Bayesian Rule Posteriors from a Rule Foam

Akash Kumar Panda[1] and Bart Kosko[1]

Electrical and Computer Engineering Department
University of Southern California
kosko@usc.edu

Abstract. A rule foam converts a neural black-box classifier into a probabilistic rule-based ontology where a fresh Bayesian posterior describes the relative rule firings for each input pattern. The rules define a generalized probability mixture that in turns yields the Bayesian posterior over the rules or subsystems. The rules and posterior explain each observed input-output pair and further give a confidence measure of the predicted output in terms of the mixture's conditional variance. A random foam creates and combines several foams by randomly sampling the neural classifier. It's mixture structure gives a Bayesian posterior over the constituent foam systems and a finer-grained Bayesian posterior over each system's rules. We illustrate the foam technique on a deep neural classifier trained on the CIFAR-10 image dataset.

Keywords: Rule foam · probability mixtures · Bayesian rule posterior.

1 Converting Neural Classifiers to Rule Foams

A rule foam is an adaptive and statistical rule base that can approximate and help explain the mapping structure of a sampled neural classifier. The rule if-part fuzzy sets often resemble bubbles of varying diameters as in Figure 1. A generalized probability mixture $p(y|x)$ governs the rule structure and gives rise to a Bayesian posterior over the rules [9].

The neural classifier is a deep black box that maps input patterns to output class labels coded as unit bit vectors. The neural network lacks a statistical explanation both of its throughput mapping structure and of its output classification prediction. The rule foam's mixture $p(y|x)$ gives a proxy or explainer system where the mixed probability densities correspond to rules. This gives a fresh Bayesian posterior distribution $p(j|y,x)$ over its rules for each input pattern x and the input's corresponding observed output classification y. The mixture's second moment also gives a conditional covariance that serves as a confidence measure of the predicted output given what the foam system has learned.

The rule foam samples from the trained classifier and approximates it with m if-then rules $R_{A_1 \to B_1}, \ldots, R_{A_m \to B_m}$. The foam acts as a type of *proxy ontology* for the neural black box. Figure 2 shows a sample rule from a 5000-rule foam that approximates the neural classifier trained on the CIFAR-10 image dataset. The if-parts A_j and then-parts B_j are quite general and can define probability densities or fuzzy sets.

A generalized probability mixture $p(y|x)$ describes the m if-then rules $R_{A_1 \to B_1}$, $\dots$, $R_{A_m \to B_m}$ [5]. The mixture $p(y|x)$ absorbs the m rules $R_{A_1 \to B_1}, \dots , R_{A_m \to B_m}$ into $p(y|x) = p_1(x)\ p_{B_1}(y|x) + \cdots + p_m(x)\ p_{B_m}(y|x)$ as we explain in the next section. The mixture structure gives at once a Bayesian posterior $p(j|y, x)$ that describes the relative rule firing of the jth rule for each pattern input x and observed output y. Figure 3 shows the 7 most-probable rules for the trained foam's classification of the frog-pattern input. This probabilistic information helps describe the system's classification performance and can assist other classification algorithms such as pruning.

A rule foam and its governing mixture $p(y|x)$ differ from Bayesian ontologies that encode independence assumptions among random graphical random variables [2]. It also differs from using a conditional probability [7] to directly model rules because the generalized mixing weights $p_j(x)$ give the relative importance of each rule likelihood density $p_{B_j}(y|x)$ and because the resulting mixture structure gives rise to the Bayesian posterior over the rules as they fire. The proxy system of [3] is interpretable but does not give a confidence measure of the classifier's output. The rule system in [1] does use fuzzy sets but it does not have a probabilistic structure over the rules as in a rule foam.

A *random* foam creates independent foams by randomly sampling with replacement from the same neural network. It then combines these independent foam systems combining their throughput rules or their outputs [9]. The mixture structure combines with the additive structure of the foam systems to give telescoped Bayesian posteriors and conditional variances over both over the combined systems and over their constituent rules. Random foams can combine their throughput rule structure while random forests combine only their outputs. Figure 4 shows a 10-foam random foam and both its between-foam posteriors and within-foam rule posteriors on the MNIST data set of hand-written digits.

