Distributed Compositionality: Formal Foundations for Distributional Semantics

James Pustejovsky
Brandeis University

Elisabetta Ježek
University of Pavia

May 12, 2026
LREC 2026
Palma, Mallorca

What This Tutorial Offers

- A **theory of compositionality** that spans symbolic and distributional approaches
- A survey of where **current formal and neural models** succeed and fail at composition
- An overview of the **Generative Lexicon** framework: Type Theory, Qualia Structure, Event Structure
- New mechanisms: **QES** (Qualia Event Structure) and **DOM** (Dynamic Object Model)
- **Distributed Compositionality** as a bridge: composition networks and controllers
- A **vector-space realization** via VSA/HRR, with a view toward exact algebraic binding
- Applications to NLI, state tracking, coercion, and multimodal grounding

Audience

Computational semanticists, cognitive linguists, and NLP practitioners. No advanced math required; familiarity with formal semantics or distributional modeling helpful.

Lecture Roadmap – 180 Minutes

§	Topic	Time
	PART I – Motivating Distributed Compositionality	
1	Motivation: Composition in the Neural Age	10 min
2	Historical & Formal Background	20 min
3	The Computational Landscape: Vectors, Neural Models, Hybrids	30 min
4	Phenomena Resisting Classical Compositionality	15 min
5	Conceptual Synthesis: Why Generative Lexicon	10 min
	PART II – Dual-Aspect Theory & Computational Realization	
6	GL Foundations: Qualia Structure & Event Types	12 min
	Coffee Break	4:00 pm
7	Qualia Event Structure (QES)	15 min
8	Argument Structure: True, Latent	8 min
9	Dynamic Object Modeling (DOM)	8 min
10	Distributed Compositionality: Networks & Controllers	15 min
11	Vector Realization: TPR, HRR, VSA, and Beyond	20 min
12	Applications, Discussion, Q&A	14 min

§1: Motivation: Composition in the Neural Age

Why this tutorial, why now

The Moment We Are In

A striking empirical fact

Large language models produce fluent, contextually appropriate, often truth-preserving language output – without any explicit representation of how meaning is built compositionally (predicate-argument structure, events, types).

Two possible conclusions:

1. *Compositionality was a useful abstraction* that neural systems have now obviated.
2. *Compositionality is still doing the work* – we just cannot yet see how.

This tutorial takes the second view

- Fluency $\neq$ compositional competence
- Systematic failures (SCAN, COGS) tell us what is missing
- A distributed theory of composition is **required**, not optional

The Core Problem

Compositionality (Frege, Montague)

The meaning of a complex expression is a function of the meanings of its parts and the way they are combined.

What this leaves unexplained

- *bake a cake* vs. *bake a potato* – same verb, different event type
- *Mary began the novel* – where did the "reading" event come from?
- *The prisoner escaped* – what makes *prisoner* a stage-level property?
- *Mary wiped the counter dry* – *dry* is not selected by *wipe*

The Fregean principle is *correct but incomplete*. Composition is not a single binary operation; it is a **distributed process** of mutual constraint among lexical items, event structure, and context.

Two Paradigms, One Question

Symbolic / Formal

- Frege, Montague, Partee, Kamp
- Meaning = logical form
- Composition = λ-application
- + Interpretability, entailment
- + Systematic generalization
- – Brittle, hand-built lexicons
- – No gradient phenomena

Distributional / Neural

- Harris, Firth $\rightarrow$ word2vec $\rightarrow$ LLMs
- Meaning = vector / activation
- Composition = learned transformation
- + Coverage, robustness, scale
- + Handles gradient phenomena
- – Opaque
- – No explicit type structure

The two traditions have been separate for three decades. The question this tutorial addresses: **can we have both?**

The Scale of the Empirical Gap

What LLMs get right

Plausibility judgments, selectional preferences, lexical similarity, paraphrase equivalence, contextualized sense disambiguation, analogical completion, some systematic generalization.

What LLMs still get wrong

- **SCAN / COGS:** systematic failures on novel combinations of familiar primitives
- **Compositional inference:** *built a house* $\Rightarrow$ house exists: inferred stochastically, not compositionally
- **State tracking:** procedural reasoning across multiple events
- **Type coercion:** handled by pattern matching, not type-driven recovery
- **Novel combinations:** exponential drop-off with depth of composition

Diagnosis

The mechanism of composition in neural models is **real but invisible**. We have no formal account of what it is doing.

The Interpretability Argument

Why we still need formal semantics

Pavlick (2023), Piantadosi et al (2024), Boleda & Korhonen (2025) each argue in different ways that LLMs have implicit cognitive/semantic structure that is **worth characterizing formally**. Without formal primitives, we lack:

- **A vocabulary** for describing what LLMs encode
- **Diagnostics** that distinguish genuine composition from memorization
- **Probes** that test for specific semantic primitives (events, types)
- **Guarantees** about inferential behavior
- **A theory of transfer** across architectures, tasks, and languages

The formal tradition is not a competitor to neural modeling. It is the **theoretical substrate** that tells us *what the model should compute*.

Three Senses of "Distributed"

Terminology, disambiguated

These are frequently conflated. Keep them separate.

Term	Meaning
Distributed representation	A concept is encoded across many dimensions (a vector) rather than a single unit (localist). Substrate-level claim.
Distributional semantics	Word meaning is approximated by co-occurrence statistics in a corpus. Methodological claim.
Distributed compositionality	Meaning is assembled by parallel interacting constraints across lexical, event, and contextual layers. Architectural claim.

What We Will Build

The theory

- GL Foundations
  - Qualia Structure
  - Event Structure
- Distributed GL
  - Qualia Event Structure (QES)
  - Dynamic Object Model (DOM)
  - Distributed Compositionality (DC)

The realization

- Tensor Product Representations
- Holographic Reduced Representations
- Vector Symbolic Architectures
- Role-filler binding for qualia
- DOM state vectors
- QES as tensor sequences
- Framework comparison: DisCoCat, HRR, FGA

By the end: a worked example of *bake a cake* computed first symbolically, then in HRR, showing **the same mechanism in two substrates**.

A Claim and a Commitment

The claim

Distributed compositionality is **substrate-independent**. The linguistic primitives (qualia, events, object states) and the compositional mechanisms (inner/outer application, controllers) can be realized symbolically, algebraically, or neurally. They are the same theory.

The commitment

We will not ask you to choose between formal and distributional approaches. We will show you a theory that honors both, and an implementation path that connects them.

One caveat

This is an active research program. Not every connection is final. We will flag what is settled, what is in progress, and what is speculative.

§2: Historical & Formal Background

From Frege to dynamic semantics

The Fregean Legacy

Frege (1892, 1918): Sense, Reference, Composition

- Sentence meaning built from word meanings via function application
- Senses compose; references do not always compose (opacity)
- Principle of Compositionality: the *foundational* commitment of formal semantics

The logical form tradition

- Russell, Carnap: truth-conditional semantics via first-order logic
- Tarski: model-theoretic interpretation; truth in a model
- Church: typed λ-calculus as the engine of composition

This sets up the architecture: **syntax assembles a logical form**, **semantics interprets it**, and the bridge is a homomorphism.

Montague's Grand Synthesis

Montague (1970–73)

The definitive formalization: a **homomorphism** from syntactic categories to semantic types. Every syntactic rule has a parallel semantic rule.

$$\text{Synt: } \alpha \cdot \beta \quad \rightarrow \quad \text{Sem: } [\![\alpha]\!]([\![\beta]\!])$$

Types

- e = entity; t = truth value
- $\langle \sigma, \tau \rangle$ = function $\sigma \rightarrow \tau$
- NP: $\langle \langle e, t \rangle, t \rangle$ (generalized quantifier)
- VP: $\langle e, t \rangle$; S: t

Mechanism

- **Composition = function application**
- Lexical items get typed λ-terms
- Possible worlds for intensionality
- Quantifier raising for scope

This is still the backbone of formal semantics. Everything we discuss in this tutorial either extends or complicates this picture.

Type-Shifting Enriches Compositional Mechanisms

Partee and Rooth(1983), Partee (1995)

The insight

The same NP (e.g. *Mary*) can appear in different positions demanding different types. Rather than multiplying lexical entries, **type-shift** the NP.

Type-shifting operators

- `lift`: $e \rightarrow \langle\langle e, t\rangle, t\rangle$
- `lower`: $\langle\langle e, t\rangle, t\rangle \rightarrow e$
- `ident`: $e \rightarrow \langle e, t\rangle$
- `BE`: predicative uses
- `A`: existential closure

Example: *Mary*

- Subject of *runs*: type e
- Conjoined with a quantifier NP: type $\langle\langle e, t\rangle, t\rangle$ via `lift`
- Predicative *is Mary*: type $\langle e, t\rangle$ via `ident`

Foreshadowing

Partee's type-shifting is the **first crack** in strict Montagovian compositionality. If types shift, then meaning isn't just function application – it's function application *plus* repair mechanisms. GL's type coercion is a descendant of this insight.

Dynamic Semantics: Meaning as Update

Kamp (1981), Heim (1982), Groenendijk & Stokhof (1991)

The shift

Sentence meaning is not a truth value but a **context-change potential**: a function from input context to output context.

$[\![S]\!] : c_{\text{in}} \rightarrow c_{\text{out}}$

Representative systems

- DRT (Kamp): sentences build Discourse Representation Structures
- File Change Semantics (Heim): sentences update "files"
- DPL (Groenendijk & Stokhof): dynamic predicate logic

Why this matters for DC

- Meaning has **state**
- Composition is a **sequence of updates**
- **Object persistence** becomes a first-class concept
- DC inherits this architecture directly

Dynamic semantics is where meaning-as-process first replaces meaning-as-value. DC generalizes this from discourse to all of composition.

Frame Semantics and Situation-Based Approaches

Fillmore (1982): Frame Semantics

Word meanings are not atomic but evoke structured **frames** – rich schematic background knowledge. Understanding *buy* requires the Commercial Transaction frame (buyer, seller, goods, money).

Related traditions

- Fillmore: frames, FrameNet
- Barsalou: perceptual symbols
- Jackendoff: conceptual semantics
- Talmy: cognitive semantics
- Croft & Cruse: Construction Grammar

Shared insight

- Words carry **structured background**
- Composition **activates** parts of that background
- Meaning is **context-sensitive** at the lexical level itself

GL's Qualia Structure is in this lineage: word meanings are not atomic symbols but **structured resources** that composition selectively activates.

Generative Lexicon: A Reference Point

The move that anticipated modern type-theoretic semantics

Before introducing contemporary typed frameworks, a brief orientation. The **Generative Lexicon** (Pustejovsky 1991, 1995) is the earliest proposal to treat lexical items as carrying *structured type content*, not atomic labels.

GL's distinctive claim

- Lexical types are **constructed** from relational *qualia*: Formal (what it is), Constitutive (what it is made of), Telic (what it is for), Agentive (how it came to be).

What GL introduced first

- Type coercion (aspectual, complement)
- Dot types for logical polysemy
- Co-composition and selective binding
- Qualia-enriched lexical entries

A type *is* its qualia configuration, not an atom labeled by one. Composition operates over qualia roles, not just truth-conditional content

Type Constructors $\otimes_Q$ and •

In GL richer types are built up out of simpler type structures

Assuming the formal role to be the basic type from which all other types are constructed, we can introduce types containing other qualia with a type constructor, **tensor**, symbolized as $\otimes$.

- The tensor adds constitutive ($\otimes_C$), agentive ($\otimes_A$) and telic ($\otimes_T$) relations to the head formal type.
- A cake is not merely a physical object but a physical object enriched with Agentive and Telic specifications, $\texttt{phys} \otimes_A \texttt{bake} \otimes_T \texttt{eat}$.

The type-constructor • reifies two elements into a new complex type. The constituents of a complex type pick up distinct aspects of the object (for instance *lunch* picks up *event* • *food* etc.).

The Contemporary Type-Theoretic Landscape

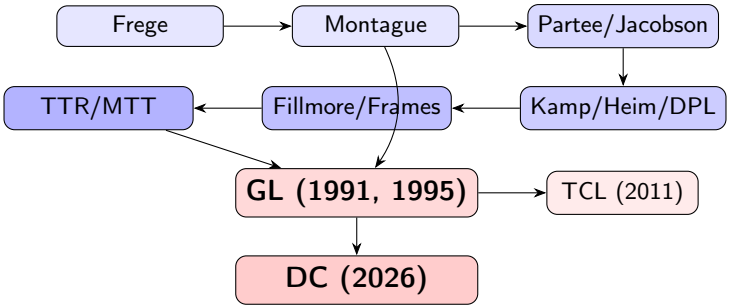
In dialogue with GL

Since the 2000s, several typed frameworks have taken up GL's program of enriching semantic types – some formalizing GL's mechanisms, some extending them in new directions.

