SGRM for H-hat: Types, Typing Rules, IR, and Semantics

0. Aim (H-hat view)

We propose *Synergy-Guided Refraction Map (SGRM)* as a small set of **types** and **instructions** expressible in H-hat's Heather dialect, lowering to IR that prunes/elides gates based on pilot measurements. Everything below respects H-hat's rule: *quantum roots may contain classical, classical may not contain quantum.*

Core idea: run a pilot; compute simple local interaction labels $\ell \in \{-1, 0, +1\}$ (**beta**, **nil**, **alpha**); compile a *refraction mask* $M \in \{0,1\}^D$ over gate positions; *reuse* M to prune small rotations and skip entanglers across "beta" edges. Mask learning is classical; execution is quantum at cast-sites.

1. Types (H-hat type table entries)

$$
\begin{aligned}
\textsf{QLabel} \; &::= \; \{\alpha,\ 0,\ \beta\} \quad \text{(ternary edge/node label)}\\
\textsf{RefractionPolicy} \; &::= \; \{k_\alpha : f32,\ k_0 : f32,\ k_\beta : f32,\ s_{\mathrm{cx}|\beta} : f32,\ \theta_{\mathrm{small}} : f32\}\\
\textsf{RefractionMask} \; &::= \; \{0,1\}^D\\
\textsf{Samples} \; &::= \; \text{multiset of } z \in \{0,1\}^n\\
\textsf{Metrics} \; &::= \; \{\text{tv\_k} : f32,\ \text{l1\_1q} : f32,\ \text{l1\_2q} : f32,\ \ldots\}\\
\textsf{@Circuit},\ \textsf{@State} \; &\text{(quantum-rooted)} \qquad \textsf{CircuitOps} \; ::= \; \langle G_1, \ldots, G_D \rangle
\end{aligned}
$$

2. Behavioral interfaces (instruction signatures)

We separate quantum ($\textsf{QInstr}$) from classical ($\textsf{CInstr}$) instructions.

$\textsf{pilot\_run} : (\textsf{@Circuit},\ \text{shots} : u64,\ \rho : \text{Rounding}) \to \textsf{Samples}$	QInstr
$\textsf{score\_edges} : (\textsf{Samples},\ E,\ \mathcal{M} : \text{Metric}) \to \{g_e \in \mathbb{R}\}_{e \in E}$	CInstr
$\textsf{label\_edges} : (\{g_e\},\ \tau_+,\ \tau_-) \to \{\ell_e \in \{-1, 0, +1\}\}_{e \in E}$	CInstr
$\textsf{build\_policy} : (\{\ell_e\},\ \text{params}) \to \textsf{RefractionPolicy}$	CInstr
$\textsf{compile\_mask} : (\textsf{CircuitOps},\ \textsf{RefractionPolicy},\ \text{seed} : u64) \to \textsf{RefractionMask}$	CInstr
$\textsf{execute\_masked} : (\textsf{@Circuit},\ \textsf{RefractionMask},\ \rho,\ \text{coherence?}) \to \textsf{Samples}$	QInstr
$\textsf{compare\_ref} : (\textsf{Samples}_{\text{ref}},\ \textsf{Samples}) \to \textsf{Metrics}$	CInstr

Metrics: e.g. Pearson, MI, HSIC. The interface is pluggable.

3. Typing judgements

We write $\Gamma \vdash e : \tau$ for typing under environment Γ. Classical types τ_c, quantum types $@\tau\_c$ (quantum-rooted).

