cup \{0\}$, $u \in [N]$, $i, j \in \{0, 1\}$ and $a \in [K]$, we have:*

$$\alpha_t(u, i, j)_a = \tau_{i,j}([\![i = 0]\!] + [\![i = 1]\!]\xi_{t,u,a})\,.$$

*Proof.* We prove by induction on $t$. By the initialization algorithm we have that:

$$\begin{aligned}
\alpha_0(u, i, j)_a &= \tau_{i,j}\\
&= \tau_{i,j}([\![i = 0]\!] + [\![i = 1]\!])\\
&= \tau_{i,j}([\![i = 0]\!] + [\![i = 1]\!]\xi_{0,u,a})
\end{aligned}$$

so the result holds for $t = 0$. Now suppose the result holds for $t = s$ (for some $s \in [T] \cup \{0\}$). By the update algorithm on trial $s$ and the inductive hypothesis we then have:

- If $u \neq u_s$ then:

$$\begin{aligned}
\alpha_{s+1}(u, i, j)_a &:= \alpha_s(u, i, j)_a\\
&= \tau_{i,j}([\![i = 0]\!] + [\![i = 1]\!]\xi_{s,u,a})\\
&= \tau_{i,j}[\![i = 0]\!] + \tau_{i,j}[\![i = 1]\!]\prod_{q\in[s-1]:u=u_q}\pi_{q,a}\\
&= \tau_{i,j}[\![i = 0]\!] + \tau_{i,j}[\![i = 1]\!]\prod_{q\in[s]:u=u_q}\pi_{q,a}\\
&= \tau_{i,j}[\![i = 0]\!] + \tau_{i,j}[\![i = 1]\!]\xi_{s+1,u,a}
\end{aligned}$$

- If $u = u_s$ then:

$$\alpha_{s+1}(u, i, j)_a$$

$$
\begin{aligned}
&:= [\![i=0]\!]\alpha_s(u,i,j)_a + [\![i=1]\!]\pi_{s,a}\alpha_s(u,i,j)_a \\
&= [\![i=0]\!]\tau_{i,j}([\![i=0]\!] + [\![i=1]\!]\xi_{s,u,a}) + [\![i=1]\!]\pi_{s,a}\tau_{i,j}([\![i=0]\!] + [\![i=1]\!]\xi_{s,u,a}) \\
&= [\![i=0]\!]\tau_{i,j} + [\![i=1]\!]\pi_{s,a}\tau_{i,j}\xi_{s,u,a} \\
&= [\![i=0]\!]\tau_{i,j} + [\![i=1]\!]\tau_{i,j}\pi_{s,a}\prod_{q\in[s-1]:u=u_q}\pi_{q,a} \\
&= [\![i=0]\!]\tau_{i,j} + [\![i=1]\!]\tau_{i,j}\prod_{q\in[s]:u=u_q}\pi_{q,a} \\
&= \tau_{i,j}[\![i=0]\!] + \tau_{i,j}[\![i=1]\!]\xi_{s,u,a}
\end{aligned}
$$

as required. ☐

**Lemma 49.** *For all $t \in [T] \cup \{0\}$, all internal, non-root vertices $n$ of $\mathcal{C}$, all $i, j \in \{0,1\}$ and all $a \in [K]$ we have:*

$$\alpha_t(n,i,j)_a = \sum_{k\in\{0,1\}} \alpha_t(\triangleleft(n),i,k)_a\alpha_t(\triangleright(n),k,j)_a\,.$$

*Proof.* We prove by induction on $t$. For $t = 0$ the result is direct from the initialization algorithm. Now suppose the result holds for $t = s$ (for some $s \in [T-1] \cup \{0\}$). We now show that it holds for $t = s+1$, completing the proof. First, suppose $n \neq \nu_{s,d}$ for all $d \in [h]$. Since, in this case, we must also have that $\triangleleft(n), \triangleright(n) \neq \nu_{s,d}$ for all $d \in [h]$ we then have, by the update algorithm, that, for $k \in \{0,1\}$, $\alpha_{s+1}(\triangleleft(n),i,k)_a := \alpha_s(\triangleleft(n),i,k)_a$ and $\alpha_{s+1}(\triangleright(n),k,j)_a := \alpha_s(\triangleright(n),k,j)_a$ so hence, by the inductive hypothesis and the update algorithm:

$$
\begin{aligned}
\alpha_{s+1}(n,i,j)_a &:= \alpha_s(n,i,j)_a \\
&= \sum_{k\in\{0,1\}} \alpha_s(\triangleleft(n),i,k)_a\alpha_s(\triangleright(n),k,j)_a \\
&= \sum_{k\in\{0,1\}} \alpha_{s+1}(\triangleleft(n),i,k)_a\alpha_{s+1}(\triangleright(n),k,j)_a
\end{aligned}
$$

as required. On the other hand, it is clear from the update algorithm that if $n = \nu_{t,d}$ for some $d \in [h]$ then:

$$\alpha_{s+1}(n,i,j)_a := \sum_{k\in\{0,1\}} \alpha_{s+1}(\triangleleft(n),i,k)_a\alpha_{s+1}(\triangleright(n),k,j)_a$$

as required. ☐

With lemmas 48 and 49 at hand we prove the result by reverse induction on the depth of $n$ (i.e. from depth $h$ to depth 1). We first consider the case that $n$ is of depth $h$. In this case we have that $n$ is a leaf and have $\blacktriangleleft(n) = \blacktriangleright(n) = n$. This means that $I(\blacktriangleleft(n), \blacktriangleright(n)+1)$ is the set of functions from $\{n, n+1\}$ into $\{0,1\}$ and hence the set $\{f \in I(\blacktriangleleft(n), \blacktriangleright(n)+1) \mid f(\blacktriangleleft(n)) = i\,, f(\blacktriangleright(n)+1) = j\}$ contains a single function $f$ with $f(n) = i$ and $f(n+1) = j$. We then have, by Lemma 48 , that:

$$
\begin{aligned}
\Omega_{t,a}(f) &= \tau_{f(n),f(n+1)}([\![f(n)=0]\!] + [\![f(n)=1]\!]\xi_{t,n,a}) \\
&= \tau_{i,j}([\![i=0]\!] + [\![i=1]\!]\xi_{t,n,a}) \\
&= \alpha_t(n,i,j)_a
\end{aligned}
$$

as required. So the inductive hypothesis holds for $n$ at depth $h$. Now suppose the inductive hypothesis holds for $n$ at depth $d$ (for some $d \in [h]$). We now show it holds for $n$ at depth $d-1$ which will complete the proof. For $k \in \{0,1\}$ let:

$$F_k := A_t(\triangleleft(n),i,k) = \{f \in I(\blacktriangleleft(\triangleleft(n)), \blacktriangleright(\triangleleft(n))+1) \mid f(\blacktriangleleft(\triangleleft(n))) = i\,, f(\blacktriangleright(\triangleleft(n))+1) = k\}\,.$$

and let:

$$G_k := A_t(\triangleright(n),k,j) = \{f \in I(\blacktriangleleft(\triangleright(n)), \blacktriangleright(\triangleright(n))+1) \mid f(\blacktriangleleft(\triangleright(n))) = k\,, f(\blacktriangleright(\triangleright(n))+1) = j\}\,.$$

Since $\triangleleft(n)$ and $\triangleright(n)$ are at depth $d$ we have, by the inductive hypothesis and Lemma 49 that:

$$\alpha_t(n,i,j)_a = \sum_{k\in\{0,1\}} \alpha_t(\triangleleft(n),i,k)_a\alpha_t(\triangleright(n),k,j)_a$$

$$= \sum_{k\in\{0,1\}} \left(\sum_{f\in F_k} \Omega_{t,a}(f)\right) \left(\sum_{f\in G_k} \Omega_{t,a}(f)\right) .$$

Since $\blacktriangleright(\triangleleft(n)) + 1 = \blacktriangleleft(\triangleright(n))$ we then have, by Lemma 16, that:

$$\begin{aligned}
\alpha_t(n,i,j)_a &= \sum_{k\in\{0,1\}} \sum_{f\in I(\blacktriangleleft(\triangleleft(n)),\blacktriangleright(\triangleright(n))+1):f(\blacktriangleleft(\triangleleft(n)))=i\,,f(\blacktriangleleft(\triangleright(n)))=k\,,f(\blacktriangleright(\triangleright(n))+1)=j} \Omega_{t,a}(f) \\
&= \sum_{f\in I(\blacktriangleleft(\triangleleft(n)),\blacktriangleright(\triangleright(n))+1):f(\blacktriangleleft(\triangleleft(n)))=i\,,f(\blacktriangleright(\triangleright(n))+1)=j} \Omega_{t,a}(f) \\
&= \sum_{f\in I(\blacktriangleleft(n),\blacktriangleright(n)+1):f(\blacktriangleleft(n))=i\,,f(\blacktriangleright(n)+1)=j} \Omega_{t,a}(f)
\end{aligned}$$

as required. ■

### I.4 Proof of Lemma 18

**Result**. For all $t \in [T]$ , $d \in [h] \cup \{0\}$ , $i \in \{0,1\}$ and $a \in [K]$ we have:

$$\beta^{\Rightarrow}_t(\nu_{t,d}, i)_a = \sum_{f\in B^{\Rightarrow}_t(\nu_{t,d},i)} \Omega_{t,a}(f) ,$$

where:

$$B^{\Rightarrow}_t(\nu_{t,d}, i) := \{f \in I(\blacktriangleright(\nu_{t,d}) + 1, N + 1) \mid f(\blacktriangleright(\nu_{t,d}) + 1) = i\} ,$$

and:

$$\beta^{\Leftarrow}_t(\nu_{t,d}, i)_a = \sum_{f\in B^{\Leftarrow}_t(\nu_{t,d},i)} \iota_{f(1)}\Omega_{t,a}(f) ,$$

where:

$$B^{\Leftarrow}_t(\nu_{t,d}, i) := \{f \in I(1, \blacktriangleleft(\nu_{t,d})) \mid f(\blacktriangleleft(\nu_{t,d})) = i\} .$$

**Proof:**

**Lemma 50.** *For all $t \in [T]$, $d \in [h] \cup \{0\}$, $i \in \{0,1\}$ and $a \in [K]$ we have:*

$$\beta^{\Rightarrow}_t(\nu_{t,d}, i)_a = \sum_{f\in I(\blacktriangleright(\nu_{t,d})+1,N+1):f(\blacktriangleright(\nu_{t,d})+1)=i} \Omega_{t,a}(f) .$$

*Proof.* We prove by induction over $d$. In the case that $d = 0$ we have that $\nu_{t,d}$ is the root so $\blacktriangleright(\nu_{t,d}) + 1 = N + 1$. From the algorithm we see that $\beta^{\Rightarrow}_t(\nu_{t,0}, i)_a = 1$ so we have, since $\{u \in \mathbb{N} \mid N+1 \leq u < N+1\} = \emptyset$ and $\{f \in I(N+1, N+1) \mid f(N+1) = i\}$ contains the single function $f'$ which maps $N + 1$ to $i$ , that:

$$\begin{aligned}
\beta^{\Rightarrow}_t(\nu_{t,0}, i)_a &= 1 \\
&= \Omega_{t,a}(f') \\
&= \sum_{f\in I(N+1,N+1):f(N+1)=i} \Omega_{t,a}(f) \\
&= \sum_{f\in I(\blacktriangleright(\nu_{t,0})+1,N+1):f(\blacktriangleright(\nu_{t,0})+1)=i} \Omega_{t,a}(f)
\end{aligned}$$

as required. Now suppose the result holds for some $d = q$ (for some $q \in [h-1] \cup \{0\}$) we now show that it holds for $d = q + 1$ which will complete the proof. We have two cases:

- If $\nu_{t,q+1} = \triangleright(\nu_{t,q})$ then from the algorithm we have $\beta^{\Rightarrow}_t(\nu_{t,q+1}, i)_a := \beta^{\Rightarrow}_t(\nu_{t,q}, i)_a$ so since in this case $\blacktriangleright(\nu_{t,q+1}) = \blacktriangleright(\nu_{t,q})$ we have, by the inductive hypothesis:

$$\beta^{\Rightarrow}_t(\nu_{t,q+1}, i)_a := \beta^{\Rightarrow}_t(\nu_{t,q}, i)_a$$

$$= \sum_{f \in I(\blacktriangleright(\nu_{t,q})+1, N+1) : f(\blacktriangleright(\nu_{t,q})+1)=i} \Omega_{t,a}(f)$$
$$= \sum_{f \in I(\blacktriangleright(\nu_{t,q+1})+1, N+1) : f(\blacktriangleright(\nu_{t,q+1})+1)=i} \Omega_{t,a}(f)$$

as required.