Framework	Core move (relative to GL)
TTR (Cooper 2005, 2023)	Types as *records*; situations as record-typed entities. Compatible with qualia as record fields.
MTT (Luo 2012, 2020)	Modern Type Theory with *coercive subtyping*. Common nouns as types; adjectives as subtypes.
System F (Retoré 2014)	*Polymorphic* types for lexical flexibility; second-order quantification.
TCL (Asher 2011)	*Type Composition Logic*: a formal proof-theoretic meta-theory for GL's coercion, dot types, and predication.

Each framework adds structure *to* types. GL's distinctive move – building types *from* qualia relations – remains the underlying theoretical commitment these frameworks formalize or extend.

GL as Synthesis: A Historical Position



- Each step **enriched the type system** or **loosened the composition rule**
- GL makes the distinctive move: **types constructed from qualia relations**
- TCL formalizes this; DC extends it to a distributed substrate

End of §2: What to Take Forward

Summary

1. Compositionality is a **principle**, realized by many architectures – each enriching the type system or loosening the composition rule.
2. Contemporary type-theoretic semantics (TTR, MTT, System F, TCL) all converge on **richer types**. GL goes further: types are **constructed from qualia relations**.
3. The relational structure of qualia is what makes GL **implementable in distributed substrates** – the claim §11 makes precise.
4. Outstanding gaps: no substrate story yet, no learning story yet, no account of gradient phenomena. Those motivate the next sections.

Next

§3 surveys the computational landscape – what distributional methods compute, where they succeed, and where they fail.

§3: The Computational Landscape

Vectors, neural models, hybrids – what they compute and what they miss

The Distributional Hypothesis

Harris (1954), Firth (1957)

"You shall know a word by the company it keeps." Word meaning is approximated by aggregated co-occurrence statistics across large corpora.

The basic empirical finding

- Words appearing in similar contexts have similar meanings
- Context can be window-based, syntactic, or deep
- Scales to billions of tokens

What this gives us

- Similarity by cosine distance
- Analogies by vector arithmetic
- Soft clustering
- Robust handling of rare words

The hypothesis is descriptive and holds empirically. The question is whether it is also **explanatorily** adequate for composition.

The Progression of Distributional Models

Era	Model	Captures	Misses
1990s	LSA, HAL	Topical similarity	Syntax, composition
2003	Bengio LM	Predictive density	Limited context
2013–14	word2vec, GloVe	Analogy, static senses	Polysemy, context
2018	ELMo, BERT	Contextual tokens	Role structure, types
2020+	GPT, Claude	Emergent behavior	Explicit type/event str

- Each step adds **context sensitivity**
- None explicitly represents qualia, events, or types
- Type structure is **implicit**, probed not declared

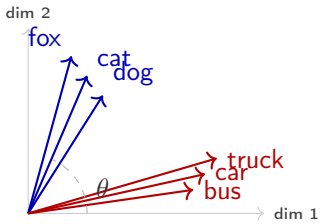
The trajectory is clear: **more context, deeper structure, less transparency**. The structure is there – we just have no formal vocabulary for it.

Words as Vectors: The Linear-Algebraic Setting

Harris (1954); Firth (1957); Turney & Pantel (2010)

Definition

Each word w is mapped to a vector $\vec{w} \in \mathbb{R}^d$. **Distributional similarity** becomes *geometric proximity*, measured by $\cos\theta = \frac{\vec{u}\cdot\vec{v}}{\|\vec{u}\|\,\|\vec{v}\|}$.



Words used alike ⇒ small angle ⇒ high $\cos\theta$

What we gain

- Continuous similarity in $[-1, 1]$
- Generalization to rare words
- Linear algebra is now available

What it costs

- Polysemy flattened (*bank*)
- No order, no roles – a bag
- Composition must be **built**

LSA → word2vec → contextual embeddings: the space gets richer, but the algebra stays *linear*.

Static Word Embeddings

word2vec, GloVe, fastText

A word w is a vector $\vec{v}_w \in \mathbb{R}^d$ learned from co-occurrence. Training objectives: skip-gram, CBOW, ratio-of-ratios.

The famous analogies

$$\vec{v}_{\text{king}} - \vec{v}_{\text{man}} + \vec{v}_{\text{woman}} \approx \vec{v}_{\text{queen}}$$

$$\vec{v}_{\text{Paris}} - \vec{v}_{\text{France}} + \vec{v}_{\text{Italy}} \approx \vec{v}_{\text{Rome}}$$

Fundamental limits

- One vector per word – no polysemy
- No argument structure
- No event decomposition
- **Composition unsolved** at this level

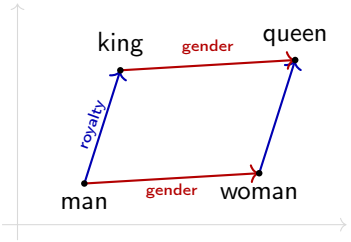
Static embeddings encode *something real* about word meaning. But they are a **starting point**, not a semantic theory.

The Parallelogram of Analogy

Mikolov et al. (2013); Levy & Goldberg (2014)

Definition

Analogy $a : b :: c : d$ is solved by *vector arithmetic*: $\vec{d} \approx \vec{b} - \vec{a} + \vec{c}$. The classic case: $\vec{queen} \approx \vec{king} - \vec{man} + \vec{woman}$.



Equal sides ⇒ same direction ⇒ same relation

Why it matters

- **Gender** as a *direction*, not a feature
- Capitals, tenses, plurals – all parallelograms

What's actually happening

- $\vec{king} - \vec{man}$ = "royal" direction
- Add it to $\vec{woman}$ ⇒ near $\vec{queen}$
- *Linearity* of the space does the work

Caveat: the parallelogram is a *single-word* phenomenon. Composing **phrases** requires more than addition – as we discuss next.

Algebraic Composition in Vector Spaces

The composition question

If *red* is $\vec{r}$ and *car* is $\vec{c}$, what is *red car*?

Approach	Operation	Failure mode
Additive (Mitchell & Lapata)	$\vec{r} + \vec{c}$	Commutative, order-blind
Multiplicative	$\vec{r} \odot \vec{c}$ (Hadamard)	No role differentiation
Matrix-based (Baroni & Zamparelli)	$M_{\text{red}} \cdot \vec{c}$	Parameter blowup
Tensor product (Smolensky)	$\vec{r} \otimes \vec{c}$	Dimension grows
Circular conv. (HRR)	$\vec{r} \circledast \vec{c}$	Noise, approximate

Each operation trades off **exactness**, **capacity**, and **dimensional stability**. We will return to these tradeoffs in §11.

Composition as Addition: Bag-of-Words Geometry

Mitchell & Lapata (2008, 2010)

Definition

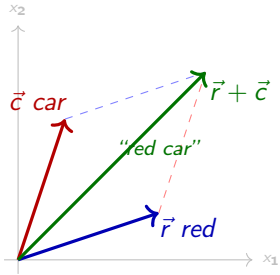
$(\vec{r} + \vec{c})_i = r_i + c_i$. The phrase vector lives in the *same* space as the words.

What it captures

- Same-dimension output ($\mathbb{R}^d \to \mathbb{R}^d$)
- Cheap, parameter-free
- Topical similarity preserved

What it loses

- **Commutative**: $\vec{r} + \vec{c} = \vec{c} + \vec{r}$
- No order, no roles, no asymmetry
- *"dog bites man"* = *"man bites dog"*



Both diagonals coincide ⇒ word order erased.

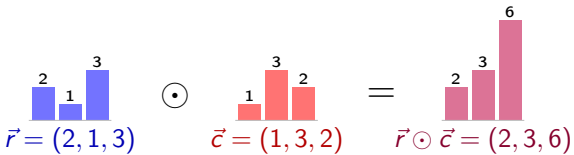
Geometrically, the **parallelogram law**: composition becomes a feature average.

Composition as Multiplication: Element-wise Filtering

Mitchell & Lapata (2008); Erk & Padó (2008)

Definition

$(\vec{r} \odot \vec{c})_i = r_i \cdot c_i$. Output is again in $\mathbb{R}^d$.



What it captures

- "AND-like" filtering – output high only where *both* inputs are high
- Same dimension; cheap

What it loses

- **Still commutative**; no role differentiation
- Zeros annihilate inputs

Hadamard intersects features without naming them: near-orthogonal words yield near-zero composition.

Matrix-based Composition: Adjectives as Functions

Baroni & Zamparelli (2010); Socher et al. (2012)

Definition

Each modifier is a *matrix*: $M_{\text{red}} \in \mathbb{R}^{d \times d}$. Composition: $\vec{p} = M_{\text{red}}\,\vec{c}$. The adjective is a **function**.

$$\underbrace{\begin{bmatrix} 1.0 & 0.5 & 0.0 \\ 0.0 & 1.0 & 0.0 \\ 0.2 & 0.0 & 0.8 \end{bmatrix}}_{M_{\text{red}}\ (d \times d)} \cdot \underbrace{\begin{bmatrix} 1 \\ 3 \\ 2 \end{bmatrix}}_{\vec{c}} = \underbrace{\begin{bmatrix} 2.5 \\ 3.0 \\ 1.8 \end{bmatrix}}_{\vec{p}}$$

red reshapes *car*: each output coord is a learned linear combination of the input.

What it captures

- **Asymmetry**: modifier $\neq$ head
- Order matters: $M_{\text{red}}\vec{c} \neq M_{\text{car}}\vec{r}$
- Adjective semantics learned per-lexeme

What it costs

- **Parameter blowup**: d^2 per adjective ($\sim 9 \times 10^4$ at d=300)
- Higher arity needs rank-3+ tensors

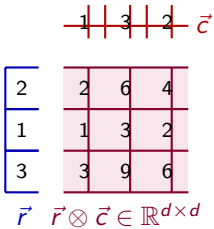
First step toward **categorial composition**: lexical category fixes algebraic shape.

Tensor Product: Roles and Fillers as Outer Products

Smolensky (1990); Smolensky & Legendre (2006)

Definition

The outer product $\vec{r} \otimes \vec{c} \in \mathbb{R}^{d \times d}$ has entries $(\vec{r} \otimes \vec{c})_{ij} = r_i \cdot c_j$.



Cell $(i, j) = r_i \cdot c_j$. Encodes *every pairwise interaction*.

What it captures

- **Exact** and **invertible**: recover $\vec{c}$ from $\vec{r} \otimes \vec{c}$ by inner product
- Order-sensitive: $\vec{r} \otimes \vec{c} = (\vec{c} \otimes \vec{r})^\top$
- **Role/filler binding**: $subj \otimes dog \neq obj \otimes dog$

What it costs

- **Dimension grows**: $d \otimes d = d^2$; k-ary $= d^k$

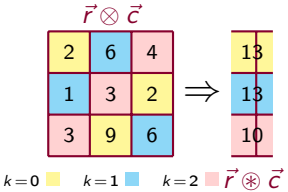
Smolensky's TPR: gold standard for exact symbolic binding – at the price of dimensional growth.

Circular Convolution: Compressing the Tensor Product

Plate (1995); Holographic Reduced Representations

Definition

$\circledast$ folds $\vec{r} \otimes \vec{c}$ back into $\mathbb{R}^d$ by summing along its **wrap-around anti-diagonals**: $(\vec{r} \circledast \vec{c})_k = \sum_{i+j \equiv k \,(\bmod\, d)} r_i \, c_j$.



Each color marks one wrap-around anti-diagonal; their sums are the coordinates of the output.

What we gain

- **Fixed dimension**: still $\mathbb{R}^d$
- Approximate retrieval: $\vec{r}^{\dagger} \circledast (\vec{r} \circledast \vec{c}) \approx \vec{c}$
- FFT computation in $O(d \log d)$

What we pay

- Retrieval is **lossy** – needs clean-up memory
- No type discipline; binding is opportunistic

Circular convolution = tensor product, *folded*. HRR is the lossy-but-tractable sibling of TPR.

DisCoCat: Categorical Composition

Coecke, Sadrzadeh, Clark (2010)

The program

Borrow **type grammar** from Lambek's pregroups and tensor semantics from quantum mechanics. Composition = tensor contraction guided by syntactic type.

