Chapter 12

Energy, Cost, and Entropy Accounting in QFM

Deterministic Resource Semantics for Distributed QVM Execution

Honza Rožek

Abstract

This document specifies an energy, cost, and entropy accounting framework for the Quantum Virtual Machine (QVM) operating over Quansistor Field Mathematics (QFM). Resource consumption is defined as a semantic property of operator application and state evolution rather than as a runtime- or hardware-dependent measurement.

Unlike classical cost models based on execution time, instruction counts, or gas units, the framework assigns deterministic energy values to operator application, explicit non-additive cost to interference, and irreversible work to collapse operations. These quantities are invariant under distribution, replayable by construction, and independent of physical deployment topology.

The accounting model introduces entropy as a semantic measure of dispersion of admissible future state evolutions and enforces execution through declared entropy budgets. By bounding semantic dispersion, the framework provides stability constraints for distributed execution and enables predictable, regulator-grade resource accountability.

Within the QFC stack, this chapter defines the canonical semantic resource accounting layer underlying optimization, governance, and distributed execution. It enables predictable pricing, incentive-aligned system design, and legally meaningful auditability of computational cost without reliance on physical measurement, probabilistic estimation, or runtime heuristics.

1 Motivation: Why Classical Cost Models Fail

1.1 The Cost Blindness of Classical Computation

Classical computation has historically treated cost as an external concern. Execution time, instruction counts, memory usage, or hardware utilization are measured empirically and interpreted after execution. Such measurements are inherently dependent on hardware architecture, runtime optimization, and deployment conditions.

As computation becomes distributed, virtualized, and increasingly decoupled from physical hardware, these cost models lose explanatory power. Identical programs may exhibit radically different performance profiles depending on scheduling, load, or infrastructure, despite being semantically identical.

This creates a fundamental disconnect between what a computation means and what it costs.

1.2 Gas Models and Their Limitations

Blockchain systems introduced gas models in an attempt to internalize computation cost. While gas provides predictability and prevents resource exhaustion, it remains a proxy measure tied to instruction counts or execution steps.

Gas models do not account for semantic complexity. Two computations with identical gas cost may differ significantly in structural complexity, interference, or irreversibility. Conversely, computations with similar semantic impact may incur widely different gas costs due to implementation artifacts.

As a result, gas pricing incentivizes optimization toward execution patterns rather than toward semantic efficiency.

1.3 FLOPs, Benchmarks, and the Illusion of Objectivity

Metrics such as FLOPs, throughput, or benchmark scores attempt to quantify computational power by measuring physical operations. These metrics assume that cost is proportional to the number of low-level operations performed.

In distributed and operator-based systems, this assumption breaks down. The same semantic operation may be realized through different microarchitectural paths without changing its meaning. Counting physical operations therefore measures implementation, not computation.

Such metrics are unsuitable for systems that require reproducibility, legal accountability, or economic predictability.

1.4 Absence of Semantic Cost

None of the classical cost models assign cost to meaningful operations. They do not distinguish between:

- reversible and irreversible transformations;
- independent and interfering operations;
- local updates and globally constraining collapses.

As a result, they cannot express the true structural cost of a computation, nor can they explain why certain operations should be considered intrinsically more expensive than others.

1.5 Why Distributed Systems Amplify the Problem

In distributed systems, cost ambiguity is amplified. Runtime, latency, and resource usage depend on network conditions, replication strategies, and fault tolerance mechanisms.

Two executions with identical semantic structure may incur different physical costs, while two executions with different semantic impact may appear similar under runtime metrics.

This ambiguity makes it impossible to reason about cost in a way that is deterministic, replayable, or legally defensible.

1.6 Need for Semantic Energy and Entropy Accounting

To support predictable cost, fair pricing, and accountable computation, cost must be elevated from an observational metric to an architectural property.

This requires:

- assigning energy to operator semantics rather than execution time;
- accounting for interference and superposition explicitly;
- treating irreversible operations as intrinsic work;
- bounding system evolution through entropy budgets.

Cost must be something the system defines, not something it measures after the fact.

1.7 Design Position of This Work

This work introduces an energy, cost, and entropy accounting framework for QVM–QFM in which resource consumption is derived from semantic structure rather than physical execution.

Energy is associated with operator application. Interference carries explicit cost. Collapse represents irreversible work. Entropy budgets bound system evolution.

