Weight Discretization for Quotient Model Abstraction in Spiking Neural Network Verification

Research Note — February 2026
(Fifth Revision)

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 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

What's New in Revision 5:

- Proper mathematical notation throughout ($\lfloor\cdot\rfloor/\lceil\cdot\rceil$ brackets, operator names)
- Leak factor ℓ replaces retention rate r as primary variable for consistency
- Leak factor dynamics analysis: signal contribution, spike difficulty, expected spike count (§4.3)
- Probabilistic formulation of biological property preservation (§5.3)
- Completed soundness proof with zero-potential non-spiking guarantee (§3.4)
- Four simulation diagram placeholders with exact parameter configurations (§8)
- Variable dependency graph updated for leak factor formulation (§6)

2. Preliminaries

This section introduces key concepts and notation used throughout the proofs.

2.1. Notation Summary

Symbol	Meaning
W	Discretization parameter: the number of positive weight levels
$w_{\max}$	Maximum absolute weight in the original model (typically 100)
δ_W	Weight discretization function: maps original weights to discrete values
T	Original firing threshold of a neuron
T_d	Discretized firing threshold
m	Fan-in: the number of incoming synapses to a neuron
S	Weighted sum of spiking inputs in the original model
S_d	Weighted sum of spiking inputs in the discretized model
γ	Class width: potential range represented by each equivalence class
λ_d	Discretized leak factor (always ≤ 0), *dependent on* T_d
k	Number of threshold levels (equivalence classes)
ℓ	Leak factor: fraction of potential lost per step ($\ell \in [0,1]$). Related to retention rate by $\ell = 1 - r$

Table 1: Summary of notation used in this document

2.2. The Rounding Property

The standard mathematical rounding function $\text{round}(x)$ rounds x to the nearest integer. A crucial property we use throughout is the *rounding bound*:

Definition. For any real number x, the rounding function satisfies:

$$x - \frac{1}{2} \leq \text{round}(x) \leq x + \frac{1}{2}$$

Equivalently: $\text{round}(x) \geq x - \frac{1}{2}$ (lower bound) and $\text{round}(x) \leq x + \frac{1}{2}$ (upper bound).

💡 **Intuition:** Rounding can shift a value by at most $\frac{1}{2}$ in either direction. When we sum m rounded values, the total error is bounded by $\frac{m}{2}$ — this is the *cumulative rounding error.*

2.3. The Clamp Function

Definition. The *clamp function* restricts a value to a specified range $[a, b]$ where $a < b$:

$$\text{clamp}(x, a, b) = \max(a, \min(b, x)) = \begin{cases} a \text{ if } x < a \\ x \text{ if } a \leq x \leq b \\ b \text{ if } x > b \end{cases}$$

💡 **Intuition:** Clamping prevents values from exceeding safe bounds. In our model, we clamp class indices to $[0, k]$ and class deltas to $[-k, k]$ to avoid invalid states or runaway accumulation.

2.4. Fan-in

Definition. The *fan-in* of a neuron, denoted m, is the number of incoming synaptic connections. A neuron with fan-in m receives input from m presynaptic neurons.

Remark. Fan-in is critical because the cumulative rounding error scales linearly with m. High fan-in neurons require finer discretization to maintain accuracy.

3. Weight Discretization

3.1. 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* $\delta_W : \mathbb{R} \to \mathbb{Z}$ is defined as:

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

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

💡 **Intuition:** We scale the original weight by $\frac{W}{w_{\max}}$ to map the range $[-w_{\max}, w_{\max}]$ to $[-W, W]$, then round to get an integer. This preserves the *relative magnitude* of weights while reducing the number of distinct values.