- If $\nu_{t,q+1} = \triangleleft(\nu_{t,q})$ then for all $j, k \in \{0,1\}$ define:

  $$F_j := A_t(\triangleright(\nu_{t,q}), i, j) = \{f \in I(\blacktriangleleft(\triangleright(\nu_{t,q})), \blacktriangleright(\triangleright(\nu_{t,q})) + 1) \mid f(\blacktriangleleft(\triangleright(\nu_{t,q}))) = i \, , f(\blacktriangleright(\triangleright(\nu_{t,q})) + 1) = j\} \, ,$$

  and:

  $$G_{j,k} := \{f \in I(\blacktriangleright(\nu_{t,q}) + 1, N + 1) \mid f(\blacktriangleright(\nu_{t,q}) + 1) = j \, , f(N + 1) = k\} \, .$$

  From the algorithm we have:

  $$\beta_t^{\Rightarrow}(\nu_{t,q+1}, i)_a := \sum_{j \in \{0,1\}} \alpha_t(\triangleright(\nu_{t,q}), i, j)_a \beta_t^{\Rightarrow}(\nu_{t,q}, j)_a \, .$$

  From Lemma 17 we have, for all $j \in \{0,1\}$ that:

  $$\alpha_t(\triangleright(\nu_{t,q}), i, j)_a = \sum_{f \in F_j} \Omega_{t,a}(f) \, ,$$

  and from the inductive hypothesis we have, for all $j \in \{0,1\}$ that:

  $$\begin{aligned} \beta_t^{\Rightarrow}(\nu_{t,q}, j)_a &= \sum_{f \in I(\blacktriangleright(\nu_{t,q})+1, N+1) : f(\blacktriangleright(\nu_{t,q})+1)=j} \Omega_{t,a}(f) \\ &= \sum_{f \in G_{j,0} \cup G_{j,1}} \Omega_{t,a}(f) \\ &= \sum_{k \in \{0,1\}} \sum_{f \in G_{j,k}} \Omega_{t,a}(f) \, , \end{aligned}$$

  so since $\blacktriangleright(\triangleright(\nu_{t,q})) + 1 = \blacktriangleright(\nu_{t,q}) + 1$ we have, by Lemma 16, that:

  $$\begin{aligned} &\beta_t^{\Rightarrow}(\nu_{t,q+1}, i)_a \\ &:= \sum_{j \in \{0,1\}} \alpha_t(\triangleright(\nu_{t,q}), i, j)_a \beta_t^{\Rightarrow}(\nu_{t,q}, j)_a \\ &= \sum_{j \in \{0,1\}} \left( \sum_{f \in F_j} \Omega_{t,a}(f) \right) \left( \sum_{k \in \{0,1\}} \sum_{f \in G_{j,k}} \Omega_{t,a}(f) \right) \\ &= \sum_{j \in \{0,1\}} \sum_{k \in \{0,1\}} \left( \sum_{f \in F_j} \Omega_{t,a}(f) \right) \left( \sum_{f \in G_{j,k}} \Omega_{t,a}(f) \right) \\ &= \sum_{j \in \{0,1\}} \sum_{k \in \{0,1\}} \sum_{f \in I(\blacktriangleleft(\triangleright(\nu_{t,q})), N+1) : f(\blacktriangleleft(\triangleright(\nu_{t,q})))=i, f(\blacktriangleright(\nu_{t,q})+1)=j, f(N+1)=k} \Omega_{t,a}(f) \\ &= \sum_{f \in I(\blacktriangleleft(\triangleright(\nu_{t,q})), N+1) : f(\blacktriangleleft(\triangleright(\nu_{t,q})))=i} \Omega_{t,a}(f) \\ &= \sum_{f \in I(\blacktriangleright(\triangleleft(\nu_{t,q}))+1, N+1) : f(\blacktriangleright(\triangleleft(\nu_{t,q}))+1)=i} \Omega_{t,a}(f) \\ &= \sum_{f \in I(\blacktriangleright(\nu_{t,q+1})+1, N+1) : f(\blacktriangleright(\nu_{t,q+1})+1)=i} \Omega_{t,a}(f) \end{aligned}$$

  as required.

$\square$

**Lemma 51.** *For all $t \in [T]$, $d \in [h] \cup \{0\}$, $i \in \{0,1\}$ and $a \in [K]$ we have:*

$$\beta^{\Leftarrow}_t(\nu_{t,d}, i)_a = \sum_{f \in I(1, \blacktriangleleft(\nu_{t,d})) : f(\blacktriangleleft(\nu_{t,d}))=i} \iota_{f(1)} \Omega_{t,a}(f)\,.$$

*Proof.* We prove by induction over $d$. In the case that $d = 0$ we have that $\nu_{t,d}$ is the root so $\blacktriangleleft(\nu_{t,d}) = 1$. From the algorithm we see that $\beta^{\Rightarrow}_t(\nu_{t,0}, i)_a = \iota_i$ so we have, since $\{u \in \mathbb{N} \mid 1 \leq u < 1\} = \emptyset$ and $\{f \in I(1,1) \mid f(1) = i\}$ contains the single function $f'$ which maps $1$ to $i$, that:

$$\begin{aligned}
\beta^{\Leftarrow}_t(\nu_{t,0}, i)_a &:= \iota_i \\
&= \iota_i \Omega_{t,a}(f') \\
&= \sum_{f \in I(1,1) : f(1)=i} \iota_i \Omega_{t,a}(f) \\
&= \sum_{f \in I(1, \blacktriangleleft(\nu_{t,0})) : f(\blacktriangleleft(\nu_{t,0}))=i} \iota_i \Omega_{t,a}(f)
\end{aligned}$$

as required. Now suppose the result holds for $d = q$ (for some $q \in [h-1] \cup \{0\}$). We now show that it holds for $d = q + 1$ which will complete the proof. We have two cases:

- If $\nu_{t,q+1} = \triangleleft(\nu_{t,q})$ then from the algorithm we have $\beta^{\Leftarrow}_t(\nu_{t,q+1}, i)_a := \beta^{\Leftarrow}_t(\nu_{t,q}, i)_a$ so since in this case $\blacktriangleleft(\nu_{t,q+1}) = \blacktriangleleft(\nu_{t,q})$ we have, by the inductive hypothesis:

$$\begin{aligned}
\beta^{\Leftarrow}_t(\nu_{t,q+1}, i)_a &:= \beta^{\Leftarrow}_t(\nu_{t,q}, i)_a \\
&= \sum_{f \in I(1, \blacktriangleleft(\nu_{t,q})) : f(\blacktriangleleft(\nu_{t,q}))=i} \iota_{f(1)} \Omega_{t,a}(f) \\
&= \sum_{f \in I(1, \blacktriangleleft(\nu_{t,q+1})) : f(\blacktriangleleft(\nu_{t,q+1}))=i} \iota_{f(1)} \Omega_{t,a}(f)
\end{aligned}$$

  as required.

- If $\nu_{t,q+1} = \triangleright(\nu_{t,q})$ then for all $j, k \in \{0,1\}$ define:

$$\begin{aligned}
G_j &:= A_t(\triangleleft(\nu_{t,q}), j, i) \\
&= \{f \in I(\blacktriangleleft(\triangleleft(\nu_{t,q})), \blacktriangleright(\triangleleft(\nu_{t,q})) + 1) \mid f(\blacktriangleleft(\triangleleft(\nu_{t,q}))) = j\,, f(\blacktriangleright(\triangleleft(\nu_{t,q})) + 1) = i\}\,,
\end{aligned}$$

  and:

$$F_{k,j} = \{f \in I(1, \blacktriangleleft(\nu_{t,q})) \mid f(1) = k\,, f(\blacktriangleleft(\nu_{t,q})) = j\}\,.$$

  From the algorithm we have:

$$\beta^{\Leftarrow}_t(\nu_{t,q+1}, i)_a := \sum_{j \in \{0,1\}} \beta^{\Leftarrow}_t(\nu_{t,q}, j)_a \alpha_t(\triangleleft(\nu_{t,q}), j, i)_a\,.$$

  From Lemma 17 we have, for all $j \in \{0,1\}$ that:

$$\alpha_t(\triangleleft(\nu_{t,q}), j, i)_a = \sum_{f \in G_j} \Omega_{t,a}(f)\,,$$

  and from the inductive hypothesis we have, for all $j \in \{0,1\}$, that:

$$\begin{aligned}
\beta^{\Leftarrow}_t(\nu_{t,q}, j)_a &= \sum_{f \in I(1, \blacktriangleleft(\nu_{t,q})) : f(\blacktriangleleft(\nu_{t,q}))=j} \iota_{f(1)} \Omega_{t,a}(f) \\
&= \sum_{f \in F_{0,j} \cup F_{1,j}} \iota_{f(1)} \Omega_{t,a}(f) \\
&= \sum_{k \in \{0,1\}} \sum_{f \in F_{k,j}} \iota_{f(1)} \Omega_{t,a}(f)\,,
\end{aligned}$$

so since $\blacktriangleleft(\triangleleft(\nu_{t,q})) = \blacktriangleleft(\nu_{t,q})$ we have, by Lemma 16, that:

$$\begin{aligned}
&\beta_t^{\Leftarrow}(\nu_{t,q+1}, i)_a \\
&:= \sum_{j\in\{0,1\}} \beta_t^{\Leftarrow}(\nu_{t,q}, j)_a \alpha_t(\triangleleft(\nu_{t,q}), j, i)_a \\
&= \sum_{j\in\{0,1\}} \left(\sum_{k\in\{0,1\}} \sum_{f\in F_{k,j}} \Omega_{t,a}(f)\right)\left(\sum_{f\in G_j} \Omega_{t,a}(f)\right) \\
&= \sum_{j\in\{0,1\}} \sum_{k\in\{0,1\}} \left(\sum_{f\in F_{k,j}} \Omega_{t,a}(f)\right)\left(\sum_{f\in G_j} \Omega_{t,a}(f)\right) \\
&= \sum_{j\in\{0,1\}} \sum_{k\in\{0,1\}} \sum_{f\in I(1,\blacktriangleright(\triangleleft(\nu_{t,q}))+1):f(1)=k,f(\blacktriangleleft(\nu_{t,q}))=j,f(\blacktriangleright(\triangleleft(\nu_{t,q}))+1)=i} \Omega_{t,a}(f) \\
&= \sum_{f\in I(1,\blacktriangleright(\triangleleft(\nu_{t,q}))+1):f(\blacktriangleright(\triangleleft(\nu_{t,q}))+1)=i} \Omega_{t,a}(f) \\
&= \sum_{f\in I(1,\blacktriangleleft(\triangleright(\nu_{t,q}))):f(\blacktriangleleft(\triangleright(\nu_{t,q})))=i} \Omega_{t,a}(f) \\
&= \sum_{f\in I(1,\blacktriangleleft(\nu_{t,q+1})):f(\blacktriangleleft(\nu_{t,q+1}))=i} \Omega_{t,a}(f)
\end{aligned}$$

as required.