Consider a K-class image classification problem. The input image x belongs to one of the K classes $C_1, ..., C_K$ that partition in the input pattern space. The rule foam has m if-then rules $R_{A_1 \to B_1}$, $\dots$,$R_{A_m \to B_m}$ in its rule base. Let $R_j \in R$ denote the j-th rule for simplicity. The if-part of the R_j is a reference image I_c if we let such a class centroid act as a proxy for the entire fuzzy class of C_j (the fuzzy class or set defines a bubble in the foam). The then-part of R_j is the class C_k and thus a unit bit vector. So the rules are in general subsets of the input-output product space. The jth rule R_j or $R^j_{I_c \to C_k}$ with if-part I_c and then-part C_k states that

$$R^j_{I_c \to C_k} = R_j(I_c, C_k) = \text{IF (input } X \text{ looks like } I_c\text{) THEN } (X \in C_k). \tag{1}$$

Further information about the image could also help explain the rule and thereby the system. The rule foam fires all m rules for each input x. It then weights and combines these rules to give the final output $F(x)$. We show below that this corresponds to computing the conditional expectation $E[Y|X = x]$ with respect to the system's generalized mixture $p(y|x)$. The rule foam's output $F(x) \in [0, 1]^K$ is a K-dimensional probability vector. The kth component of this vector gives the predicted class probability $P(x \in C_k)$.

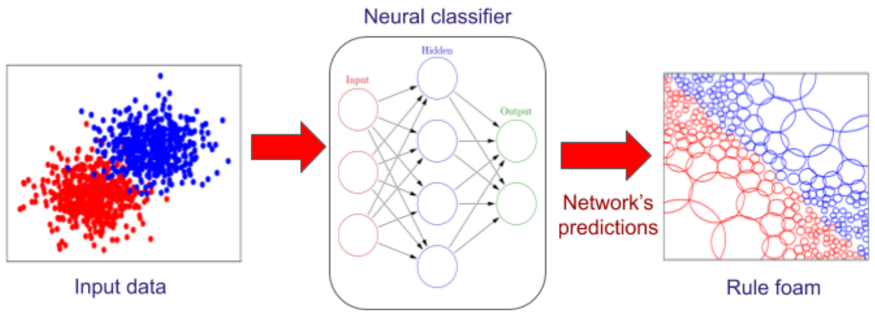

Fig. 1. Rule foam: The neural classifier trains on the 2-class dataset. Then the rule foam trains on this neural classifier. Each circle or bubble in the rule foam represents the if-part fuzzy set of an if-then rule. The rule foam has larger rules in the class interior and smaller rules near the class boundary. This helps mitigate rule explosion.

The rule foam also defines a Bayesian posterior distribution over all its rules for each input-output pair. The next section shows how this posterior arises from the system's mixture $p(y|x)$. The rule foam gives $P(R_j|C_k, x)$ for all $j = 1, ..., m$ given the observed input pattern x and output y or C_k. These posteriors measure how much each rule contribute to the output $F(x)$ and thus to the final classification output C_k. This makes the rule foam interpretable in a probabilistic sense as well as a modular rule-based sense.

Suppose that the foam misclassifies the pattern input x. The Bayesian posteriors point to the rule that contributed the most towards this misclassification. Figure 3 shows why studying these rule weights can give insight into the misclassification. The Bayesian posterior can also help identify and prune useless or harmful rules.

The rule foam's rule base represents the knowledge about this K-class image classification problem. The rule foam also gives a Bayesian posterior distribution over its rule-base R. So the rule foam defines a type of proxy Bayesian ontology for this knowledge domain given the trained deep network. The next section gives the mathematical details.

2 Mixture Structure of Additive Rule-based Systems

A standard additive model (SAM) fuzzy system approximates the function $f : R^d \to R$ using m "if-then" fuzzy rules. The j-th rule has the fuzzy if-part set $A_j \subset R^d$ and the fuzzy then-part set $B_j \subset R$. The fuzzy sets A_j and B_j have their respective unit-interval-valued indicator functions $a_j : R^d \to [0, 1]$ and $b_j : R \to [0, 1]$ such that $a_j(x) = \text{Degree}(x \in A_j)$ and $b_j(y) = \text{Degree}(y \in B_j)$. The jth fired then-part set $B_j(x)$ has set function $b_j(y|x) = a_j(x)b_j(y)$ because the system is standard.