Example. For $W = 3$ (giving 7 discrete levels: $-3, -2, -1, 0, 1, 2, 3$):

- $\delta_3(100) = \operatorname{round}(100 \cdot \frac{3}{100}) = \operatorname{round}(3) = 3$ (strong excitatory)
- $\delta_3(67) = \operatorname{round}(67 \cdot \frac{3}{100}) = \operatorname{round}(2.01) = 2$ (medium excitatory)
- $\delta_3(33) = \operatorname{round}(33 \cdot \frac{3}{100}) = \operatorname{round}(0.99) = 1$ (weak excitatory)
- $\delta_3(0) = \operatorname{round}(0) = 0$ (negligible)
- $\delta_3(-50) = \operatorname{round}(-50 \cdot \frac{3}{100}) = \operatorname{round}(-1.5) = -2$ (medium inhibitory)
- $\delta_3(-100) = \operatorname{round}(-3) = -3$ (strong inhibitory)

3.2. 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:

$$T_d = \left\lceil T \cdot \frac{W}{w_{\max}} \right\rceil$$

💡 **Intuition:** We use ceiling (round up) rather than standard rounding to ensure the discretized threshold is *at least as hard* to reach as the original. This prevents false positives: if a discretized neuron fires, we can be confident the original would too.

3.3. Threshold Preservation Theorem (Completeness)

Theorem (Threshold Preservation — Completeness)**.** 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. We prove this in six steps.

Step 1 (Setup)**:** Let $\boldsymbol{y}^*$ be an input pattern that causes the original neuron $\mathcal{N}$ to fire. Define:

- $S = \sum_{i=1}^m w_i \cdot y_i^*$ — the weighted sum of spiking inputs in the original model
- By assumption, $S \geq T$ (the neuron fires)

Step 2 (Discretized contribution)**:** The corresponding weighted sum in the discretized model is:

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

We need to show $S_d \geq T_d$ to prove the discretized neuron also fires.

Step 3 (Apply the rounding property)**:** By the rounding property (see §2.2), for each weight:

$$\delta_W(w_i) = \text{round}\left(w_i \cdot \frac{W}{w_{\max}}\right) \geq \frac{w_i \cdot W}{w_{\max}} - \frac{1}{2}$$

Since $y_i^* \in \{0,1\}$, when $y_i^* = 1$ this bound applies, and when $y_i^* = 0$ the term is zero. Summing over all inputs:

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

Step 4 (Bound the cumulative error)**:** Let $m^* = \sum_{i=1}^m y_i^*$ be the number of active inputs. Then:

$$S_d \geq S \cdot \frac{W}{w_{\max}} - \frac{m^*}{2}$$

Since $m^* \leq m$ (at most m inputs can be active):

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

The term $\frac{m}{2}$ is the *cumulative rounding error* — the worst-case total error from rounding m weights.

Step 5 (Derive the firing condition)**:** For the discretized neuron to fire, we need $S_d \geq T_d$. By the ceiling property:

$$T_d = \left\lceil T \cdot \frac{W}{w_{\max}} \right\rceil \leq T \cdot \frac{W}{w_{\max}} + 1$$

Combining with our bound on S_d:

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

Since $S \geq T$:

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

A sufficient condition for firing is:

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

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

Step 6 (Boundary case and parameter choice)**:** For the critical boundary case where $S = T$ exactly, we need:

$$0 \geq \frac{m}{2} + 1$$

This is never satisfied, so at the exact boundary, firing is not guaranteed. However, if S exceeds T by a small margin, or we choose W large enough, firing is preserved.

Specifically, for $W \geq w_{\max} \cdot \frac{\frac{m}{2}+1}{T}$, the discretized neuron fires whenever the original does.

Practical example: With $W = 3$, $w_{\max} = 100$, $m \leq 10$, and $T = 100$:

$$W \geq 100 \cdot \frac{5+1}{100} = 6$$

Thus $W = 7$ (discrete range $[-3, 3]$) is sufficient for most practical networks.

$\square$

Remark. Critical Constraint: The proof shows that weight discretization introduces a cumulative error of $-\frac{m}{2}$. If the fan-in m is large relative to W (specifically if $m > 2W$), the rounding noise may exceed the signal of the smallest synaptic weight.

For high-fanin neurons, we strictly recommend:
1. Using a finer discretization ($W \geq \frac{m}{2}$)
2. Applying a threshold correction factor: $T_{d'} = T_d - \left\lfloor \frac{m}{2W} \right\rfloor$ (Note: This may increase false positive firings).

3.4. Soundness Theorem (Safety)

We now prove the converse property: if the original neuron should *not* fire, the discretized neuron should also *not* fire.