By defining cost semantically, QVM–QFM enables deterministic, replayable, and regulator-grade resource accounting for distributed computation.

2 Energy as an Operator Property in QFM

2.1 From Runtime Cost to Semantic Energy

In QVM–QFM, energy is not a measurement of physical resource consumption, execution time, or hardware utilization. Instead, energy is defined as an intrinsic semantic property of operator application within the QFM state field.

This distinction is fundamental. Whereas runtime cost depends on how an operation is executed, semantic energy depends only on what the operation means: the structure, scope, and irreversibility of the state transformation it induces.

Two executions that differ in physical realization but apply the same operator to the same state incur identical semantic energy cost.

2.2 Operator-Induced State Transformation

Let O denote a QFM operator acting on an architectural state Ψ. Operator application induces a state transition

$$\Psi' = O(\Psi). \tag{12.1}$$

The energy cost $E(O)$ of the operator is defined independently of execution strategy and reflects the semantic complexity of this transformation.

Energy is therefore associated with the mapping $\Psi \to \Psi'$, not with intermediate computation artifacts.

2.3 Deterministic Energy Assignment

Each operator class in QVM–QFM is assigned a deterministic energy function

$$E : O \mapsto \mathbb{R}_{\geq 0}, \tag{12.2}$$

defined by architectural specification rather than empirical measurement.

The energy function may depend on:

- the type and dimensionality of affected state components;
- the degree of coupling introduced between components;
- declared precision, normalization, or stabilization parameters;
- whether the transformation is reversible or irreversible.

Energy assignment may not depend on execution time, hardware capabilities, or runtime optimization.

2.4 Scope-Dependent Energy

Operator energy may scale with the scope of application. An operator applied to a larger or more complex subset of the state field may incur higher energy than the same operator applied to a smaller scope.

Such scaling rules are explicitly declared as part of operator semantics. Scope-dependent energy remains deterministic and replayable, as scope resolution itself is deterministic.

2.5 Additivity and Composition

For a sequence of operators $O_1, O_2, \ldots, O_n$, the total execution energy is defined as

$$E_{\text{total}} = \sum_{i=1}^{n} E(O_i), \tag{12.3}$$

unless modified by explicitly defined interaction terms.

Energy additivity provides a simple and predictable accounting model. Non-additive behavior, such as interference effects, must be declared explicitly and is addressed in subsequent sections.

2.6 Energy Invariance Under Distribution

Because energy is defined semantically, it is invariant under distribution. An operator applied locally or across multiple canisters incurs the same energy cost provided its semantic effect is identical.

This invariance ensures that energy accounting remains stable across deployment topologies and supports deterministic replay and independent verification.

2.7 Energy Visibility and Auditability

All operator energy costs are architecturally visible. Energy consumption is reflected explicitly in execution accounting and may be recorded as part of the replay trace.

No hidden or implicit energy expenditure is permitted. If an operation has semantic effect, it must have declared energy cost.

2.8 Relationship to Economic and Regulatory Use

By defining energy as an operator property, QVM–QFM enables:

- predictable pricing independent of infrastructure;
- fair comparison of computational workloads;
- regulator-grade explanation of resource usage;
- separation of economic cost from hardware performance.

This semantic notion of energy forms the foundation for cost, interference, and entropy accounting introduced in subsequent sections.

2.9 Summary

Energy in QVM–QFM is a deterministic, semantic attribute of operator application. By divorcing energy from runtime behavior and binding it to state transformation meaning, the framework establishes a stable and verifiable basis for resource accounting in distributed computation.

3 Interference Cost and Superposition Accounting

3.1 Interference as a Source of Non-Additive Cost

In QVM–QFM, not all computational cost is additive. While independent operator applications contribute linearly to total energy, operations that introduce coupling, overlap, or dependency between state components generate additional structural cost.

This additional cost is referred to as interference cost. Interference arises whenever multiple operators or state components cannot be treated as independent under semantic composition.

Interference is not an implementation artifact; it is a property of meaningful interaction within the QFM state field.

3.2 Superposition and Coupling of State Components

Let Ψ denote the architectural state field and let $\{\psi_i\} \subset \Psi$ be a collection of state components. A superposition exists when the semantic effect on one component depends on the state of another.

Operators that act on overlapping or mutually dependent components induce coupling. Such coupling increases semantic complexity and therefore incurs interference cost.