$\square$

The result then comes directly from lemmas 50 and 51.

### I.5 Proof of Lemma 19

**Result**. For all $t \in [T]$ and $a \in [K]$ we have:

$$\bar{p}_{t,a} = \sum_{U\in\mathcal{E}:u_t\in U} \bar{\kappa}_t(a, U)\,.$$

**Proof:**

**Lemma 52.** *For all $t \in [T]$ and $a \in [K]$ we have:*

$$\bar{p}_{t,a} = \frac{1}{K} \sum_{f\in I(1,N+1):f(u_t)=1} \iota_{f(1)}\Omega_{t,a}(f)\,.$$

*Proof.* Noting that $u_t = \nu_{t,h}$ and, since $\nu_{t,h}$ is a leaf, $\blacktriangleleft(\nu_{t,h}) = \blacktriangleright(\nu_{t,h}) = \nu_{t,h} = u_t$ we have, by Lemma 18, that for all $i \in \{0, 1\}$ :

$$\beta_t^{\Rightarrow}(u_t, i)_a = \sum_{f\in I(u_t+1,N+1):f(u_t+1)=i} \Omega_{t,a}(f)$$

and:

$$\beta_t^{\Leftarrow}(u_t, 1)_a = \sum_{f\in I(1,u_t):f(u_t)=1} \iota_{f(1)}\Omega_{t,a}(f)\,.$$

Also note that by Lemma 17 we have, for all $i \in \{0,1\}$, that $\alpha_t(u_t, 1, i)_a = \tau_{1,i}\xi_{t,u_t,a}$. Hence we have, by the prediction algorithm and definition of $\Omega_{t,a}$ , that:

$$K\bar{p}_{t,a} := \sum_{i\in\{0,1\}} \beta_t^{\Leftarrow}(u_t, 1)_a \alpha_t(u_t, 1, i)_a \beta_t^{\Rightarrow}(u_t, i)_a$$

$$
\begin{aligned}
&= \beta_t^{\Leftarrow}(u_t,1)_a \sum_{i\in\{0,1\}} \alpha_t(u_t,1,i)_a \beta_t^{\Rightarrow}(u_t,i)_a \\
&= \beta_t^{\Leftarrow}(u_t,1)_a \sum_{i\in\{0,1\}} \tau_{1,i}\xi_{t,u_t,a} \sum_{f\in I(u_t+1,N+1):f(u_t+1)=i} \Omega_{t,a}(f) \\
&= \beta_t^{\Leftarrow}(u_t,1)_a \sum_{i\in\{0,1\}} \sum_{f\in I(u_t+1,N+1):f(u_t+1)=i} (\tau_{1,i}\xi_{t,u_t,a})\Omega_{t,a}(f) \\
&= \beta_t^{\Leftarrow}(u_t,1)_a \sum_{i\in\{0,1\}} \sum_{f\in I(u_t,N+1):f(u_t)=1,f(u_t+1)=i} \Omega_{t,a}(f) \\
&= \beta_t^{\Leftarrow}(u_t,1)_a \sum_{f\in I(u_t,N+1):f(u_t)=1} \Omega_{t,a}(f) \\
&= \left(\sum_{f\in I(1,u_t):f(u_t)=1} \iota_{f(1)}\Omega_{t,a}(f)\right)\left(\sum_{f\in I(u_t,N+1):f(u_t)=1} \Omega_{t,a}(f)\right) \\
&= \left(\sum_{i\in\{0,1\}} \sum_{f\in I(1,u_t):f(1)=i,f(u_t)=1} \iota_{f(1)}\Omega_{t,a}(f)\right)\left(\sum_{j\in\{0,1\}} \sum_{f\in I(u_t,N+1):f(u_t)=1,f(N+1)=j} \Omega_{t,a}(f)\right) \\
&= \left(\sum_{i\in\{0,1\}} \iota_i \sum_{f\in I(1,u_t):f(1)=i,f(u_t)=1} \Omega_{t,a}(f)\right)\left(\sum_{j\in\{0,1\}} \sum_{f\in I(u_t,N+1):f(u_t)=1,f(N+1)=j} \Omega_{t,a}(f)\right) \\
&= \sum_{i\in\{0,1\}} \sum_{j\in\{0,1\}} \iota_i \left(\sum_{f\in I(1,u_t):f(1)=i,f(u_t)=1} \Omega_{t,a}(f)\right)\left(\sum_{f\in I(u_t,N+1):f(u_t)=1,f(N+1)=j} \Omega_{t,a}(f)\right).
\end{aligned}
$$

By Lemma 16 we then have:

$$
\begin{aligned}
K\bar{p}_{t,a} &= \sum_{i\in\{0,1\}} \sum_{j\in\{0,1\}} \iota_i \sum_{f\in I(1,N+1):f(1)=i,f(u_t)=1,f(N+1)=j} \Omega_{t,a}(f) \\
&= \sum_{i\in\{0,1\}} \sum_{j\in\{0,1\}} \sum_{f\in I(1,N+1):f(1)=i,f(u_t)=1,f(N+1)=j} \iota_{f(1)}\Omega_{t,a}(f) \\
&= \sum_{f\in I(1,N+1):f(u_t)=1} \iota_{f(1)}\Omega_{t,a}(f)
\end{aligned}
$$

as required. □

**Lemma 53.** *Given $t \in [T]$, $a \in [K]$, $i \in \{0,1\}$ and $U \in \mathcal{E}$, then if $f \in I(1, N+1)$ is defined so that for all $u \in [N]$ we have $f(u) := [\![u \in \mathcal{E}]\!]$ and we have $f(N+1) := i$ then:*

$$\iota_{f(1)}\Omega_{t,a}(f) = K\bar{\kappa}_t(a,U)\tau_{[\![N\in U]\!],i}\,.$$

*Proof.* We prove by induction on $t$. For $t = 1$ we have, for all $u \in [N]$, that $\xi_{1,u,a} = 1$ so:

$$
\begin{aligned}
\iota_{f(1)}\Omega_{1,a}(f) &= \iota_{f(1)} \prod_{u\in[N]} \tau_{f(u),f(u+1)}\xi_{1,u,a} \\
&= \iota_{f(1)} \prod_{u\in[N]} \tau_{f(u),f(u+1)} \\
&= \tau_{f(N),f(N+1)}\iota_{f(1)} \prod_{u\in[N-1]} \tau_{f(u),f(u+1)} \\
&= \tau_{[\![N\in U]\!],i}\iota_{[\![1\in U]\!]} \prod_{u\in[N-1]} \tau_{[\![u\in\mathcal{E}]\!],[\![u+1\in\mathcal{E}]\!]} \\
&= \tau_{[\![N\in U]\!],i}\sigma(U) \\
&= \tau_{[\![N\in U]\!],i}K\bar{\kappa}_1(a,U)
\end{aligned}
$$

as required. Now suppose the inductive hypothesis holds for $t = s$ (for some $s \in [T]$). We now show that it holds for $t = s + 1$, completing the proof. From the fact that $\xi_{s+1,u,a} = \xi_{s,u,a}$ for all $u \neq u_s$ we have:

$$\begin{aligned} \frac{\Omega_{s+1,a}(f)}{\Omega_{s,a}(f)} &= \frac{[\![f(u_s) = 0]\!] + [\![f(u_s) = 1]\!]\xi_{s+1,u_s,a}}{[\![f(u_s) = 0]\!] + [\![f(u_s) = 1]\!]\xi_{s,u_s,a}} \\ &= \frac{[\![u_s \notin U]\!] + [\![u_s \in U]\!]\xi_{s+1,u_s,a}}{[\![u_s \notin U]\!] + [\![u_s \in U]\!]\xi_{s,u_s,a}}, \end{aligned} \tag{17}$$

so we have two cases:

- If $u_s \notin U$ then, by Equation 17, $\Omega_{s+1,a}(f)/\Omega_{s,a}(f) = 1$ so $\iota_{f(1)}\Omega_{s+1,a}(f) = \iota_{f(1)}\Omega_{s,a}(f)$ which, by the inductive hypothesis, is equal to $K\bar{\kappa}_s(a,U)\tau_{[\![N\in U]\!],i}$. From the definition of $\bar{\kappa}_{s+1}(a,U)$ we have $\bar{\kappa}_{s+1}(a,U) := \bar{\kappa}_s(a,U)$ so $\iota_{f(1)}\Omega_{s+1,a}(f) = K\bar{\kappa}_{s+1}(a,U)\tau_{[\![N\in U]\!],i}$ as required.
- If $u_s \in U$ then, by Equation 17, $\Omega_{s+1,a}(f)/\Omega_{s,a}(f) = \xi_{s+1,u_s,a}/\xi_{s,u_s,a} = \pi_{s,a}$ so $\iota_{f(1)}\Omega_{s+1,a}(f) = \pi_{s,a}\iota_{f(1)}\Omega_{s,a}(f)$ which, by the inductive hypothesis, is equal to $K\pi_{s,a}\bar{\kappa}_s(a,U)\tau_{[\![N\in U]\!],i}$. From the definition of $\bar{\kappa}_{s+1}(a,U)$ we have $\bar{\kappa}_{s+1}(a,U) := \pi_{s,a}\bar{\kappa}_s(a,U)$ so $\iota_{f(1)}\Omega_{s+1,a}(f) = K\bar{\kappa}_{s+1}(a,U)\tau_{[\![N\in U]\!],i}$ as required.

□

By lemmas 52 and 53 we have:

$$\begin{aligned} \bar{p}_{t,a} &= \frac{1}{K} \sum_{f\in I(1,N+1):f(u_t)=1} \iota_{f(1)}\Omega_{t,a}(f) \\ &= \frac{1}{K} \sum_{i\in\{0,1\}} \sum_{f\in I(1,N+1):f(u_t)=1,f(N+1)=i} \iota_{f(1)}\Omega_{t,a}(f) \\ &= \frac{1}{K} \sum_{i\in\{0,1\}} \sum_{U\in\mathcal{E}:u_t\in U} K\bar{\kappa}_t(a,U)\tau_{[\![N\in U]\!],i} \\ &= \sum_{i\in\{0,1\}} \sum_{U\in\mathcal{E}:u_t\in U} \bar{\kappa}_t(a,U)\tau_{[\![N\in U]\!],i} \\ &= \sum_{U\in\mathcal{E}:u_t\in U} \bar{\kappa}_t(a,U)(\tau_{[\![N\in U]\!],0} + \tau_{[\![N\in U]\!],1}) \\ &= \sum_{U\in\mathcal{E}:u_t\in U} \bar{\kappa}_t(a,U) \end{aligned}$$

as required. ■

### I.6 Proof of Lemma 20

**Result**. GABA-I implements PROTOGABA with $\mathcal{E}$ and $\sigma$ defined as in Definition 10.

**Proof:**

**Definition 54.** *For all $t \in [T]$ let $\kappa_t$, $\boldsymbol{p}_t$, $z_t$, and $\lambda_t$ be as defined in the* PROTOGABA *algorithm when run with $\mathcal{E}$ and $\sigma$ defined as in Definition 10, and assuming that, for all $t \in [T]$, we have that $a_t$ is equal to that selected by* GABA-I.