Theorem (Asymptotic Silence — Soundness)**.** Let $\mathcal{N}'$ be a discretized neuron with current potential $P_t < T_d$ and leak factor $\lambda_d \le -1$. If the input sequence is empty (i.e., total weighted input $S_d = 0$) for all steps $t' \ge t$, then $\mathcal{N}'$ will never fire.

Proof. We prove this by showing the potential strictly decreases to zero and that zero potential guarantees non-spiking.

Step 1 (Dynamics without input)**:** In the absence of input ($S_d = 0$), the discretized update rule simplifies to:

$$P_{t+1} = \max(0, P_t + \lambda_d)$$

This is applying the potential update with only leak (no external contribution).

Step 2 (Strict decay guarantee)**:** Since $\lambda_d \le -1$, if $P_t > 0$, then:

$$P_{t+1} \le P_t + \lambda_d \le P_t - 1$$

The potential strictly decreases by at least 1 unit per step.

Step 3 (Convergence to zero)**:** Starting from any $P_t < T_d$, the sequence $\{P_t\}$ is strictly decreasing until it reaches the absorbing state $P = 0$ (due to the $\max(0, ...)$ clamp).

Step 4 (Firing impossibility during decay)**:** Since $P_t < T_d$ initially and the sequence is non-increasing, the firing condition $P \ge T_d$ is never met during the decay trajectory. The neuron cannot fire while the potential is decreasing.

Step 5 (Zero potential guarantees non-spiking)**:** Once $P = 0$, the neuron is at resting potential. Since the firing probability is determined by the potential's position relative to the discrete threshold levels, $P = 0$ maps to level $L = 0$, which corresponds to a firing probability of 0. The probability of *not* spiking is therefore 1 (i.e., 100%). Combined with the absence of input, the neuron remains permanently at $P = 0$ with zero probability of emitting a spike.

$\square$

💡 **Intuition:** This theorem provides a *safety guarantee*: the discretization does not introduce spurious spikes. If there's no input to drive the potential up, leak ensures it decays to rest, and rest guarantees zero firing probability. This prevents rounding errors from accidentally triggering false firings.

4. Class Transition with Weighted Contributions

4.1. Contribution-Based Class Evolution

In the original quotient model, class evolution used a binary rule which loses weight information. We replace it with *weighted contribution-based* class evolution.

> **Definition.** The *weighted contribution* for neuron n with incoming discretized weights $\{w_1^d, ..., w_m^d\}$ 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:
>
> $$\Delta(C) = \operatorname{clamp}\left(\operatorname{round}\left(\frac{C}{\gamma}\right), -k, k\right)$$
>
> where:
> - γ is the *class width* (typically $\gamma = \frac{T_d}{k}$) — the potential range each class represents
> - k is the number of threshold levels
> - The clamp ensures the class change stays within valid bounds

> 💡 **Intuition:** The class delta converts a weighted sum of inputs into a class change. We divide by γ to express the contribution in "class units", round to get an integer, and clamp to prevent impossibly large jumps.

4.2. Threshold-Dependent Leak Factor

The quotient model must also account for membrane potential decay (leak). We define the leak factor using the *leak parameter* ℓ, which represents the fraction of potential lost per step.

> **Definition.** The *discretized leak factor* λ_d is:
>
> $$\lambda_d = -\max(1, \lfloor \ell \cdot T_d \rfloor)$$
>
> where:
> - $\ell \in [0, 1]$ is the leak factor (fraction of potential lost per step, with $\ell = 1 - r$ where r is the retention rate)
> - T_d is the discretized threshold
> - The $\max(1, ...)$ ensures a minimum decay of 1 unit per step (preventing infinite energy trapping)
> - The negative sign ensures leak *decreases* potential

> 💡 **Intuition:** By linking the leak factor to T_d rather than the number of classes k, we ensure the decay scales correctly with the calibrated threshold. A high threshold means larger potential values, so the decay must be proportionally larger to maintain realistic behavior.

Thinking in terms of ℓ (how much leaks away) rather than r (how much is retained) provides a more direct relationship: higher ℓ means more aggressive decay.

Example. For $\ell = 0.1$ (10% leak, i.e., 90% retention) and $T_d = 10$:

$$\lambda_d = -\max(1, \lfloor 0.1 \cdot 10 \rfloor) = -\max(1, \lfloor 1.0 \rfloor) = -1$$

Result: Potential decreases by 1 per step without input.

