## A. Axiomatization of Semantic Loss: Details

This appendix provides further details on our axiomatization of semantic loss. We detail here a complete axiomatization of semantic loss, which will involve restating some axioms and propositions from the main paper.

The first axiom says that there is no loss when the logical constraint $\alpha$ is always true (it is a logical tautology), independent of the predicted probabilities $\mathsf{p}$.

**Axiom 5** (Truth)**.** The semantic loss of a true sentence is zero: $\forall \mathsf{p}, \mathrm{L}^{\mathrm{s}}(\mathit{true}, \mathsf{p}) = 0$.

Next, when enforcing two constraints on disjoint sets of variables, we want the ability to compute semantic loss for the two constraints separately, and sum the results for their joint semantic loss.

**Axiom 6** (Additive Independence)**.** Let $\alpha$ be a sentence over $\mathbf{X}$ with probabilities $\mathsf{p}$. Let $\beta$ be a sentence over $\mathbf{Y}$ disjoint from $\mathbf{X}$ with probabilities $\mathsf{q}$. The semantic loss between sentence $\alpha \wedge \beta$ and the joint probability vector $[\mathsf{p}\,\mathsf{q}]$ decomposes additively: $\mathrm{L}^{\mathrm{s}}(\alpha \wedge \beta, [\mathsf{p}\,\mathsf{q}]) = \mathrm{L}^{\mathrm{s}}(\alpha, \mathsf{p}) + \mathrm{L}^{\mathrm{s}}(\beta, \mathsf{q})$.

It directly follows from Axioms 5 and 6 that the probabilities of variables that are not used on the constraint do not affect the semantic loss.

Proposition 5 formalizes this intuition.

**Proposition 5** (Locality)**.** *Let $\alpha$ be a sentence over $\mathbf{X}$ with probabilities $\mathsf{p}$. For any $\mathbf{Y}$ disjoint from $\mathbf{X}$ with probabilities $\mathsf{q}$, the semantic loss $\mathrm{L}^{\mathrm{s}}(\alpha, [\mathsf{p}\,\mathsf{q}]) = \mathrm{L}^{\mathrm{s}}(\alpha, \mathsf{p})$.*

*Proof.* Follows from the additive independence and truth axioms. Set $\beta = \mathit{true}$ in the additive independence axiom, and observe that this sets $\mathrm{L}^{\mathrm{s}}(\beta, \mathsf{q}) = 0$ because of the truth axiom. □

To maintain logical meaning, we postulate that semantic loss is monotone in the order of implication.

**Axiom 7** (Monotonicity)**.** If $\alpha \models \beta$, then the semantic loss $\mathrm{L}^{\mathrm{s}}(\alpha, \mathsf{p}) \geq \mathrm{L}^{\mathrm{s}}(\beta, \mathsf{p})$ for any vector $\mathsf{p}$.

Intuitively, as we add stricter requirements to the logical constraint, going from $\beta$ to $\alpha$ and making it harder to satisfy, semantic loss cannot decrease. For example, when $\beta$ enforces the output of an neural network to encode a subtree of a graph, and we tighten that requirement in $\alpha$ to be a path, semantic loss cannot decrease. Every path is also a tree and any solution to $\alpha$ is a solution to $\beta$.

A first consequence following the monotonicity axiom is that logically equivalent sentences must incur an identical semantic loss for the same probability vector $\mathsf{p}$. Hence, the semantic loss is indeed a semantic property of the logical sentence, and *does not depend on the syntax* of the sentence.

**Proposition 6.** *If $\alpha \equiv \beta$, then the semantic loss $\mathrm{L}^{\mathrm{s}}(\alpha, \mathsf{p}) = \mathrm{L}^{\mathrm{s}}(\beta, \mathsf{p})$ for any vector $\mathsf{p}$.*

A second consequence is that semantic loss must be non-negative.