The rule-based system's total fired then-part set $B(x)$ sums the fired rules by summing the fired then-part sets: $B(x) = \sum_{j=1}^{m} w_j a_j(x) B_j$ for rule weight

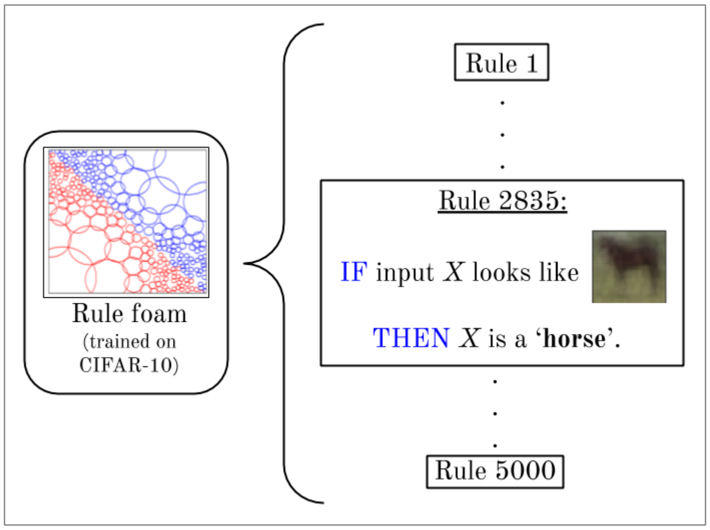

Fig. 2. Example of one rule from a 5000-rule foam. The neural classifier trained on the CIFAR-10 dataset. Then the rule foam trained on random samples from the trained neural classifier. The if-part of the 2835th rule resembles a horse. The corresponding then-part set is the class 'horse' and the unit bit vector that codes for it.

w_j. Then the fuzzy rules define the generalized probability mixture $p(y|x)$ [5]:

$$p(y|x) = \frac{b(y|x)}{\int b(y|x)dy} = \sum_{j=1}^{m} \frac{w_j a_j(x) V_j}{\sum_{j=1}^{m} w_j a_j(x) V_j} \frac{b_j(y)}{V_j} = \sum_{j=1}^{m} p_j(x) p_{B_j}(y) \quad (2)$$

where a_j and b_j are the jth rule's respective if-part and then-part fuzzy set functions. The volume $V_j = \int b_j(y)dy > 0$ is the volume of then-part set B_j. The SAM structure $b_j(y|x) = a_j(x)b_j(y)$ gives the likelihood as $b_j(y|x) = b_j(y)$.

The fuzzy system $F : R^d \to R$ arises naturally as the first moment of $p(y|x)$:

$$F(x) = E[Y|X = x] = \int y \; p(y|x) \; dy = \sum_{j=1}^{m} p_j(x) c_j \quad (3)$$

where c_j is the j-th then-part centroid. The system output $F(x)$ also arises as $F(x) = Centroid(B(y|x))$. Mixing just two normal bell curves can always exactly represent any bounded non-constant real function f such that $f(x) = E[Y|X = x]$ for curves centered at the infimum α and supremum β of f [5]: $p(y|x) = w(x)N(y|\alpha, \sigma_\alpha^2) + (1 - w(x))N(y|\beta, \sigma_\beta^2)$ for any variances σ_α^2 and σ_β^2 if $w(x) = \frac{\beta - f(x)}{\beta - \alpha}$ defines the Watkins coefficients as in the exact two-rule SAM representation of a bounded real function [11]. We recover the Watkins SAM representation in the deterministic limit when the variances σ_α^2 and σ_β^2 go to zero and thus when the bell curves reduce to delta spikes [6]. Consider the exponential target function e^{-x} with $\alpha = 0$ and $\beta = 1$. Then the representing mixture is $p(y|x) = (1 - e^{-x})N(y|0, 1) + e^{-x}N(y|1, 1)$. This gives $E[Y|X = x] = e^{-x}$.