For $\ell = 0.5$ (50% leak) and $T_d = 10$:

$$\lambda_d = -\max(1, \lfloor 0.5 \cdot 10 \rfloor) = -\max(1, \lfloor 5.0 \rfloor) = -5$$

Result: Potential decreases by 5 per step without input — aggressive decay.

For $\ell = 0.05$ (5% leak, i.e., 95% retention) and $T_d = 10$:

$$\lambda_d = -\max(1, \lfloor 0.05 \cdot 10 \rfloor) = -\max(1, \lfloor 0.5 \rfloor) = -\max(1, 0) = -1$$

Result: Even with very low leak, we enforce minimum decay of 1.

Remark. Key Insight: The previous formulation $\lambda_d = -\operatorname{round}(\ell \cdot k)$ was problematic because:
1. It depended on k (number of classes) rather than T_d (actual potential scale)
2. Could result in $\lambda_d = 0$ for low leak, violating the Soundness theorem

The new formulation ensures $|\lambda_d| \geq 1$ always, guaranteeing the Asymptotic Silence property.

The class evolution rule becomes:

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

4.3. Leak Factor Dynamics

This section characterizes the three key relationships governed by the leak factor ℓ, providing insight into how leak affects neuron behavior across the model.

4.3.1. Signal Contribution

The leak factor directly controls how much of the *current* input signal contributes to potential accumulation at each time step. From the membrane potential update:

$$P_{t+1} = \max(0, (1 - \ell) \cdot P_t + C_t)$$

The retained potential from the previous step is scaled by $(1 - \ell)$, while the current input C_t is added in full. The effective weight of past vs. present signal is therefore:

Leak factor ℓ	**Retention $(1-\ell)$**	**Effect on signal**
$\ell \to 0$	≈ 1	Past and present signals weighted equally — full memory
$\ell = 0.1$	0.9	Mild decay; accumulated history dominates
$\ell = 0.5$	0.5	Current input and history contribute equally
$\ell \to 1$	≈ 0	Only the current input matters — no memory

Table 2: Signal contribution as a function of leak factor

4.3.2. Spike Emission Difficulty

The leak factor determines the *minimum sustained input* required for a neuron to ever reach threshold. From the feasibility analysis (§5.1), a spike is only possible if the net gain per step is positive:

$$C_{\text{in}} > |\lambda_d| = \max(1, \lfloor \ell \cdot T_d \rfloor)$$

This establishes a *noise floor*: any input below $|\lambda_d|$ is completely absorbed by the leak and cannot contribute to potential accumulation. As ℓ increases, this floor rises, making spike emission increasingly difficult:

Definition. The *minimum excitation for spiking* is:

$$C_{\text{min}} = |\lambda_d| + 1 = \max(1, \lfloor \ell \cdot T_d \rfloor) + 1$$

Any constant input $C_{\text{in}} < C_{\text{min}}$ will never cause the neuron to fire, regardless of duration.

4.3.3. Expected Spike Frequency

For a neuron receiving constant input $C_{\text{in}} > |\lambda_d|$, the *expected number of spikes* over N time steps can be approximated. The inter-spike interval (ISI) is:

$$N_{\text{steps}} = \left\lceil \frac{T_d}{C_{\text{in}} - |\lambda_d|} \right\rceil$$

Therefore the expected spike count is:

$$\mathbb{E}[\text{spikes}] \approx \frac{N}{N_{\text{steps}}} = N \cdot \frac{C_{\text{in}} - |\lambda_d|}{T_d}$$

As ℓ increases, $|\lambda_d|$ increases, the net gain $(C_{\text{in}} - |\lambda_d|)$ decreases, and consequently the expected spike count decreases. This relationship is monotonic: *leakier neurons fire less frequently.*

Remark. Summary of leak factor correlations:

Property	**As ℓ increases**	**Formula**
Current signal contribution	↑ (present input dominates)	Weight: $1 - \ell$
Spike emission difficulty	↑ (harder to fire)	$C_{\min} = \lvert\lambda_d\rvert + 1$
Expected spike count	↓ (fires less often)	$\mathbb{E}[\text{spikes}] \approx \frac{N(C_{\text{in}} - \lvert\lambda_d\rvert)}{T_d}$

Table 3: Summary of leak factor ℓ correlations with model behavior

5. Threshold Feasibility Analysis

5.1. 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.