**Proposition 7** (Non-Negativity)**.** *Semantic loss is non-negative.*

*Proof.* Because $\alpha \models \mathit{true}$ for all $\alpha$, the monotonicity axiom implies that $\forall \mathsf{p}, \mathrm{L}^{\mathrm{s}}(\alpha, \mathsf{p}) \geq \mathrm{L}^{\mathrm{s}}(\mathit{true}, \mathsf{p})$. By the truth axiom, $\mathrm{L}^{\mathrm{s}}(\mathit{true}, \mathsf{p}) = 0$, and therefore $\mathrm{L}^{\mathrm{s}}(\alpha, \mathsf{p}) \geq 0$ for all choices of $\alpha$ and $\mathsf{p}$. □

A state $\mathbf{x}$ is equivalently represented as a data vector, as well as a logical constraint that enforces a value for every variable in $\mathbf{X}$. When both the constraint and the predicted vector represent the same state (for example, $X_1 \wedge \neg X_2 \wedge X_3$ vs. $[1\,0\,1]$), there should be no semantic loss.

**Axiom 8** (Identity)**.** For any state $\mathbf{x}$, there is zero semantic loss between its representation as a sentence, and its representation as a deterministic vector: $\forall \mathbf{x}, \mathrm{L}^{\mathrm{s}}(\mathbf{x}, \mathbf{x}) = 0$.

The axioms above together imply that any vector satisfying the constraint must incur zero loss. For example, when our constraint $\alpha$ requires that the output vector encodes an arbitrary total ranking, and the vector $\mathbf{x}$ correctly represents a single specific total ranking, there is no semantic loss.

**Proposition 8** (Satisfaction)**.** *If $\mathbf{x} \models \alpha$, then the semantic loss $\mathrm{L}^{\mathrm{s}}(\alpha, \mathbf{x}) = 0$.*

*Proof of Proposition 8.* The monotonicity axiom specializes to say that if $\mathbf{x} \models \alpha$, we have that $\forall \mathsf{p}, \mathrm{L}^{\mathrm{s}}(\mathbf{x}, \mathsf{p}) \geq \mathrm{L}^{\mathrm{s}}(\alpha, \mathsf{p})$. By choosing $\mathsf{p}$ to be $\mathbf{x}$, this implies $\mathrm{L}^{\mathrm{s}}(\mathbf{x}, \mathbf{x}) \geq \mathrm{L}^{\mathrm{s}}(\alpha, \mathbf{x})$. From the identity axiom, $\mathrm{L}^{\mathrm{s}}(\mathbf{x}, \mathbf{x}) = 0$, and therefore $0 \geq \mathrm{L}^{\mathrm{s}}(\alpha, \mathbf{x})$. Proposition 7 bounds the loss from below as $\mathrm{L}^{\mathrm{s}}(\alpha, \mathbf{x}) \geq 0$. □

As a special case, logical literals ($x$ or $\neg x$) constrain a single variable to take on a single value, and thus play a role similar to the labels used in supervised learning. Such constraints require an even tighter correspondence: semantic loss must act like a classical loss function (i.e., cross entropy).

**Axiom 9** (Label-Literal Correspondence)**.** The semantic loss of a single literal is proportionate to the cross-entropy loss for the equivalent data label: $\mathrm{L}^{\mathrm{s}}(x, p) \propto -\log(p)$ and $\mathrm{L}^{\mathrm{s}}(\neg x, p) \propto -\log(1-p)$.

Next, we have the symmetry axioms.

**Axiom 10** (Value Symmetry)**.** For all $\mathsf{p}$ and $\alpha$, we have that $\mathrm{L}^{\mathrm{s}}(\alpha, \mathsf{p}) = \mathrm{L}^{\mathrm{s}}(\bar{\alpha}, 1 - \mathsf{p})$ where $\bar{\alpha}$ replaces every variable in $\alpha$ by its negation.

**Axiom 11** (Variable Symmetry)**.** Let $\alpha$ be a sentence over $\mathbf{X}$ with probabilities $\mathsf{p}$. Let $\pi$ be a permutation of the variables $\mathbf{X}$, let $\pi(\alpha)$ be the sentence obtained by replacing variables $x$ by $\pi(x)$, and let $\pi(\mathsf{p})$ be the corresponding permuted vector of probabilities. Then, $\mathrm{L}^{\mathrm{s}}(\alpha, \mathsf{p}) = \mathrm{L}^{\mathrm{s}}(\pi(\alpha), \pi(\mathsf{p}))$.

The value and variable symmetry axioms together imply the equality of the multiplicative constants in the label-literal duality axiom for all literals.