**Lemma 55.** *For all $t \in [T]$ and all $(a,U) \in [K] \times \mathcal{E}$ we have $\bar{\kappa}_t(a,U) = \kappa_t(a,U)$ and $\bar{p}_{t,a} = p_{t,a}$.*

*Proof.* We prove by induction on $t$. For $t = 1$ it is clear, from the definitions of $\bar{\kappa}_1(a, U)$ and $\kappa_1(a, U)$, that $\bar{\kappa}_1(a, U) := \kappa_1(a, U)$ so also, by Lemma 19, we have that:

$$\bar{p}_{1,a} := \sum_{U \in \mathcal{E} : u_1 \in U} \bar{\kappa}_1(a, U) = \sum_{U \in \mathcal{E} : u_1 \in U} \kappa_1(a, U) := p_{1,a} ,$$

so the result holds for $t = 1$. Now suppose the result holds for $t = s$ (for some $s \in [T]$). We will now show that it holds for $t = s + 1$, which will complete the proof. By the inductive hypothesis we have, for all $b \in [K]$:

$$\begin{aligned}
c_{s,b} &:= \exp\left(-\eta [\![ b = a_s ]\!] \ell_{s,a_s} \|\bar{\boldsymbol{p}}_s\|_1 / \bar{p}_{s,b}\right) \\
&= \exp\left(-\eta [\![ b = a_s ]\!] \ell_{s,a_s} \|\boldsymbol{p}_s\|_1 / p_{s,b}\right) \\
&= [\![ b \neq a_s ]\!] + [\![ b = a_s ]\!] \exp\left(-\eta \ell_{s,a_s} \|\boldsymbol{p}_s\|_1 / p_{s,b}\right) \\
&= [\![ b \neq a_s ]\!] + [\![ b = a_s ]\!] \lambda_s ,
\end{aligned}$$

so also, by the inductive hypothesis:

$$\begin{aligned}
\bar{\boldsymbol{p}}_s \cdot \boldsymbol{c}_s &= \sum_{b \in [K]} \bar{p}_{s,b} \cdot c_{s,b} \\
&= \sum_{b \in [K]} p_{s,b} \cdot c_{s,b} \\
&= \sum_{b \in [K]} p_{s,b} ([\![ b \neq a_s ]\!] + [\![ b = a_s ]\!] \lambda_s) \\
&= \sum_{b \in [K] \setminus \{a_s\}} p_{s,b} + p_{s,a_s} \lambda_s \\
&= (\|\boldsymbol{p}_s\|_1 - p_{s,a_s}) + p_{s,a_s} \lambda_s \\
&= \|\boldsymbol{p}_s\|_1 - (1 - \lambda_s) p_{s,a_s} \\
&= \frac{\|\boldsymbol{p}_s\|_1}{z_s} ,
\end{aligned}$$

and hence, again by the inductive hypothesis:

$$\begin{aligned}
\pi_{s,a} &:= \frac{\|\bar{\boldsymbol{p}}_s\|_1 c_{s,a}}{\bar{\boldsymbol{p}}_s \cdot \boldsymbol{c}_s} \\
&= \frac{\|\boldsymbol{p}_s\|_1 c_{s,a}}{\bar{\boldsymbol{p}}_s \cdot \boldsymbol{c}_s} \\
&= z_s c_{s,a} \\
&= z_s ([\![ a \neq a_s ]\!] + [\![ a = a_s ]\!] \lambda_s) .
\end{aligned}$$

By the inductive hypothesis and definitions of $\bar{\kappa}_{s+1}(a, U)$ and $\kappa_{s+1}(a, U)$ we then have:

$$\begin{aligned}
\bar{\kappa}_{s+1}(a, U) &:= [\![ u_s \notin U ]\!] \bar{\kappa}_s(a, U) + [\![ u_s \in U ]\!] \pi_{s,a} \bar{\kappa}_s(a, U) \\
&= [\![ u_s \notin U ]\!] \kappa_s(a, U) + [\![ u_s \in U ]\!] \pi_{s,a} \kappa_s(a, U) \\
&= [\![ u_s \notin U ]\!] \kappa_s(a, U) + [\![ u_s \in U ]\!] z_s ([\![ a \neq a_s ]\!] + [\![ a = a_s ]\!] \lambda_s) \kappa_s(a, U) \\
&:= \kappa_{s+1}(a, U) .
\end{aligned}$$

By Lemma 19, we then have that:

$$\bar{p}_{s+1,a} := \sum_{U \in \mathcal{E} : u_{s+1} \in U} \bar{\kappa}_{s+1}(a, U) = \sum_{U \in \mathcal{E} : u_{s+1} \in U} \kappa_{s+1}(a, U) := p_{s+1,a}$$

as required. This completes the proof. □

Since, by Lemma 55, we have $\bar{\boldsymbol{p}}_t = \boldsymbol{p}_t$ we then have, by the prediction algorithm, that $a_t$ is drawn with probability $\mathbb{P}[a_t = a] = p_{t,a} / \|\boldsymbol{p}_t\|_1$ which implies the result. ■

### I.7 Proof of Lemma 22

**Result**. We have that:

- For all $u \in [N]$ there exists a unique pair $(a, U) \in \mathcal{A}$ with $u \in U$ .
- For all $u \in [N]$ and $(a, U) \in \mathcal{A}$ with $u \in U$ , we have that $a = y(u)$ .

**Proof:**

For all $u \in [N]$ and $a \in [K]$ we have that $u \in \{v \in [N] \mid y(v) = a\}$ if and only if $a = y(u)$ so there exists an unique pair $(b, U) \in \mathcal{A}$ with $u \in U$ and furthermore this pair satisfies $b = y(u)$ as required.
■

### I.8 Proof of Theorem 23

**Result**. Given $\Psi \leq (N-1)/4$ and the parameters are tuned as:

$$\phi := 4\Psi/(K(N-1)) \,,$$

and:

$$\eta := \sqrt{\frac{10\Psi \ln(KN/\Psi)}{KT}} \,,$$

we have, for GABA-I:

$$R \in \mathcal{O}\left(\sqrt{\ln\left(\frac{KN}{\Psi}\right)\Psi KT}\right) \,.$$

**Proof:**

**Definition 56.** *Define:*

$$\Gamma := \sum_{u \in [N-1]} [\![ y(u) \neq y(u+1) ]\!] \,,$$

*and for all $a \in [K]$ and $u \in [N]$ define:*

$$\epsilon(a, u) = [\![ y(u) = a ]\!] \,.$$

**Lemma 57.** *We have:*

$$R \leq \frac{1}{\eta}\mathbb{E}\left[\sum_{a \in [K]} \ln\left(\frac{1}{\sigma(\{u \in [N] \mid y(u) = a\})}\right)\right] + \frac{\eta KT}{2} \,.$$

*Proof.* Lemmas 20 and 22 allow us to invoke Theorem 9 so, noting that $|\mathcal{A}| = K$, we have:

$$\begin{aligned}
R &\leq \frac{1}{\eta}\sum_{(a,U) \in \mathcal{A}} \ln\left(\frac{K}{|\mathcal{A}|\sigma(U)}\right) + \frac{\eta KT}{2} \\
&= \frac{1}{\eta}\sum_{(a,U) \in \mathcal{A}} \ln\left(\frac{1}{\sigma(U)}\right) + \frac{\eta KT}{2} \\
&= \frac{1}{\eta}\sum_{(a,U) \in \mathcal{A}} \ln\left(\frac{1}{\sigma(\{u \in [N] \mid y(u) = a\})}\right) + \frac{\eta KT}{2} \\
&= \frac{1}{\eta}\sum_{a \in [K]} \ln\left(\frac{1}{\sigma(\{u \in [N] \mid y(u) = a\})}\right) + \frac{\eta KT}{2} \,.
\end{aligned}$$

Taking expectations then gives us the result. □

**Lemma 58.** *We have:*

$$\sum_{a \in [K]} \ln(\iota_{\epsilon(a,1)}) = \ln\left(\frac{1}{K}\right) + (K-1)\ln\left(1 - \frac{1}{K}\right) \,.$$

*Proof.* We have $\epsilon(y(1),1) = [\![y(1) = y(1)]\!] = 1$ so $\iota_{\epsilon(y(1),1)} = 1/K$. Also, for all $a \in [K] \setminus \{y(1)\}$, we have $\epsilon(a,1) = [\![y(1) = a]\!] = 0$ so $\iota_{\epsilon(a,1)} = 1 - 1/K$. Hence, we have:

$$\begin{aligned}\sum_{a\in[K]} \ln(\iota_{\epsilon(a,1)}) &= \ln(\iota_{\epsilon(y(1),1)}) + \sum_{a\in[K]\setminus\{y(1)\}} \ln(\iota_{\epsilon(a,1)}) \\ &= \ln\left(\frac{1}{K}\right) + \sum_{a\in[K]\setminus\{y(1)\}} \ln\left(1 - \frac{1}{K}\right) \\ &= \ln\left(\frac{1}{K}\right) + (K-1)\ln\left(1 - \frac{1}{K}\right)\end{aligned}$$

as required. □

**Lemma 59.** *For all $u \in [N-1]$ with $y(u) = y(u+1)$ we have:*

$$\sum_{a\in[K]} \ln\left(\tau_{\epsilon(a,u),\epsilon(a,u+1)}\right) = K\ln(1-\phi)\,.$$

*Proof.* We have, for all $a \in [K]$, that $\epsilon(a,u) = [\![y(u) = a]\!] = [\![y(u+1) = a]\!] = \epsilon(a,u+1)$ so $\tau_{\epsilon(a,u),\epsilon(a,u+1)} = 1 - \phi$. This implies that:

$$\sum_{a\in[K]} \ln(\tau_{\epsilon(a,u),\epsilon(a,u+1)}) = \sum_{a\in[K]} \ln(1-\phi) = K\ln(1-\phi)$$

as required. □

**Lemma 60.** *For all $u \in [N-1]$ with $y(u) \neq y(u+1)$ we have:*

$$\sum_{a\in[K]} \ln\left(\tau_{\epsilon(a,u),\epsilon(a,u+1)}\right) = (K-2)\ln(1-\phi) + 2\ln(\phi)\,.$$