💡 **Intuition:** Feasibility asks: "Can this neuron ever fire?" If the leak is too strong relative to the available excitation, the potential can never build up enough to reach threshold — the neuron is permanently silent.

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

$$\sum_{w_i^d > 0} w_i^d > |\lambda_d|$$

And specifically, for reliable firing:

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

Proof.

Step 1 (Define maximum excitation)**:** Let $E = \sum_{w_i^d > 0} w_i^d$ be the maximum possible excitatory contribution per step (achieved when all excitatory presynaptic neurons fire simultaneously).

Step 2 (Single-step case)**:** If $E \geq T_d$, the neuron can fire in a single step by receiving all excitatory inputs simultaneously. Feasibility is trivially satisfied.

Step 3 (Accumulation case)**:** If $E < T_d$, the neuron must accumulate potential over multiple steps. With leak factor $\lambda_d \leq 0$, each step adds a net contribution of:

$$\text{Net gain} = E + \lambda_d$$

(Note: λ_d is negative, so this is $E - |\lambda_d|$)

For accumulation to be possible, we require:

$$E + \lambda_d > 0 \quad \Rightarrow \quad E > |\lambda_d|$$

If this holds, the minimum steps to reach threshold is:

$$n = \left\lceil \frac{T_d}{E + \lambda_d} \right\rceil = \left\lceil \frac{T_d}{E - |\lambda_d|} \right\rceil$$

Step 4 (Necessity: why excitation ≤ leak implies impossibility)**:** If $E \leq |\lambda_d|$, then the net gain per step is $E - |\lambda_d| \leq 0$. The potential cannot grow — any excitation is immediately cancelled (or overpowered) by leak. The neuron can never reach threshold.

□

5.2. Implementation

The feasibility check should be performed at PRISM generation time:

```
fn check_feasibility(
    weights: &[i32],  // discretized weights
    threshold: i32,
    leak_factor: i32, // expected to be <= 0 (e.g. -1, -2)
) -> Feasibility {
    // Sum only positive excitatory weights
    let max_excitation: i32 = weights.iter()
        .filter(|&&w| w > 0)
        .sum();

    // Ensure we are working with the magnitude of the leak
    let leak_magnitude = leak_factor.abs();

    // Safety Check (Soundness): Input must overcome leak
    if max_excitation <= leak_magnitude {
        return Feasibility::Impossible;
    }

    let min_required = threshold / (1 + leak_magnitude);

    if max_excitation >= threshold {
        Feasibility::SingleStep
    } else if max_excitation >= min_required {
        // Steps = ⌈Threshold / Net_Gain⌉
        let net_gain = max_excitation - leak_magnitude;
        let steps = (threshold + net_gain - 1) / net_gain;
        Feasibility::MultiStep { min_steps: steps }
    } else {
        Feasibility::Impossible
    }
}
```

5.3. Biological Property Preservation

We verify the quotient model against standard Leaky Integrate-and-Fire properties defined in the literature [2]. Since the model uses probabilistic firing thresholds, all properties are stated in terms of *spike emission probability*.

Definition. The following biological properties should be preserved by the discretized model:

Tonic Spiking: Under constant input C_{in}, the neuron has a *non-zero probability* of emitting a spike at each step if and only if:

$$C_{\text{in}} > |\lambda_d|$$

When this condition holds, the potential accumulates to a level $L > 0$ in the threshold discretization, yielding a firing probability $P_{\text{fire}} = \frac{\text{thresholds}[L]}{100} > 0$ at each step where the potential exceeds the first threshold level. The *expected* inter-spike interval gives rise to periodic probabilistic spiking.

