Knowledge State and Decision Models

Knowledge State and Decision Models

The judgment layer

Most trigger conditions are computable in code (round/tick counts, HP fractions, allies alive, distance). A few are genuinely judgment-shaped:

- "Is this party vulnerable / outmatched?"
- "Is the player kiting?" (kiting = attacking from range while retreating to stay out of reach - a behavior pattern over time, not a single state; a hardcoded detector misfires on feints, terrain, and repositioning)
- "Is that prep a feint or a commitment?" (especially once fake-cancel feats exist - pure state can't answer this; it's about habits and context)

Typed decision models (Jev/Kev class)

The model supplies judgment; your code owns policy. Ask typed questions over a compact state snapshot; get back probabilities/distributions, not text.

- **noul** booleans for non-computable conditions (one per condition - they must not compete for probability mass).

- **choice** for posture selection; **sample from the distribution**, don't take argmax - seeded for reproducible variety. This is the "every run differs" lever.
- Cost is fractions of a cent per call (tiny state envelopes); latency is a network round trip per monster decision.
- **Provider-swappable**: hosted Jev for dev, local Kev (0.8B fits Terra's P2200; 4B fits Saturn's M2 Pro via MLX) at play time, pure-code fallback for offline. Small local models are option-order sensitive - pin option order per question pack.
- Valid output never means correct judgment - the model's answer is a suggestion the profile policy resolves, not the final word.

What stays in code

- The Ammann profile and its trigger table: always evaluated in code.
- Thresholds, weights, branches - reviewable and tunable without re-prompting.
- Outcome determination and any side effects.

Revision #2
Created 2026-09-27 16:48:30 UTC by Rowan Flora
Updated 2026-09-27 19:09:51 UTC by Rowan Flora