*Proof.* We have that:

$$\epsilon(y(u),u) = [\![y(u) = y(u)]\!] = 1 \neq 0 = [\![y(u+1) = y(u)]\!] = \epsilon(y(u),u+1)\,,$$

and that:

$$\epsilon(y(u+1),u) = [\![y(u+1) = y(u)]\!] = 0 \neq 1 = [\![y(u+1) = y(u+1)]\!] = \epsilon(y(u+1),u+1)\,.$$

so $\tau_{\epsilon(y(u),u),\epsilon(y(u),u+1)} = \phi$ and $\tau_{\epsilon(y(u+1),u),\epsilon(y(u+1),u+1)} = \phi$. We also have, for all $a \in [K] \setminus \{y(u), y(u+1)\}$, that:

$$\epsilon(a,u) = [\![y(u) = a]\!] = 0 = [\![y(u+1) = a]\!] = \epsilon(a,u+1)\,,$$

so $\tau_{\epsilon(a,u),\epsilon(a,u+1)} = 1 - \phi$. Combining these equalities gives us:

$$\begin{aligned}&\sum_{a\in[K]} \ln\left(\tau_{\epsilon(a,u),\epsilon(a,u+1)}\right) \\ &= \ln\left(\tau_{\epsilon(y(u),u),\epsilon(y(u),u+1)}\right) + \ln\left(\tau_{\epsilon(y(u+1),u),\epsilon(y(u+1),u+1)}\right) + \sum_{a\in[K]\setminus\{y(u),y(u+1)\}} \ln\left(\tau_{\epsilon(a,u),\epsilon(a,u+1)}\right) \\ &= \ln(\phi) + \ln(\phi) + \sum_{a\in[K]\setminus\{y(u),y(u+1)\}} \ln(1-\phi) \\ &= 2\ln(\phi) + (K-2)\ln(1-\phi)\end{aligned}$$

as required. □

**Lemma 61.** *We have:*

$$\sum_{u\in[N-1]}\sum_{a\in[K]} \ln(\tau_{\epsilon(a,u),\epsilon(a,u+1)}) = 2\Gamma\ln(\phi) + (K(N-1) - 2\Gamma)\ln(1-\phi)\,.$$

*Proof.* From lemmas 59 and 60 we have:

$$
\begin{aligned}
&\sum_{u\in[N-1]}\sum_{a\in[K]}\ln(\tau_{\epsilon(a,u),\epsilon(a,u+1)})\\
&=\sum_{u\in[N-1]:y(u)=y(u+1)}\sum_{a\in[K]}\ln(\tau_{\epsilon(a,u),\epsilon(a,u+1)})+\sum_{u\in[N-1]:y(u)\neq y(u+1)}\sum_{a\in[K]}\ln(\tau_{\epsilon(a,u),\epsilon(a,u+1)})\\
&=\sum_{u\in[N-1]:y(u)=y(u+1)}K\ln(1-\phi)+\sum_{u\in[N-1]:y(u)\neq y(u+1)}((K-2)\ln(1-\phi)+2\ln(\phi))\\
&=(N-1-\Gamma)K\ln(1-\phi)+\Gamma((K-2)\ln(1-\phi)+2\ln(\phi))\\
&=((N-1-\Gamma)K+\Gamma(K-2))\ln(1-\phi)+2\Gamma\ln(\phi)\\
&=((N-1)K-2\Gamma)\ln(1-\phi)+2\Gamma\ln(\phi)
\end{aligned}
$$

as required. $\square$

**Lemma 62.** *Given $\phi\leq 1/2$ we have:*

$$
\mathbb{E}\left[\sum_{u\in[N-1]}\sum_{a\in[K]}\ln(\tau_{\epsilon(a,u),\epsilon(a,u+1)})\right]\geq 4\Psi\ln(\phi)+(K(N-1)-4\Psi)\ln(1-\phi)\,.
$$

*Proof.* Since $0<\phi\leq 1/2$ and hence $1-\phi\geq\phi$ we have $\ln(\phi)-\ln(1-\phi)\leq 0$ so since, by Lemma 2, we have $\mathbb{E}[\Gamma]\leq 2\Psi$ , we must have:

$$
(\ln(\phi)-\ln(1-\phi))\mathbb{E}[\Gamma]\geq 2(\ln(\phi)-\ln(1-\phi))\Psi\,.
$$

So, from Lemma 61, we have:

$$
\begin{aligned}
\mathbb{E}\left[\sum_{u\in[N-1]}\sum_{a\in[K]}\ln(\tau_{\epsilon(a,u),\epsilon(a,u+1)})\right]&=\mathbb{E}\left[2\Gamma\ln(\phi)+(K(N-1)-2\Gamma)\ln(1-\phi)\right]\\
&=\mathbb{E}\left[K(N-1)\ln(1-\phi)+2\Gamma(\ln(\phi)-\ln(1-\phi))\right]\\
&=K(N-1)\ln(1-\phi)+2(\ln(\phi)-\ln(1-\phi))\mathbb{E}[\Gamma]\\
&\geq K(N-1)\ln(1-\phi)+4(\ln(\phi)-\ln(1-\phi))\Psi\\
&=4\Psi\ln(\phi)+(K(N-1)-4\Psi)\ln(1-\phi)
\end{aligned}
$$

as required. $\square$

**Lemma 63.** *Given $f,f'\in\mathbb{R}^+$, if we set $f^\dagger:=f/(f+f')$ then:*

$$
-f\ln(f^\dagger)-f'\ln(1-f^\dagger)\leq f\ln\left(\frac{1}{f^\dagger}\right)+f\,.
$$

*Proof.* We recall the following standard inequality about the binary entropy of $f^\dagger$:

$$
-f^\dagger\ln(f^\dagger)-(1-f^\dagger)\ln(1-f^\dagger)\leq f^\dagger\ln(1/f^\dagger)+f^\dagger\,.
$$

Using this and the fact that $f=(f+f')f^\dagger$ and $f'=(f+f')(1-f^\dagger)$ gives us:

$$
\begin{aligned}
-f\ln(f^\dagger)-f'\ln(1-f^\dagger)&=(f+f')(-f^\dagger\ln(f^\dagger)-(1-f^\dagger)\ln(1-f^\dagger))\\
&\leq(f+f')(f^\dagger\ln(1/f^\dagger)+f^\dagger)\\
&=(f+f')\left(\frac{f}{f+f'}\ln\left(\frac{1}{f^\dagger}\right)+\frac{f}{f+f'}\right)\\
&=f\ln\left(\frac{1}{f^\dagger}\right)+f
\end{aligned}
$$

as required. $\square$

**Lemma 64.** *We have:*

$$-\sum_{a\in[K]}\ln(\iota_{\epsilon(a,1)})\leq \ln(K)+1\,.$$

*Proof.* Direct from lemmas 58 and 63 with $f:=1$ , $f':=K-1$ and $f^{\dagger}=1/K$. $\square$

**Lemma 65.** *Given* $\phi:=4\Psi/(K(N-1))\leq 1/2$ *and* $\Psi>0$ *, we have:*

$$-\mathbb{E}\left[\sum_{u\in[N-1]}\sum_{a\in[K]}\ln(\tau_{\epsilon(a,u),\epsilon(a,u+1)})\right]\leq 4\Psi\ln\left(\frac{KN}{\Psi}\right)\,.$$

*Proof.* Direct from lemmas 62 and 63 with $f:=4\Psi$ , $f':=(K(N-1)-4\Psi)$ and $f^{\dagger}:=4\Psi/(K(N-1))=\phi$ , we have:

$$\begin{aligned}-\mathbb{E}\left[\sum_{u\in[N-1]}\sum_{a\in[K]}\ln(\tau_{\epsilon(a,u),\epsilon(a,u+1)})\right]&\leq 4\Psi\ln\left(\frac{K(N-1)}{4\Psi}\right)+4\Psi\\&\leq 4\Psi\ln\left(\frac{K(N-1)}{\Psi}\right)+(1-\ln(4))4\Psi\\&\leq 4\Psi\ln\left(\frac{K(N-1)}{\Psi}\right)\\&\leq 4\Psi\ln\left(\frac{KN}{\Psi}\right)\end{aligned}$$

as required. $\square$

**Lemma 66.** *Given* $\phi:=4\Psi/(K(N-1))\leq 1/2$ *and* $0<\Psi\leq(N-1)/4$ *we have:*

$$\mathbb{E}\left[\sum_{a\in[K]}\ln\left(\frac{1}{\sigma(\{u\in[N]\mid y(u)=a\})}\right)\right]\leq 5\Psi\ln\left(\frac{KN}{\Psi}\right)\,.$$

*Proof.* For all $a\in[K]$, we have, from the definition of $\sigma$ , that:

$$\begin{aligned}&\ln(\sigma(\{u\in[N]\mid y(u)=a\}))\\&=\ln(\iota_{[\![1\in\{v\in[N]\mid y(v)=a\}]\!]})+\sum_{u\in[N-1]}\ln(\tau_{[\![u\in\{v\in[N]\mid y(v)=a\}]\!],[\![u+1\in\{v\in[N]\mid y(v)=a\}]\!]})\\&=\ln(\iota_{[\![y(1)=a]\!]})+\sum_{u\in[N-1]}\ln(\tau_{[\![y(u)=a]\!],[\![y(u+1)=a]\!]})\\&=\ln(\iota_{\epsilon(a,1)})+\sum_{u\in[N-1]}\ln(\tau_{\epsilon(a,u),\epsilon(a,u+1)})\,,\end{aligned}$$

so by lemmas 64 and 65 we have:

$$\begin{aligned}&\mathbb{E}\left[\sum_{a\in[K]}\ln\left(\frac{1}{\sigma(\{u\in[N]\mid y(u)=a\})}\right)\right]\\&=-\sum_{a\in[K]}\mathbb{E}\left[\ln(\sigma(\{u\in[N]\mid y(u)=a\}))\right]\\&=-\sum_{a\in[K]}\mathbb{E}\left[\ln(\iota_{\epsilon(a,1)})+\sum_{u\in[N-1]}\ln(\tau_{\epsilon(a,u),\epsilon(a,u+1)})\right]\\&=-\sum_{a\in[K]}\ln(\iota_{\epsilon(a,1)})-\mathbb{E}\left[\sum_{u\in[N-1]}\sum_{a\in[K]}\ln(\tau_{\epsilon(a,u),\epsilon(a,u+1)})\right]\end{aligned}$$

$$\leq (\ln(K) + 1) + 4\Psi \ln\left(\frac{KN}{\Psi}\right)$$
$$= \ln(eK) + 4\Psi \ln\left(\frac{KN}{\Psi}\right).$$

Since we have $\Psi \leq (N-1)/4 < N/e$ we also have that $KN/\Psi > eK$ so $\ln(eK) < \ln(KN/\Psi)$. Substituting this into the above inequality gives us the result. $\square$

Since $\Psi \leq (N-1)/4$ implies that $4\Psi/(K(N-1)) \leq 1/2$, combining lemmas 57 and 66 gives us the fact that if $0 \leq \Psi \leq (N-1)/4$ and $\phi := 4\Psi/(K(N-1))$, we have:

$$R \leq \frac{5}{\eta}\Psi \ln\left(\frac{KN}{\Psi}\right) + \frac{\eta KT}{2},$$

so, choosing:

$$\eta := \sqrt{\frac{10\Psi \ln(KN/\Psi)}{KT}}$$

we have:

$$R \leq \sqrt{10\Psi \ln\left(\frac{KN}{\Psi}\right) KT}$$

as required. $\blacksquare$

# J GABA-II Proofs

## J.1 Proof of Lemma 29

**Result**. Given $t \in [T]$, $n \in \mathcal{C}$ and $a \in [K]$ we have:

$$\bar{\kappa}_t(a,n) = \frac{\mu_t(n)\theta_t(n,a)}{(2N-1)K}.$$

**Proof:**

We prove by induction on $t$. For $t = 1$ we have $\mu_1(n) := 1$ and $\theta_1(n,a) := 1$ so:

$$\bar{\kappa}_1(a,n) := \frac{1}{(2N-1)K} = \frac{\mu_1(n)\theta_1(n,a)}{(2N-1)K}$$

as required. Now suppose the result holds for $t = s$ (for some $s \in [T]$). We now show that it holds for $t = s+1$, completing the proof. We first note that by the update algorithm we have, for all $n \in \Uparrow(u_s)$:

$$\mu_{s+1}(n) := \mu_s(n)\frac{\psi_s}{\psi_s - (1-\bar{\lambda}_s)\varrho_s} = \bar{z}_s\mu_s(n),$$

whilst for $n \notin \Uparrow(u_s)$ we have:

$$\mu_{s+1}(n) := \mu_s(n),$$

so in general:

$$\mu_{s+1}(n) = (\llbracket n \notin \Uparrow(u_s)\rrbracket + \llbracket n \in \Uparrow(u_s)\rrbracket \bar{z}_s)\mu_s(n).$$