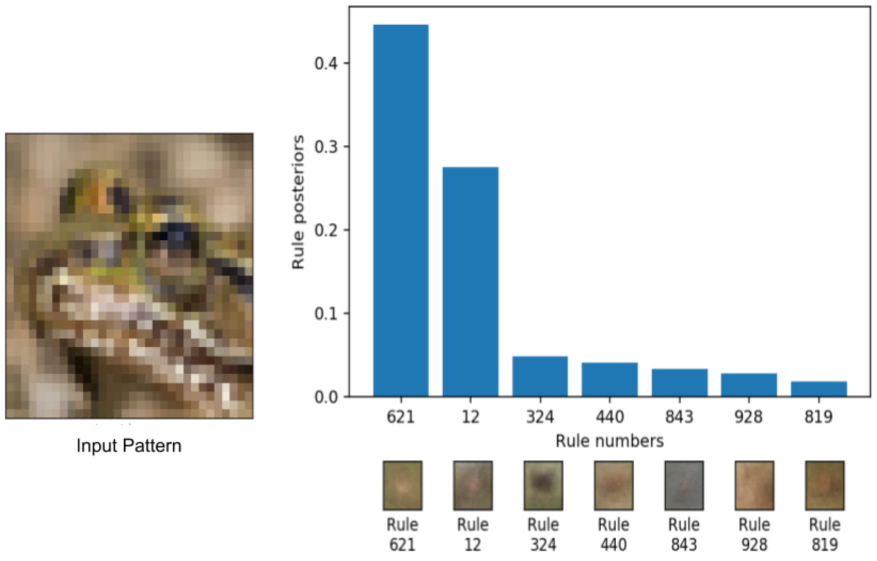

Fig. 3. A snapshot of the rule foam"s Bayesian posterior over all its rules. The input pattern is on the left. The seven highest posteriors are on the right. The x-axis lists the rule numbers. The y-axis lists the corresponding posterior probability. The figure also shows the if-parts of the seven rules. Rule 621 contributes the most towards classifying the input pattern.

The SAM rule-based system measures the uncertainty in its output through the conditional variance

$$V[Y|X=x] = \sum_{j=1}^{m} p_j(x)\sigma_{B_j}^2 + \sum_{j=1}^{m} p_j(x)(c_j - F(x))^2 \tag{4}$$

where σ_{B_j} is the j-th rule's then-part dispersion. The second term in equation (4) imposes an interpolation penalty on the system for guessing with respect to the given set of rules. We are more confident in the system's output relative to the stored rules if the variance is low and less so when the variance is high.

The same mixture structure gives rise to the Bayesian posterior over the rules. This holds because the generalized mixture $p(y|x)$ in (2) has the same form as the elementary theorem on total probability: $p(y|x) = p_1(x)\ p_{B_1}(y|x) + \cdots + p_m(x)\ p_{B_m}(y|x)$. The mixing weights $p_j(x)$ define generalized rule priors and the $p_{B_j}(y)$ define generalized rule likelihoods. This total-probability structure gives at once a posterior probability density over the rules from Bayes theorem:

$$p(j|y,x) = \frac{P(Z=j, Y=y|X=x)}{p(y|x)} = \frac{p_j(x)p_{B_j}(y)}{p(y|x)} = \frac{p_j(x)p_{B_j}(y)}{\sum_{j=1}^{m} p_j(x)p_{B_j}(y)} \tag{5}$$

for the j-th rule firing given observed input x and output y. These posteriors also give the contribution of each rule to the final output. Figure 3 shows 7 of the

5,000 rule-posterior probabilities for a given CIFAR-10 input image. See [8] for further details on how to implement a classifier rule foam using adaptive vector quantization.

3 Combining *q*-many Rulebased Subsystems

Additive fuzzy systems combine through a convex combination of their fired then-part sets. So the q additive systems $F_1, \ldots, F_q$ combine by adding their weighted rule firings $v^k b^k(y|x)$:

$$b(y|x) = \sum_{k=1}^{q} v^k b^k(y|x) = \sum_{k=1}^{q} \sum_{j=1}^{m_k} v^k w_j^k a_j^k(x) b_j^k(y) \tag{6}$$

where v^k is the weight of the kth SAM system F_k and where $b^k(y|x)$ sums all rule firings in this subsystem. The jth rule of this SAM has weight w_j^k, the if-part set function $a_j^k(x)$, and the then-part set function $b_j^k(y)$.

