Weight Discretization for Quotient Model Abstraction in Spiking Neural Network Verification

Research Note — January 2026

1. Introduction

This note extends the filtration-based quotient model abstraction [1] to preserve *synaptic weight information* during formal verification. While the original quotient model (see `research_note_filtration.typ`) abstracts membrane potentials into equivalence classes, it treats all synapses uniformly, losing essential structural information.

We address this by introducing a *weight discretization scheme* that:
1. Maps continuous weights to a finite discrete range
2. Preserves the relative contribution of synapses to membrane potential
3. Ensures threshold feasibility is maintained
4. Retains weight visibility in the generated PRISM model

2. Weight Discretization

2.1. Formal Definition

Definition. Given a weight range $[w_{\min}, w_{\max}] = [-100, 100]$ and a discretization parameter $W \in \mathbb{N}^{+}$, the *weight discretization function* $\delta_W : \mathbb{Z} \to \mathbb{Z}$ is defined as:

$$\delta_{W(w)} = \operatorname{round}\left(w \cdot \frac{W}{w_{\max}}\right)$$

The discretized weight range is $[-W, W] \subset \mathbb{Z}$.

Example. For $W = 3$ (7 levels total):
- $\delta_3(100) = 3$ (strong excitatory)
- $\delta_3(67) = 2$ (medium excitatory)
- $\delta_3(33) = 1$ (weak excitatory)
- $\delta_3(0) = 0$ (negligible)
- $\delta_3(-50) = -2$ (medium inhibitory)
- $\delta_3(-100) = -3$ (strong inhibitory)

2.2. Threshold Calibration

The key challenge is ensuring that threshold reachability is preserved after discretization. We must calibrate the *discretized threshold* T_d to be consistent with the original threshold T.

Definition. The *discretized threshold* for a neuron with original threshold T and weight discretization parameter W is:

$$T_d = \text{ceil}\left(T \cdot \frac{W}{w_{\max}}\right)$$

Theorem (Threshold Preservation)**.** Let $\mathcal{N}$ be a neuron with incoming weights $\{w_1, ..., w_m\}$ and threshold T. Let $\mathcal{N}'$ be the discretized version with weights $\{\delta_W(w_1), ..., \delta_W(w_m)\}$ and threshold T_d.

If $\mathcal{N}$ can fire in a single step (i.e., $\exists$ input pattern $\boldsymbol{y} \in \{0,1\}^m$ such that $\sum_{i=1}^{m} w_i \cdot y_i \geq T$), then $\mathcal{N}'$ can also fire in a single step.

Proof. Let $\boldsymbol{y}^*$ be an input pattern that causes $\mathcal{N}$ to fire:

$$S = \sum_{i=1}^{m} w_i \cdot y_i^* \geq T$$

The discretized contribution is:

$$S_d = \sum_{i=1}^{m} \delta_W(w_i) \cdot y_i^*$$

By the rounding property of δ_W:

$$\delta_W(w_i) \geq \frac{w_i \cdot W}{w_{\max}} - \frac{1}{2}$$

Therefore:

$$S_d \geq \left(\sum_{i=1}^{m} w_i \cdot y_i^*\right) \cdot \frac{W}{w_{\max}} - \frac{m}{2} = S \cdot \frac{W}{w_{\max}} - \frac{m}{2}$$

Since $S \geq T$:

$$S_d \geq T \cdot \frac{W}{w_{\max}} - \frac{m}{2}$$

For the discretized model to fire, we need $S_d \geq T_d$. By choosing:

$$T_d = \text{ceil}\left(T \cdot \frac{W}{w_{\max}}\right) \leq T \cdot \frac{W}{w_{\max}} + 1$$

A sufficient condition is:

$$S \cdot \frac{W}{w_{\max}} - \frac{m}{2} \geq T \cdot \frac{W}{w_{\max}} + 1$$

Which simplifies to:

$$(S - T) \cdot \frac{W}{w_{\max}} \geq \frac{m}{2} + 1$$

This holds when $S - T$ is sufficiently large relative to m. For the boundary case $S = T$, we need $W \geq w_{\max} \cdot (\frac{m}{2} + 1)/T$.

In practice, with $W = 3$, $w_{\max} = 100$, and typical networks ($m \leq 10$, $T = 100$):

$$W \geq 100 \cdot \frac{6}{100} = 6$$

Thus $W = 7$ (range $[-3, 3]$) is sufficient for most practical networks. For larger fan-in, a higher W may be required. □

Remark. The proof shows that weight discretization can introduce a *margin of error* proportional to the fan-in m. For high-fanin neurons ($m > 10$), we recommend either:
1. Using a finer discretization ($W \geq 5$)
2. Applying a threshold correction factor: $T_{d'} = T_d - \text{floor}\left(\frac{m}{2W}\right)$

3. Class Transition with Weighted Contributions

3.1. Contribution-Based Class Evolution

In the original quotient model, class evolution used a binary rule:
- Any input fires → class increases by 1
- No input fires → class decreases by 1 (leak)

This loses weight information. We replace it with *weighted contribution-based* class evolution.

Definition. The *weighted contribution* for neuron n with incoming discretized weights $\left\{w_1^d, ..., w_m^d\right\}$ is:

$$C_n = \sum_{i=1}^{m} w_i^d \cdot y_i$$

where $y_i \in \{0, 1\}$ is the spike output of presynaptic neuron i.