We note that also, by the update algorithm, we have, for all $n \in \Uparrow(u_s)$, that if $a = a_t$ then

$$\theta_{s+1}(n,a) := \bar{\lambda}_t\theta_s(n,a),$$

whilst if $n \notin \Uparrow(u_s)$ or $a \neq a_t$ we have that:

$$\theta_{s+1}(n,a) := \theta_s(n,a),$$

so in general:

$$\theta_{s+1}(n,a) = (\llbracket n \notin \Uparrow(u_s)\rrbracket + \llbracket n \in \Uparrow(u_s) \wedge a \neq a_t\rrbracket + \llbracket n \in \Uparrow(u_s) \wedge a = a_t\rrbracket \bar{\lambda}_t)\theta_s(n,a).$$

Multiplying these two equalities and invoking the inductive hypothesis gives us:

$$\begin{aligned}
&\mu_{s+1}(n)\theta_{s+1}(n,a) \\
&= ([\![n \notin \Uparrow(u_s)]\!] + [\![n \in \Uparrow(u_s) \wedge a \neq a_t]\!]\bar{z}_s + [\![n \in \Uparrow(u_s) \wedge a = a_t]\!]\bar{z}_s\bar{\lambda}_t)\mu_s(n)\theta_s(n,a) \\
&= ([\![n \notin \Uparrow(u_s)]\!] + [\![n \in \Uparrow(u_s) \wedge a \neq a_t]\!]\bar{z}_s + [\![n \in \Uparrow(u_s) \wedge a = a_t]\!]\bar{z}_s\bar{\lambda}_t)\bar{\kappa}_s(a,n)(2N-1)K\,,
\end{aligned}$$

which, by the definition of $\bar{\kappa}_{s+1}(a,n)$ is equal to $\bar{\kappa}_{s+1}(a,n)(2N-1)K$ as required. This completes the proof. ■

### J.2 Proof of Lemma 30

**Result**. Given $t \in [T]$ , $n \in \mathcal{C}$ and $m \in \mathcal{B}$ we have:

$$\theta_t(n,m) = \sum_{a \in \Downarrow(m) \cap [K]} \theta_t(n,a)\,.$$

**Proof:**

We prove by induction on $t$. For $t = 1$ we have the result directly from the initialization algorithm. Now suppose the result holds for $t = s$ (for some $s \in [T]$). We now show that the result holds for $t = s + 1$ which will complete the proof. We first consider the case that $m$ is a leaf of $\mathcal{B}$. We have two cases:

- If $m \in [K]$ then $\Downarrow(m) \cap [K] = \{m\}$ so the result is immediate.
- If $m \notin [K]$ then $\Downarrow(m) \cap [K] = \emptyset$ so by the inductive hypothesis we have:

$$\begin{aligned}
\theta_s(n,m) &= \sum_{a \in \Downarrow(m) \cap [K]} \theta_s(n,a) \\
&= 0 \\
&= \sum_{a \in \Downarrow(m) \cap [K]} \theta_{s+1}(n,a)\,.
\end{aligned}$$

  From the update algorithm we have (since $m$ is not an ancestor of $a_s$) that $\theta_{s+1}(n,m) := \theta_s(n,m)$ , which then gives us the result.

So the result holds in the case that $m$ is a leaf of $\mathcal{B}$. Now suppose that $m$ is an internal vertex of $\mathcal{B}$. We have two cases:

- If $m \notin \Uparrow(a_t)$ or $n \notin \Uparrow(u_t)$ we have, from the update algorithm, that $\theta_{s+1}(n,m) := \theta_s(n,m)$ and, since also either $n \notin \Uparrow(u_t)$ or both children of $m$ are not in $\Uparrow(a_t)$ we also have, from the update algorithm, that $\theta_{s+1}(n,\triangleleft(m)) := \theta_s(n,\triangleleft(m))$ and $\theta_{s+1}(n,\triangleright(m)) := \theta_s(n,\triangleright(m))$ so by the inductive hypothesis:

$$\begin{aligned}
\theta_{s+1}(n,m) &:= \theta_s(n,m) \\
&= \sum_{a \in \Downarrow(m) \cap [K]} \theta_s(n,a) \\
&= \sum_{a \in \Downarrow(\triangleleft(m)) \cap [K]} \theta_s(n,a) + \sum_{a \in \Downarrow(\triangleright(m)) \cap [K]} \theta_s(n,a) \\
&= \theta_s(n,\triangleleft(m)) + \theta_s(n,\triangleright(m)) \\
&= \theta_{s+1}(n,\triangleleft(m)) + \theta_{s+1}(n,\triangleright(m)) \\
&= \sum_{a \in \Downarrow(\triangleleft(m)) \cap [K]} \theta_{s+1}(n,a) + \sum_{a \in \Downarrow(\triangleright(m)) \cap [K]} \theta_{s+1}(n,a) \\
&= \sum_{a \in \Downarrow(m) \cap [K]} \theta_{s+1}(n,a)
\end{aligned}$$

  as required.

- If $m \in \Uparrow(a_t)$ and $n \in \Uparrow(u_t)$ then $m = \zeta_{t,g-d}$ for some $d \in [g]$ so we prove by induction on $d$. For $d = 1$ we note that $\triangleleft(m)$ and $\triangleright(m)$ are both leaves and we proved above that the result held for leaves. Hence, by the update algorithm, we have:

$$\begin{aligned}\theta_{s+1}(n,m) &:= \theta_{s+1}(n,\triangleleft(m)) + \theta_{s+1}(n,\triangleright(m))\\ &= \sum_{a\in\Downarrow(\triangleleft(m))\cap[K]} \theta_{s+1}(n,a) + \sum_{a\in\Downarrow(\triangleright(m))\cap[K]} \theta_{s+1}(n,a)\\ &= \sum_{a\in\Downarrow(m)\cap[K]} \theta_{s+1}(n,a)\end{aligned}$$

  as required. Now suppose the result holds for $d = q$ (for some $q \in [g]$). We now show that it holds for $d = q + 1$ which will complete the proof. We have that one of the children of $m$ is an ancestor of $a_t$ so without loss of generality assume this is its left child. By above we proved the result holds for all internal vertices that are not ancestors of $a_s$ so specifically the result holds for $\triangleright(m)$. Noting that $\triangleleft(m) = \zeta_{t,g-q}$ we also have, by the inductive hypothesis, that the result holds for $\triangleleft(m)$ so, by the update algorithm:

$$\begin{aligned}\theta_{s+1}(n,m) &:= \theta_{s+1}(n,\triangleleft(m)) + \theta_{s+1}(n,\triangleright(m))\\ &= \sum_{a\in\Downarrow(\triangleleft(m))\cap[K]} \theta_{s+1}(n,a) + \sum_{a\in\Downarrow(\triangleright(m))\cap[K]} \theta_{s+1}(n,a)\\ &= \sum_{a\in\Downarrow(m)\cap[K]} \theta_{s+1}(n,a)\end{aligned}$$

  as required

This completes the proof. ■

### J.3 Proof of Lemma 31

**Result**. For all $t \in [T]$ and $a \in [K]$ we have:

$$\mathbb{P}\left[a_t = a\right] = \frac{\bar{p}_{t,a}}{\|\bar{\boldsymbol{p}}_t\|_1}.$$

**Proof:**

**Lemma 67.** *For all trials $t \in [T]$ and all $d \in [g-1] \cup \{0\}$ we have, given $\delta_t$ and $\zeta_{t,d}$, that:*

$$\mathbb{P}\left[\zeta_{t,d+1} = m\right] = \frac{\theta_t(\delta_t, m)}{\theta_t(\delta_t, \uparrow(m))},$$

*for $m \in \{\triangleleft(\zeta_{t,d}), \triangleright(\zeta_{t,d})\}$.*

*Proof.* From the algorithm we have $\mathbb{P}\left[\zeta_{t,d+1} = m\right] \propto \theta_t(\delta_t, m)$ for $m \in \{\triangleleft(\zeta_{t,d}), \triangleright(\zeta_{t,d})\}$ so $\mathbb{P}\left[\zeta_{t,d+1} = m\right] = \theta_t(\delta_t, m)/Z$ where $Z := \theta_t(\delta_t, \triangleleft(\zeta_{t,d})) + \theta_t(\delta_t, \triangleright(\zeta_{t,d}))$ By Lemma 30, we then have:

$$\begin{aligned}Z &:= \theta_t(\delta_t, \triangleleft(\zeta_{t,d})) + \theta_t(\delta_t, \triangleright(\zeta_{t,d}))\\ &= \sum_{a\in\Downarrow(\triangleleft(\zeta_{t,d}))\cap[K]} \theta_t(\delta_t, a) + \sum_{a\in\Downarrow(\triangleright(\zeta_{t,d}))\cap[K]} \theta_t(\delta_t, a)\\ &= \sum_{a\in\Downarrow(\zeta_{t,d})\cap[K]} \theta_t(\delta_t, a)\\ &= \theta_t(\delta_t, \zeta_{t,d})\\ &= \theta_t(\delta_t, \uparrow(m))\end{aligned}$$

as required. □

**Lemma 68.** *For all $n \in \mathcal{C}$ and all $m \in \mathcal{B}$ at depth $d \in [g] \cup \{0\}$ we have:*

$$\mathbb{P}\left[\zeta_{t,d} = m \mid \delta_t = n\right] = \frac{\theta_t(n,m)}{\theta_t(n,r)} .$$

*Proof.* We prove by induction on $d$. For $d = 0$ we have $m = r$ and always $\zeta_{t,0} = r$ so:

$$\mathbb{P}\left[\zeta_{t,0} = m \mid \delta_t = n\right] = 1 = \frac{\theta_t(n,r)}{\theta_t(n,r)} = \frac{\theta_t(n,m)}{\theta_t(n,r)}$$

as required. Now assume the result holds for $d = q$ (for some $q \in [g-1] \cup \{0\}$). We now show it holds for $d = q+1$ which completes the proof. We have, by the inductive hypothesis and Lemma 67, that:

$$\begin{aligned}
\mathbb{P}\left[\zeta_{t,q+1} = m \mid \delta_t = n\right] &= \mathbb{P}\left[\zeta_{t,q+1} = m \mid \delta_t = n \wedge \uparrow(m) = \zeta_{t,q}\right] \mathbb{P}\left[\zeta_{t,q} = \uparrow(m) \mid \delta_t = n\right] \\
&= \mathbb{P}\left[\zeta_{t,q+1} = m \mid \delta_t = n \wedge \uparrow(m) = \zeta_{t,d}\right] \frac{\theta_t(n,\uparrow(m))}{\theta_t(n,r)} \\
&= \frac{\theta_t(n,m)}{\theta_t(n,\uparrow(m))} \frac{\theta_t(n,\uparrow(m))}{\theta_t(n,r)} \\
&= \frac{\theta_t(n,m)}{\theta_t(n,r)}
\end{aligned}$$

as required. □

By Lemma 68, the law of total probability, the prediction algorithm, and Lemma 29 we have:

$$\begin{aligned}
\mathbb{P}\left[a_t = a\right] &= \mathbb{P}\left[\zeta_{t,g} = a\right] \\
&= \sum_{n \in \Uparrow(u_t)} \mathbb{P}\left[\delta_t = n\right] \mathbb{P}\left[\zeta_{t,g} = a \mid \delta_t = n\right] \\
&= \sum_{n \in \Uparrow(u_t)} \mathbb{P}\left[\delta_t = n\right] \frac{\theta_t(n,a)}{\theta_t(n,r)} \\
&\propto \sum_{n \in \Uparrow(u_t)} (\mu_t(n)\theta_t(n,r)) \frac{\theta_t(n,a)}{\theta_t(n,r)} \\
&= \sum_{n \in \Uparrow(u_t)} \mu_t(n)\theta_t(n,a) \\
&\propto \sum_{n \in \Uparrow(u_t)} \bar{\kappa}_t(a,n) \\
&= \bar{p}_{t,a}
\end{aligned}$$

as required. ■

### J.4 Proof of Lemma 32

**Result**. For all $t \in [T]$ we have:

$$\bar{\lambda}_t = \exp\left(\frac{-\eta \ell_{t,a_t} \|\bar{\boldsymbol{p}}_t\|_1}{\bar{p}_{t,a_t}}\right) ,$$

and:

$$\bar{z}_t = \frac{\|\bar{\boldsymbol{p}}_t\|_1}{\|\bar{\boldsymbol{p}}_t\|_1 - (1-\bar{\lambda}_t)\bar{p}_{t,a_t}} .$$

