NGF Engineering Playbook: Scaling Across Aptitude Tests

Once intellectual primitives are identified for a given domain (ARC, HellaSwag, math, code, etc.), the NGF pipeline can be applied systematically to enforce deterministic reasoning: Adapter → Warp → Detect → Denoise → Execute → Verify.

1. Primitive Sets by Benchmark Family

ARC / Visual Reasoning

- Geometry: flip_h, flip_v, rotate_90/180/270, transpose, scale, translate
- Pattern ops: flood_fill, draw_line, copy_patch, repeat_tile, grow/shrink
- Symbolic: color_remap, count, argmax-color, shape_match

HellaSwag / Commonsense MC

- Discourse: temporal_continuity, causal_link, goal_satisfaction, physical_affordance
- Semantics: entail, contradict, exclude_alt, plausibility_filter

GSM8K / Math Word Problems

- Arithmetic: add, subtract, multiply, divide
- Structure: group_terms, carry_borrow, unit_convert, round, simplify_fraction
- Control: choose_operation, check_consistency, estimate_bound

Code (LeetCode-style)

- Control flow: branch_if, loop_iter, recursion_call/return
- Data ops: index, slice, push/pop, sort, hash_lookup
- Reasoning: invariant_check, complexity_choice, off_by_one_guard

Raven's / Abstract Matrices

- Transforms: add_shape, remove_shape, move, rotate, mirror
- Rules: progression, XOR/OR/AND_shape, count_parity, texture_change

MMLU / Factual+Logic

- Logic: entail, contradict, analogize, define_term, rule_apply
- Heuristics: eliminate_implausible, prior_strengthen, exception_detect

2. Repeatable NGF Pipeline

- Adapter: convert raw input → latent traces per primitive (or per choice).
- Warp: PCA/whiten → funnel fit to enforce a single dominant basin.
- Detect: matched filter + null-calibrated thresholds to pick active primitives.

- Denoise: inhibition, phantom-guard, jitter averaging for stability.
- Execute: apply detected primitives/ops in order.
- Verify: confirm across examples, abstain if gates fail.

3. Registry Schema Example

{id: "rotate_90", arity: 1, domain: ["ARC","RAVEN"], proto: "latent_direction_or_kernel_id", detector: "adapter_fn_name", executor: "apply_on_input_fn", compose: [{"after":"flip_h","equiv":["flip_h","rotate_270"]}], aliases: ["turn_right", "quarter_turn"]}

4. Build Plan

- Seed the registry: ~15–25 primitives per domain, reuse across domains where possible.
- Write thin adapters: ARC/Raven (grid ops), HellaSwag (choices), GSM8K (math steps), Code (AST control ops).
- Calibrate funnels + nulls once per family, reuse for batch evaluation.
- Benchmark: accuracy + NGF-native metrics (hallucination %, omission %, phantom index, margin).
- Always enable abstain mode for uncertain cases.

5. Risks & Mitigations

- Primitive coverage gaps → iterative registry expansion with logged unknown ops.
- Combinatorial blowup → operator algebra (compose rules) + NGF ordering.
- Adapter bias → rely on null calibration + abstain instead of forcing guesses.
- Cross-domain drift → keep per-domain funnel fits, but reuse shared primitives where valid.

6. Outcome

A unified reasoning core (NGF) with a growing operator registry. Adapters per benchmark family. Deterministic, hallucination-resistant reasoning by design once primitives are in place.