Architecture

- Nouns live in N
- Transitive verbs live in $N^r \otimes S \otimes N^l$
- Syntactic composition contracts adjacent type pairs

Strengths

- Mathematically elegant
- Compositional by design
- Maps cleanly to quantum circuits
- Categorical framework

Limits

Works in controlled settings. Data-hungry for verb tensors. No qualia. No event structure. No object dynamics.

Contextualized Representations: BERT and After

The shift

Token representations are **contextualized**: the vector for *bank* differs in *river bank* vs. *bank account*. Meaning is a function of position in a sequence, not a lookup.

What this solves

- Polysemy at the token level
- Sensitivity to local context
- Sentence-level similarity
- Transfer across tasks

What this does not solve

- Explicit predicate-argument structure
- Event decomposition
- Object persistence across sentences
- Inferential transparency

Contextualization is **distributional sense disambiguation at scale**. It is not compositional semantics.

LLMs as Cognitive Architectures

Pavlick (2023), Piantadosi et al (2024)

The provocation

LLMs behave, at scale, as if they had structured semantic representations – even though the architecture contains no explicit structure beyond attention patterns and feedforward layers.

What the evidence shows

- Probing finds structure: POS, syntax, roles
- Emergent few-shot generalization
- Robust pragmatic inference
- Cross-task transfer

What remains unexplained

- What *exactly* is encoded?
- Which primitives are universal?
- How is **composition performed**?
- Why do specific failures persist?

Pavlick's position: semantic structure *is there*. We just need the theoretical vocabulary to describe it. That vocabulary is what GL provides.

Diagnostic 1: Compositional Generalization (SCAN)

Lake & Baroni (2018)

The test

Train on simple sequences of commands (*jump*, *walk twice*). Test on novel combinations using the same primitives (*jump twice and walk*).

The finding

- Standard seq2seq: near-zero accuracy on held-out combinations
- Performance scales **non-linearly** with training size
- Systematic **compositional gaps**
- Not resolved by scale alone

What's missing

- Separation of content and combinator
- Explicit representation of "once more with X"
- Type-driven slot filling

The SCAN failures are **diagnostic**: the model has not learned a general combinator, only patterns of combinations seen in training.

Diagnostic 2: COGS and Structural Generalization

Kim & Linzen (2020), Hupkes et al. (2023)

COGS: Compositional Generalization Dataset

Test structural generalization: novel role assignments, novel depths of embedding, novel combinations of verb classes and argument types.

Representative gaps

- Subject → Object role swap: fails
- Deeper recursion than in training: degrades
- Novel verb-object pairs with familiar types: underperforms

Diagnosis

- Types are not explicit
- Roles are not first-class
- Structural abstraction is **shallow**

COGS shows what neural models are missing in terms **closely aligned with classical formal semantics**: argument types, roles, scope.

Diagnostic 3: Polysemy and Coercion

The probe

Show that contextualized models handle *frequent* polysemy (e.g. *bank*) but fail on *structural* polysemy and coercion that requires qualia reasoning.

What works

- *bank*: river vs. financial
- *bat*: animal vs. equipment
- Distributionally frequent senses

What breaks

- *enjoyed the book* → reading
- *began the novel* → reading/writing
- *a quick lunch* → event vs. object
- Novel dot-type combinations

The failures cluster around **Telic coercion** – exactly the mechanism GL was designed to handle. Models succeed when the sense is distributionally frequent and fail when it requires **structural reasoning over qualia**.

Diagnostic 4: Event Entailment

The probe

Given *John built a house*, does a model infer that *a house exists*? Given *John was building a house*, does a model infer that *the house does not yet exist*?

Findings

- Accomplishment $\Rightarrow$ existence: high accuracy
- Progressive $\not\Rightarrow$ existence: **much lower**
- Inferences are stochastic, not compositional
- Performance drops on rare verbs

Missing mechanism

- Explicit culmination node (e_2)
- Result-state predicate tied to Formal quale
- Aspect-dependent entailment

Event entailment is a **structural property** of accomplishment predicates. Neural models approximate it stochastically; GL encodes it explicitly.

Diagnostic 5: State Tracking

Bosselut & Choi (2022), Tandon et al. (2020)

The probe

Given a procedural text (*Mary put water in the pot. She lit the stove. After ten minutes the water boiled.*), track the state of the water at each step.

Findings

- Short sequences: acceptable
- Long sequences: **drift**
- Multi-object tracking: degrades fast
- Implicit state changes: poorly recovered

Missing mechanism

- First-class object state records
- Event-to-object update functions
- Persistence over discourse

This is precisely what **DOM** provides. LLMs can be augmented with DOM-style state vectors to stabilize procedural reasoning.

Hybrid Neurosymbolic Architectures

The proposal

Combine neural representations with explicit symbolic reasoning. Neural for perception, association, robustness; symbolic for inference, structure, guarantees.

System	Contribution
Neural Module Networks (Andreas)	Task-specific networks composed by parser
Neurosymbolic Concept Learner (Mao)	Joint visual + symbolic concept learning
Tensor Product Networks (Smolensky)	Exact role-filler binding
Vector Symbolic Architectures (Kleyko)	Fixed-dim algebraic representations
DeepProbLog, Logic Tensor Nets	Probabilistic logic + neural

These systems show that distribution and structure can coexist. What's typically missing is a **linguistically grounded** theory of what the symbolic component should represent.

Vector-Symbolic Architectures (VSA)

Plate (1995), Kanerva (2009), Kleyko et al. (2023)

Core commitments

Represent structured data as **fixed-dimensional vectors**, with algebraic operations for binding (role-filler) and superposition (combination).

Key operations

- **Bind**: $\vec{r} \circledast \vec{f}$ (circular convolution, MAP, etc.)
- **Unbind**: approximate inverse
- **Bundle**: $\vec{v}_1 + \vec{v}_2 + \ldots$
- **Protect**: permutation-based

Variants

- HRR (Plate): circular convolution
- Binary Spatter Code (Kanerva)
- MAP (Gayler): component-wise mult
- FHRR: Fourier HRR

VSA gives us a principled bridge from symbolic structure to distributed representation. §11 builds on this framework.

Structured Distributional Semantics

Baroni et al. (2014), Boleda & Korhonen (2025)

The middle path

Keep vector representations, but **impose linguistic structure**. Vectors get typed; composition respects types; specific operations align with syntactic relations.

Framework	Structural commitment
Frege in space (Baroni 2014)	Typed operators for comp distributional
Formal Distr Semantics (Boleda 2016)	Model-theoretic grounding
DisCoCat	Pregroup type system
SRNN (Socher)	Recursive syntax-driven composition
BabelNet + embeddings	Sense-disambiguated vectors

Structured DS gets part-way there. What's missing: **qualia roles**, **event decomposition**, **object dynamics** – the content that GL supplies.(EJ: Moreover, if faces scaling challenges because of high tensor order,

Multimodal Grounding

The enrichment

Couple linguistic representations with perceptual signals : images, video, action sequences, robotic sensorimotor data.

Representative work

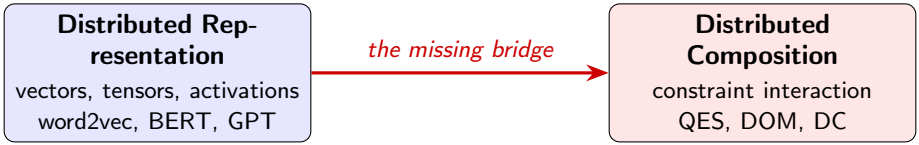
- CLIP, Flamingo, Gemini: image-text
- Alikhani et al.: gesture-language
- VoxML (Pustejovsky & Krishnaswamy)
- Embodied dialogue systems

What grounding buys you

- Situated reference
- Affordances for objects
- Event grounding in action
- Anchor for shared common ground

Multimodal grounding *provides* the perceptual data that GL's qualia and DOM describe structurally. The two traditions are **complementary**.

The Representation / Composition Gap



G1 Distributed *representation* is mature: we know how to encode meaning in vectors at scale.

G2 Distributed *composition* is not: we lack a principled theory of how those vectors combine.

G3 The gap is **linguistic**, not computational: we know how to train networks. We do not know *what they should compute*.

Five Failure Patterns

Failure	Symptom	What's missing
Polysemy collapse	*bake a cake* / *bake a potato* share embedding	Qualia-driven verb specialization
Event opacity	Accomplishment entailment stochastic	Explicit culmination + result state
State amnesia	*Mary sliced the bread*: slices lost	Object lifecycle tracking (DOM)
Coercion failure	*enjoyed the novel* $\not\rightarrow$ reading	Qualia-based type shifting
Structural rigidity	Novel constructions underperform	Multi-constituent licensing

Each failure corresponds to a specific GL mechanism to be introduced later

What the Two Traditions Need From Each Other

Formal tradition needs

- Scale to open vocabulary
- Gradient phenomena (plausibility, acceptability)
- A learning story
- A substrate story
- Empirical robustness

Distributional tradition needs

- Explicit types
- Explicit event structure
- Explicit object persistence
- Inferential transparency
- A vocabulary for what emerges

Each tradition has what the other lacks. DC is a framework designed around this complementarity – not by choosing one or the other but by **specifying a theory that both can implement**.

End of Computational Landscape

The state of play

1. Distributional models encode *something real* about meaning, at scale.
2. Composition remains the hard problem: algebraic, categorical, neural attempts all partial.
3. Systematic failures cluster around **missing linguistic structure**: qualia, events, object state.
4. Hybrid neurosymbolic systems point the way, but lack linguistic grounding.
5. VSA gives us an algebraic substrate; structured DS gives us type discipline.

Next

Part I's positive phenomena: the cases where classical compositionality visibly breaks down. These motivate the specific mechanisms GL provides.

§4: Phenomena Resisting Classical Compositionality

The empirical case that motivates a richer theory

The Fregean Assumption Revisited

Principle of Compositionality

The meaning of a complex expression is a function of the meanings of its parts and the way they are syntactically combined.

- Semantic interpretation is a homomorphism from syntax to semantics
- The operative mechanism: $f(a)$ where $f : \tau_1 \rightarrow \tau_2$ and $a : \tau_1$
- Works for predicates and their explicit arguments
- **Systematically breaks down** in linguistically revealing ways

The claim

Classical compositionality is *too narrow*. It cannot account for polysemy, type mismatches, light verb constructions, stage-level predicates, implicit arguments, or construction-level meaning – without additional machinery.

A Catalog of Resistant Phenomena

Phenomenon	Example	Problem
Inherent polysemy	*Mary left after lunch*	*lunch* = event or food?
Type mismatch	*Mary left after her coffee*	NP shifts to temporal meaning
Light verbs	*take a pill / take a train*	verb meaning shaped by NP
Stage-level preds	*The prisoner escaped*	culmination negates a property
Measure phrases	*John ran three miles*	distance not a V argument
Resultatives	*Mary wiped the counter dry*	AP not selected by verb
Way construction	*John laughed his way in*	idiomatic path from nothing
Aspectual coercion	*John sneezed all afternoon*	punctual → iterative
Co-predication	*The book is expensive and boring*	physical + informational
Implicit arguments	*The boss fired* (*whom)	obligatorily expressed

In each case, **multiple constituents contribute simultaneously** to interpretation. Direction of composition is not fixed.

Stage-Level Change Predicates

Stage-level (contingent)

- ✓ *The prisoner escaped*
  ⇒ no longer a prisoner
- ✓ *Mary fixed the leaky faucet*
  ⇒ faucet no longer leaky
- ✓ *The balloon deflated*
  ⇒ no longer inflated

Individual-level (essential)

- × *The tall man escaped*
  ⇏ no longer tall
- × *The mathematician escaped*
  ⇏ not a mathematician
- × *Mary fixed the blue faucet*
  ⇏ faucet not blue

Stage-Level Generalization

A stage-level predicate identifies a property ϕ of its argument that is **negated** at the culmination of the event – but *only* when ϕ is contingent (stage-level), not essential (individual-level).

Polysemy: The Central Challenge

The same word, different types

Polysemy is the **signature phenomenon** that distinguishes GL from classical compositional approaches. Classical semantics posits multiple lexical entries; GL posits one entry with **multiple accessible facets**.

Logical polysemy (dot types)

- *book*: physical object • informational content
- *school*: institution • building • event
- *newspaper*: publisher • article • physical copy
- *lunch*: food • event

Co-predication diagnostics

- ✓ *The book is heavy and interesting*
- ✓ *Lunch was delicious and lasted three hours*
- ✓ *The newspaper was sued and then burned*

Co-predication shows that **both facets remain accessible** after composition. Multiple entries cannot predict this; dot types can.