**Proof:**

**Lemma 69.** *We have:*

$$\varrho_t = (2N-1)K\bar{p}_{t,a_t} .$$

*Proof.* From Lemma 29 and the update algorithm we have:

$$\begin{aligned}\varrho_t &:= \sum_{n \in \Uparrow(u_t)} \mu_t(n)\theta(n, a_t) \\ &= \sum_{n \in \Uparrow(u_t)} \bar{\kappa}_t(a_t, n)(2N-1)K \\ &= (2N-1)K \sum_{n \in \Uparrow(u_t)} \bar{\kappa}_t(a_t, n) \\ &= (2N-1)K\bar{p}_{t,a_t}\end{aligned}$$

as required. □

**Lemma 70.** *We have:*

$$\psi_t = (2N-1)K\|\bar{\boldsymbol{p}}_t\|_1 \,.$$

*Proof.* From lemmas 29 and 30 and the update algorithm we have:

$$\begin{aligned}\psi_t &:= \sum_{n \in \Uparrow(u_t)} \mu_t(n)\theta(n, r) \\ &= \sum_{n \in \Uparrow(u_t)} \mu_t(n) \sum_{a \in \Downarrow(r) \cap [K]} \theta(n, a) \\ &= \sum_{n \in \Uparrow(u_t)} \mu_t(n) \sum_{a \in [K]} \theta(n, a) \\ &= \sum_{a \in [K]} \sum_{n \in \Uparrow(u_t)} \mu_t(n)\theta(n, a) \\ &= \sum_{a \in [K]} \sum_{n \in \Uparrow(u_t)} \bar{\kappa}_s(a, n)(2N-1)K \\ &= (2N-1)K \sum_{a \in [K]} \sum_{n \in \Uparrow(u_t)} \bar{\kappa}_t(a, n) \\ &= (2N-1)K \sum_{a \in [K]} \bar{p}_{t,a} \\ &= (2N-1)K\|\bar{\boldsymbol{p}}_t\|_1\end{aligned}$$

as required. □

From lemmas 69 and 70 and the update algorithm we have:

$$\bar{\lambda}_t := \exp\left(\frac{-\eta\ell_{t,a_t}\psi_t}{\varrho_t}\right) = \exp\left(\frac{-\eta\ell_{t,a_t}\|\bar{\boldsymbol{p}}_t\|_1}{\bar{p}_{t,a_t}}\right) \,,$$

and:

$$\bar{z}_t := \frac{\psi_t}{\psi_t - (1-\bar{\lambda}_t)\varrho_t} = \frac{\|\bar{\boldsymbol{p}}_t\|_1}{\|\bar{\boldsymbol{p}}_t\|_1 - (1-\bar{\lambda}_t)\bar{p}_{t,a_t}}$$

as required. ■

### J.5 Proof of Lemma 33

**Result**. GABA-II implements PROTOGABA with $\mathcal{E}$ and $\sigma$ defined as in Definition 25

**Proof:**

**Definition 71.** *For all $t \in [T]$ let $\kappa_t$, $\boldsymbol{p}_t$, $z_t$, and $\lambda_t$ be as defined in the* PROTOGABA *algorithm when run with $\mathcal{E}$ and $\sigma$ defined as in Definition 25, and assuming that, for all $t \in [T]$, we have that $a_t$ is equal to that selected by* GABA-II.

**Lemma 72.** *For all $t \in [T]$, $a \in [K]$ and $n \in \mathcal{C}$ we have:*

$$\bar{\kappa}_t(a,n) = \kappa_t(a, \Downarrow(n))\,.$$

*Proof.* We prove by induction over $t$. For $t = 1$ we have:

$$\kappa_1(a, \Downarrow(n)) := \frac{\sigma(\Downarrow(n))}{K} = \frac{1}{(2N-1)K} := \bar{\kappa}_1(a,n)$$

as required. Now suppose the result holds for $t = s$ (for some $s \in [T]$). We now show that it holds for $t = s+1$ which will complete the proof. Note first that, by the PROTOGABA algorithm and the inductive hypothesis, we have, for all $b \in [K]$, that:

$$\begin{aligned}
p_{s,b} &:= \sum_{U \in \mathcal{E} : u_s \in U} \kappa_s(b, U) \\
&= \sum_{n \in \mathcal{C} : u_s \in \Downarrow(n)} \kappa_s(b, \Downarrow(n)) \\
&= \sum_{n \in \Uparrow(u_s)} \kappa_s(b, \Downarrow(n)) \\
&= \sum_{n \in \Uparrow(u_s)} \bar{\kappa}_s(b, n) \\
&:= \bar{p}_{s,b}\,,
\end{aligned}$$

so $\bar{\boldsymbol{p}}_s = \boldsymbol{p}_s$. By Lemma 32 we then have:

$$\bar{\lambda}_s = \exp\left(\frac{-\eta \ell_{s,a_s} \|\bar{\boldsymbol{p}}_s\|_1}{\bar{p}_{s,a_s}}\right) = \exp\left(\frac{-\eta \ell_{s,a_s} \|\boldsymbol{p}_s\|_1}{p_{s,a_s}}\right) := \lambda_s\,,$$

and hence, also by Lemma 32, we have:

$$\bar{z}_s = \frac{\|\bar{\boldsymbol{p}}_s\|_1}{\|\bar{\boldsymbol{p}}_s\|_1 - (1-\bar{\lambda}_s)\bar{p}_{s,a_s}} = \frac{\|\boldsymbol{p}_s\|_1}{\|\boldsymbol{p}_s\|_1 - (1-\lambda_s)p_{s,a_s}} := z_s\,,$$

so, by the PROTOGABA algorithm and the inductive hypothesis, we have:

$$\begin{aligned}
&\kappa_{s+1}(a, \Downarrow(n)) \\
&:= (\llbracket u_s \notin \Downarrow(n) \rrbracket + \llbracket u_s \in \Downarrow(n) \wedge a \neq a_s \rrbracket z_s + \llbracket u_s \in \Downarrow(n) \wedge a = a_s \rrbracket \lambda_s z_s)\kappa_s(a, \Downarrow(n)) \\
&= (\llbracket n \notin \Uparrow(u_s) \rrbracket + \llbracket n \in \Uparrow(u_s) \wedge a \neq a_s \rrbracket z_s + \llbracket n \in \Uparrow(u_s) \wedge a = a_s \rrbracket \lambda_s z_s)\kappa_s(a, \Downarrow(n)) \\
&= (\llbracket n \notin \Uparrow(u_s) \rrbracket + \llbracket n \in \Uparrow(u_s) \wedge a \neq a_s \rrbracket z_s + \llbracket n \in \Uparrow(u_s) \wedge a = a_s \rrbracket \lambda_s z_s)\bar{\kappa}_s(a, n) \\
&= (\llbracket n \notin \Uparrow(u_s) \rrbracket + \llbracket n \in \Uparrow(u_s) \wedge a \neq a_s \rrbracket \bar{z}_s + \llbracket n \in \Uparrow(u_s) \wedge a = a_s \rrbracket \bar{\lambda}_s \bar{z}_s)\bar{\kappa}_s(a, n) \\
&:= \bar{\kappa}_{s+1}(a, n)
\end{aligned}$$

as required. This completes the proof. □

From the PROTOGABA algorithm and Lemma 72 we have, for all $a \in [K]$ and $t \in [T]$, that:

$$\begin{aligned}
p_{t,a} &:= \sum_{U \in \mathcal{E} : u_t \in U} \kappa_t(a, U) \\
&= \sum_{n \in \mathcal{C} : u_t \in \Downarrow(n)} \kappa_t(a, \Downarrow(n)) \\
&= \sum_{n \in \Uparrow(u_t)} \kappa_t(a, \Downarrow(n)) \\
&= \sum_{n \in \Uparrow(u_t)} \bar{\kappa}_t(a, n) \\
&:= \bar{p}_{t,a}\,,
\end{aligned}$$

so $\bar{\boldsymbol{p}}_t = \boldsymbol{p}_t$. By Lemma 31 we then have that :

$$\mathbb{P}[a_t = a] = \frac{\bar{p}_{t,a}}{\|\bar{\boldsymbol{p}}_t\|_1} = \frac{p_{t,a}}{\|\boldsymbol{p}_t\|_1}\,,$$

so the selections of GABA-II equal those of PROTOGABA. This completes the proof. ■

### J.6 Proof of Lemma 35

**Result**. We have that:

- For all $u \in [N]$ there exists a unique pair $(a, U) \in \mathcal{A}$ with $u \in U$.
- For all $u \in [N]$ and $(a, U) \in \mathcal{A}$ with $u \in U$, we have that $a = y(u)$.

**Proof:**

Suppose we have some $u \in [T]$. Let $n$ be the ancestor of $u$ of least depth in $\mathcal{C}$ which satisfies $y(v) = y(u)$ for all $v \in \Downarrow(n)$. Note that such a $n$ exists as $u$ is an ancestor of $u$ with $y(v) = y(u)$ for all $v \in \Downarrow(u)$ so the set we're selecting from is non-empty. Suppose we now take some $(a, n') \in [K] \times \mathcal{C}$ with $n' \in \Uparrow(u)$. We have the following cases:

- Suppose $a \neq y(u)$. Then $u \in \Downarrow(n')$ and $y(u) \neq a$ so $(a, n') \notin \mathcal{A}^\dagger$.
- Suppose $a = y(u)$ and $n' \neq n$. We now have two subcases:
  - Suppose $n'$ is a descendant of $n$. Then $\uparrow(n')$ is also a descendant of $n$ so since then $\Downarrow(\uparrow(n'))$ is a subset of $\Downarrow(n)$ we have, by definition of $n$, that for all $v \in \Downarrow(\uparrow(n'))$ we have $y(v) = y(u) = a$. By definition of $\mathcal{A}^\dagger$ we must then have $(a, n') \notin \mathcal{A}^\dagger$.
  - Suppose $n'$ is an ancestor of $n$. Then $n'$ is of lower depth that $n$ so, by definition of $n$, we must have that there exists some $v \in \Downarrow(n')$ with $y(v) \neq y(u) = a$ and so, by definition of $\mathcal{A}^\dagger$, that $(a, n') \notin \mathcal{A}^\dagger$.