This notion of superposition is abstract and does not require quantum hardware or probabilistic interpretation. It refers strictly to semantic interdependence.

3.3 Definition of Interference Cost

For a set of operators $O_1, O_2, \ldots, O_n$, the interference cost I is defined as the excess energy beyond simple additivity:

$$I = E(O_1 \circ O_2 \circ \cdots \circ O_n) - \sum_{i=1}^{n} E(O_i). \tag{12.4}$$

If operators commute semantically and act on disjoint state scopes, then $I = 0$. If their composition introduces coupling or dependency, then $I > 0$.

3.4 Deterministic Interference Accounting

Interference cost is computed deterministically based on declared operator semantics and resolved state scope.

Interference accounting may consider:

- degree of overlap between operator scopes;
- introduction of shared constraints or normalization;
- reduction of independent degrees of freedom;
- coupling strength declared by operator semantics.

No interference cost may depend on execution timing, scheduling, or runtime conditions.

3.5 Interference Across Distributed Execution

Interference is invariant under distribution. Operators executed on separate canisters may still interfere if they introduce semantic coupling in the global state field.

Conversely, physically concurrent execution does not imply interference if operators are semantically independent.

This property ensures that interference cost reflects meaning rather than deployment.

3.6 Pay-Per-Interference Principle

Because interference represents structural complexity rather than execution effort, QVM–QFM enables a pay-per-interference model.

Under this model:

- independent computation is inexpensive;
- highly coupled computation incurs higher cost;
- system incentives favor modular and decoupled design.

This aligns economic incentives with semantic clarity and system stability.

3.7 Interference Visibility and Replay

Interference cost is architecturally visible and replayable. Given an execution trace, a verifier can recompute interference cost exactly by reconstructing operator composition and scope overlap.

No hidden or emergent interference is permitted. All coupling must be declared or derivable from semantics.

3.8 Relationship to Collapse and Entropy

Interference cost reflects reversible coupling. When interference is resolved through irreversible operations, such as collapse, additional cost is incurred. This transition is addressed in the following section.

3.9 Summary

Interference cost captures the non-additive semantic complexity arising from coupled computation. By accounting for interference explicitly, QVM–QFM provides a deterministic and meaningful measure of structural cost beyond simple operator energy.

4 Collapse Cost and Irreversibility

4.1 Irreversibility as Semantic Work

In QVM–QFM, not all state transformations are reversible. Certain operations explicitly eliminate admissible alternative evolutions of the architectural state. Such operations perform irreversible semantic work.

Irreversibility is not defined by loss of information at the hardware level, but by the deliberate reduction of the space of admissible future states. This reduction constitutes intrinsic work and therefore incurs cost.

4.2 Definition of Collapse Operations

A collapse operation is an operator that maps a set of admissible states $\{\Psi^{(i)}\}$ to a strictly smaller set, typically a singleton:

$$\{\Psi^{(1)}, \Psi^{(2)}, \ldots\} \longrightarrow \{\Psi^*\}. \tag{12.5}$$

Collapse operations include, but are not limited to:

- conflict resolution that selects a single outcome;
- commitment decisions that finalize prior superpositions;
- pruning of speculative or forked execution branches;
- enforcement of hard constraints.

4.3 Collapse Cost

The collapse cost C associated with a collapse operation is defined as the semantic work required to eliminate alternative admissible states.

Collapse cost may depend on:

- the number of eliminated alternatives;
- the structural diversity of those alternatives;
- the depth of causal history being collapsed;
- declared collapse strength parameters.

Collapse cost is deterministic and derived solely from semantic structure.

4.4 Distinction from Interference Cost

Interference cost accounts for reversible coupling and superposition. Collapse cost accounts for irreversible resolution.

While interference increases complexity without committing to an outcome, collapse reduces complexity by destroying alternatives. The two costs are complementary and non-substitutable.

4.5 Collapse Across Distributed Execution

Collapse operations may occur across distributed execution boundaries. A globally visible collapse has the same semantic effect regardless of where it is physically executed.

Distribution does not dilute collapse cost. Eliminating alternatives in a distributed system incurs the same collapse cost as in a local system.

4.6 Replay and Auditability

Because collapse operations are explicit and deterministic, collapse cost is replayable and auditable.

A replay verifier can:

- identify when a collapse occurred;
- enumerate the eliminated alternatives;
- recompute the associated collapse cost.

