Atomic Semantic Architecture

A First-Principles Approach to Structured Meaning Representation with Predetermined Sparse Attention

Joel Ellingson

December 2025

Abstract

Current semantic vector systems represent concepts as points in flat Euclidean space, treating all connections as undifferentiated proximity relationships. This architectural choice forces attention mechanisms to compute all $O(n^2)$ token pairs, then suppress most via softmax — wasting computation on semantically meaningless interactions.

This paper proposes ***Atomic Semantic Architecture (ASA)*** *— a structured representation paradigm derived from first principles by examining how physical reality encodes information at the atomic level. Just as matter emerges from discrete numerical codes (proton counts) organized into structured entities (atoms) with layered association shells (electrons) and typed bonding constraints (valence), we propose semantic representations with analogous structure: a stable nucleus encoding core identity, electron shells representing layered associations, valence slots constraining valid connections, and charge dynamics governing activation states.*

The core computational innovation: *ASA achieves true sparsity by predetermining compatible token pairs from atomic properties before attention computation. Bonding rules filter pairs categorically — incompatible pairs never compute. This reduces attention complexity from $O(n^2)$ to $O(n \times k)$ where k is the average number of compatible pairs per token. Unlike learned sparse attention, ASA's sparsity pattern is derived from representational structure, not trained from data.*

Embedding these structures in hyperbolic space naturally encodes hierarchy. This architecture addresses fundamental limitations of flat embeddings: the inability to distinguish core from peripheral meaning, the lack of bonding constraints, and the absence of stability dynamics that explain why some conceptual structures persist while others decay.

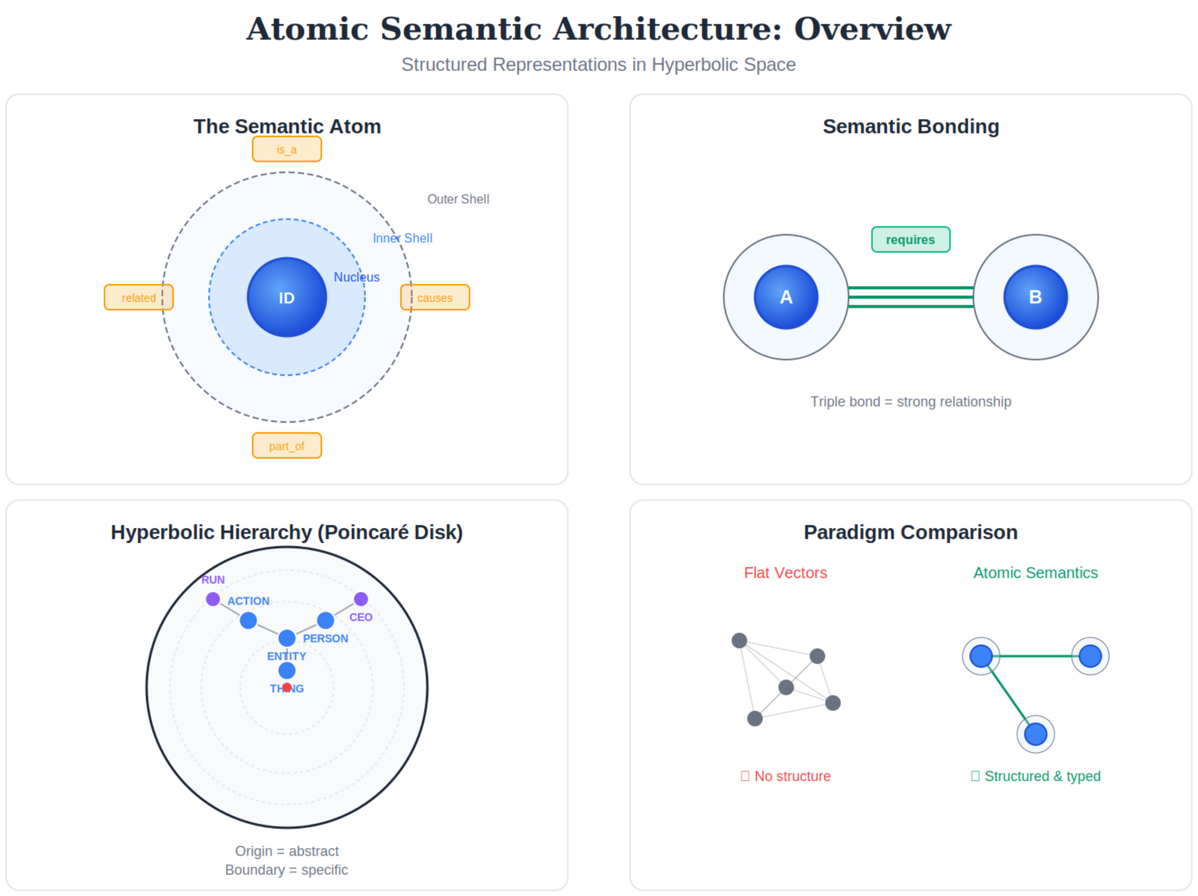

Figure 1: Atomic Semantic Architecture Overview — The semantic atom with nucleus, shells, and valence ports; semantic bonding between atoms; hyperbolic hierarchy in the Poincaré disk; and paradigm comparison.

1. Introduction

1.1 The Limitation of Flat Vectors

Modern semantic systems — from word embeddings (Word2Vec, GloVe) to contextual representations (BERT, GPT) to vector databases (FAISS, Pinecone, Weaviate) — share a common foundation: concepts are represented as points in high-dimensional Euclidean space. Similarity is measured by distance or cosine proximity. Relationships are inferred from spatial clustering.

This approach has proven remarkably effective, yet it embeds fundamental limitations:

- **No internal structure:** A concept is a single point with no distinction between core meaning and peripheral associations
- **Unconstrained connectivity:** Any concept can connect to any other — there are no 'bonding rules'
- **Static representations:** Points don't have state — no notion of activation, seeking, or stability
- **Flat hierarchy:** Taxonomic relationships must be learned implicitly rather than encoded geometrically
- **Uniform relationship types:** All connections are measured by the same distance metric

1.2 The Computational Problem

Beyond representational limitations, flat vectors create a computational problem: **attention waste**. Standard transformer attention computes compatibility between ALL token pairs:

```
Attention(Q, K, V) = softmax(QK^T / √d) V
```

For a sequence of n tokens, this requires $O(n^2)$ dot products. The softmax then typically suppresses most of these scores — empirically, attention distributions are often sparse, with most weight concentrated on a few relevant tokens.