Integrator: The *probability of immediate firing* on simultaneous inputs n reaches 1.0 (deterministic) if and only if:

$$\sum_{i=1}^{n} \delta_W(w_i) \geq T_d$$

Below this threshold, the firing probability is determined by the discrete threshold level mapping: $P_{\text{fire}} = \frac{\text{thresholds}[L]}{100}$ where $L = \min\left(N-1, \max\left(0, \left\lfloor \frac{P}{T_d} \cdot N \right\rfloor\right)\right)$ and N is the number of threshold levels. This creates a *graded probabilistic response* rather than a binary all-or-nothing firing.

Excitability: The *expected* inter-spike interval (ISI) decreases monotonically as input frequency increases. Specifically, the expected number of steps between spikes is:

$$\mathbb{E}[N_{\text{steps}}] = \left\lceil \frac{T_d}{C_{\text{in}} - |\lambda_d|} \right\rceil$$

which decreases as C_{in} increases. Individual spike times are stochastic — drawn from the threshold-level probability distribution at each step — but the *expected rate* increases monotonically with input strength.

💡 **Intuition:** These properties ensure our discretized model behaves like a real LIF neuron, with all spike emissions governed by probability:

- *Tonic spiking* means sustained input produces a sustained *probability* of output — the neuron fires at a stochastic rate
- *Integrator* means accumulated input raises the firing *probability* continuously, reaching certainty ($P = 1.0$) only at full threshold
- *Excitability* means stronger input increases the *expected* firing rate monotonically

Remark. Immersion Memory: After emitting a spike, the potential resets to 0. At $P = 0$, the firing probability is exactly 0 (level $L = 0$), so the neuron has zero probability of spiking again until new input arrives. The neuron has no memory of past input signals — only the accumulated potential matters. This is preserved by our model since we explicitly reset $P = 0$ after firing.

6. Variable Dependencies

This section clarifies the relationships between the key variables in our discretization scheme.

6.1. Dependency Graph

```
┌─────────────────────────────────────────────────────────────────┐
│                    Variable Dependency Graph                     │
├─────────────────────────────────────────────────────────────────┤
│                                                                 │
│    ┌───────────┐                                                │
│    │  w_max    │ (Original max weight, typically 100)           │
│    └─────┬─────┘                                                │
│          │                                                      │
│          ▼                                                      │
│    ┌───────────┐      ┌───────────┐                             │
│    │     W     │◄─────│     m     │ (Fan-in constraint: W ≥ m/2)│
│    └─────┬─────┘      └───────────┘                             │
│          │                                                      │
│          ├──────────────────┐                                   │
│          ▼                  ▼                                   │
│    ┌───────────┐      ┌───────────┐                             │
│    │  δ_W(w)   │      │    T_d    │ (Discretized threshold)     │
│    └───────────┘      └─────┬─────┘                             │
│                             │                                   │
│          ┌──────────────────┤                                   │
│          ▼                  ▼                                   │
│    ┌───────────┐      ┌───────────┐                             │
│    │     ℓ     │─────►│    λ_d    │ (Leak depends on T_d & ℓ)  │
│    └───────────┘      └───────────┘                             │
│    (Leak factor)            │                                   │
│                             ▼                                   │
│                       ┌────────────┐                            │
│                       │ Feasibility│                            │
│                       └────────────┘                            │
│                                                                 │
└─────────────────────────────────────────────────────────────────┘
```

Listing 1: Variable dependencies in the discretization scheme. The crucial link is λ_d depending on T_d and the leak factor ℓ, rather than on the number of classes k.

6.2. Summary Table

Variable	Previous Definition	Revised Definition (v5)
λ_d	$-\operatorname{round}(\ell \cdot k)$ (Depends on classes k)	$-\max(1, \lfloor \ell \cdot T_d \rfloor)$ (Depends on threshold T_d)
T_d	$\left\lceil T \cdot \frac{W}{w_{\max}} \right\rceil$	Unchanged
k	Independent parameter	Remains for class discretization, but λ_d now independent of k
ℓ	Not used (retention rate r used instead)	Primary leak variable: $\ell = 1 - r$