Type Coercion

The phenomenon

A predicate selects for one type; the argument appears in another. The mismatch is resolved by **implicit type-shifting** guided by the argument's lexical structure.

Aspectual coercion

- *Mary began the novel*
  → began reading the novel
- *John enjoyed the film*
  → enjoyed watching it
- *She finished the book*
  → finished reading it

The mechanism

- Predicate expects event
- Argument is object type
- **Telic quale** supplies event
- *novel*'s Telic = `read`
- Coerced reading recovered

Coercion is **qualia-driven**. This is one of GL's most predictive contributions: the mechanism specifies *which* event is recovered, not just that something is.

Co-Compositionality: *bake a cake* vs *bake a potato*

The puzzle

The same verb *bake* produces two different event types depending on the argument. With *a cake*: creation. With *a potato*: change of state.

Creation reading (*bake a cake*)

- Before: no cake exists
- After: a cake exists
- Material $\neq$ result type
- **cake**'s Agentive = `bake`

Change-of-state (*bake a potato*)

- Before: raw potato
- After: baked potato
- Same type, changed state
- **potato** has no baking Agentive

The NP's qualia **shape the verb's event type**. Composition goes *both ways*: not just $V(NP)$ but mutual specialization.

Implicit Arguments

Participants without syntactic realization

Some arguments play a role in interpretation but cannot or do not surface.

Type	**Example**	**Status**
Default	*John built the house [out of bricks]*	Optional, surfacing allowed
Shadow	*She phoned [*on the phone]*	Incorporated, blocked
Hidden	*She photographed the scene* $\rightarrow$ *photos*	Never surfaces, accessible
Bridging	*Mary sliced the bread* $\rightarrow$ *slices*	Inferred from event
Covert goal	*John climbed* [up]	Path in verb semantics

Classical compositionality cannot say where these participants come from. GL's argument structure types, extended with **Generalized Result Roles** (GRR), derive them systematically.

Resultatives and Constructional Meaning

The phenomenon

In *Mary wiped the counter dry*, the AP *dry* is not selected by *wipe*. The **construction itself** contributes an accomplishment template.

More examples

- *John ran himself exhausted*
- *She painted the door red*
- *He shouted himself hoarse*
- *The kids laughed the speaker off the stage*

The structural contribution

$$[e_1 : \text{P-process}] \xrightarrow{V} [e_2 : \text{AP-state}]$$

- Verb gives process
- AP gives result state
- **Construction gives the accomplishment skeleton**

Construction Grammar insight: meaning is not only in words but in **patterns of combination**. DC accommodates this directly.

Scope, Binding, and Long-Distance Dependencies

Not all compositional puzzles are lexical

Some challenges arise from the interaction of **quantifier scope**, **binding**, and **non-local dependency**.

Examples

- *Every student read a book* – scope ambiguity
- *John thinks he is smart* – binding possibilities
- *Which book did Mary say John read t* – long-distance filler

What's required

- Storage and retrieval mechanisms (Cooper storage)
- Variable binding across distance
- Coherent discourse referents

These phenomena are largely *orthogonal* to lexical compositional effects, but any complete theory must handle both. DC's network view naturally accommodates long-distance constraints.

Gradience and Selectional Preferences

Beyond binary compositionality

Classical semantics treats composition as succeed-or-fail. Actual linguistic behavior is graded: some compositions are *better* than others without being outright ungrammatical.

Graded acceptability

- *The dog barked* (natural)
- *The idea barked* (metaphorical)
- *The triangle barked* (coerced)
- *Sincerity barked* (bizarre but parseable)

What grades this?

- Selectional preferences (Resnik)
- Coercion cost
- Plausibility
- Genre and context

Distributional models capture gradience natively. Symbolic models struggle. DC needs to accommodate both **discrete structure** and **graded preference**.

Five Principles for a Richer Theory of Compositionality

T1 Multiple constituents contribute in parallel.
Verbs, arguments, modifiers, and implicit roles simultaneously shape interpretation.

T2 Direction of function application is not fixed.
Sometimes the verb specifies the noun; sometimes the noun specializes the verb.

T3 Lexical items carry latent semantic resources.
Qualia roles, event structure fragments, and affordances are selectively activated.

T4 Composition is constraint-driven, not strictly hierarchical.
Interpretation emerges from interacting constraints, not a fixed bottom-up traversal.

T5 Composition admits gradient phenomena.
Acceptability, plausibility, and coercion cost are part of composition, not external filters.

End of Phenomena Survey

What we have established

1. Classical compositionality is **correct in outline** but incomplete.
2. Polysemy, coercion, co-composition, and constructions all require **richer lexical content** and **more flexible composition**.
3. Implicit arguments and gradient phenomena require mechanisms that classical approaches lack.
4. The theory we need: **distributed** constraint satisfaction across lexical, event, and contextual layers.

Next

The conceptual synthesis: we argue that GL provides exactly the mechanisms these phenomena demand – and that those mechanisms are **substrate-independent**, implementable symbolically or distributionally.

§5: Conceptual Synthesis

Why Generative Lexicon

What a Compositional Theory Needs

From Part I, a set of requirements

1. Rich typed lexical entries (Partee, Cooper, Luo, Asher)
2. Type-shifting and coercion as legitimate operations (Partee, Asher)
3. Event structure as first-class (Kamp, Heim, dynamic semantics)
4. Object persistence across composition (DRT, File Change Semantics)
5. Multiple simultaneous constraints (Frame Semantics, Construction Grammar)
6. Gradient acceptability (distributional models)
7. Substrate realizable (neural, algebraic, or symbolic)

The claim

No existing framework does all seven. GL, extended with QES and DOM, and framed as distributed compositionality, does.

GL as Synthesis

Requirement	GL mechanism/component
Rich lexical entries	Qualia Structure (F, C, T, A)
Type-shifting, coercion	Qualia-driven coercion operators
Event structure	Dynamic Event Structure, QES
Object persistence	Dynamic Object Model (DOM)
Multiple simultaneous constraints	Composition networks with controllers
Gradient acceptability	Constraint satisfaction with preference weighting
Substrate realizable	QES/DOM as tensor structures; HRR role-binding

GL is not a new theory in 2026 – it dates to 1995. What is new is: (a) the QES and DOM extensions, (b) the explicit articulation as *distributed* compositionality, and (c) the account of how this framework interfaces with vector-based representations.

Dual-Aspect Semantics

The core architecture

Meaning has **two coupled aspects**: a *situational* aspect (what happens: events, processes, transitions) and an *object* aspect (what endures: entities, their properties, their evolving states).

- Events **modify** objects (DOM records the updates)
- Object states **condition** which events can apply
- Composition is **mutual update** between the two aspects

Distributed Compositionality as Equilibrium

The process model

Rather than a fixed bottom-up traversal, DC treats composition as **iterative adjustment** among multiple interacting constraints. Interpretation is the fixed point of this process.

What interacts

- Verb's event template
- Argument's qualia structure
- Modifier constraints
- Constructional frames
- Discourse context

How they interact

- Mutual type specialization
- Qualia role activation
- Constraint propagation
- Controller licensing
- Convergence to stable reading

This process view is **substrate-independent**. It can be implemented as symbolic constraint satisfaction or as iterative updates in vector space. Part II makes this precise.

A Roadmap for Part II

1. §6 – Classic GL foundations: Qualia Structure and Dynamic Event Structure
2. §7 – Qualia Event Structure (QES): integrating qualia into event representation
3. §8 – Argument Structure types and Generalized Result Roles
4. §9 – Dynamic Object Modeling (DOM): resource-result structures
5. §10 – Distributed Compositionality: composition networks, inner/outer application, controllers
6. §11 – Vector realization: TPR, HRR, VSA, and toward exact algebraic binding
7. §12 – Applications and open questions

Parts §§6–10 build the symbolic theory; §11 shows how the same theory implements in distributed substrates; §12 connects back to the tasks surveyed in Part I.

Bridge to Part II

What to keep in mind as we proceed

1. Every mechanism we introduce is motivated by a **specific empirical phenomenon** from Part I.
2. The symbolic formulation is not the final form – it is a **specification** that can also be implemented in vectors.
3. The computational payoff (§11) depends on the symbolic theory being **exactly right** about what the mechanism does.
4. At the end, we will return to the diagnostic tasks (§3) and show how GL-informed DC addresses each one.

Break

A short break is recommended before Part II.

§6: GL Foundations: Qualia Structure & Event Types

Pustejovsky (1995/2013), Jezek & Pustejovsky (2018), Pustejovsky & Batiukova (2019)

The Classic GL Architecture

Argument Structure (AS)

Specifies number, type, and syntactic realization of a predicate's arguments. Distinguishes *true*, *default*, *shadow*, and *hidden* argument types.

Qualia Structure (QS)

Decomposes lexical meaning into four qualia roles: **Formal** (what it is), **Constitutive** (what it is made of), **Telic** (what it is for), **Agentive** (how it came to be).

Event Structure (ES)

Defines the temporal constitution of the situation. Identifies subevents, their ordering, and prominence. Provides event templates: state, achievement, process, accomplishment.

Lexical Type Structure

Captures hierarchical and relational aspects: type inheritance, dot types, type coercion, within a polymorphic type system.

The Four Qualia Roles

Role	Aristotelian	What it encodes
Formal ($\varepsilon\hat{\imath}\delta o\varsigma$)	formal cause	Basic type; taxonomic position; salient properties
Constitutive ($\upsilon\lambda\eta$)	material cause	Parts, constitution, material; mass/count distinction
Telic ($\tau\acute{\varepsilon}\lambda o\varsigma$)	final cause	Intended use or function; purpose
Agentive ($\alpha\acute{\imath}\tau\iota o\nu$)	efficient cause	Origin, mode of creation; natural vs. artifactual

```
bread
  F = food
  C = grain-product
  T = eat(x)
  A = bake(e, y, x)
```

```
knife
  F = physical-object
  C = {blade, handle}
  T = cut(e, x, z)
  A = manufacture(e, y, x)
```

Telic Role: Driving Modifier Interpretation

Telic selectors

Adjectives that pick out purpose-relevant properties:

- *fast food* (to eat)
- *a slow oven* (to cook)
- *a short novel* (to read)
- *a complex question* (to answer)
- *a good doctor* (to heal)
- *an effective antibiotic* (to cure)

Implicit complement recovery

- *This is a difficult problem.* → (to solve)
- *This is a difficult question.* → (to answer)

Telic in compounds

Head's telic provides the semantic relation:

- *book-shop* (selling)
- *wine glass* (holding)
- *school bus* (transporting)
- *tooth brush* (cleaning)
- *hair brush* (combing)

Note

tooth brush vs. hair brush: same head, different telic – same surface pattern, different semantic relation to the modifier.

Agentive Role and the Informativity Constraint

Informativity (Batiukova & Pustejovsky 2013)

Modification is anomalous when the modifier merely **repeats information already encoded** in the head noun's Agentive qualia.

Redundant – Agentive repeat

* *baked bread* (A = bake)
* *a built house* (A = build)
* *a written book* (A = write)

cf. * *a male bachelor*
* *a female woman*

Well-formed – extra information

✓ *freshly baked bread*
✓ *a well-built house*
✓ *a beautifully written book*

Additional dimension (manner, degree, evaluation) beyond the bare agentive value is required to license the modification.

Constitutive Role: Parts, Matter, Mass/Count

Mass nouns: Formal = Constitutive

```
water
  F / C = liquid
  A = nil (no origin)
```

- Homogeneous: no distinct part-type
- $F = C$ predicts unbounded reference

Count nouns: Formal ≠ Constitutive

```
room
  F = space
  C = {walls, floor, ceiling,...}
  C_l = building
```

```
plastic bag
  F = bag
  C = plastic
```

```
grocery bag
  F = bag
  C = grocery
```

Constitutive in compounds: the non-head fills the C role.

Qualia specific constructions

Formal

NP *such as* NP: *events such as lectures, walks, tours and meetings*

NP *and other* NP: *rum and other spirits*

NP *or other* NP: *insects or other animals*

Constitutive

NP_2 is a part of NP_2: $C(NP_2, NP_1)$ *brain is a very sensitive part of the body.*

NP_1 made of NP_2: $C(NP_2, NP_1)$ *monuments made of stone and marble.*

NP_1 containing NP_2: $C(NP_2, NP_1)$ *a forest containing dead trees.*

Telic

an NP worth V-ing: $T(V, NP)$ *a question worth asking.*

enjoy/prefer V-ing NP: $T(V, NP)$ *enjoy listening to music / prefer watching television.*

an Adj NP to V: $T(V, NP)$ *a difficult question to ask.*

Identifying Qualia Values

Qualia values are not stipulated on formal Grounds. Linguistic evidence determines what information is stated to be lexically associated with the Qualia Structure of an expression.