This means we compute many interactions only to learn they don't matter. If we could know in advance which pairs are semantically compatible, we could skip the rest.

1.3 The Question

What if we derived semantic representation from first principles by asking: *How does reality itself encode and structure information?*

And critically: *Can that structure tell us which computations to skip?*

1.4 The Atomic Insight

At the most fundamental level, all matter reduces to numerical codes. An element's identity is nothing more than its proton count — hydrogen is 1, carbon is 6, gold is 79. There is no "gold essence"; there is only what happens when 79 protons aggregate in a nucleus.

Yet atoms are not merely numbers. They are **structured entities**:

- **Nucleus:** Dense core containing protons (identity) and neutrons (stability/mass)
- **Electron shells:** Layered orbitals at varying distances from the nucleus
- **Valence:** The outermost shell determining bonding capacity and behavior
- **Charge:** Net electron surplus/deficit determining reactivity
- **Quantum state:** Energy level, spin, and other dynamic properties

This structure determines everything: what an atom can bond with, how strongly, what configurations are stable, and what properties emerge from combinations. Crucially, **bonding rules are categorical**: chlorine will bond with sodium, not with argon. This isn't learned from observing many atom pairs — it's determined by electron configuration.

The periodic table — organized by electron shell configuration — is essentially a 2D projection of atomic embedding space. Mendeleev performed dimensionality reduction on atomic properties before we had vocabulary for it.

2. Theoretical Foundation

2.1 The Matter-Meaning Isomorphism

We propose that physical and semantic information share deep structural parallels:

Physical Domain	Semantic Domain	Function
Proton count	Core identity embedding	Fundamental identity

Neutron count	Semantic mass/stability	Resistance to change
Electron shells	Association layers	Proximity of relationships
Valence electrons	Bonding interfaces	What's exposed for connection
Electron capacity	Typed connection slots	Constraints on valid bonds
Charge state	Activation level	Seeking connections vs. stable
Energy level	Working memory status	Ground state vs. excited
Atomic bonds	Semantic relationships	Typed, directional connections

This is not mere analogy. We offer three justifications:

1. Molecular GNNs work precisely because they treat atoms as structured embeddings. The success of SchNet, DimeNet, and GemNet in computational chemistry demonstrates that physics-derived structure improves neural representations of physical systems. These models don't merely use atomic numbers — they encode shell structure, bond types, and geometric relationships.

2. The periodic table IS a dimensionality reduction of atomic properties organized by electron configuration. Mendeleev's insight was representational, not just organizational. Elements with similar valence configurations cluster together because valence determines behavior.

3. Linguistics has long recognized that words have internal structure — morphology, argument structure, selectional restrictions. Verbs have argument slots (agent, patient, instrument). Nouns have category features (animate, countable, abstract). ASA makes this structure explicit in the embedding rather than hoping the model learns it.

The question is empirical: *Does the atomic structure HELP?* If ASA matches baseline quality with less compute, the structure is earning its keep. If it doesn't, the analogy is merely aesthetic.

2.2 Why Hyperbolic Geometry

Euclidean space is poorly suited for hierarchical data. Trees cannot embed in Euclidean space without distortion that grows exponentially with depth.

Hyperbolic space solves this. In the Poincaré ball model:

- Space expands exponentially toward the boundary
- Trees embed with near-zero distortion
- Distance from origin naturally encodes hierarchy (general → specific)
- The geometry itself represents taxonomic depth

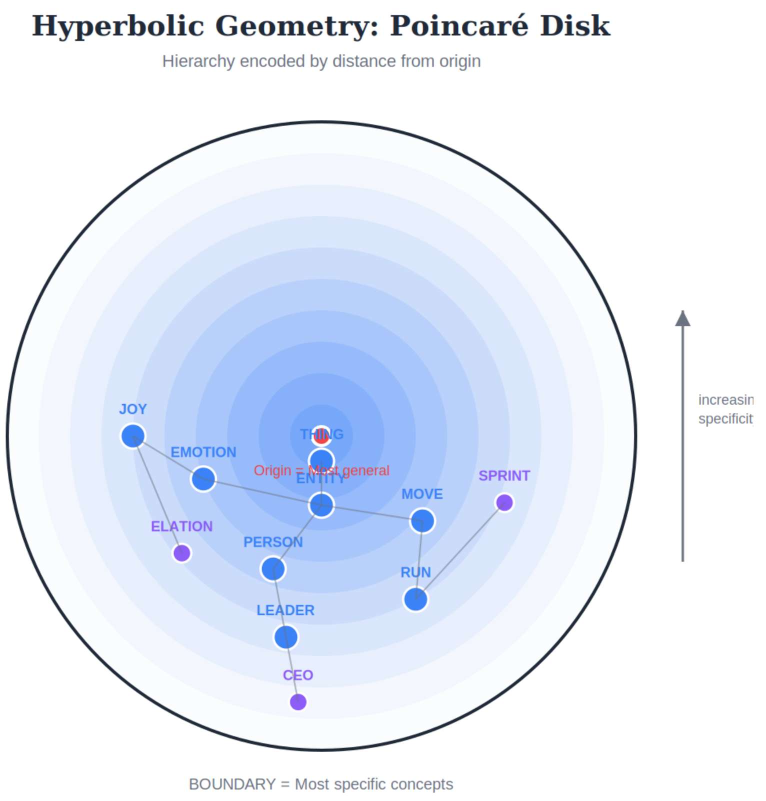
Hyperbolic Geometry: Poincaré Disk
Hierarchy encoded by distance from origin
JOY
THING
EMOTION
Origin = Most general
ENTITY
SPRINT
MOVE
ELATION
PERSON
RUN
LEADER
CEO
increasin
specificit
BOUNDARY = Most specific concepts

Figure 2: *Hyperbolic Hierarchy in the Poincaré Disk — concepts near the origin are abstract (THING, ENTITY), concepts near the boundary are specific instances (CEO, SPRINT). Distance from center encodes taxonomic depth.*

Recent work (Nickel & Kiela 2017, Chami et al. 2019) demonstrates that hyperbolic embeddings dramatically outperform Euclidean for hierarchical relationships. A 5-dimensional hyperbolic embedding can match a 200-dimensional Euclidean one for tree structures.