Table 4: Summary of variable definition changes between revisions

7. PRISM Model Structure

The weighted quotient model generates PRISM code with explicit weight constants and contribution formulas. Note that weights are static constants, minimizing state explosion.

```
// Threshold-Dependent Leak Calculation
const int T_d = ceil(T_orig * W / w_max);
const int leak_percent = 10;  // Leak factor as percentage (ℓ = 0.10)
const int lambda_d = -max(1, floor(leak_percent * T_d / 100));

// 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

// 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));

module Neuron
    p : [0..T_d + 5] init 0;
    spike : bool init false;

    // Update with Safety Clamp (max(0, ...))
    [tick] true ->
        (p' = max(0, min(T_d + 5, p + contrib_2 + lambda_d)))
        & (spike' = (p' >= T_d));
endmodule
```

8. Visualizations

The following four diagrams demonstrate model dynamics across two key parameter axes: *threshold* (low vs. high) and *leak factor* (low vs. high). Weights are held constant across all diagrams to isolate the effects of threshold and leak on neuron behavior.

Common setup for all diagrams:

- **Topology:** 1 input neuron → 1 output neuron (single excitatory synapse)
- **Weight:** $w = 100$ (original), discretized with $W = 3$ giving $\delta_3(100) = 3$
- **Input pattern:** AlwaysOn (constant input at every time step)
- **Threshold levels:** $N = 4$ (probabilities: $0.25, 0.5, 0.75, 1.0$)
- **Refractory periods:** Disabled (ARP = off, RRP = off) to focus on core dynamics
- **Simulation duration:** 50 ms
- **Seed:** Fixed (e.g., 42) for deterministic reproduction

📊 **Diagram Placeholder: Low Threshold, Low Leak**

Parameters:

- $T_d = 3$ (discretized threshold), CogSpike `p_rth = 3`
- $\ell = 0.1$ (10% leak), CogSpike `leak_r = 90`
- $\lambda_d = -\max(1, \lfloor 0.1 \cdot 3 \rfloor) = -\max(1, 0) = -1$

- **Net gain per step:** $3 - 1 = 2$ (strong positive)

CogSpike `ModelConfig`: `threshold_levels = 4`, `p_rth = 3`, `leak_r = 90`, `enable_arp = false`, `enable_rrp = false`

Expected observations:
- With input weight 3 and threshold 3, the neuron reaches full threshold ($L = N - 1$) in a single step
- Firing probability = 1.0 at each step → deterministic firing every step after reset
- Raster plot should show near-continuous spiking
- Membrane potential trace: rapid sawtooth — rises to threshold immediately, resets, repeats

What this proves: With low threshold and low leak, the model exhibits maximum excitability. The single-step firing condition ($E \geq T_d$) is satisfied, confirming the Feasibility Criterion (§5.1) for the SingleStep case.

📊 **Diagram Placeholder: Low Threshold, High Leak**

Parameters:
- $T_d = 3$ (discretized threshold), CogSpike `p_rth = 3`
- $\ell = 0.5$ (50% leak), CogSpike `leak_r = 50`
- $\lambda_d = -\max(1, \lfloor 0.5 \cdot 3 \rfloor) = -\max(1, 1) = -1$
- **Net gain per step:** $3 - 1 = 2$ (still positive)

CogSpike `ModelConfig`: `threshold_levels = 4`, `p_rth = 3`, `leak_r = 50`, `enable_arp = false`, `enable_rrp = false`

Expected observations:
- Still fires in a single step (net gain $= 2 > 0$, and $E = 3 \geq T_d = 3$)
- **Difference from Diagram 1:** Despite higher leak, the low threshold means single-step firing still dominates. The leak factor is noticeable in the slightly lower resting potential between spikes, but firing behavior is similar because $E \geq T_d$.
- This demonstrates that for low-threshold neurons, leak primarily affects *accumulation-regime* behavior rather than single-step firing.

What this proves: When the excitatory input meets or exceeds the threshold in a single step, the leak factor has minimal impact on firing frequency — it only affects the multi-step accumulation regime.