No hidden or implicit collapse is permitted.

4.7 Relationship to Logical Time

Collapse operations occur at specific logical times and contribute to the global execution cost at those times.

The temporal placement of collapse is semantically significant, as it affects which alternatives remain admissible at subsequent logical steps.

4.8 Irreversibility and Responsibility

By making collapse explicit and costly, QVM–QFM establishes clear responsibility for irreversible decisions.

Any actor or process triggering a collapse can be identified and held accountable for the irreversible semantic work performed.

4.9 Summary

Collapse cost captures the intrinsic work associated with irreversible semantic decisions. By accounting for collapse explicitly, QVM–QFM enables deterministic, replayable, and legally meaningful attribution of irreversible computational actions.

5 Entropy and Budget Constraints

5.1 Semantic Entropy in QVM–QFM

In QVM–QFM, entropy is a semantic quantity representing the dispersion of admissible future states of the architectural system. Entropy does not correspond to thermodynamic entropy and does not measure physical disorder or heat.

Instead, semantic entropy quantifies the number and diversity of execution paths that remain possible given the current state and declared constraints.

5.2 Admissible State Space

Let $\mathcal{A}_t$ denote the set of admissible architectural states reachable from the current state Ψ_t under the execution semantics.

Semantic entropy S_t is a function of $\mathcal{A}_t$:

$$S_t = \mathcal{S}(\mathcal{A}_t), \tag{12.6}$$

where $\mathcal{S}$ is a deterministic measure of state-space dispersion defined by architectural specification.

5.3 Entropy Evolution

Entropy evolves as execution proceeds:

- reversible coupling and superposition tend to increase entropy by expanding $\mathcal{A}_t$;
- collapse operations reduce entropy by eliminating admissible states;
- neutral operations may leave entropy unchanged.

Entropy evolution is deterministic and replayable.

5.4 Entropy Budgets

An entropy budget is a declared upper bound

$$S_t \leq S_{\max} \tag{12.7}$$

that constrains the admissible evolution of the system.

Entropy budgets may be applied globally, per execution, per subsystem, or per policy domain. Violation of an entropy budget constitutes a semantic execution error.

5.5 Purpose of Entropy Budgets

Entropy budgets serve multiple purposes:

- preventing unbounded speculative branching;
- limiting uncontrolled superposition growth;
- enforcing predictability of execution outcomes;
- bounding future collapse cost.

Budgets transform entropy from an abstract quantity into an enforceable architectural constraint.

5.6 Interaction with Energy and Collapse

Entropy, energy, and collapse cost are interrelated but distinct:

- energy accounts for operator application;
- interference accounts for reversible coupling;
- collapse accounts for irreversible resolution;
- entropy measures remaining freedom.

Entropy budgets may force collapse when entropy growth approaches limits, thereby incurring explicit collapse cost.

5.7 Distributed Entropy Accounting

In distributed execution, entropy is accounted globally. Local execution domains do not maintain independent entropy measures.

This ensures that speculative expansion in one domain cannot silently shift entropy burden to another.

5.8 Replay and Verification

Entropy evolution and budget enforcement are replayable. A verifier can reconstruct $\mathcal{A}_t$ and recompute S_t at each logical time.

Any divergence between declared and reconstructed entropy constitutes a violation of execution semantics.

5.9 Policy and Governance Implications

Entropy budgets provide a powerful tool for policy enforcement. They enable governance decisions based on semantic complexity rather than runtime metrics.

Examples include:

- limiting speculative financial computation;
- bounding AI search spaces;
- enforcing regulatory predictability.

5.10 Summary

Semantic entropy and entropy budgets bound the admissible evolution of QVM–QFM executions. By making entropy explicit and enforceable, the framework ensures predictable, accountable, and governable computation.

6 Total Cost Composition and Accounting

6.1 Components of Total Semantic Cost

In QVM–QFM, the total cost of an execution is a semantic quantity composed of multiple structurally distinct contributions. These contributions reflect different aspects of computational meaning rather than execution effort.

The total semantic cost K_{total} is composed of:

- operator energy cost;
- interference cost from semantic coupling;
- collapse cost from irreversible decisions;
- entropy budget enforcement effects.

Each component is deterministic, explicit, and replayable.