  So in either subcase we have $(a, n') \notin \mathcal{A}^\dagger$.
- Suppose $a = y(u)$ and $n' = n$. Then we have two subcases:
  - Suppose $n$ is the root of $\mathcal{C}$. By definition of $n$, we have, for all $v \in \Downarrow(n) = \Downarrow(n')$, that $y(v) = y(u) = a$ and $n'$ is the root of $\mathcal{C}$. By definition of $\mathcal{A}^\dagger$ we must then have $(a, n') \in \mathcal{A}^\dagger$.
  - Suppose $n$ is not the root of $\mathcal{C}$. Then $\uparrow(n')$ is the parent of $n$ so is at lower depth than $n$ and hence, by definition of $n$, there exists some $v \in \Downarrow(\uparrow(n'))$ with $y(v) \neq y(u) = a$. By definition of $n$, we also have that all $v \in \Downarrow(n')$ satisfy $y(v) = y(u) = a$. By definition of $\mathcal{A}^\dagger$ we must then have $(a, n') \in \mathcal{A}^\dagger$.

  So in either subcase we must have $(a, n') \in \mathcal{A}^\dagger$.

We have hence shown that $(a, n') \in \mathcal{A}^\dagger$ if and only if $a = y(u)$ and $n' = n$ so, by definition of $\mathcal{A}$ we have that $(a, \Downarrow(n')) \in \mathcal{A}$ if and only if $a = y(u)$ and $n' = n$ which implies the result. ■

### J.7 Proof of Lemma 36

**Result**. We have:

$$\mathbb{E}\left[|\mathcal{A}|\right] \leq 4\Psi \log_2\left(\frac{eN}{\Psi}\right).$$

**Proof:**

**Definition 73.** *Let:*

$$\Gamma := \sum_{u \in [N-1]} [\![ y(u) \neq y(u+1) ]\!].$$

**Lemma 74.** *For all $n \in \mathcal{C}$ there exists at most one $a \in [K]$ such that $(a, n) \in \mathcal{A}^\dagger$.*

*Proof.* Suppose $(a, n) \in \mathcal{A}^\dagger$ for some $a \in [K]$ and take any $b \in [K] \setminus \{a\}$. Then by definition of $\mathcal{A}^\dagger$ we have $y(u) = a \neq b$ for all (and hence since $\Downarrow(n) \neq \emptyset$, for some) $u \in \Downarrow(n)$ so by definition of $\mathcal{A}^\dagger$ we have $(b, n) \notin \mathcal{A}^\dagger$. This implies the result. □

**Lemma 75.** *For all $(a, n) \in \mathcal{A}^\dagger$ such that $n$ is not the root of $\mathcal{C}$ there exists some $u \in \Downarrow(\uparrow(n))$ with $y(u) \neq y(u+1)$.*

*Proof.* Suppose, for contradiction, the contrary: that $y(u) = y(u+1)$ for all $u \in \Downarrow(\uparrow(n))$. Then since $\Downarrow(\uparrow(n))$ is a complete interval of natural numbers we have, by a simple induction, that there exists $b \in [K]$ such that for all $u \in \Downarrow(\uparrow(n))$ we have $y(u) = b$. By definition of $\mathcal{A}^\dagger$ we must have that all $u \in \Downarrow(n)$ satisfy $y(u) = a$ and hence, as $\emptyset \neq \Downarrow(n) \subseteq \Downarrow(\uparrow(n))$ , we have that there exists $u \in \Downarrow(\uparrow(n))$ with $y(u) = a$. So $b = a$ and hence all $u \in \Downarrow(\uparrow(n))$ satisfy $y(u) = a$. But this contradicts the fact that $(a, n) \in \mathcal{A}^\dagger$ which completes the proof. □

**Lemma 76.** *Given $d \in [h]$, the cardinality of the set of all $n \in \mathcal{C}$ at depth $d$, such that there exists $a \in [K]$ with $(a, n) \in \mathcal{A}^\dagger$, is bounded above by $2\Gamma$.*

*Proof.* Given some $n' \in \mathcal{C}$ at depth $d-1$ with a child $n \in \mathcal{C}$ and some $a \in [K]$ with $(a, n) \in \mathcal{A}^\dagger$ we have, from Lemma 75, that there exists some $u \in \Downarrow(n')$ with $y(u) \neq y(u+1)$. Now, since all such $\Downarrow(n')$ are pairwise disjoint, we must have that the cardinality of the set of all such $n'$ is bounded above by $\Gamma$. Since each such $n$ has only two children the result follows. □

Suppose we have $\Gamma \geq 1$. Since there are no more than $4\Gamma$ vertices at depth at most $\log_2(\Gamma) + 1$ we have, by Lemma 74, that the number of pairs $(a, n) \in \mathcal{A}^\dagger$ in which $n$ is at depth at most $\log_2(\Gamma) + 1$ is no more that $4\Gamma$. Also, by lemmas 74 and 76, we have that for all $d \in [h]$ there are at most $2\Gamma$ pairs $(a, n) \in \mathcal{A}^\dagger$ in which $n$ is at depth $d$, which implies that there are at most

$$2\Gamma(h - \log_2(\Gamma)) = 2\Gamma(\log_2(N) - \log_2(\Gamma)) = 2\Gamma \log_2\left(\frac{N}{\Gamma}\right)$$

pairs $(a, n) \in \mathcal{A}^\dagger$ in which $n$ is at depth greater than $\log_2(\Gamma) + 1$. So the total cardinality of $\mathcal{A}^\dagger$ is bounded above by:

$$|\mathcal{A}^\dagger| \leq 2\Gamma \log_2\left(\frac{N}{\Gamma}\right) + 4\Gamma = 2\Gamma\left(\log_2\left(\frac{N}{\Gamma}\right) + \log_2(4)\right) = 2\Gamma \log_2\left(\frac{4N}{\Gamma}\right) < 2\Gamma \log_2\left(\frac{2eN}{\Gamma}\right) .$$

Since the function $x \to 2x \log_2(2eN/x)$ is concave and monotonic increasing for $x \leq 2N$ and $2\Psi < 2N$ we then have, by definition of $\mathcal{A}$, Jenson's inequality, and Lemma 2, that:

$$\mathbb{E}\left[|\mathcal{A}|\right] = \mathbb{E}\left[|\mathcal{A}^\dagger|\right] \leq \mathbb{E}\left[2\Gamma \log_2\left(\frac{2eN}{\Gamma}\right)\right] \leq 2\mathbb{E}\left[\Gamma\right] \log_2\left(\frac{2eN}{\mathbb{E}\left[\Gamma\right]}\right) \leq 4\Psi \log_2\left(\frac{eN}{\Psi}\right)$$

as required. ■

### J.8 Proof of Theorem 37

**Result**. Given $\Psi \leq N/2$ and setting:

$$\eta := \sqrt{\frac{8\Psi \log_2\left(eN/\Psi\right) \ln\left(3KN/2\Psi\right)}{KT}} ,$$

we have, for GABA-II:

$$R \in \mathcal{O}\left(\sqrt{\ln\left(\frac{N}{\Psi}\right) \ln\left(\frac{KN}{\Psi}\right) \Psi KT}\right) .$$

**Proof:**

Lemmas 33 and 35 allow us to invoke Theorem 9 giving us, by definition of $\sigma$ :

$$\begin{aligned}
R &\leq \frac{1}{\eta} \sum_{(a,U) \in \mathcal{A}} \ln\left(\frac{K}{|\mathcal{A}|\sigma(U)}\right) + \frac{\eta KT}{2} \\
&= \frac{1}{\eta} \sum_{(a,U) \in \mathcal{A}} \ln\left(\frac{K(2N-1)}{|\mathcal{A}|}\right) + \frac{\eta KT}{2} \\
&\leq \frac{1}{\eta} \sum_{(a,U) \in \mathcal{A}} \ln\left(\frac{2KN}{|\mathcal{A}|}\right) + \frac{\eta KT}{2}
\end{aligned}$$

$$
\begin{aligned}
&= \frac{1}{\eta}|\mathcal{A}|\ln\left(\frac{2KN}{|\mathcal{A}|}\right) + \frac{\eta KT}{2} \\
&< \frac{1}{\eta}|\mathcal{A}|\ln\left(\frac{6KN}{|\mathcal{A}|}\right) + \frac{\eta KT}{2}\,. \qquad (18)
\end{aligned}
$$

By the change of base rule for logarithms we have:

$$
4\Psi\log_2\left(\frac{eN}{\Psi}\right) = \frac{4\Psi}{\ln(2)}\ln\left(\frac{eN}{\Psi}\right) = \frac{4eN}{\ln(2)}\left(\frac{\Psi}{eN}\right)\ln\left(\frac{eN}{\Psi}\right)\,,
$$

so since the function $x\ln(1/x)$ has a maximum value of $1/e$ we have that $4\Psi\log_2\left(\frac{eN}{\Psi}\right)$ is no greater than $4eN/(\ln(2)e) < 12N/e \leq 6KN/e$. Note also that by definition $|\mathcal{A}| \leq N < 6KN/e$.

Hence, since the function $x \to x\ln(6KN/x)$ is concave and monotonic increasing for $x \leq 6KN/e$ we have, from Jenson's inequaltiy and Lemma 36, that:

$$
\mathbb{E}\left[|\mathcal{A}|\ln\left(\frac{6KN}{|\mathcal{A}|}\right)\right] \leq \mathbb{E}\left[|\mathcal{A}|\right]\ln\left(\frac{6KN}{\mathbb{E}\left[|\mathcal{A}|\right]}\right) \leq 4\Psi\log_2\left(\frac{eN}{\Psi}\right)\ln\left(\frac{3KN}{2\Psi\log_2(eN/\Psi)}\right)\,.
$$

Substituting into Equation 18 and noting that since $\Psi \leq N$ we have $\log_2(eN/\Psi) \geq \log_2(e) > 1$ we then have, by taking expectations:

$$
R < \frac{1}{\eta}4\Psi\log_2\left(\frac{eN}{\Psi}\right)\ln\left(\frac{3KN}{2\Psi\log_2(eN/\Psi)}\right) + \frac{\eta KT}{2} \leq \frac{1}{\eta}4\Psi\log_2\left(\frac{eN}{\Psi}\right)\ln\left(\frac{3KN}{2\Psi}\right) + \frac{\eta KT}{2}
$$

so setting:

$$
\eta := \sqrt{\frac{8\Psi\log_2\left(eN/\Psi\right)\ln\left(3KN/2\Psi\right)}{KT}}
$$

gives us:

$$
R < \sqrt{8\Psi\log_2\left(\frac{eN}{\Psi}\right)\ln\left(\frac{3KN}{2\Psi}\right)KT}\,.
$$

Enforcing $\Psi \leq N/2$ ensures that $N/\Psi$ and $KN/\Psi$ never fall below a positive constant so:

$$
\log_2\left(\frac{eN}{\Psi}\right) = \log_2(e) + \frac{1}{\ln(2)}\ln\left(\frac{N}{\Psi}\right) \in \mathcal{O}\ln\left(\frac{N}{\Psi}\right)\,,
$$

and

$$
\ln\left(\frac{3KN}{2\Psi}\right) = \ln\left(\frac{3}{2}\right) + \ln\left(\frac{KN}{\Psi}\right) \in \mathcal{O}\left(\ln\left(\frac{KN}{\Psi}\right)\right)\,,
$$

which implies the result. ■