Four Event Types

Pustejovsky & Moszkowicz (2011)

Core idea: Event types are built from two primitives – **State** and **Simple Transition** – by iteration and combination.

Type	Schema	Example
State	$[e_\phi]$	*John loves Mary.*
Achievement	$[e\ [e_1 \neg\phi] \xrightarrow{\alpha} [e_2 \phi]]$	*She arrived.*
Process	$[e\ [e_1 \phi_1] \cdots \xrightarrow{\alpha} [e_n \phi_n]]$	*Mary walked.*
Accomplishment	$[e\ [e_1 \neg\psi : [e_{1.1} \phi_1] \cdots] \xrightarrow{\alpha} [e_2 \psi]]$	*John built a house.*

- Process = *iteration* of simple transitions
- Accomplishment = process + achievement (culmination)
- α labels the arc (agentive action)
- Node labels encode state oppositions (Formal quale)

Event Structure in Lexical Entries: **break**

$$\left[\begin{array}{l}\textbf{build} \\ \text{AS} = \left[\begin{array}{l}\text{ARG}_1 = \mathbf{x} \\ \text{ARG}_2 = \mathbf{y} \\ \text{ARG}_3 = \mathbf{z}\end{array}\right] \\ \text{ES} = \left[\text{E} = \left[\ [e_1 \neg\mathcal{A}(y) : [e_{1.1}\ \mathcal{A}\_\textbf{prog(y)}] \cdots \xrightarrow{b\_act(x,z)} [e_{1.n}\ \mathcal{A}\_\textbf{prog(y)}]\ \right] \xrightarrow{b\_act(x,z)} [e_2\ \ \mathcal{A}(y)]]\right]\end{array}\right]$$

- **Formal quale** encodes the result state: *broken*(y) at e_2
- **Agentive quale** encodes the causal action
- The agentive arc is *optionally expressed* – breakage can be spontaneous

Alternation

✓ *The pot broke.*
(no agentive arc: spontaneous)

✓ *The girl broke the pot.*
(agentive arc: A:Act(x,y))

Four Types of Argument

Type	Syntactic status	Characterization
True	Obligatory	Must be expressed; predicate is incomplete without it
Default	Optional	Participates in qualia; need not surface
Shadow	Blocked	Semantically incorporated; surfacing is redundant
Hidden	Never surfaces	Plays a role in interpretation; not expressible

True Arg – lock

* *After she locked, she slept.*
* *I forgot to lock.*
* *She is locking.*
(object NP is obligatory)

Shadow Arg – phone

✓ *She phoned the office.*
* *She phoned the office on the phone.*
(phone incorporated in qualia)

Default and Hidden Arguments

Default Argument – **build**

John built the house out of bricks.

out of bricks fills the **default material argument** – expressible when informative, incorporated in the Agentive quale otherwise.

```
build:
 ARG1 = x:agent
 ARG2 = y:artifact
 D-ARG1 = z:material
```

Hidden Argument – **photograph**

John photographed the whole scene.
The shots are amazing!

The photos are never expressed syntactically yet:

- accessible via bridging inference (*the shots*)
- introduced by the GRR mechanism (next slide)
- not a result of type coercion

Defaulting is a compositional operation rather than an argument type (Jezek 2018, 2021), lexically or pragmatically licenced (see i.e. objects omissions). **Hidden args (Badia & Sauri 2001; Jezek & Melloni 2011; Pustejovsky & Jezek 2011)** are an extension of the classic 1995 model. They participate in interpretation but *cannot surface under any condition*.

Generalized Result Roles (GRR)

Rim and Pustejovsky (2023)

Definition

Any event predicate that induces a **modification, transformation, or creation** of an object introduces a GRR argument $\bar{z}$, binding the change undergone by the object.

General form

$$\lambda y\, \lambda x\, [P(x, y)] \;\rightarrow\; \lambda \bar{z}\, \lambda y\, \lambda x\, [R(x, \bar{z}, y)]$$

Creation verb (encoded GRR):

$$\lambda \bar{z}\, \lambda y\, \lambda x\, [\text{build}(x, \bar{z}, y)]$$

John built a house. – build selects result-type from ARG_2

Diagnostics

- *Anaphora: Mary translated the poem.* ***It*** *is beautiful.* [= translation]
- *Bridging: Karen sliced the bread. We each got one* ***slice****.*
- *Inert: ?John swept the room and put* ***it*** *in the trash.* [ambiguous]

Dot Types and Logical Polysemy

Pustejovsky (1995), Asher (2011)

A **dot type** $\tau_1 \bullet \tau_2$ captures lexical items whose meaning integrates two distinct but related types.

Canonical dot types

- *book*: physical • informational
- *school*: institution • building • event
- *city*: location • community
- *newspaper*: publisher • copy • article
- *lunch*: event • food

Co-predication tests

- *The book is heavy and informative*
- *The city elected a mayor and flooded*
- *Lunch was delicious and lasted two hours*

Dot types are accessed by predicates that select different facets. They are a **structural property** of the lexical entry, not a disambiguation choice. Co-predication encompasses different syntactic structures and may involve coercions (Jezek & Vieu 2014).

The Classic Composition Mechanisms

Mechanism	What it does
Pure selection	Standard function application: $f(a)$ where types match
Type coercion	Shift argument type to match selector (*began the novel*)
Co-composition	Mutual specialization of verb and argument (*bake a cake*)
Selective binding	Modifier picks out a facet of the head's qualia (*fast food*)
Qualia unification	Compound interpretation via shared qualia (*book-shop*)

These mechanisms are the classical engine. QES and DOM (next sections) extend them to cover richer phenomena and to support a distributed account.

Summary: GL as Foundation

- **Four levels** of lexical representation: AS, ES, QS, Type Structure
- **Qualia** as Aristotelian modes of predication, systematically encoded
- **Dynamic event types** built from state and transition primitives
- **Composition** via selection, coercion, co-composition, selective binding, unification
- **Dot types** for logical polysemy and co-predication

What comes next

QES integrates qualia labels directly into event structure; DOM tracks object states across events; DC coordinates both through composition networks.

§7: Qualia Event Structure (QES)

Pustejovsky & Jezek (2027)

QES – Motivation and Structure

Core Proposal

QES **integrates qualia roles directly into event structure**: each arc and node in the event graph carries quale labels that constrain what sub-events represent and how they relate.

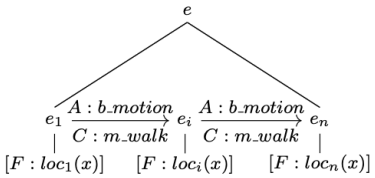
Quale	Position	Function
F (Formal)	Node label	State/property holding at that sub-event
A (Agentive)	Arc label	Action type initiating the transition
C (Constitutive)	Arc label	Manner constraint on the action
T (Telic)	Goal-node label	Purpose/intended outcome of the event

Why this matters

QES provides a **structured, fine-grained** account of how verbs encode stages, manners, and opposition relations – moving beyond the flat template approach of classic GL.

QES Analysis: **walk** (Process)

Event graph



$$\left[\begin{array}{l}\textbf{walk}\\ \text{AS} = \left[\ \text{ARG}_1 = \mathbf{x}\ \right]\\ \text{ES} = \left[\ \text{E} = \left[\ [e_{1,1}\, F : \mathbf{loc(x)}_1] \cdots \xrightarrow[C:m\_walk]{A:b\_motion} [e_{1,n}\, F : \mathbf{loc(x)}_n]\ \right]\right]\end{array}\right]$$

- A: **b_motion** – agentive label: the *type* of action is bipedal motion
- C: **m_walk** – constitutive label: the *manner* distinguishing *walk* from *run*, *stroll*, *saunter*
- **Manner subtype**: $\beta \sqsubseteq_C \alpha$ iff β specifies a particular manner of enacting α, encoded in the C value of β

Manner hierarchy

b-motion $\succ$ walk, run, stroll, saunter

QES Analysis: **build** (Accomplishment)

Pustejovsky & Jezek (2027)

$$\left[\begin{array}{l} \textbf{build} \\ \text{AS} = \left[\begin{array}{l} \text{ARG}_1 = \mathbf{x} \\ \text{ARG}_2 = \mathbf{y} \\ \text{ARG}_3 = \mathbf{z} \end{array}\right] \\ \text{ES} = \left[\text{E} = \left[\ [e_1 \neg\mathcal{A}(y) : [e_{1.1}\, \mathcal{A}\_\textbf{prog(y)}] \cdots \xrightarrow{b\_act(x,z)} [e_{1.n}\, \mathcal{A}\_\textbf{prog(y)}]\ \right] \xrightarrow{b\_act(x,z)} [e_2\ \ \mathcal{A}(y)]\right]\right] \end{array}\right]$$

- The **agentive arc** *b*-act(*x*,*z*) spans both the process and transition phases
- $\mathcal{A}$-prog(*y*): object *y* exists in a *progressive* (partial) state during e_1
- The culmination $\mathcal{A}(y)$ is the Agentive fact: artifact *y* now fully exists

Implication

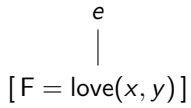
John is building a house $\not\Rightarrow$ a house exists (yet)

John built a house $\Rightarrow$ a house exists (now)

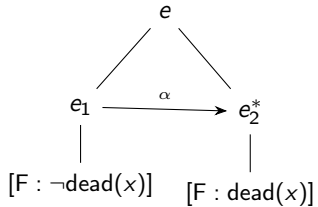
The $\mathcal{A}$-prog nodes represent the house-in-progress.

Encoding Formal and Agentive in Event Structure

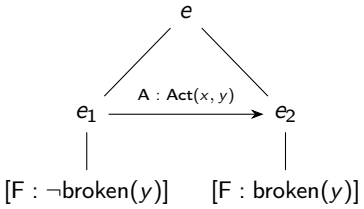
State: love



Achievement: die



Accomplishment: break



- **Formal** labels nodes (state predicate)
- **Agentive** labels arcs (causing action)
- e_2^* = instantaneous transition (achievement)

QES: Manner and the Constitutive Quale

The manner hierarchy

Within a single event type, the Constitutive quale **differentiates manners**. *walk*, *run*, *saunter*, *stroll* share an Agentive (bipedal motion) but differ in Constitutive.

Manner specifications

- *walk*: steady pace
- *run*: elevated pace, airborne phase
- *saunter*: leisurely, relaxed
- *trudge*: heavy, effortful
- *tiptoe*: quiet, cautious

Subtype relation

$$\beta \sqsubseteq_{\mathsf{C}} \alpha$$

β is a *manner specialization* of α: β inherits α's Agentive and refines its Constitutive.

QES lets us say precisely what it means for one verb to be a **manner subtype** of another. The relation is first-class in the theory.

QES: Telic and the Goal Node

Telic as goal-node label

For purpose-directed events, the Telic quale labels the culmination node. This distinguishes purposeful from accidental culminations.

Purpose-directed

- *Mary built a house* [T: inhabited]
- *John baked a cake* [T: eaten]
- *She wrote a letter* [T: read]

Non-purpose-directed

- *The pot broke* [no telic]
- *John fell* [no telic]
- *The vase shattered* [no telic]

The Telic quale is what licenses inferences about **why** an event happened, and what licenses modifier constructions like *built for family use* or *baked for her birthday*.

QES Summary

Quale	Position	Function	Example
F Formal	Node label	State/property at sub-event	F : broken(y) loc(x)
A Agentive	Arc label	Action type initiating transition	A : b_motion, b_act(x,z)
C Constitutive	Arc label	Manner constraint on action	C : m_walk, m_build
T Telic	Goal-node label	Purpose/intended outcome	T : eat(x), inhabited(z)

Key Insight

QES provides a **structured, fine-grained** account of how verbs encode stages, manners, and opposition relations in change predicates – moving beyond the flat template approach of classic GL.

QES and Aspectual Composition

How QES informs aspect

Aspectual properties (telicity, durativity, dynamicity) **derive** from the QES representation rather than being stipulated as features.

Aspect class	**QES signature**
State	Single node, Formal only
Achievement	Two nodes, instantaneous arc, Formal change
Activity	Iterated transitions, no culmination, Agentive arc
Accomplishment	Process subevent + culmination, Agentive + Formal

Coercions between aspectual classes (e.g. *John ran for an hour* $\rightarrow$ activity from accomplishment) are **QES transformations** that add or remove culmination nodes.