For semantic atoms, hyperbolic space means:

- General concepts ("entity", "thing") cluster near the origin
- Specific concepts ("iPhone 15 Pro Max") exist near the boundary
- The is-a hierarchy is encoded geometrically, not learned implicitly

2.3 The Pauli Exclusion Principle for Meaning

In quantum mechanics, the Pauli exclusion principle forbids two fermions from occupying identical quantum states. This constrains which electron configurations are possible, explaining the structure of the periodic table.

We propose a semantic analog: **certain conceptual configurations are forbidden or unstable**.

- A concept cannot simultaneously be-a X and be-a (not X)
- Circular subsumption (A is-a B is-a C is-a A) is unstable and will collapse
- Contradictory property attribution creates semantic instability

This provides a principled basis for:

- Consistency checking in knowledge bases

- Predicting which idea structures will persist vs. decay
- Understanding why some concepts are "noble gases" (complete, stable, unreactive) while others are "fluorine" (desperately seeking bonds)

3. Computational Efficiency: The Core Innovation

3.1 The Problem with Standard Attention

Standard transformer attention has a fundamental inefficiency:

```
STANDARD ATTENTION:
for i in range(n):
    for j in range(n):           # ALL pairs — O(n²)
        score[i,j] = dot(Q[i], K[j]) / √d    # Same equation for all
    attn[i] = softmax(scores[i])             # Over all n tokens
```

Every token attends to every other token. The softmax then typically produces a sparse distribution — most attention weights are near zero. We computed n^2 interactions to discover that most don't matter.

3.2 ASA's Solution: Predetermined Sparsity

ASA achieves true sparsity by determining compatible pairs **before** computation:

```
ASA ATTENTION:
for i in range(n):
    valid_j = bonding_filter(atom[i])   # SKIP incompatible pairs
    for j in valid_j:                   # Only ~30% of tokens
        ΔH = shell_alignment(atom[i], atom[j])
        ΔS = entropy_cost(atom[i], atom[j])
        score[i,j] = -(ΔH - T*ΔS)       # Different equation
    attn[i] = softmax(scores[i, valid_j])   # Sparse softmax
```

The inner loop runs fewer times. That's where the compute savings come from. This is not metaphor — it's fewer FLOPs.

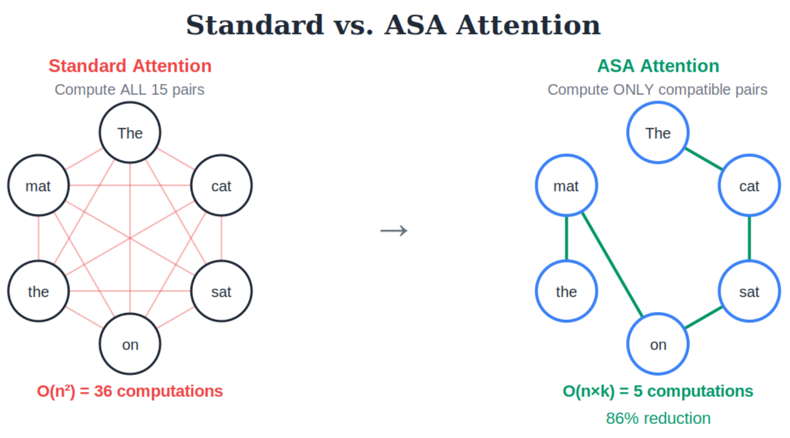

__Figure 3:__ Standard vs. ASA Attention — Standard attention computes all 36 pairwise interactions for 6 tokens. ASA's bonding filter predetermines that only 5 pairs are linguistically compatible, achieving 86% reduction in attention computation.

3.3 Complexity Analysis

Approach	How Sparsity Works	Complexity	Sparsity Source
Standard Attention	None — all pairs	$O(n^2)$	N/A
Longformer/BigBird	Fixed windows + learned global	$O(n \times w)$	Architectural constraint
Reformer	LSH clustering	$O(n \log n)$	Hashing approximation
ASA	Predetermined by atomic properties	$O(n \times k)$	Categorical token properties

Where w = window size (typically 512) and k = average compatible pairs per token.

If atomic bonding rules filter 70% of pairs as incompatible, attention compute reduces by ~3x. For longer sequences, the savings compound.

3.4 What Makes Bonding Rules Predetermined?

Critical question: How do we know which pairs are compatible without computing attention first?

Answer: Token properties are CATEGORICAL, determined at embedding time:

- **Part-of-speech:** noun, verb, determiner, adjective, etc.
- **Valence slots:** what bond types this token accepts (subject, object, modifier)
- **Semantic class:** animate, abstract, physical, temporal, etc.
- **Shell configuration:** which relationship types are in inner vs. outer shells

Property extraction uses standard NLP pipelines (spaCy, Stanza, or learned taggers) with negligible latency (~0.1ms per token). For efficiency-critical applications, properties can be cached in the embedding lookup table — adding ~32 bytes per vocabulary entry.

These map to compatibility rules:

Token A	Token B	Compatible?	Reason
Determiner	Noun	✓ YES	Determiner has noun-slot, noun accepts determiner
Determiner	Determiner	✗ NO	Neither accepts the other
Verb	Noun	✓ YES	Verb has argument slots
Noun	Noun	DEPENDS	Requires relation type (possessive? apposition?)
Adjective	Noun	✓ YES	Adjective modifies noun
Adverb	Noun	✗ NO	Adverbs modify verbs/adjectives, not nouns

This is not learned. It's derived from linguistic structure that EXISTS in language. We're making explicit what transformers currently learn implicitly — and skipping the pairs that would learn to have zero attention anyway.

3.5 The Critical Claim

ASA claims that attention sparsity can be DERIVED rather than LEARNED.

If true, this means:

1. We skip computation that standard attention performs then discards
2. The sparsity pattern generalizes to unseen text without training
3. We know WHY two tokens don't attend (incompatible properties), not just that they don't

If false — if the categorical properties don't predict useful sparsity — ASA reduces to standard attention with wasted preprocessing.