6.2 Formal Cost Composition

Let $\{O_t\}$ denote the sequence of committed operators at logical times t. The total cost is defined as

$$K_{\text{total}} = \sum_t E(O_t) \; + \; \sum_\alpha I_\alpha \; + \; \sum_\beta C_\beta, \tag{12.8}$$

where:

- $E(O_t)$ are operator energy costs;
- I_α are interference costs for coupled operator sets;
- C_β are collapse costs for irreversible operations.

Entropy budgets constrain admissible sequences but do not themselves add cost unless they trigger collapse.

6.3 Temporal Localization of Cost

All cost components are localized in logical time. Each cost contribution is associated with a specific logical step or interval.

This temporal localization enables:

- fine-grained replay analysis;
- attribution of cost to specific decisions;
- partial execution accounting.

No cost is amortized implicitly across time.

6.4 Additivity and Non-Additivity

Operator energy costs are additive by default. Interference and collapse costs introduce controlled non-additivity.

Non-additivity is never implicit. Any deviation from additivity must be declared and derivable from operator semantics.

This preserves predictability while allowing expressive accounting of complex structure.

6.5 Determinism and Replayability

Total cost computation is deterministic. Given an execution trace, a verifier can recompute K_{total} exactly without observing runtime behavior.

Replay reproduces not only state evolution but also complete cost accounting.

6.6 Infrastructure Independence

Total semantic cost is independent of:

- hardware architecture;
- execution speed;
- network latency;
- deployment topology.

This independence enables consistent cost comparison across heterogeneous environments.

6.7 Cost as a First-Class Architectural Signal

Total cost may be exposed as an architectural signal usable by:

- schedulers and admission controllers;
- pricing and billing mechanisms;
- governance and regulatory controls;
- optimization and planning tools.

Because cost is semantic, such uses do not compromise determinism.

6.8 Failure and Cost Accounting

Failures do not erase cost. Any operator that commits before failure contributes to total cost.

Failure handling operations themselves may incur energy, interference, or collapse cost and are accounted accordingly.

6.9 Summary

Total cost in QVM–QFM is a deterministic composition of semantic energy, interference, and collapse costs constrained by entropy budgets. This accounting model provides a complete, replayable, and legally robust foundation for resource attribution in distributed computation.

7 Policy, Pricing, and Regulatory Alignment

7.1 From Semantic Cost to Policy Enforcement

Because cost in QVM–QFM is defined semantically and deterministically, it can be used directly as an input to policy enforcement mechanisms. Unlike runtime-based metrics, semantic cost is known, bounded, and explainable at the level of architectural meaning.

Policies may therefore be expressed in terms of allowable energy, interference, collapse, or entropy consumption rather than in terms of execution time or resource usage.

7.2 Deterministic Pricing Models

Semantic cost enables deterministic pricing models in which the price of an execution is a direct function of its structural complexity.

Pricing rules may be defined as mappings

$$P : K_{\text{total}} \mapsto \mathbb{R}_{\geq 0}, \tag{12.9}$$

where K_{total} is the total semantic cost.

Because K_{total} is replayable, pricing decisions are reproducible and auditable.

7.3 Separation of Pricing from Infrastructure

Pricing based on semantic cost is independent of infrastructure efficiency, hardware performance, or operational optimizations.

This separation ensures that:

- users pay for what a computation means, not how it is executed;
- infrastructure providers are free to optimize execution without affecting pricing semantics;
- economic incentives align with semantic clarity rather than low-level micro-optimizations.

7.4 Regulatory Explainability

Semantic cost accounting provides regulator-grade explainability. For any execution, it is possible to reconstruct:

- which operations contributed to cost;
- why interference or collapse occurred;
- how entropy constraints influenced execution.

This enables formal justification of pricing, throttling, or denial decisions in regulated environments.

7.5 Carbon and Sustainability Mapping

Although QVM–QFM does not measure physical energy consumption, semantic cost can be mapped to sustainability policies through externally defined conversion functions.

Such mappings allow:

- carbon-aware policy enforcement without runtime measurement;
- comparison of workloads based on semantic complexity;
- long-term planning independent of hardware evolution.

The mapping layer is explicitly separated from core execution semantics.

7.6 Governance and Budget Enforcement

Semantic energy and entropy budgets can be used as governance instruments. Governance rules may:

- limit allowable entropy growth per execution;
- cap total semantic energy consumption;

- restrict irreversible collapse operations.