QES: Looking Ahead

What QES enables downstream

- §9 (DOM) uses QES nodes as points where object states update
- §10 (DC) uses QES arcs as sites of constraint interaction
- §11 (Vector Realization) encodes QES graphs as tensor sequences
- §12 applies QES to entailment and state tracking

QES is the **structural backbone** of the dual-aspect semantics. Every downstream mechanism references QES graphs directly or indirectly.

§8: Argument Structure Types

True and Latent Arguments

True Arguments (formal)

A verb's type signature $V : \tau_1 \to \tau_2 \to \cdots \to \tau_n \to \mathrm{Event}$ specifies arguments that must be explicitly satisfied during composition.

admire – true args only: $admire : e \to (e \to (s_s \to t))$; experiencer x, target y.

$$[\![admire(x, y)]\!] \;=\; \exists e \left[\, admire\text{-}state(e, x, y) \,\right]$$

Latent Arguments (formal)

Latent args are incorporated via implicit quantification in the denotation:
$[\![V(x_1, \ldots, x_n)]\!] = \exists y_1 \cdots y_m.\, V'(x_1, \ldots, x_n,\, y_1, \ldots, y_m)$.

build – true args + latent result z:
$build : e \to (e \to (s_{\mathrm{acc}} \to t))$; agent x, material y; latent $z : \mathrm{obj}_{\mathrm{q}}$.

$$[\![build(x, y)]\!] \;=\; \exists e\, \exists z \left[\, build\text{-}act(e, x, y) \wedge create(e, z) \,\right]$$

The s_{acc} vs. s_s distinction is downstream

Both verbs superficially type as $e \to (e \to t)$. The event-type contrast is a *consequence*, not the source: accomplishment types carry a result slot requiring a referent, and the latent z in *build* fills it. Stative types carry no such slot; *admire* has no latent argument.

GRR and the Inventory of Result Types

What kinds of results are bound?

GRR is general but its values are constrained by event type.

Event class	Result type	Example
Creation	New artifact	*built a house*
Transformation	Modified type	*baked a cake* (dough → cake)
Scalar change	Same type, new degree	*cooled the soup*
Destruction	Negation of type	*demolished the wall*
Division	Derivative parts	*sliced the bread* → slices
Representation	Image/record	*photographed* → photos

The GRR argument's type is **derived from the event type**. This is a predictive contribution: for each

Summary: Argument Structure Machinery

- **GRR** – result argument for change-of-state and creation verbs
- **Type signature** distinguishes creation (introduces quantized object) from relation (does not)
- Argument structure interacts with QES (which subevents each arg participates in) and DOM (how the object updates)

Next: DOM, which tracks the **evolving object states** that argument structure and QES together predict.

§9: Dynamic Object Modeling (DOM)

Meaning as Object Change – Resource-Result Structures

Dual-Aspect Semantics

The core architecture

Meaning has **two coupled aspects**: a *situational* aspect (what happens: events, processes, transitions) and an *object* aspect (what endures: entities, their properties, their evolving states).

- Events **modify** objects (DOM records the updates)
- Object states **condition** which events can apply
- Composition is **mutual update** between the two aspects

DOM – Core Idea

Central Claim

DOM reframes verb meaning as **object change**. Events create, transform, and remove objects. Every event is modeled as a **labeled transition on objects** rather than just a truth-value function.

Key DOM concepts

- **Labeled Transition System:** objects move through typed states
- **Resources:** objects consumed or transformed
- **Results:** objects created or significantly modified
- **Quantized objects O:** fully individuated, countable
- **Intermediate objects $\tilde{O}$:** non-quantized

Result functions (partial)

$\text{res}_i(e) \rightarrow$ non-quantized result
$\text{res}_q(e) \rightarrow$ quantized result

Example:
Mary baked bread from flour.
Resource: flour (material)
Result: bread (quantized artifact)

$$\text{flour} \xrightarrow{\text{bake}} \text{bread}$$

Modes of Change: Extrinsic vs. Intrinsic

Extrinsic Change

Defined by the object's **relation to an external reference frame**.

The change is not about an internal property of x but about its location, position, or relational status.

- *John moved the table.* [location change]
- *The prisoner escaped.* [relational status]

Intrinsic Change

Internal to the object, independent of external frame. Transformation is inherent – shifts the object's internal condition.

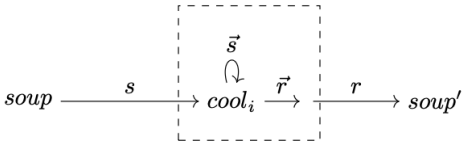
- *The soup cooled.* [temperature $\downarrow$]
- *Mary baked a cake.* [material $\rightarrow$ artifact]
- *The workers widened the road.* [width $\uparrow$]

This distinction structures the typology of result structures: **type-preserving** (scalar) vs. **type-changing** (creation/destruction).

DOM Diagrams – Type-Preserving Changes

Scalar change: cool

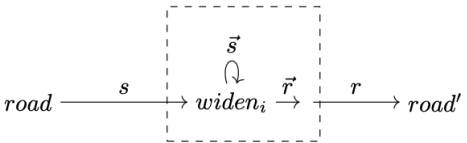
The soup cooled quickly.



TYPE(soup) = TYPE(soup′)

Widening: widen

The workers widened the road.

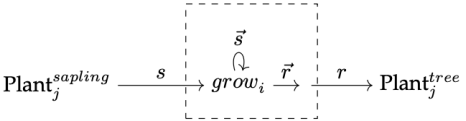


TYPE(road) = TYPE(road′)

$width_1 < width_2$

Natural growth: grow

The sapling grew into a tree.



Type-preserving pattern:
Object identity maintained; only a scalar or relational property changes.

$\text{Agent} + \text{Obj} \xrightarrow{V_i} \text{Obj}'$

DOM Diagrams – Type-Changing (Creation)

Resource manipulation: knit

Mary knitted for hours.

$$\text{yarn} \xrightarrow{\text{knit}_i} \tilde{o}_{\text{knit}} \quad \text{(intermediate)}$$

Mary knitted a sweater.

$$\text{yarn} \xrightarrow{\text{knit}_i} \text{sweater} \quad \text{(quantized)}$$

General creation schema:

$$\text{Agent} + \text{Material} \xrightarrow{V_i} \text{Artifact}$$

$$\text{type(material)} \neq \text{type(artifact)}$$

Representational creation

Two sources fused into a representational artifact:

- **write**: Agent + Info $\rightarrow$ [Info • Phys]
- **photograph**: Agent + Scene $\rightarrow$ Photo
- **draw/paint/film**: alternating

Result functions:
$\text{res}_i(e)$: non-quantized (in-progress)
$\text{res}_q(e)$: quantized (completed artifact)

DOM as a Type-Theoretic Signature: A Catalogue

Each DOM is a typed source-to-result mapping

DOM schema

Each DOM is a typed mapping from source state to result state: $\mathrm{DOM} : \tau_{\mathrm{source}} \to \tau_{\mathrm{result}}$. Types may be empty ($\bot$), bare (*obj*), or qualified: *obj*: *s* (object in state *s*), *obj*: *l* (at location *l*), *m*: *f* (material *m* in form *f*), obj_q (object qualified by quale *q*).

DOM	Type signature	Example verbs
Create	$\bot \to obj_q$	*build, bake (a cake), make, produce*
Destroy	$obj \to \bot$	*eat, consume, destroy, burn away*
Persist	$(obj{:}\, s) \to (obj{:}\, s)$	*admire, see, hold, contain*
Change-State	$(obj{:}\, s_1) \to (obj{:}\, s_2)$	*fix, dry, harden, heat, ripen*
Change-Loc	$(obj{:}\, l_1) \to (obj{:}\, l_2)$	*put, place, push, move, send*
Change-Form	$(m{:}\, f_1) \to (m{:}\, f_2)$	*bake (a potato), shape, carve*

What this buys

The DOM signature *predicts the derivation grade*: Create/Change-Form need a quale to anchor the result (Grade 2, *bake a cake*); Change-State in resultatives needs constructional contribution (Grade 3, *wipe the counter dry*); Persist admits classical FA (Grade 0, *admire the painting*).

Modeling Verbal Polysemy: **bake**

The Classic Puzzle

The verb **bake** is ambiguous between a **type-preserving** reading (change-of-state) and a **type-changing** reading (creation). DOM makes this formally precise.

Type-preserving

Mary baked the potato.

$\text{potato} \xrightarrow{\text{bake}_i} \text{potato}'$

$\text{type}(\text{potato}) = \text{type}(\text{potato}')$
material $\rightarrow$ *material*

No creation; the potato undergoes a scalar change of state.

Type-changing

Mary baked a cake.

$\text{dough} \xrightarrow{\text{bake}_i} \text{cake}$

$\text{type}(\text{dough}) \neq \text{type}(\text{cake})$
material $\rightarrow$ *artifact*

Creation event: a new artifact type emerges.

The **Agentive quale of the NP** (cake's A = bake) drives the type-changing interpretation via inner application (see §10).

DOM as Record Update

Formal architecture

A DOM object is a record $\langle T, \sigma \rangle$ where T is the base type and σ is the current state. An event e induces a transition:

$$\langle T, \sigma_i \rangle \xrightarrow{e} \langle T', \sigma_f \rangle$$

State transition cases

- $T = T'$: type-preserving (scalar change)
- $T \neq T'$: type-changing (creation, transformation)
- $\sigma_f = \emptyset$: destruction
- $\sigma_f =$ new: creation

Discourse updates

- After *Mary baked a cake*: cake record active
- After *She served it*: cake record unchanged, new event
- After *We ate it*: cake record $\rightarrow$ destruction

This is the **DRT/File Change inheritance**. DOM is dynamic semantics generalized to object lifecycle within and across sentences.

§10: Distributed Compositionality

The engine that puts QES and DOM to work

What Distributed Compositionality Claims

The thesis

Phrasal meaning is **not** the value of a single functor applied to a single argument at each node. It is the **joint** contribution of the verb, its arguments, their qualia, and – where present – the construction itself. The semantic load is genuinely *distributed*.

Three commitments

1. Lexical items carry **latent resources** (qualia, event templates, result schemas) that composition can activate.
2. Composition has **two modes**: **Outer Application** (OA) and **Inner Application** (IA).
3. Whether IA fires is determined by **licensing conditions**, not by the verb alone.

What this buys us

- One verb **bake**, two readings – without lexical ambiguity.
- Type coercion explained without ad-hoc lexical rules.
- Resultatives, stage-level contrasts, light verbs – all the same mechanism.
- A graded measure of compositional complexity.

Outer Application (OA) – The Classical Mode

Every theory of compositional semantics since Frege has had a single rule for combining a predicate with its argument: function application. We call this *Outer Application*.

Outer Application

Given $f : \tau_1 \to \tau_2$ and $a : \tau_1$, return $f(a) : \tau_2$.

$$\textit{the rock fell} = \textit{fell}(\textit{the-rock}), \qquad \textit{Mary slept} = \textit{slept}(\textit{Mary}).$$

What OA does

- Combines functor with argument under **type discipline**.
- Driven by syntactic argument structure.
- Available in every framework: Montague, type-logical, DRT, categorial, DisCoCat.

What OA does *not* do

- Cannot **inspect the internal structure** of its argument.
- Argument is a *black box* of type τ_1: qualia, affordances, event history all invisible.
- Inadequate for *bake a cake*, *begin the book*, *wipe the counter dry*...

OA is the **floor**. Everything DC adds is *on top of* OA – it never replaces OA.

Inner Application (IA) – The Novel Contribution

The claim

Inner Application is new to compositional semantics. No version of Montague, categorial grammar, DRT, DisCoCat, or vector composition provides anything equivalent. It is what DC contributes.

Inner Application

Given predicate f and argument a, IA produces a *specialized* predicate f' **before** OA applies:

$$f' \;=\; Q_a^q(f), \quad Q_a^q : (\tau_1 \to \tau_2) \to (\tau_1 \to \tau_2), \quad q \in \{\mathsf{A}, \mathsf{T}, \mathsf{C}, \mathsf{F}\}.$$

Q_a^q is a **higher-order operator** read off the q quale of a.

What IA does

- Lets the predicate **look inside** the argument.
- Selectively probes *one* quale – not all four.
- Specializes f before OA delivers a value.
- Identity when no quale is exploitable: classical FA is the special case.