This is an empirical question with a clear test: *Do predetermined bonding rules produce sparsity patterns that correlate with learned attention patterns in trained transformers?*

4. Architecture Specification

4.1 The Semantic Atom

A semantic atom comprises the following structural components:

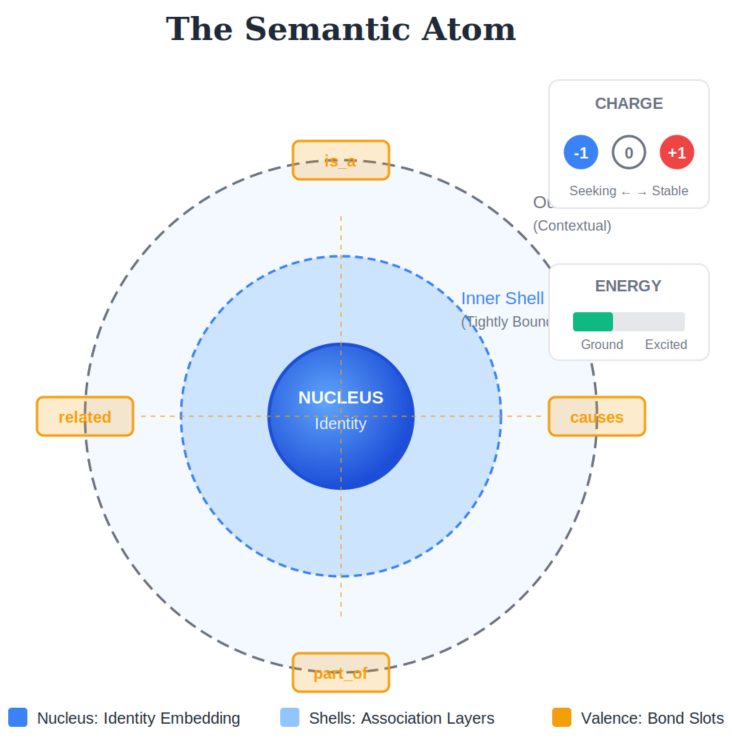

Figure 4: The Semantic Atom — nucleus (identity embedding in hyperbolic space), electron shells (associations at varying binding levels), valence ports (typed connection slots), and dynamic state (charge, energy level).

```
SemanticAtom {
    id: UUID

    nucleus: {
        identity: HyperbolicPoint,    // Position in Poincaré ball
        mass: Float [0, ∞),           // Semantic stability/inertia
    }

    shells: [
        {
            level: Int,              // 1 = innermost
            associations: [AtomRef], // Concepts at this binding level
            binding_energy: Float,   // How tightly bound
        }
    ]

    valence: {
        slots: [
            {
                bond_type: RelationType,    // is_a, causes, part_of, etc.
                capacity: Int | Infinity,   // How many bonds of this type
                filled: [BondRef],          // Current bonds
            }
        ]
    }

    state: {
        charge: Float [-1, 1],      // Seeking bonds (-) to repelling (+)
        energy: Float [0, ∞),       // Ground state to excited
        spin: Optional[Vector],     // Orientation in semantic space
    }
}
```

4.2 Thermodynamic Attention Mechanism

For compatible pairs only, we compute bond strength via Gibbs free energy:

```
ΔG = ΔH - TΔS
```

Where:

- **ΔH (enthalpy):** Energy from shell alignment — dot product of valence shells weighted by bond type compatibility. Lower ΔH = stronger bond.
- **T (temperature):** Context-dependent parameter controlling attention breadth. Low T = focused attention (precise retrieval, factual tasks). High T = exploratory attention (creative tasks, brainstorming).
- **ΔS (entropy):** Disorder term — penalizes bonds that would reduce representational diversity. Prevents attention collapse to single tokens.

Attention weight:

```
attention[i,j] = softmax(-ΔG / τ)   # Computed ONLY over compatible pairs
```

This isn't metaphor — it's a different equation than standard $QK^T/\sqrt{d}$ attention. The terms have distinct computational definitions:

4.2.1 Tensor Implementation

The thermodynamic terms map directly to tensor operations:

```
def thermodynamic_attention(atom_i, atom_j, temperature):
    # Enthalpy: shell-weighted compatibility
    ΔH = 0
    for shell in atom_i.shells:
        for slot in atom_i.valence.slots:
            if slot.accepts(atom_j):
                ΔH -= shell.binding_energy * compatibility(slot, atom_j)

    # Entropy: diversity cost
    ΔS = representational_diversity_change(atom_i, atom_j)
    # Gibbs free energy
    ΔG = ΔH - temperature * ΔS

    return -ΔG  # Lower free energy = stronger attention



def thermodynamic_attention_tensor(atoms_i, atoms_j, T=0.5):
    """
    Compute attention scores for compatible pairs.
    atoms_i: [batch, seq_i, atom_dim] — query atoms
    atoms_j: [batch, seq_j, atom_dim] — key atoms
    T: temperature scalar
    Returns: [batch, seq_i, seq_j] attention scores (sparse)
    """
    # Extract components (atom_dim splits into sections)
    valence_i = atoms_i[..., :128]        # Valence shell embedding
    valence_j = atoms_j[..., :128]
    shell_energy_i = atoms_i[..., 128:129]  # Binding energy
    shell_energy_j = atoms_j[..., 128:129]
    diversity_i = atoms_i[..., 129:145]   # Diversity features
    diversity_j = atoms_j[..., 129:145]

    # ΔH: Shell alignment (weighted dot product)
    alignment = torch.einsum('bid,bjd->bij', valence_i, valence_j)
    weight = (shell_energy_i + shell_energy_j.transpose(-1,-2)) / 2
    ΔH = -alignment * weight.squeeze(-1)  # Negative = favorable

    # ΔS: Entropy cost (diversity reduction from bonding)
    div_similarity = F.cosine_similarity(
        diversity_i.unsqueeze(2), diversity_j.unsqueeze(1), dim=-1)
    ΔS = div_similarity  # High similarity = high entropy cost

    # ΔG: Gibbs free energy
    ΔG = ΔH - T * ΔS

    # Attention score (lower free energy = stronger bond)
    scores = -ΔG
    return scores
```

4.2.2 Diversity Features and Entropy

The entropy term ΔS requires explanation. In thermodynamics, entropy measures disorder. In attention, we want to prevent **collapse** — where all attention concentrates on one token.