Violations of such limits are deterministic semantic errors rather than runtime failures.

7.7 Dispute Resolution and Audit

In the event of disputes regarding pricing, cost attribution, or policy enforcement, semantic cost accounting provides an objective basis for resolution.

An independent auditor can replay the execution, recompute all cost components, and verify policy application without access to runtime logs or infrastructure details.

7.8 Non-Claims

This framework does not claim:

- optimal pricing strategies;
- fairness of economic outcomes;
- automatic regulatory compliance;
- direct measurement of environmental impact.

It provides only the deterministic semantic substrate upon which such concerns may be addressed.

7.9 Summary

By elevating cost to a semantic property, QVM–QFM enables deterministic pricing, transparent policy enforcement, and regulator-grade auditability. Economic and governance decisions become replayable consequences of execution meaning rather than opaque artifacts of runtime behavior.

8 Conclusion

This document has introduced a semantic framework for energy, cost, and entropy accounting in distributed computation executed by a Quantum Virtual Machine (QVM) operating over Quansistor Field Mathematics (QFM).

The framework departs fundamentally from classical cost models by rejecting runtime-dependent metrics such as execution time, instruction counts, or physical resource utilization. Instead, cost is defined as an intrinsic property of computational meaning, derived from operator semantics, structural coupling, and irreversible decision-making.

By defining energy as an operator property, interference as non-additive semantic coupling, collapse as explicit irreversible work, and entropy as a bound on admissible future evolution, the framework provides a complete and deterministic accounting system. All cost components are explicit, replayable, and independent of hardware or deployment topology.

Entropy budgets introduce enforceable limits on speculative growth and ensure predictability of execution evolution. Collapse costs establish responsibility for irreversible decisions. Interference accounting aligns economic incentives with modularity and semantic clarity.

Because total cost is deterministic and localized in logical time, it can be used as a first-class architectural signal for pricing, governance, regulatory explainability, and dispute resolution. Independent auditors can reconstruct and verify cost attribution without access to runtime logs or infrastructure details.

The framework deliberately limits its claims to semantic accounting. It does not measure physical energy, optimize performance, or guarantee fairness. Its contribution lies in establishing a stable, legally robust substrate for reasoning about computational cost in distributed systems.

Taken together, energy, cost, and entropy accounting in QVM–QFM transform resource usage from an observational artifact into an architectural property. Computation becomes not only executable, but explainable, governable, and accountable by construction.

Appendix A

A.1 Formal Definitions of Energy, Interference, Collapse, and Entropy

A.1.1 Purpose and Normative Status

This section provides formal definitions of semantic energy, interference cost, collapse cost, and entropy as used throughout the QVM–QFM energy and cost accounting framework.

The definitions in this section are normative. Any execution claiming compliance with this framework must satisfy these definitions exactly.

A.1.2 Architectural State Space

Let Ψ denote the architectural state of a QVM–QFM execution. Let $\mathcal{A}_t$ denote the set of admissible states reachable from Ψ_t under declared execution semantics.

All semantic quantities defined below are functions of Ψ_t, $\mathcal{A}_t$, and declared operator semantics, and are independent of physical execution.

A.1.3 Operator Energy

Let O be an operator applicable to state Ψ. Operator energy is a function

$$E : O \to \mathbb{R}_{\geq 0}, \tag{12.1}$$

defined by architectural specification.

$E(O)$ depends only on the semantic structure of the induced transformation $\Psi \mapsto O(\Psi)$ and may not depend on runtime, hardware, or scheduling.

A.1.4 Energy of an Execution

For a sequence of committed operators $\{O_t\}$, the total operator energy is

$$E_{\text{ops}} = \sum_t E(O_t), \tag{12.2}$$

unless modified by explicitly defined interference terms.

A.1.5 Interference Cost

Let $\{O_1, \ldots, O_n\}$ be a set of operators whose joint semantic effect cannot be decomposed into independent applications. The interference cost I is defined as

$$I = E(O_1 \circ \cdots \circ O_n) - \sum_{i=1}^{n} E(O_i). \tag{12.3}$$

$I = 0$ if and only if the operators commute semantically and act on disjoint state scopes.

A.1.6 Collapse Operations

A collapse operation is an operator C that maps a set of admissible states $\mathcal{A}_t$ to a strict subset

$$\mathcal{A}_{t+1} \subset \mathcal{A}_t, \tag{12.4}$$

with $|\mathcal{A}_{t+1}| < |\mathcal{A}_t|$.