📊 **Diagram Placeholder: High Threshold, Low Leak**

Parameters:
- $T_d = 6$ (discretized threshold), CogSpike `p_rth = 6`
- $\ell = 0.1$ (10% leak), CogSpike `leak_r = 90`
- $\lambda_d = -\max(1, \lfloor 0.1 \cdot 6 \rfloor) = -\max(1, 0) = -1$
- **Net gain per step:** $3 - 1 = 2$ (positive)

- **Expected steps to threshold:** $\lceil \frac{6}{2} \rceil = 3$ steps

CogSpike `ModelConfig`: `threshold_levels = 4`, `p_rth = 6`, `leak_r = 90`, `enable_arp = false`, `enable_rrp = false`

Expected observations:
- Neuron enters the *accumulation regime* — cannot fire in a single step ($E = 3 < T_d = 6$)
- Potential builds over 3 steps: $0 \to 2 \to 4 \to 6$ (reaching threshold at step 3)
- During accumulation, the neuron passes through threshold levels, gaining increasing firing *probability* at each step (e.g., $P = 0.25$ at level 1, $P = 0.5$ at level 2, etc.)
- Raster plot: periodic spiking with ISI ≈ 3 steps
- Membrane potential trace: staircase pattern rising to threshold

What this proves: Demonstrates the Excitability property (§5.3): the neuron integrates input over multiple steps, with the ISI determined by $\lceil \frac{T_d}{C_{\text{in}} - |\lambda_d|} \rceil$. Also validates the probabilistic graded response during accumulation.

📊 **Diagram Placeholder: High Threshold, High Leak**

Parameters:
- $T_d = 6$ (discretized threshold), CogSpike `p_rth = 6`
- $\ell = 0.5$ (50% leak), CogSpike `leak_r = 50`
- $\lambda_d = -\max(1, \lfloor 0.5 \cdot 6 \rfloor) = -\max(1, 3) = -3$
- **Net gain per step:** $3 - 3 = 0$ (zero!)

CogSpike `ModelConfig`: `threshold_levels = 4`, `p_rth = 6`, `leak_r = 50`, `enable_arp = false`, `enable_rrp = false`

Expected observations:
- **Critical case:** Net gain is exactly 0 — the neuron is at the *feasibility boundary*
- The potential plateaus: each step adds $+3$ (input) and -3 (leak), resulting in no accumulation
- Potential hovers at a low level, never reaching threshold
- Raster plot: silence (no spikes emitted)
- Membrane potential trace: flat line at the equilibrium point

What this proves: Demonstrates the Feasibility Criterion (§5.1): when $E \leq |\lambda_d|$ (here $E = |\lambda_d| = 3$), the neuron is threshold-infeasible. This is the exact boundary case where the Soundness Theorem (§3.4) guarantees silence. Also illustrates the Leak Factor Dynamics (§4.3): with high leak, the minimum excitation $C_{\text{min}} = |\lambda_d| + 1 = 4$ exceeds the available input of 3.

9. Conclusion

We have established a formal framework for weight discretization in quotient model abstraction. The key contributions in this revision include:

1. **Threshold-Dependent Leak:** The formulation $\lambda_d = -\max(1, \lfloor \ell \cdot T_d \rfloor)$ using the leak factor ℓ ensures leak scales with the actual potential range, not the arbitrary number of classes.

2. **Soundness Theorem:** Complete proof that no spurious spikes occur — including the guarantee that zero potential maps to zero firing probability, providing a full safety chain from sub-threshold potential through decay to permanent silence.
3. **Biological Property Preservation:** Probabilistic verification that tonic spiking, integrator behavior, and excitability are maintained in the quotient model, with all spike emission expressed in terms of firing probabilities determined by threshold level mapping.
4. **Leak Factor Dynamics:** Analysis of the three key correlations: how the leak factor ℓ controls current signal contribution, spike emission difficulty (minimum excitation C_{min}), and expected spike frequency.
5. **Variable Dependency Clarification:** Clear documentation of how λ_d depends on T_d and ℓ, breaking the previous incorrect dependency on k.
6. **Simulation Verification Plan:** Four precisely specified diagram configurations (two thresholds $\times$ two leak factors) to empirically validate the model dynamics using CogSpike's simulation engine.

Future Work:

- Generate the four simulation diagrams via CogSpike CLI export mode
- Formal PRISM model checking with the revised leak formulation
- Extend to multi-neuron networks with inter-neuron weight calibration

---

Bibliography

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

[2] E. D. Maria, C. D. Giusto, and L. Laversa, "Spiking Neural Networks Modelled as Timed Automata with Parameter Learning," *Natural Computing*, vol. 19, pp. 135–155, 2020.