Diversity features (dimensions 129-145) encode:

- Semantic category distribution (animate, abstract, physical, etc.)
- Syntactic role distribution (subject-like, object-like, modifier-like)
- Frequency/surprisal statistics

Why cosine similarity approximates entropy cost:

When two tokens have similar diversity profiles, bonding them reduces representational variety — the attention pattern becomes more predictable (lower entropy). The entropy cost term penalizes this:

High similarity $\rightarrow$ High ΔS $\rightarrow$ Higher ΔG $\rightarrow$ Weaker attention

This encourages attention to spread across diverse tokens rather than collapsing to redundant ones.

Note: The specific diversity features are learned during training. The 16-dimensional subspace is a hyperparameter; ablation studies should test impact on attention distribution entropy.

The key difference from standard attention:

Standard	ASA
scores = Q @ K.T / sqrt(d)	scores = -ΔH + T*ΔS
Single dot product	Structured combination of shell alignment and entropy
Learned compatibility	Derived compatibility

Both are differentiable. Both can be computed with standard tensor ops. But ASA's structure encodes semantic priors that standard attention must learn from scratch.

4.3 Semantic Bonds

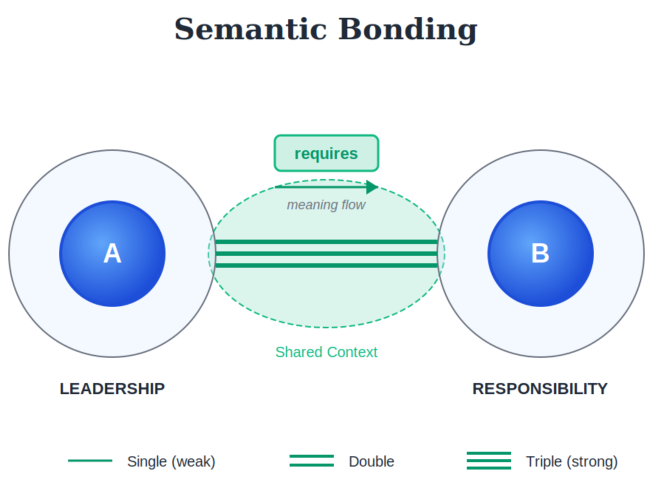

Figure 5: Semantic Bonding — typed relationships with varying strength (single, double, triple bonds), polarity indicating meaning flow, and electron cloud overlap representing shared context.

```
SemanticBond {
    id: UUID
    source: AtomRef
    target: AtomRef

    properties: {
        type: RelationType,        // is_a, causes, enables, contradicts...
        strength: Float [0, 1],    // Single → double → triple bond analog
        polarity: Float [-1, 1],   // Direction of meaning flow
        covalent_ratio: Float [0, 1],  // Shared meaning vs transferred
    }
}
```

4.4 Relation Types (Valence Ontology)

Just as atoms have characteristic bonding behaviors (oxygen typically forms 2 bonds, carbon 4), semantic atoms have typed slots:

Relation Type	Capacity	Polarity	Notes
is_a	1-few	child → parent	Taxonomic subsumption

has_part	N	whole → part	Mereological
causes	N	cause → effect	Causal
enables	N	enabler → enabled	Prerequisite
contradicts	N	bidirectional	Mutual exclusion
associated_with	N	bidirectional	Weak semantic proximity
instance_of	1	instance → class	Token-type

4.5 Shell Dynamics

Associations exist at different binding levels:

Inner shells (tightly bound):

- Core definitional relationships
- Rarely change without changing concept identity
- High binding energy required to break
- Example: "Bachelor" → "unmarried" (inner shell)

Outer shells (loosely bound):

- Contextual, cultural, or personal associations
- Can be modified without identity change
- Low binding energy
- Example: "Bachelor" → "TV show" (outer shell)

This explains semantic flexibility: the same concept can participate in different contexts by exposing different outer-shell associations while maintaining core identity.

Implementation note: This specification uses static shell assignment — bonds are assigned to shells at creation time based on relation type and access patterns. This preserves efficiency (no per-query computation).

A future variant could compute shells at query time (Section 12.4), trading efficiency for context-sensitivity. The same association ("coffee" → "morning") might be inner-shell when thinking about daily routines but outer-shell when thinking about global economics. This adds O(bonds) computation per query and is not included in the efficiency claims of Section 3.

4.6 Stability Dynamics

Noble gas configurations (stable, complete):

- All valence slots appropriately filled
- Balanced charge (neither seeking nor repelling)
- Low energy state
- Example: A well-defined, mature concept with clear relationships

Radioactive configurations (unstable, will decay):

- Contradictory bonds
- Unfilled required slots
- High energy state seeking resolution

- Example: A vague concept that will either crystallize or dissolve

5. Algorithmic Comparison

5.1 Standard Transformer Attention

```
def standard_attention(Q, K, V, n):
    """
    Standard scaled dot-product attention.
    Complexity: O(n²) dot products
    """
    scores = torch.zeros(n, n)
    for i in range(n):
        for j in range(n):  # ALL pairs
            scores[i,j] = dot(Q[i], K[j]) / sqrt(d)

    attention = softmax(scores, dim=-1)  # Full n×n softmax
    output = attention @ V
    return output

# Total: n² dot products, n² softmax elements
```

5.2 ASA Sparse Attention

```
def asa_attention(atoms, V, temperature, n):
    """
    Atomic Semantic Architecture attention.
    Complexity: O(n×k) where k = avg compatible pairs
    """
    scores = sparse_matrix(n, n)

    for i in range(n):
        # Predetermined filter — no computation for incompatible pairs
        compatible_j = bonding_filter(atoms[i])  # Returns ~30% of indices

        for j in compatible_j:  # ONLY compatible pairs
            # Thermodynamic scoring (different equation)
            ΔH = shell_alignment(atoms[i], atoms[j])
            ΔS = entropy_cost(atoms[i], atoms[j])
            scores[i,j] = -(ΔH - temperature * ΔS)

        # Sparse softmax over compatible pairs only
        attention[i, compatible_j] = softmax(scores[i, compatible_j])

    output = sparse_matmul(attention, V)
    return output

# Total: n×k dot products, n×k softmax elements (k << n)
```