Collapse operations are irreversible by definition.

A.1.7 Collapse Cost

The collapse cost associated with a collapse operation C is a deterministic function

$$C_{\text{cost}} = \mathcal{C}(\mathcal{A}_t, \mathcal{A}_{t+1}), \tag{12.5}$$

representing the semantic work required to eliminate admissible alternatives.

A.1.8 Semantic Entropy

Semantic entropy at logical time t is defined as

$$S_t = \mathcal{S}(\mathcal{A}_t), \tag{12.6}$$

where $\mathcal{S}$ is a deterministic measure of dispersion over admissible states.

Entropy measures freedom of future evolution, not physical disorder.

A.1.9 Entropy Evolution Rules

Entropy evolution satisfies the following constraints:

- reversible coupling may increase S_t;
- collapse operations strictly decrease S_t;
- neutral operators may leave S_t unchanged.

Entropy evolution is fully determined by execution semantics.

A.1.10 Entropy Budgets

An entropy budget is a declared bound

$$S_t \leq S_{\max}, \tag{12.7}$$

violation of which constitutes a semantic execution error.

Entropy budgets constrain admissible execution trajectories but do not themselves add cost unless they induce collapse.

A.1.11 Total Semantic Cost

The total semantic cost of an execution is defined as

$$K_{\text{total}} = \sum_t E(O_t) + \sum_\alpha I_\alpha + \sum_\beta C_\beta, \tag{12.8}$$

subject to entropy budget constraints.

A.1.12 Determinism and Replayability

All quantities defined in this section are deterministic functions of execution semantics. Given an execution trace, an independent verifier can reconstruct all energy, interference, collapse, and entropy values exactly.

A.1.13 Summary

This section establishes formal, enforceable definitions of semantic energy, interference, collapse, and entropy. Together, these definitions form the mathematical foundation of deterministic cost accounting in QVM–QFM.

Appendix B

B.1 Semantic Cost Contract

B.1.1 Purpose of the Contract

This section defines the semantic cost contract governing all executions that claim compliance with the QVM–QFM energy, cost, and entropy accounting framework.

The contract specifies mandatory obligations under which cost attribution, budget enforcement, and accounting results are considered valid, deterministic, replayable, and legally verifiable.

Any execution violating this contract is non-compliant by definition, regardless of observed runtime behavior.

B.1.2 Scope of the Contract

The semantic cost contract applies to:

- operator energy assignment and accumulation;
- interference detection and accounting;
- collapse identification and cost attribution;
- entropy computation and budget enforcement;
- total cost composition and reporting.

The contract is independent of implementation strategy, hardware architecture, execution speed, and deployment topology.

B.1.3 Operator Energy Obligations

An execution must satisfy the following obligations with respect to operator energy:

- every committed operator has a declared semantic energy value;
- operator energy depends only on semantic structure and resolved scope;
- operator energy is invariant under distribution and execution strategy.

Any undeclared or implicit energy expenditure constitutes a contract violation.

B.1.4 Interference Accounting Obligations

Interference cost must satisfy the following:

- interference is accounted if and only if semantic coupling exists;
- interference cost is zero for semantically independent operators;
- interference cost is deterministic and replayable;
- interference may not be inferred from runtime concurrency.

Failure to declare or account for semantic coupling violates the contract.

B.1.5 Collapse Obligations

Collapse operations must satisfy the following:

- collapse is explicit and architecturally declared;
- collapse strictly reduces the admissible state set;
- collapse incurs deterministic collapse cost;
- collapse is temporally localized in logical time.

Implicit or silent elimination of alternatives is prohibited.

B.1.6 Entropy Obligations

Entropy accounting must satisfy:

- entropy is computed from admissible future state space;
- entropy evolution follows declared semantic rules;
- entropy budgets are enforced deterministically;
- entropy violations result in semantic execution errors.

Entropy may not be approximated or inferred heuristically.

B.1.7 Total Cost Obligations

Total semantic cost must:

- be computable solely from execution semantics;
- include all energy, interference, and collapse contributions;
- exclude runtime, timing, or hardware-dependent factors;
- be identical across replay executions.

Any divergence in total cost under replay constitutes a violation.