What IA explains

- *bake a cake* via A: creation reading.
- *begin the book* via T: coerced event.
- *good knife* via T: qualia-relative gradability.
- *fix the leaky faucet* via F: stage-level transition.

Pipeline: $\underbrace{f' = Q_a^q(f)}_{\text{IA: specialize}} \implies \underbrace{f'(a_1)(a_2)}_{\text{OA: apply}}$ IA *licensed by the argument*; OA *required by syntax*.

When is Inner Application Licensed?

Three licensing conditions on $Q_a^q(f)$

Given quale $q \in \{\mathrm{A}, \mathrm{T}, \mathrm{C}, \mathrm{F}\}$ on argument a and predicate f, IA fires *iff* all three hold:

(L1) Availability. The projection $\pi_q(a)$ is *defined*: argument a has a non-empty q quale.
Example: *cake* has A = bake-act. *potato* does not.

(L2) Compatibility. The content of $\pi_q(a)$ is consistent with the event template of f.
Example: *cake*'s A involves baking; **bake**'s event template applies heat – compatible.

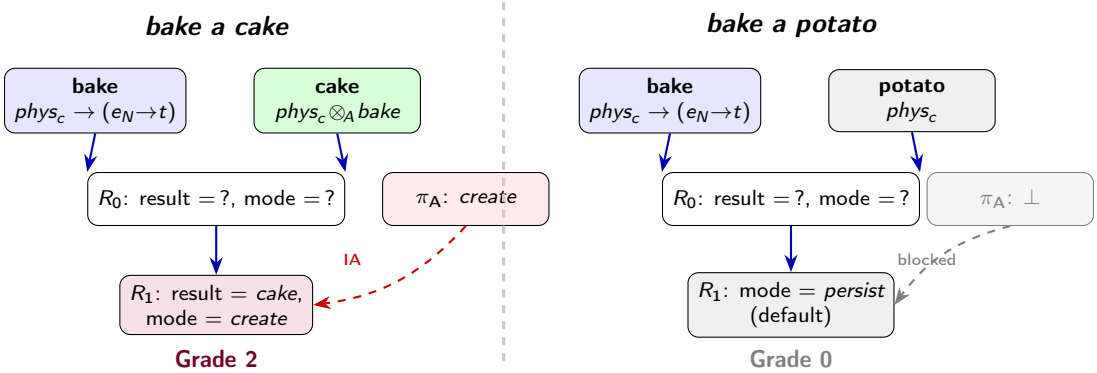
(L3) Informativity. The result $Q_a^q(f)$ is strictly more specified than f alone.
No-op IA is filtered out.

If any condition fails

IA is **not licensed**. The derivation proceeds with OA alone, and any gaps in f's event template are filled by defaults.

Diagnostic: *bake a cake* vs. *bake a potato*

One verb. One signature. The qualia of the object decide whether IA fires.



Same verb, same OA spine. The argument's qualia structure determines whether IA is licensed – and that is where the creation reading comes from.

Walking Through *Mary baked a cake*

Lexical entries

bake : $phys_c \to (e_N \to t)$
$\lambda x{:}phys_c\ \lambda y{:}e_N\ \lambda e\ [apply\text{-}heat(e,x) \wedge change\text{-}state(e,x) \wedge agent(e,y)]$

cake : $(phys_c \otimes_A bake) \otimes_T eat$
F = *baked-good*,
A = $\lambda e'\ [bake\text{-}act(e',x) \wedge create(e',x)]$

Derivation

(a) OA: **bake** composes with *a cake*.
$\pi_C(cake) = phys_c$ ✓

(b) Check IA on A:
L1 ✓, L2 ✓, L3 ✓.

(c) IA fires: meet R_0 with $\pi_A(cake)$.

(d) DOM update: mode = *create*.

Contrast: *Mary baked a potato*

potato : $phys_c$, no A

(a) OA: same step as above. ✓
(b) Check IA on A:
L1: $\pi_A(potato)$ undefined.
⇒ **IA blocked.**

(c) DOM default: mode = *persist*.

Outcome

Mary baked a cake → creation, **Grade 2**.

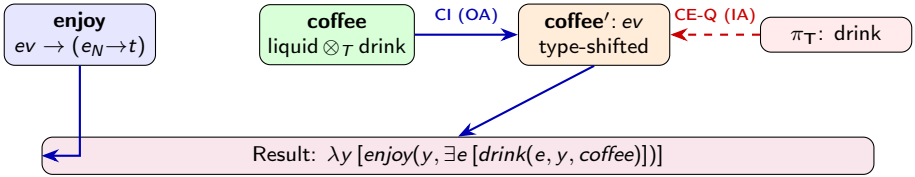
Mary baked a potato → change-of-state, **Grade 0**.

The verb does not change. The argument's qualia do all the work.

Coverage I – Type Coercion as OA + IA

Type coercion is the paradigm of OA and IA *working together*. The verb selects type τ_1; its argument has type τ_2. OA shifts the type via **Coercion by Introduction** (CI); IA fills the shifted shell from a quale via **Coercion by Exploitation** (CE-Q).

Mary enjoyed her coffee



Aspectual

began the novel

π_T(novel) = read

Complement

finished the brief

implicit reading event

Intentional

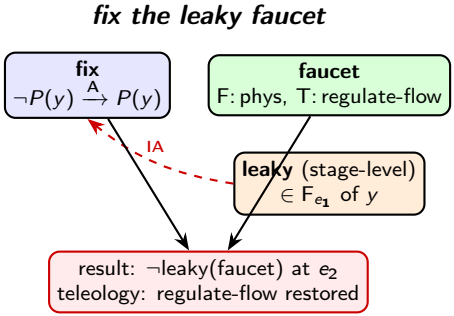
enjoyed the film

π_T(film) = watch

Coercion is not a separate mechanism. It is OA and IA running in tandem: OA introduces the event shell; IA fills it from a quale.

Coverage II – Stage-Level Predicates

Stage-level vs. individual-level adjectives modify the event structure of the noun differently – and DC licenses each at the right point in the derivation.



Why this works

- *leaky* is a **stage-level** predicate: lives in F at a specific event stage.
- Fits the input state of **fix**'s template via IA.
- DOM update: faucet transitions from leaky → not-leaky.

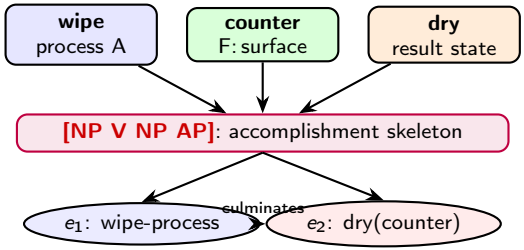
Contrast: *fix the blue faucet*

blue is **individual-level** (F, not stage-bound).
⇒ The controller does *not* feed it to **fix**'s input state.
⇒ No entailment that the faucet ceases to be blue.
The reading is just locative/identificational.

Coverage III – Resultatives

The construction itself contributes a result state; verb and AP each supply a piece of the event template.

Mary wiped the counter dry



Why DC is needed

- **wipe** alone is atelic: no culmination point.
- *dry* alone is stative: no process.
- **Neither lexical item** licenses the resultative reading on its own.
- The **construction** contributes the accomplishment skeleton $[e_1] \xrightarrow{\text{culm}} [e_2]$.
- DC binds verb-process to e_1, AP-state to e_2; the NP threads through both via DOM.

Coverage IV – The Wider Landscape

Phenomenon	Example	What DC contributes
Co-composition	*bake a cake / a potato*	Argument's A licenses IA; same verb, two readings.
Type coercion	*enjoyed the coffee*	OA (CI) shifts type; IA (CE-Q) fills shell from T.
Stage-level mod.	*fix the leaky faucet*	Stage predicate fits verb's input state.
Resultative	*wipe the counter dry*	Construction contributes accomplishment skeleton.
Light verb	*take a pill*	Verb maximally underspecified; object's A fills event content.
Subject coercion	*article angered Mary*	Subject's content supplies the cause; IA on verb.
Cospecification	*shuttle dropped sat.*	Multiple arguments jointly enrich the event template.

Unifying claim: all of these are **the same machinery** – OA + licensed IA – varying only in *which* quale is exploited and *which* constituents contribute it.

Grades of Compositional Complexity

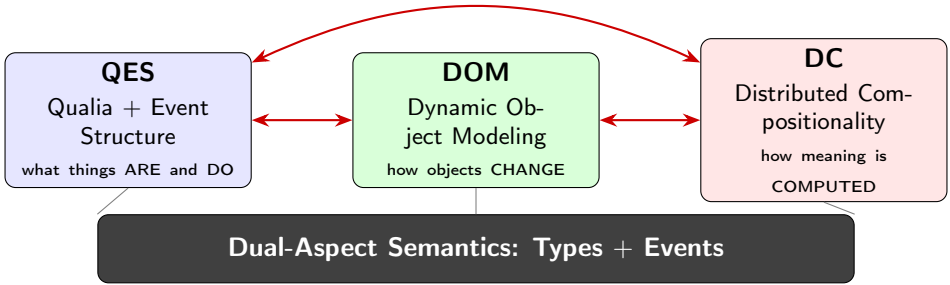
Each derivation gets a **grade** reflecting how much IA work was needed. Classical FA is the floor.

Grade	Operations	Diagnostic	Examples
0	OA only	No IA licensed.	*bake a potato*, *the rock fell*
1	OA + 1 IA	Single quale exploited.	*begin the book*, *good knife*
2	OA + co-composition	Argument and verb mutually specialize.	*bake a cake*, *take a pill*
3	OA + multi-controller	Multiple constituents jointly enrich; construction-level contribution.	*wipe the counter dry*, *fix the leaky faucet*

Grade is *computed* from the derivation, not stipulated. The system reduces to classical compositional semantics at Grade 0 – DC is **conservative** over Montague.

Dual-Aspect Semantics – The Full Picture

Three pillars of the framework, working together:



- **Static aspect:** types, categories, properties, relations – QES.
- **Dynamic aspect:** events, transformations, state changes – DOM.
- **Compositional engine:** OA, IA, controllers, grades – DC.

GL provides the formal machinery to model all three simultaneously. **The next section asks: how do we *realize* this in vectors?**

§11: Vector Realization of DC

TPR, HRR, VSA – and toward exact algebraic binding

Why Vector Realization?

The claim of §§6–10 restated

Qualia, events, argument structure, and DOM are **linguistic primitives** specified by a formal theory. That theory is *substrate-independent*: it constrains what a compositional system must represent, not how.

What vector realization buys

- Compatibility with modern neural models
- Gradient preferences native
- Scale through learned embeddings
- Integration with LLM hidden states
- A substrate for the composition networks of §10

What it must preserve

- Role-filler distinctness
- Type discipline
- Exact retrieval of constituents
- Compositional transparency
- Interpretable operations

Tensor Product Representations (TPR)

Smolensky (1990)

Core idea

Represent a role-filler binding as the outer product $\vec{r} \otimes \vec{f}$. A structure is a superposition of such bindings.

$$\text{bind}(\vec{r}, \vec{f}) = \vec{r} \otimes \vec{f} \qquad S = \sum_i \vec{r}_i \otimes \vec{f}_i$$

Virtues

- Exact retrieval: $\vec{f}_i = S \cdot \vec{r}_i$
- No interference if roles orthogonal
- Multiple bindings compose linearly
- Clear algebraic semantics

Limits

- Dimension grows: $d \cdot d$ per binding
- Deep structure $\rightarrow$ huge tensors
- Typically impractical for LLM-scale
- Approximations needed

TPR sets the **algebraic standard**: what binding and superposition *should* look like. HRR and VSA are lossy but tractable approximations.

Holographic Reduced Representations (HRR)

Plate (1995)

Fixed-dimensional binding

Replace outer product with **circular convolution** $\circledast$. Binding keeps the same dimension. Unbinding approximates retrieval via correlation $\oslash$.

$$\vec{b} = \vec{r} \circledast \vec{f} \qquad \vec{f}' = \vec{r} \oslash \vec{b} \approx \vec{f}$$

Mechanics

- $\circledast$: circular convolution
- $\oslash$: circular correlation
- Random unit vectors for roles
- Clean-up memory for exact recovery

Tradeoffs

- Fixed dim d throughout
- Retrieval is approximate
- Error accumulates with depth
- Superposition capacity $\propto d$

HRR is the most direct computational realization of a dual-aspect lexical entry: roles as vectors, qualia fillers as vectors, bound by $\circledast$.

VSA: The Broader Family

Kanerva, Gayler, Kleyko et al.