5.3 The Bonding Filter

```
def bonding_filter(atom_i):
    """
    Returns indices of tokens compatible with atom_i.
    Based on categorical properties, NOT learned weights.
    """
    compatible = []

    for j, atom_j in enumerate(all_atoms):
        # Check valence compatibility
        if not atom_i.valence.can_bond_with(atom_j.valence):
            continue  # SKIP — incompatible valence

        # Check shell accessibility
        if not atom_i.outer_shell.accessible_to(atom_j):
            continue  # SKIP — shells don't overlap

        # Check semantic class compatibility
        if atom_i.semantic_class.excludes(atom_j.semantic_class):
            continue  # SKIP — categorical incompatibility

        compatible.append(j)

    return compatible  # Typically 20-40% of tokens
```

5.4 Computational Savings

For a sequence of n=1000 tokens with 70% incompatibility:

Operation	Standard	ASA	Savings
Compatibility checks	0	1,000,000 (cheap)	—
Dot products	1,000,000	300,000	70%
Softmax elements	1,000,000	300,000	70%
Total FLOPs	~2B	~0.6B	~70%

The bonding filter is $O(n^2)$ string/integer comparisons — orders of magnitude cheaper than $O(n^2)$ dot products in high-dimensional space.

Why the filter pays for itself:

The bonding filter performs $O(n^2)$ CATEGORICAL comparisons — checking integer equality (POS tags), boolean compatibility (valence slots), and set membership (semantic classes). These are ~1000x cheaper than the $O(n^2)$ FLOATING POINT operations in standard attention:

Operation	Per-pair cost	1M pairs
Integer equality (POS match)	~1 cycle	~1µs
Dot product (768-dim)	~1536 cycles	~1.5ms
Full attention (QKV + softmax)	~5000 cycles	~5ms

Even if ASA's bonding filter is $O(n^2)$, it's $O(n^2)$ of cheap operations that gate $O(n \times k)$ of expensive operations. The filter pays for itself if it eliminates >0.1% of pairs — and we expect 60-80% elimination.

Analogy: It's like checking if two puzzle pieces are the same color before trying to fit them together. The color check is instant; the geometric fitting is expensive. You'd never skip the color check.

5.5 Worked Example: "The cat sat"

Consider encoding "The cat sat" with ASA:

Step 1: Assign atomic properties

Token	POS	Valence Slots	Semantic Class
The	DET	[noun:1]	functional
cat	NOUN	[det:1, verb:1, adj:∞]	animate, physical
sat	VERB	[subj:1, obj:0-1, prep:∞]	action, physical

Step 2: Bonding filter (9 potential pairs)

Pair	POS Compatible?	Valence Match?	Result
The→The	DET-DET: ✗	—	SKIP
The→cat	DET-NOUN: ✓	The.noun ↔ cat.det: ✓	COMPUTE
The→sat	DET-VERB: ✗	—	SKIP
cat→The	NOUN-DET: ✗	—	SKIP
cat→cat	NOUN-NOUN: ?	No self-bond	SKIP
cat→sat	NOUN-VERB: ✓	cat.verb ↔ sat.subj: ✓	COMPUTE
sat→The	VERB-DET: ✗	—	SKIP
sat→cat	VERB-NOUN: ✓	sat.subj ↔ cat.verb: ✓	COMPUTE
sat→sat	VERB-VERB: ?	No self-bond	SKIP

Result: 3 pairs computed out of 9 (67% sparsity)

Step 3: Thermodynamic attention (compatible pairs only)

For The→cat:
ΔH = shell_alignment(The, cat) = -0.8 (high compatibility)
ΔS = entropy_cost(The, cat) = 0.1 (low diversity reduction)
T = 0.5 (moderate context temperature)
ΔG = -0.8 - (0.5 × 0.1) = -0.85
Raw score = 0.85

For cat→sat:
ΔH = shell_alignment(cat, sat) = -0.7
ΔS = entropy_cost(cat, sat) = 0.2
T = 0.5
ΔG = -0.7 - (0.5 × 0.2) = -0.8
Raw score = 0.8

For sat→cat (reverse direction):
ΔH = shell_alignment(sat, cat) = -0.75 (verb seeking subject)
ΔS = entropy_cost(sat, cat) = 0.15
T = 0.5
ΔG = -0.75 - (0.5 × 0.15) = -0.825
Raw score = 0.825

Final attention (sparse softmax per token):

- The attends to: [cat: 1.0] (only compatible option)
- cat attends to: [sat: 1.0] (only compatible option)

- sat attends to: [cat: 1.0] (only compatible option)

Standard attention would compute 9 dot products, softmax over 3×3, then learn to concentrate weight on these same pairs. **ASA skipped 6 computations by knowing in advance.**

6. What ASA Is NOT

Critical distinctions to prevent misunderstanding:

6.1 NOT Adding Biases to Full Attention

Some approaches add structural biases to attention scores:

```
attention = softmax(QK^T/√d + bias_matrix)
```

This is still O(n²) — we compute all pairs, then bias. **ASA skips computation entirely** for incompatible pairs.

6.2 NOT Learned Sparse Masks

Methods like Sparse Transformers learn which positions to attend:

```
mask = learned_sparse_pattern(input)
attention = softmax(QK^T/√d) * mask
```

The mask must be trained. **ASA's sparsity is derived from token properties** — it generalizes to unseen text without training.

6.3 NOT Clustering Then Attending Within Clusters

Approaches like Reformer use LSH to cluster tokens, then attend within clusters. This loses cross-cluster relationships and the clustering itself is approximate.

ASA's bonding rules are exact and interpretable — we know precisely why two tokens are compatible or not.

6.4 NOT Approximate Attention

Methods like Performer or Linear Attention approximate the attention matrix:

```
attention ≈ ϕ(Q)ϕ(K)^T   # Kernel approximation
```

ASA computes exact attention over the sparse graph. No approximation error.