Cast rule (H-hat style).

$$\frac{\Gamma \vdash e : @\tau\_c}{\Gamma \vdash \mathsf{cast}(e) : \tau\_c \times \mathsf{Meta}} \quad \text{(quantum execution occurs at cast)}$$

SGRM instruction typings.

$$\frac{\Gamma \vdash C : @\mathsf{Circuit}}{\Gamma \vdash \mathsf{pilot\_run}(C, \mathrm{shots}, \rho) : \mathsf{Samples}} \qquad \frac{\Gamma \vdash S : \mathsf{Samples}}{\Gamma \vdash \mathsf{score\_edges}(S, E, \mathcal{M}) : \mathbb{R}^{|E|}}$$

$$\frac{\Gamma \vdash g : \mathbb{R}^{|E|}}{\Gamma \vdash \mathsf{label\_edges}(g, \tau_+, \tau_-) : \mathsf{QLabel}^{|E|}} \qquad \frac{\Gamma \vdash \ell : \mathsf{QLabel}^{|E|}}{\Gamma \vdash \mathsf{build\_policy}(\ell, \_) : \mathsf{RefractionPolicy}}$$

$$\frac{\Gamma \vdash \langle G_t \rangle : \mathsf{CircuitOps} \quad \Gamma \vdash P : \mathsf{RefractionPolicy}}{\Gamma \vdash \mathsf{compile\_mask}(\langle G_t \rangle, P, \mathrm{seed}) : \mathsf{RefractionMask}}$$

$$\frac{\Gamma \vdash C : @\mathsf{Circuit} \quad \Gamma \vdash M : \mathsf{RefractionMask}}{\Gamma \vdash \mathsf{execute\_masked}(C, M, \rho, coh?) : \mathsf{Samples}}$$

These preserve the classical/quantum barrier: all SGRM artifacts (Samples, labels, policy, mask, metrics) are classical.

4. Local interaction scoring and labeling

Given pilot samples $S = \{z^{(s)}\}_{s=1}^{S}$ with $z^{(s)} \in \{0,1\}^n$ and graph E, a generic score per edge $e = (i,j)$:

$$g_e \;=\; \mathcal{M}(S; e), \quad \text{e.g. } r_{ij} = \frac{\mathrm{cov}(Z_i, Z_j)}{\sqrt{\mathrm{var}(Z_i)\mathrm{var}(Z_j)}}$$

Ternary labels (alpha/nil/beta) via thresholds $\tau_+ > \tau_- > 0$:

$$\ell_e = \begin{cases} +1 & g_e \geq \tau_+ \\ 0 & \tau_- < g_e < \tau_+ \\ -1 & g_e \leq \tau_- \end{cases} \qquad \ell_e \in \{+1, 0, -1\}.$$

Optional node label by signed vote $\tilde{\ell}_q = \mathrm{sign}\left(\sum_{e \ni q} \ell_e\right)$.

5. Refraction policy and mask semantics

Policy. Keep-scales for small rotations and CX skip on beta:

$$\kappa(+1) = k_\alpha, \quad \kappa(0) = k_0, \quad \kappa(-1) = k_\beta, \qquad s_{\mathrm{cx}|\beta} \in [0,1],$$

with angle threshold θ_{small}.

Deterministic Bernoulli. Let $\mathrm{DetBern}(p; \mathrm{seed}, t, \ldots) \in \{0, 1\}$ be a seeded hash comparison so mask compilation is reproducible.

Mask rule over gate index t: Gate G_t acts on Q_t with parameter θ_t.

$$m_t = \begin{cases} 1, & G_t \in \{\mathrm{H}\} \\ 1, & G_t = R_{x/y}(\theta_t) \ \wedge \ |\theta_t| \geq \theta_{\mathrm{small}} \\ \mathrm{DetBern}(\kappa(\tilde{\ell}_q)), & G_t = R_{x/y}(\theta_t) \text{ on } q, \ |\theta_t| < \theta_{\mathrm{small}} \\ \mathrm{DetBern}(1 - s_{\mathrm{cx}|\beta}), & G_t = \mathrm{CX}(c \to t), \ \tilde{\ell}_c = -1 \ \vee \ \tilde{\ell}_t = -1 \\ 1, & \text{otherwise.} \end{cases}$$

Refraction operator $\mathcal{R}_M$ compiles $U' = \mathcal{R}_M(\langle G_t \rangle) = \prod_{t: m_t = 1} G_t$.