**Lemma 9.** *There exists a single constant $K$ such that $\mathrm{L}^{\mathrm{s}}(X, p) = -K \log(p)$ and $\mathrm{L}^{\mathrm{s}}(\neg X, p) = -K \log(1-p)$ for any literal $x$.*

*Proof.* Value symmetry implies that $\mathrm{L}^{\mathrm{s}}(X_i, \mathsf{p}) = \mathrm{L}^{\mathrm{s}}(\neg X_i, 1 - \mathsf{p})$. Using label-literal correspondence, this implies $K_1 \log(p_i) = K_2 \log(1 - (1 - p_i))$ for the multiplicative constants $K_1$ and $K_2$ that are left unspecified by that axiom. This implies that the constants are identical. A similar argument based on variable symmetry proves equality between the multiplicative constants for different $i$. □

Finally, this allows us to prove the following form of semantic loss for a state $\mathbf{x}$.

**Lemma 10.** *For state $\mathbf{x}$ and vector $\mathsf{p}$, we have $\mathrm{L}^{\mathrm{s}}(\mathbf{x}, \mathsf{p}) \propto -\sum_{i:\mathbf{x} \models X_i} \log \mathsf{p}_i - \sum_{i:\mathbf{x} \models \neg X_i} \log(1 - \mathsf{p}_i)$.*

*Proof of Lemma 10.* A state $\mathbf{x}$ is a conjunction of independent literals, and therefore subject to the additive independence axiom. Each literal's loss in this sum is defined by Lemma 9. □

The following and final axiom requires that semantic loss is proportionate to the logarithm of a function that is additive for mutually exclusive sentences.

**Axiom 12** (Exponential Additivity)**.** Let $\alpha$ and $\beta$ be mutually exclusive sentences (i.e., $\alpha \wedge \beta$ is unsatisfiable), and let $f^s(K, \alpha, \mathsf{p}) = K^{-\mathrm{L}^{\mathrm{s}}(\alpha, \mathsf{p})}$. Then, there exists a positive constant $K$ such that $f^s(K, \alpha \vee \beta, \mathsf{p}) = f^s(K, \alpha, \mathsf{p}) + f^s(K, \beta, \mathsf{p})$.

We are now able to state and prove the main uniqueness theorem.

**Theorem 11** (Uniqueness)**.** *The semantic loss function in Definition 1 satisfies all axioms in Appendix A and is the only function that does so, up to a multiplicative constant.*

*Proof of Theorem 11.* The truth axiom states that $\forall \mathsf{p}, f^s(K, true, \mathsf{p}) = 1$ for all positive constants $K$. This is the first Kolmogorov axiom of probability. The second Kolmogorov axiom for $f^s(K, ., \mathsf{p})$ follows from the additive independence axiom of semantic loss. The third Kolmogorov axiom (for the finite discrete case) is given by the exponential additivity axiom of semantic loss. Hence, $f^s(K, ., \mathsf{p})$ is a probability distribution for some choice of $K$, which implies the definition up to a multiplicative constant. □

## B. Specification of the Convolutional Neural Network Model

Table 6 shows the slight architectural difference between the CNN used in ladder nets and ours. The major difference lies in the choice of ReLu. Note we add standard padded cropping to preprocess images and an additional fully connected layer at the end of the model, neither is used in ladder nets. We only make those slight modification so that the baseline performance reported by Rasmus et al. (2015) can be reproduced.

## C. Hyper-parameter Tuning Details

Validation sets are used for tuning the weight associated with semantic loss, the only hyper-parameter that causes noticeable difference in performance for our method. For our semi-supervised classification experiments, we perform a grid search over $\{0.001, 0.005, 0.01, 0.05, 0.1\}$ to find the optimal value. Empirically, $0.005$ always gives the best or nearly the best results and we report its results on all experiments.

For the FASHION dataset specifically, because MNIST and FASHION share the same image size and structure, methods developed in MNIST should be able to directly perform on FASHION without heavy modifications. Because of this, we use the same hyper-parameters when evaluating our method. However, for the sake of fairness, we subject ladder nets to a small-scale parameter tuning in case its performance is more volatile.

For the grids experiment, the only hyper parameter that needed to be tuned was again the weight given to semantic loss, which after trying $\{0.001, 0.005, 0.01, 0.05, 0.1, 0.5, 1\}$ was selected to be 0.5 based on validation results. For the preference learning experiment, we initially chose the semantic loss weight from $\{0.001, 0.005, 0.01, 0.05, 0.1, 0.5, 1\}$ to be 0.1 based on validation, and then further tuned the weight to 0.25.