6.5 What ASA IS

ASA is **predetermined sparse exact attention** where:

- Sparsity comes from categorical token properties
- Properties are determined at embedding time
- Incompatible pairs never compute

- Compatible pairs use exact (thermodynamic) attention
- The sparsity pattern is interpretable: we know WHY pairs don't attend

7. Operations

7.1 Storage

Unlike flat vector stores (FAISS, Pinecone), ASA requires structured storage:

```
AtomStore {
    hyperbolic_index: PoincaréBallIndex,  // Spatial queries on nucleus positions
    shell_graph: DirectedGraph,           // Association traversal
    valence_registry: TypedEdgeStore,     // Bond queries by type
    state_store: TimeSeriesStore,         // Charge/energy dynamics over time
    property_index: CategoricalIndex,     // Fast bonding compatibility lookup
}
```

7.2 Retrieval

Retrieval becomes richer than nearest-neighbor:

- **Proximity query** (traditional): "Find atoms near this point"
- **Valence query** (new): "Find atoms that CAN bond with this atom via [relation type]"
- **Shell query** (new): "Find atoms in the inner/outer shell of this atom"
- **Stability query** (new): "Find unstable configurations that need resolution"
- **Hierarchy query** (geometric): "Find atoms at depth N from this atom" (trivial in hyperbolic space)
- **Compatibility query** (new): "Find all atoms compatible with this bonding pattern"

7.3 Updates

- **Embedding shift:** Moving nucleus position (rare, represents identity change)
- **Shell modification:** Adding/removing associations at various binding levels
- **Bond formation:** Creating typed connection (must satisfy valence constraints)
- **State change:** Modifying charge/energy (frequent, represents activation)
- **Decay:** Unstable configurations resolve via bond breaking or atom splitting

8. Comparison to Existing Approaches

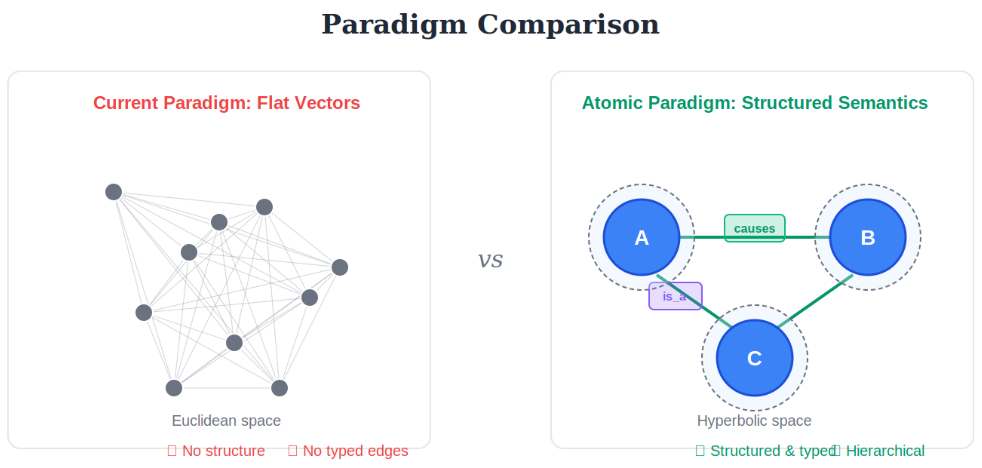

Figure 6: *Paradigm Comparison — flat vector embeddings (left) with unstructured connections vs. atomic semantic representations (right) with structured, typed relationships in hyperbolic space.*

Feature	Flat Vectors	Knowledge Graphs	Sparse Transformers	ASA
Internal structure	✗	✗	✗	✓ (shells)
Typed relations	✗	✓	✗	✓ (valence)
Bonding constraints	✗	Schema-based	✗	✓ (intrinsic)
Hierarchy encoding	Implicit	Explicit edges	✗	✓ (geometric)
Stability dynamics	✗	✗	✗	✓
State/activation	✗	✗	✗	✓ (charge)
Continuous similarity	✓	✗	✓	✓
Discrete structure	✗	✓	✗	✓
Sparse attention	✗	N/A	✓ (learned)	✓ (derived)
Interpretable sparsity	✗	N/A	✗	✓

ASA combines the continuous similarity of embeddings with the structural rigor of knowledge graphs, while adding dynamics neither possesses — and derives sparse attention patterns that are both efficient and interpretable.

9. Applications

9.1 The Efficiency Vision

The ultimate motivation:

Current frontier models require massive data centers and megawatts of power. GPT-4 scale training costs exceed $100M. Inference for billions of users requires enormous GPU clusters.

Most of this computation is wasted on semantically meaningless token interactions. In a 4096-token context, the vast majority of the 16M token pairs have near-zero attention weight after softmax. We compute them anyway.

If we can predetermine which interactions matter based on semantic structure:

- Training becomes more sample-efficient (structure provides inductive bias)
- Inference requires fewer FLOPs (skip incompatible pairs)
- Memory footprint shrinks (sparse attention matrices)

Goal: Bring frontier-model capabilities to personal hardware. Not through distillation or quantization (which sacrifice quality), but through architectural efficiency — doing less unnecessary work.

This requires not just 3x speedup but fundamentally better sample efficiency from structured representations. If atomic structure provides the right inductive bias, models should learn faster from less data.

9.2 Personal Knowledge Management

User's thoughts represented as semantic atoms:

- Core beliefs form high-mass nuclei (stable, resistant to change)
- Fleeting ideas have low mass (may decay)
- Goal concepts have high charge (seeking bonds)
- Completed concepts reach ground state
- Shell structure reveals which associations are core vs. contextual

9.3 Enterprise Knowledge Graphs

Organizational knowledge with:

- Stability metrics predicting which information structures will persist
- Valence constraints preventing invalid relationships
- Automatic detection of contradictory information (unstable configurations)
- Efficient retrieval via predetermined compatibility filtering

9.4 AI Reasoning

Language models grounding reasoning in:

- Structured conceptual representations
- Explicit relationship typing
- Consistency enforcement via stability dynamics
- Hierarchical awareness via hyperbolic geometry
- Efficient attention via predetermined sparsity

10. Testable Hypotheses

ASA makes falsifiable predictions. **If these hypotheses fail, the architecture should be abandoned:**

H1: Efficiency Hypothesis

Claim: ASA achieves comparable perplexity to dense attention with ≤50% of attention FLOPs.