6. IR hooks (Heather → IR)

Heather-level calls lower to IR with flags (CALL, DECLARE_ASSIGN, etc.). An SGRM block typically lowers as:

CALL pilot_run	$\to$ Samples
CALL score_edges	$\to \mathbb{R}^{\|E\|}$
CALL label_edges	$\to$ $\mathsf{QLabel}^{\|E\|}$
CALL build_policy	$\to$ RefractionPolicy
CALL compile_mask	$\to$ RefractionMask
CALL execute_masked	$\to$ Samples
CALL compare_ref	$\to$ Metrics

No new control-flow forms are required; *SGRM is a library-level extension* expressed with standard calls and assignments, making it dialect-friendly.

7. Fair attribution protocol (configurations)

On the same op list and seed:

$$\begin{array}{ll} \textbf{REF}: & \rho = \rho_{\mathrm{tight}}, \ \text{no refraction}, \ \text{coherence} = 0 \Rightarrow \mathsf{Samples}_{\mathrm{ref}} \\ \textbf{BASE}: & \rho = \rho_{\mathrm{fast}}, \ \text{no refraction}, \ \text{coherence} = 0 \Rightarrow \mathsf{Samples}_{\mathrm{base}} \\ \textbf{BASE+R}: & \rho = \rho_{\mathrm{fast}}, \ \underline{M \text{ from REF}}, \ \text{coherence} = 0 \Rightarrow \mathsf{Samples}_{\mathrm{br}} \\ \textbf{MOD}: & \rho = \rho_{\mathrm{fast}}, \ \underline{M \text{ from REF}}, \ \text{coherence} = 1 \Rightarrow \mathsf{Samples}_{\mathrm{mod}} \end{array}$$

Keeping M fixed isolates the effect of the coherence toggle: $\Delta_{\mathrm{coh}} = \mathsf{Samples}_{\mathrm{mod}} - \mathsf{Samples}_{\mathrm{br}}$.

8. Comparison metrics (library-level)

Top-K TV (heavy-state focus): let $S_K(p)$ be the K heaviest under p.

$$\mathrm{TV}_K(p, q) = \frac{1}{2} \sum_{z \in S_K(p)} |p(z) - q(z)| + \frac{1}{2} \left| \big(1 - \sum_{z \in S_K(p)} p(z)\big) - \big(1 - \sum_{z \in S_K(p)} q(z)\big) \right|.$$

One-qubit L^1: $L^1_{1q}(p,q) = \sum_{i=0}^{n-1} |\Pr_p(Z_i{=}1) - \Pr_q(Z_i{=}1)|$. Two-qubit L^1: average over a pair set $\mathcal{P}$ of half-L^1 on joint marginals.

9. Determinism & safety

Mask compilation uses DetBern($\cdot$) keyed by (seed, gate index, op hash); thus reproducible and IR-stable. All SGRM outputs are classical, so H-hat's quantum/classical containment rule is preserved by construction.

10. Example (Heather-ish surface form; lowers to calls)

```
let ref = pilot_run(@circ, shots=10000, rounding="tight");
let g = score_edges(ref, E="ring", metric="pearson");
let labels = label_edges(g, tau_pos=+0.005, tau_neg=-0.005);
let policy = build_policy(labels, keep_alpha=1.0, keep_0=0.0, keep_beta=0.0, skip_cx_beta=1.0,
theta_small=0.13);
let mask = compile_mask(@circ.ops, policy, seed=0xA5A5_1234);
let baseR = execute_masked(@circ, mask, rounding="fast", coherence=false);
let mod = execute_masked(@circ, mask, rounding="fast", coherence=true);
let m_baseR = compare_ref(ref, baseR);
let m_mod = compare_ref(ref, mod);
```

11. Minimal integration plan

Front-end: add types (QLabel, RefractionPolicy, RefractionMask); declare functions with above signatures in the Heather prelude.
Lowering: map calls to IR nodes with flags CALL/DECLARE_ASSIGN.
Backend: implement evaluators: pilot_run, execute_masked trigger quantum execution at cast-sites; the rest are pure classical transforms.