Additively combining *any* q rule-based systems $F^1, \ldots, F^q$ gives a generalized mixture $p(y|x)$ that mixes q conditional probability $p^1(y|x), \ldots, p^q(y|x)$:

$$p(y|x) = \frac{\sum_{k=1}^{q} v^k b^k(y|x)}{\sum_{l=1}^{q} v^l \int b^l(y|x) dy} = \frac{\sum_{k=1}^{q} v^k b^k(y|x) \frac{U^k(x)}{U^k(x)}}{\sum_{l=1}^{q} v^l U^l(x)} \tag{7}$$

$$= \sum_{k=1}^{q} [\frac{v^k}{\sum_{l=1}^{q} v^l U^l(x)}] p^k(y|x) = \sum_{k=1}^{q} u^k(x) p^k(y|x). \tag{8}$$

So even in this general case a Bayesian posterior results over the q subsystems:

$$p(k|y,x) = \frac{u^k(x) p^k(y|x)}{\sum_{l=1}^{q} u^l(x) p^l(y|x)}. \tag{9}$$

The *additive* structure of the q SAM subsystems further simplifies this overall mixture into a double sum of generalized priors times rule likelihoods:

$$p(y|x) = \sum_{k=1}^{q} \sum_{j=1}^{m_k} \frac{v^k w_j^k a_j^k(x) V_j^k}{\sum_{k=1}^{q} \sum_{j=1}^{m_k} v^k w_j^k a_j^k(x) V_j^k} \frac{b_j^k(y)}{V_j^k} \tag{10}$$

$$= \sum_{k=1}^{q} \sum_{j=1}^{m_k} p_j^k(x) p_{B_j^k}(y) \tag{11}$$

where $V_j^k = \int b_j^k(y) dy$ is the volume of the kth SAM's jth then-part set B_j^k. This gives in turn the Bayesian posterior $p(j,k|y,x)$ in (14) over the rules of the kth subsystem as in Figure 4.

This mixture's expected value gives the function approximator F:

$$F(x) = E[Y|X=x] = \int y \; p(y|x) \; dy = \sum_{k=1}^{q} \sum_{j=1}^{m_k} p_j^k(x) c_j^k. \tag{12}$$

where c_j^k is the then-part set's centroid. This combination also gives an uncertainty measure through its conditional variance:

$$V[Y|X=x] = \sum_{k=1}^{q}\sum_{j=1}^{m_k} p_j^k(x)\sigma^2_{B_j^k} + \sum_{k=1}^{q}\sum_{j=1}^{m_k} p_j^k(x)(c_j^k - F(x))^2 \tag{13}$$

where $\sigma_{B_j^k}$ is the then-part dispersion. The mixture in (11) also gives a Bayesian posterior distribution over all the SAMs and their rules.

$$p(j,k|y,x) = \frac{p_j^k(x)p_{B_j^k}(y)}{\sum_{k=1}^{q}\sum_{j=1}^{m_k} p_j^k(x)p_{B_j^k}(y)}. \tag{14}$$

A random foam trains and combines several independent foams. The random foam measures its uncertainty through its conditional variance in (13). It also measures the confidence of each constituent foam in their outputs through equation (4). Random foam gives the contribution of all the rules through the Bayesian posteriors given by (14). It also measures the contribution of each constituent foam through a posterior distribution over all the foams.

$$p(k|y,x) = \sum_{j=1}^{m_k} p(j,k|y,x) = \frac{\sum_{j=1}^{m_k} p_j^k(x)p_{B_j^k}(y)}{\sum_{k=1}^{q}\sum_{j=1}^{m_k} p_j^k(x)p_{B_j^k}(y)} \tag{15}$$

Figure 4 shows an example of such telescoping posteriors for a 10-foam random foam trained on the MNIST dataset of handwritten digits.

4 Conclusions

A rule foam can help explain a neural classifier by training on it and then using the foam's probabilistic mixture structure. This includes the rule mixture's Bayesian posterior over the rules and its conditional variance. A foam suffers from rule explosion because of its inherent graph cover. The foam's if-part or bubble structure can help mitigate this rule explosion. A random rule foam combines statistically independent foams by additively combining the rules or throughputs of its constituent foams. This gives a telescoped posteriors and variances both over the foams that help produce a final output classification and over those foams' constituent rules.