B.1.8 Failure and Recovery Obligations

Failures do not excuse cost accounting obligations. Operators that commit before failure contribute to total cost. Failure-handling operations themselves incur cost according to their semantics.

Rollback or recovery may not erase or discount incurred semantic cost.

B.1.9 Verification Criterion

An execution satisfies the semantic cost contract if and only if:

- deterministic replay reconstructs identical cost components;
- entropy budgets are enforced identically;
- total semantic cost is reproduced exactly.

Failure to meet any criterion invalidates claims of compliance.

B.1.10 Legal and Regulatory Interpretation

For legal and regulatory purposes, violation of the semantic cost contract implies that:

- cost attribution is not legally reliable;
- pricing or policy decisions lack semantic justification;
- audit results are inadmissible.

The contract therefore provides a binary standard for admissibility of cost accounting evidence.

B.1.11 Summary

This semantic cost contract establishes strict, enforceable obligations for energy, interference, collapse, and entropy accounting in QVM–QFM. It defines cost as a contractual property of execution semantics, ensuring determinism, replayability, and legal robustness.

Appendix C

C.1 Claim-Ready Summary (US / EPC)

C.1.1 Overview

This section provides a claim-ready summary of the inventive concepts disclosed in this document concerning deterministic semantic accounting of energy, cost, and entropy in distributed computation executed by a Quantum Virtual Machine (QVM) operating over Quansistor Field Mathematics (QFM).

The claims are structured to be suitable for direct translation into United States and European Patent Convention (EPC) claim formats.

C.1.2 Independent Claim

Claim 1 (Independent). A computer-implemented method for deterministic semantic accounting of computational cost, the method comprising:

- executing instructions as deterministic operators acting on an explicit architectural state field governed by Quansistor Field Mathematics;
- assigning a semantic energy value to each operator based solely on the structural properties of the induced state transformation;
- detecting semantic coupling between operators and assigning deterministic interference cost when such coupling exists;
- executing explicit collapse operations that irreversibly eliminate admissible alternative state evolutions and assigning deterministic collapse cost to such operations;
- computing semantic entropy as a measure of admissible future execution states;
- enforcing entropy budgets that bound admissible execution evolution;

wherein total computational cost is derived from semantic structure rather than physical execution characteristics.

C.1.3 Dependent Claims

Claim 2. The method of Claim 1, wherein semantic energy is invariant under distribution, concurrency, and execution topology.

Claim 3. The method of Claim 1, wherein interference cost is zero if and only if operators commute semantically and act on disjoint state scopes.

Claim 4. The method of Claim 1, wherein collapse cost is proportional to the number and structural diversity of eliminated admissible states.

Claim 5. The method of Claim 1, wherein semantic entropy increases under reversible superposition and decreases under collapse operations.

Claim 6. The method of Claim 1, wherein violation of an entropy budget constitutes a semantic execution error.

Claim 7. The method of Claim 1, wherein energy, interference, collapse, and entropy accounting are deterministic and replayable.

C.1.4 System Claims

Claim 8. A distributed computational system comprising:

- a Quantum Virtual Machine configured to execute QFM operators;
- a semantic energy accounting mechanism;
- an interference detection and accounting mechanism;
- a collapse detection and accounting mechanism;
- a semantic entropy computation and budget enforcement mechanism;

wherein the system provides replayable and auditable semantic cost attribution.

C.1.5 Use and Policy Claims

Claim 9. The system of Claim 8, wherein semantic cost accounting is used for deterministic pricing, throttling, or admission control.

Claim 10. The system of Claim 8, wherein semantic cost accounting is used to enforce governance or regulatory constraints independent of physical execution.

C.1.6 Non-Claims

The invention does not claim:

- measurement of physical energy consumption;
- correspondence to thermodynamic entropy;
- runtime-based or probabilistic cost estimation;
- performance optimization or execution efficiency;
- artificial intelligence or autonomous decision-making.

C.1.7 EPC Reformulation Notes

For European Patent Convention filings, Claim 1 may be reformulated as a computer-implemented method producing the technical effects of:

- elimination of runtime-dependent cost variability;
- deterministic and reproducible cost accounting;
- improved predictability and auditability of distributed computation.

C.1.8 Summary

This claim set defines a deterministic semantic framework for energy, cost, and entropy accounting in distributed computation. By binding cost to operator semantics rather than execution artifacts, the invention enables replayable, auditable, and legally robust resource attribution at scale.