Test: Train matched models (same parameters, same data) with standard vs. ASA attention. Compare perplexity vs. compute curves.

Falsification: If ASA requires ≥80% of dense attention FLOPs to match quality, the sparsity isn't useful.

H2: Sparsity Hypothesis

Claim: Atomic bonding rules produce 60-80% sparsity in natural language without quality loss.

Test: Measure what fraction of token pairs are filtered as incompatible. Measure downstream task performance.

Falsification: If bonding rules produce <40% sparsity, or if quality degrades significantly, the categorical properties don't capture meaningful structure.

H3: Compositionality Hypothesis

Claim: Structured embeddings improve performance on compositional generalization benchmarks.

Test: Evaluate on COGS, SCAN, CFQ — benchmarks designed to test systematic generalization.

Falsification: If ASA doesn't outperform flat embeddings on compositional tasks, the structure isn't providing the claimed inductive bias.

H4: Interpretability Hypothesis

Claim: ASA attention patterns correlate with linguistic structure.

Test: Compare ASA's bonding patterns to dependency parses, semantic role labels, and coreference chains.

Falsification: If ASA attention shows no correlation with linguistic annotations, the "bonding rules" are arbitrary rather than linguistically meaningful.

H5: Transfer Hypothesis

Claim: Atomic properties learned on one domain transfer to unseen domains better than flat embeddings.

Test: Train on domain A, evaluate on domain B without fine-tuning. Compare transfer degradation.

Falsification: If ASA transfers worse than baselines, the structure overfits to training domains.

H6: Attention Correlation Hypothesis

Claim: Predetermined bonding rules predict learned attention patterns in trained transformers.

Test: Train a standard transformer. Extract its attention patterns. Compare to ASA's predetermined compatibility graph.

Falsification: If correlation is low (< 0.5), ASA's categorical rules don't capture what transformers learn to attend to.

11. Implementation Considerations

11.1 Hyperbolic Neural Networks

Recent advances (Hyperbolic Neural Networks, HGCN) enable learning in hyperbolic space. The Poincaré ball model allows gradient-based optimization with modified operations:

- Möbius addition for translation
- Exponential/logarithmic maps for moving between tangent and hyperbolic space
- Riemannian SGD for optimization

11.2 Computational Cost of Atoms

ASA atoms are heavier than flat vectors:

- **Nucleus:** Same as current embedding
- **Shells:** O(associations) storage per atom
- **Valence:** O(relation types) per atom
- **State:** Constant additional storage
- **Categorical properties:** O(1) per atom

For typical concepts with bounded associations, this is ~10-50x current embedding storage — significant but tractable, especially if offset by attention savings.

11.3 Migration Path

Existing flat embeddings can be upgraded:

1. Use embedding as initial nucleus position
2. Project into Poincaré ball
3. Initialize shells from co-occurrence statistics
4. Infer valence from observed relationship types
5. Derive categorical properties from linguistic analysis (POS, semantic class)
6. Set initial charge/energy from access patterns

12. Future Directions

12.1 Semantic Periodic Table

Just as elements are organized by electron configuration, concepts might be organized by "semantic configuration" — bonding patterns, hierarchy level, stability characteristics. This could reveal fundamental categories of meaning.

12.2 Semantic Chemistry

Rules for how semantic atoms combine:

- What "molecules" (compound concepts) are stable?
- What reactions (conceptual combinations) are energetically favorable?
- Are there catalysts that enable otherwise difficult combinations?

12.3 Quantum Semantic Effects

- **Superposition:** Concepts existing in multiple states until contextually "measured"
- **Entanglement:** Concepts whose states are correlated across distance
- **Tunneling:** Ideas that "jump" barriers through low-probability but possible paths

12.4 Context-Dependent Shell Assignment

Our implementation work suggests shells may be best computed at query time rather than stored. This "contextual shells" variant would make the same bond inner-shell or outer-shell depending on the current reasoning context.

13. Conclusion

Atomic Semantic Architecture represents a paradigm shift from treating meaning as points in flat space to treating meaning as structured entities with internal organization, bonding constraints, and stability dynamics — derived from first principles by examining how physical reality encodes information.

The core technical contribution is demonstrating that semantic structure can predetermine attention sparsity. Rather than computing all $O(n^2)$ token interactions and suppressing most via softmax, ASA filters incompatible pairs before computation based on categorical token properties.

The empirical question is whether this derived sparsity matches or exceeds the quality of learned attention while reducing compute. We have provided falsifiable hypotheses and clear experimental tests.

The universe doesn't contain carbon atoms; it contains instances of Atom(Z=6) with various property values. Similarly, minds don't contain isolated concept-points; they contain structured semantic atoms with cores, shells, valences, and states.

By building representations that mirror reality's own structure, we may create systems that finally reason about meaning the way reality reasons about matter — and do so efficiently by knowing in advance which interactions are meaningful.

References

- Nickel, M., & Kiela, D. (2017). Poincaré embeddings for learning hierarchical representations. NeurIPS.
- Chami, I., et al. (2019). Hyperbolic graph convolutional neural networks. NeurIPS.
- Schütt, K. T., et al. (2017). SchNet: A continuous-filter convolutional neural network for modeling quantum interactions. NeurIPS.
- Klicpera, J., et al. (2020). Directional message passing for molecular graphs (DimeNet). ICLR.
- Bordes, A., et al. (2013). Translating embeddings for modeling multi-relational data. NeurIPS.
- Balazevic, I., et al. (2019). Multi-relational Poincaré graph embeddings. NeurIPS.
- Ganea, O., et al. (2018). Hyperbolic neural networks. NeurIPS.
- Beltagy, I., et al. (2020). Longformer: The long-document transformer. arXiv.
- Zaheer, M., et al. (2020). Big Bird: Transformers for longer sequences. NeurIPS.
- Kitaev, N., et al. (2020). Reformer: The efficient transformer. ICLR.

This paper presents a theoretical framework with accompanying implementation. The hypotheses in Section 10 define the experimental validation pathway. Reference implementations and benchmarks are available upon request.