A key problem remains: How well does the foam proxy or explainer system approximate the sampled neural classifier? A foam or a random foam system is in principle a uniform function approximator [9, 4, 6]. But its approximation accuracy depends on the sampling and on the foam structure. We can follow the suggestion of Rudin [10] and just discard the underlying neural network and work instead with the trained foam *if* the foam sufficiently approximates the sampled neural classifier. The trained foam may otherwise serve only an auxiliary explanatory role.

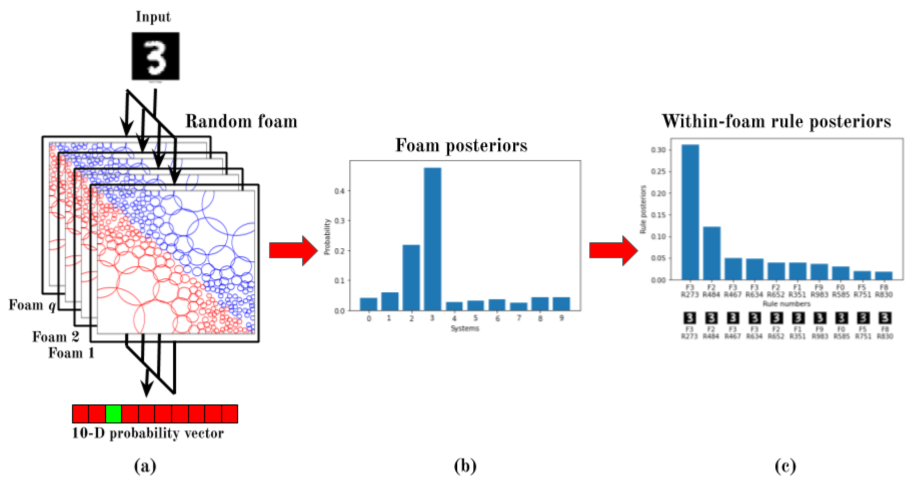

Fig. 4. Telescoping Bayesian posteriors in a Random Foam. (a) The random foams trains 10 foams and then combines them. The random foam then correctly classifies the input pattern from class ‘3’. (b) The random foam gives a Bayesian posterior over all its constituent foams for this input. The foam numbers lie along x-axis and their probabilities lie along the y-axis. Foam 3 contributes most to the classification. (c) The random foam also gives a posterior distribution over the rules present inside the foams. The image shows the 10 rules with the highest posterior probabilities. The x-axis gives the foam numbers and the rule numbers. The y-axis gives their probability. ‘F3 R273’ refers to 273rd rule of the 3rd foam. This rule contributed most to the classification. The rule if-part centroids appear as the images below their posteriors.

References

1. Baader, F., Borgwardt, S., Penaloza, R.: Decidability and complexity of fuzzy description logics. KI-Künstliche Intelligenz **31**(1), 85–90 (2017)
2. Ceylan, I.I., Peñaloza, R.: The bayesian ontology language. Journal of Automated Reasoning **58**(1), 67–95 (2017)
3. Cocarascu, O., Cyras, K., Toni, F.: Explanatory predictions with artificial neural networks and argumentation (2018)
4. Kosko, B.: Fuzzy Systems as Universal Approximators. IEEE Transactions on Computers **43**(11), 1329–1333 (November 1994)
5. Kosko, B.: Additive Fuzzy Systems: From Generalized Mixtures to Rule Continua. International Journal of Intelligent Systems **33**(8), 1573–1623 (2018)
6. Kosko, B.: Convergence of generalized probability mixtures that describe adaptive fuzzy rule-based systems. In: 2020 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE). pp. 1–8. IEEE (2020)
7. Lukasiewicz, T.: Probabilistic logic programming. In: ECAI. pp. 388–392 (1998)
8. Panda, A.K., Kosko, B.: Converting neural networks to rule foam. In: Proceedings of 2019 International Conference on Computational Science and Computational Intelligence (CSCI). pp. 519–525. IEEE (2019)
9. Panda, A.K., Kosko, B.: Random fuzzy-rule foams for explainable ai. In: Fuzzy Information Processing 2020, Proceedings of NAFIPS-2020. Springer (2020)

10. Rudin, C.: Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead. Nature Machine Intelligence **1**(5), 206–215 (2019)
11. Watkins, F.: The representation problem for additive fuzzy systems. In: Proceedings of the International Conference on Fuzzy Systems (IEEE FUZZ-95). pp. 117–122 (1995)