## D. Specification of Complex Constraint Models

**Grids** To compile our grid constraint, we first use Nishino et al. (2017) to generate a constraint for each source destination pair. Then, we conjoin each of these with indicators specifying which source and destination

(a) Confidently Correct (b) Unconfidently Correct (c) Unconfidently Incorrect (d) Confidently Incorrect

Figure 5: FASHION pictures grouped by how confidently the supervised base model classifies them correctly. With semantic loss, the final semi-supervised model predicts all correctly and confidently.

Table 6: Specifications of CNNs in Ladder Net and our proposed method.

| CNN in Ladder Net | CNN in this paper |
|---|---|
| Input 32×32 RGB image | |
| | Resizing to 36 × 36 with padding<br>Cropping Back |
| Whitening<br>Contrast Normalization<br>Gaussian Noise with std. of 0.3 | |
| 3×3 conv. 96 BN LeakyReLU<br>3×3 conv. 96 BN LeakyReLU<br>3×3 conv. 96 BN LeakyReLU | 3×3 conv. 96 BN ReLU<br>3×3 conv. 96 BN ReLU<br>3×3 conv. 96 BN ReLU |
| 2×2 max-pooling stride 2 BN | |
| 3×3 conv. 192 BN LeakyReLU<br>3×3 conv. 192 BN LeakyReLU<br>3×3 conv. 192 BN LeakyReLU | 3×3 conv. 192 BN ReLU<br>3×3 conv. 192 BN ReLU<br>3×3 conv. 192 BN ReLU |
| 2×2 max-pooling stride 2 BN | |
| 3×3 conv. 192 BN LeakyReLU<br>1×1 conv. 192 BN LeakyReLU<br>1×1 conv. 10 BN LeakyReLU | 3×3 conv. 192 BN ReLU<br>3×3 conv. 192 BN ReLU<br>1×1 conv. 10 BN ReLU |
| Global meanpool BN | |
| | Fully connected BN |
| 10-way softmax | |

pair must be used, and finally we disjoin all of these together to form our constraint.

To generate the data, we begin by randomly removing one third of edges. We then filter out connected components with fewer than 5 nodes to reduce degenerate cases, and proceed with randomly selecting pairs of nodes to create data points.

The predictive model we employ as our baseline is a 5 layer MLP with 50 hidden sigmoid units per layer. It is trained using Adam Optimizer, with full data batches (Kingma & Ba, 2015). Early stopping with respect to validation loss is used as a regularizer.

**Preference Learning** We split each user's ordering into their ordering over sushis 1,2,3,5,7,8, which we use as the features, and their ordering over 4,6,9,10 which are the labels we predict. The constraint is compiled directly from logic, as this can be done in a straightforward manner for an n-item ordering.

The predictive model we use here is a 3 layer MLP with 25 hidden sigmoid units per layer. It is trained using Adam Optimizer with full data batches (Kingma & Ba, 2015). Early stopping with respect to validation loss is used as a regularizer.

## E. Probabilistic Soft Logic Encodings

We here give both encodings on the exactly-one constraint over three $x_1, x_2, x_3$. The first encoding is:

$$(\neg x_1 \wedge x_2 \wedge x_3) \vee (x_1 \wedge \neg x_2 \wedge x_3) \vee (x_1 \wedge x_2 \wedge \neg x_3)$$

The second encoding is:

$$(x_1 \vee x_2 \vee x_3) \wedge (\neg x_1 \vee \neg x_2) \wedge (\neg x_1 \vee \neg x_3) \wedge (\neg x_2 \vee \neg x_3)$$

Both encodings extend to cases whether the number of variables is arbitrary.

The norm functions used for these experiments are as described in Kimmig et al. (2012), with the loss for an interpretation $I$ being defined as follows:

$$
\begin{aligned}
x_1 \wedge x_2 &= max\{0, I(x_1) + I(x_2) - 1\} \\
x_1 \vee x_2 &= min\{I(x_1) + I(x_2), 1\} \\
\neg x_1 &= 1 - I(x_1)
\end{aligned}
$$