Definition. The *class delta function* $\Delta : \mathbb{Z} \to \mathbb{Z}$ maps contribution to class change:

$$\Delta(C) = \text{clamp}\left(\text{round}\left(\frac{C}{\gamma}\right), -k, k\right)$$

where γ is the *class width* (typically $\gamma = \frac{T_d}{k}$) and k is the number of threshold levels.

Proposition. The class delta function preserves the ordering of contributions: if $C_1 > C_2$, then $\Delta(C_1) \geq \Delta(C_2)$.

Proof. This follows directly from the monotonicity of `round` and `clamp` operations. □

3.2. Integration with Leak Rate

The quotient model must also account for membrane potential decay (leak). We propose *discretized leak*:

Definition. The *discretized leak factor* λ_d is:

$$\lambda_d = 1 - \text{round}((1 - r) \cdot k)$$

where $r \in [0, 1]$ is the original leak rate and k is the number of classes.

The class evolution rule becomes:

$$c'_n = \text{clamp}(c_n + \Delta(C_n) + \lambda_d, 0, k)$$

Example. For $r = 0.9$ (10% decay per step) and $k = 4$:

$$\lambda_d = 1 - \text{round}(0.1 \cdot 4) = 1 - 0 = 1$$

So with no input, the class stays the same + leak = same - 0 (no change from this formula).

For $r = 0.5$ (50% decay):

$$\lambda_d = 1 - \text{round}(0.5 \cdot 4) = 1 - 2 = -1$$

So with no input, class decreases by 1 per step.

Remark. We choose to apply leak *only when no excitatory input fires* to maintain consistency with the precise model's behavior, where leak is a multiplicative factor on the existing potential.

4. Threshold Feasibility Analysis

4.1. Definition

Definition. A neuron configuration is *threshold-feasible* if there exists at least one input pattern that can cause the neuron to reach the firing threshold within a finite number of steps.

Theorem (Feasibility Criterion)**.** A neuron with discretized weights $\{w_1^d, ..., w_m^d\}$ and threshold T_d is threshold-feasible if and only if:

$$\sum_{w_i^d > 0} w_i^d \geq \frac{T_d}{1 + |\lambda_d|}$$

Proof. Let $E = \sum_{w_i^d > 0} w_i^d$ be the maximum excitatory contribution per step.

Sufficiency: If $E \geq T_d$, the neuron can fire in a single step when all excitatory inputs fire. Otherwise, the neuron accumulates potential over steps. With leak factor $\lambda_d \leq 0$, each step adds at most $E + \lambda_d$ net contribution.

For the class to reach k (firing threshold), starting from class 0:

$$n \cdot (E + \lambda_d) \geq k$$

The minimum steps required is $n = \text{ceil}\left(\frac{k}{E + \lambda_d}\right)$, which is finite iff $E + \lambda_d > 0$, i.e., $E > |\lambda_d|$.

Since $T_d = k \cdot \gamma$ and $\gamma \approx \frac{T_d}{k}$, the condition $E \geq \frac{T_d}{1 + |\lambda_d|}$ ensures firing is reachable.

Necessity: If $E < \frac{T_d}{1+|\lambda_d|}$, then even with optimal accumulation, the maximum class reached is bounded by $\frac{E}{|}\lambda_d| < k$, so firing is unreachable. □

4.2. Implementation

The feasibility check should be performed at PRISM generation time:

```
fn check_feasibility(
    weights: &[i32],  // discretized weights
    threshold: i32,
    leak_factor: i32,
) -> Feasibility {
    let max_excitation: i32 = weights.iter()
        .filter(|&&w| w > 0)
        .sum();

    let min_required = threshold / (1 + leak_factor.abs());

    if max_excitation >= threshold {
        Feasibility::SingleStep
    } else if max_excitation >= min_required {
        let steps = (threshold + max_excitation - 1) / max_excitation;
        Feasibility::MultiStep { min_steps: steps }
    } else {
        Feasibility::Impossible
    }
}
```

5. PRISM Model Structure

The weighted quotient model generates PRISM code with explicit weight constants and contribution formulas:

```
// Weight constants (discretized)
const int W = 3;  // Discretization parameter
const int W_in0_2 = 3;   // δ_3(100) = 3
const int W_n1_2 = -2;   // δ_3(-67) = -2
const int W_n3_2 = 1;    // δ_3(33) = 1

// Discretized threshold
const int T_d = 3;

// Contribution formula (evaluated at runtime)
formula contrib_2 = W_in0_2 * x0 + W_n1_2 * z1_2 + W_n3_2 * z3_2;

// Class delta (clamped)
formula delta_2 = max(-4, min(4, contrib_2));

// Class evolution with weighted contribution
[tick] y2=0 & pClass2=0 -> (pClass2' = max(0, min(4, pClass2 + delta_2)));
```

6. Conclusion

We have established a formal framework for weight discretization in quotient model abstraction:

1. **Threshold Preservation Theorem** guarantees firing reachability is maintained
2. **Contribution-based class evolution** preserves weight effects on dynamics

3. **Feasibility analysis** detects and warns about unreachable configurations
4. **Discretized leak** maintains decay behavior in a state-preserving manner

The key insight is that weights enter the PRISM model as *constants*, not *state variables*, so weight discretization does not increase state space—it only affects transition probabilities and guard conditions.

Bibliography

[1] C. Baier and J.-P. Katoen, *Principles of Model Checking*. MIT Press, 2008.