Variant	Bind operation	Key property
HRR (Plate)	Circular convolution	Real-valued, approximate
FHRR	Complex Fourier HRR	Unitary binding, exact inverse
Binary Spatter Code	Bit-wise XOR, majority	Very high dim, binary
MAP (Gayler)	Hadamard product	Simple, self-inverse
Sparse VSA	Block-wise operations	Energy-efficient

Common commitments

- **Binding**: combine role and filler into a single vector
- **Bundling**: superpose multiple bindings
- **Permutation**: protect against interference
- **Clean-up**: recover exact fillers from noisy retrieval

Encoding a GL Lexical Entry as HRR

The mapping

Each qualia role $R \in \{\mathrm{F}, \mathrm{C}, \mathrm{T}, \mathrm{A}\}$ gets a fixed role vector $\vec{r}_R$. A lexical entry becomes a superposition of role-filler bindings.

$$\vec{v}_{\text{cake}} = \vec{r}_{\mathrm{F}} \circledast \vec{f}_{\text{food}} + \vec{r}_{\mathrm{C}} \circledast \vec{f}_{\text{ingredients}} + \vec{r}_{\mathrm{T}} \circledast \vec{f}_{\text{eat}} + \vec{r}_{\mathrm{A}} \circledast \vec{f}_{\text{bake}}$$

Retrieving Agentive

$$\vec{f}'_{\mathrm{A}} = \vec{r}_{\mathrm{A}} \oslash \vec{v}_{\text{cake}} \approx \vec{f}_{\text{bake}}$$

Adding a new role

$$\vec{v}' = \vec{v}_{\text{cake}} + \vec{r}_{\text{GRR}} \circledast \vec{f}_{\text{new}}$$

This is GL's 1995 architecture, rendered in vector form. Roles are first-class, fillers are vectors, composition is algebraic.

Encoding Argument Structure in HRR

The mapping

Argument structure is also role-filler. Role vectors $\vec{r}_{\text{ARG}_i}$ distinguish subject, object, oblique. GRR, default, shadow, and hidden arguments each get their own role vector.

$$\vec{\text{event}} = \vec{r}_{\text{VERB}} \circledast \vec{f}_{\text{bake}} + \vec{r}_{\text{ARG}_1} \circledast \vec{f}_{\text{Mary}} + \vec{r}_{\text{ARG}_2} \circledast \vec{f}_{\text{cake}} + \vec{r}_{\text{GRR}} \circledast \vec{f}_{\text{cake}'}$$

- The **GRR binding** makes the resulting object accessible as a first-class argument
- Hidden arguments get bound with a **latent** role marker, never syntactically realized
- Same vector arithmetic, enriched role inventory

The argument structure typology of §8 translates directly: each class of argument gets a dedicated role vector with its own retrieval semantics.

Worked Example: *bake a cake* in HRR

Step 1 – Lexical entries

$$\vec{v}_{\text{cake}} = \sum_R \vec{r}_R \circledast \vec{f}_R^{\text{cake}}$$

$$\vec{v}_{\text{bake}} = \sum_R \vec{r}_R \circledast \vec{f}_R^{\text{bake}}$$

Step 2 – Inner application

Compatibility check: $\vec{f}_{\text{A}}^{\text{cake}} \cdot \vec{f}_{\text{A}}^{\text{bake}} > \theta$

Step 3 – Specialized verb

$$\vec{v}_{\text{bake-cake}} = \vec{v}_{\text{bake}} + \vec{r}_{\text{obj-type}} \circledast \vec{f}_{\text{cake}}$$

Step 4 – Outer application

$$\vec{e} = \vec{v}_{\text{bake-cake}} + \vec{r}_{\text{ARG}_1} \circledast \vec{\text{Mary}} + \vec{r}_{\text{ARG}_2} \circledast \vec{\text{cake}}$$

This mirrors §10/C4 exactly. The **inner application** step is a compatibility-gated superposition; the **outer application** is role-slot binding. Two algebraic operations, two compositional mechanisms.

QES as a Tensor Sequence

From event graphs to vectors

A QES graph is a sequence of typed transitions. Each node becomes a state vector; each arc becomes a transition tensor.

$$\vec{s}_{e_1} \xrightarrow{M_\alpha} \vec{s}_{e_2} \xrightarrow{M_\beta} \vec{s}_{e_3}$$

Encoding

- Node: Formal quale superposition
- Arc: Agentive + Constitutive matrix
- Culmination: Telic marker added
- Whole event: sequential composition

What this enables

- Entailment as state vector equality
- Progressive vs. perfective as positional readout
- Aspectual coercion as tensor transformation
- Direct tie to LLM hidden states

DOM in Vector Form

Object state as a dynamic vector

A DOM object is a vector $\vec{o}$ in a state space. Events act as learned transformations: $\vec{o}' = T_e \cdot \vec{o}$ or $\vec{o}' = \vec{o} + \vec{\delta}_e$.

Type-preserving events

$$\vec{o}_{\text{potato}} \xrightarrow{T_{\text{bake}}} \vec{o}_{\text{potato}'}$$

$\vec{o}$ and $\vec{o}'$ share basis type; scalar change encoded in component shift.

Type-changing events

$$\vec{o}_{\text{dough}} \xrightarrow{T_{\text{bake}}} \vec{o}_{\text{cake}}$$

Base type changes; new GRR vector added to context; old material vector decays.

DOM updates are **first-class** in the representation. This is what gives the system the state-tracking capability (§3/D12) that pure LLMs lack.

Composition Networks as Iterative Updates

Recall §10: $\mathcal{N} = \langle V, E, \tau, \lambda, \text{Ctrl} \rangle$

The network's fixed-point interpretation $[\![\alpha]\!] = \mathcal{M}^*(\text{root})$ is computed as a sequence of vector updates.

$$\vec{v}_i^{(t+1)} = \phi\left(\vec{v}_i^{(t)}, \bigoplus_{j \in N(i)} \psi(\vec{v}_j^{(t)}, e_{ji})\right)$$

Mapping

- ϕ: local update (aggregate influences)
- ψ: edge message (qualia-mediated)
- $N(i)$: neighboring constituents
- Convergence: stable interpretation

Parallels

- Graph neural networks
- Attention layers
- Constraint propagation networks
- Predictive coding

DC's network view is **computationally natural** in modern neural architectures. It does not require a new substrate – it specifies what the substrate should compute.

Training and Learning

The open problem

GL's qualia have historically been hand-specified. For vector realization to scale, qualia must be **learnable** from data.

Training objectives

- Reconstruction: recover qualia from contexts
- Role consistency: $\vec{r}_R$ stable across lexicon
- Decode fidelity: $\vec{r} \oslash (\vec{r} \circledast \vec{f}) \approx \vec{f}$
- Coercion fidelity: predicted coercions match data

Data sources

- Dictionaries + lexical resources
- Corpus-derived selectional preferences
- Multimodal grounding (VoxML-style)
- Behavioral probes on LLMs

The training story is **active research**. We have prototypes, not a finished pipeline. The formal theory specifies what success should look like.

Current Research: Functional Geometric Algebra

Toward exact algebraic binding

Clifford / geometric algebras offer bindings (wedge products, contractions, rotors) that are **exact** and type-graded, addressing HRR's approximation and interference limits.

Operations of interest

- Wedge product $\wedge$: event composition
- Contraction $\rfloor$: argument application
- Rotor R: type coercion
- Grade projection: role extraction

What this might give us

- Exact retrieval (no clean-up)
- Type-graded composition
- Coercion as geometric transformation
- Unified algebra for all mechanisms

Active research program: *Toward a Functional Geometric Algebra for Natural Language Semantics* (Pustejovsky 2026, arXiv preprint). We flag this as current work, not a settled framework.

Three Frameworks, One Theory

	DisCoCat	HRR / VSA	FGA (current)
Types	Pregroup grammar	Role vectors	Graded multivectors
Binding	Tensor contraction	Circular convolution	Wedge / contraction
Retrieval	Category morphism	Correlation (approx)	Exact grade projection
Coercion	Category functor	Linear transformation	Rotor
Dim. growth	Grows with depth	Fixed, lossy	Fixed, exact
Linguistic coverage	Syntax-driven	Arbitrary role structure	Type-graded structure
Maturity	Established	Established	Research

Observation

All three realize the **same DC theory**. They differ in algebraic substrate and tradeoffs. Choice of substrate is a computational decision; the **linguistic commitments (qualia, events, DOM)** are invariant.

Summary: Vector Realization

What we have shown

1. TPR sets the algebraic standard; HRR/VSA give tractable approximations
2. Qualia, argument structure, QES, and DOM **all translate** to vector operations
3. The *bake a cake* analysis runs symbolically and in HRR with the same mechanism
4. Composition networks are naturally expressed as iterative vector updates
5. Learning qualia from data is active work; FGA is a current research direction

Part II's core claim is now concrete: **GL is substrate-independent**. The same linguistic theory admits symbolic, algebraic, and neural realizations. Composition is the same process, computed differently.

§12: Applications, Discussion, and Q&A

Closing the loop with Part I's diagnostics

Revisiting the Diagnostics

Diagnostic (§3)	Neural failure	DC mechanism
SCAN / COGS	Combinator not explicit	Composition network with role-typed bindings
Polysemy / coercion	Frequency-driven only	Qualia-driven inner application
Event entailment	Stochastic	QES culmination + GRR
State tracking	Drift over discourse	DOM record updates
Type mismatch	Fails silently	Controller-mediated type shift
Resultatives, constructions	Underperforms	Constructional DC

Each diagnostic failure corresponds to a specific DC mechanism. The prediction: probing with qualia-aware, QES-aware, DOM-aware tests reveals where models have and have not learned the underlying structure.

Applications in Current NLP

Task	DC contribution
Natural language inference	QES-grounded entailment patterns (accomplishment, achievement)
State tracking	DOM-style object records; resource-result tracking
Coercion resolution	Qualia-driven sense recovery
Event extraction	Typed event templates; GRR for implicit results
Multimodal grounding	VoxML-style affordances align with Telic qualia
LLM interpretability	Qualia as probe targets; role-filler bindings as diagnostics
Low-resource semantics	Typed lexical resources reduce data requirements

The contributions are not necessarily *new systems*; often they are **structured probes and training signals** for existing models. DC gives us a vocabulary and a checklist.

Open Questions

1. **Learnability.** Can qualia roles be acquired from corpora without hand-specification? From multimodal data? From behavioral probes?
2. **Universality.** Are qualia cross-linguistically invariant? Are the telic/agentive distinctions culturally stable?
3. **Substrate convergence.** Do HRR, FGA, and Transformer hidden states learn the *same* qualia when trained on the same data?
4. **Boundaries.** Where does the dual-aspect theory end and pragmatics begin?
5. **Evaluation.** What are the right benchmarks for qualia-aware composition?
6. **Integration.** How do DOM and QES interface with discourse-level phenomena beyond the sentence?

What We Hope You Take Away

1. Compositionality is **real**, but classical function application captures only part of it.
2. Neural models encode **something structured**; they just don't declare what. Formal theory gives us the vocabulary.
3. GL provides **qualia**, **events**, and **object dynamics** – the missing primitives.
4. **Distributed compositionality** unifies these into a network-theoretic framework.
5. The theory is **substrate-independent**: symbolic, algebraic, and neural realizations converge.
6. Applications are concrete: inference, state tracking, coercion, grounding, interpretability.

Not a rejection of the formal tradition. Not a rejection of the distributional tradition. A **principled synthesis** that makes predictions both camps can test.

Selected References

Theoretical foundations

- Pustejovsky (1995), *The Generative Lexicon*. MIT Press.
- Pustejovsky & Batiukova (2019), *The Lexicon*. CUP.
- Pustejovsky & Jezek (2027, forthcoming), *Generative Lexicon Theory: A Modern Introduction*. OUP.
- Pustejovsky & Moszkowicz (2011), *Spatial Cognition & Computation,*.
- Asher (2011), *Lexical Meaning in Context*. CUP.
- Cooper (2023), *From Perception to Communication*. OUP.
- Jezek & Melloni (2011).
- Batiukova & Pustejovsky (2013).
- Rim & Pustejovsky (2023).

Distributional / computational

- Smolensky (1990), *AI*.
- Plate (1995), *IEEE TNN*.
- Coecke, Sadrzadeh & Clark (2010), DisCoCat.
- Baroni et al. (2014), *Frege in space*.
- Lake & Baroni (2018), SCAN.
- Hupkes et al. (2023), *Nature Machine Intelligence*.
- Pavlick (2023), *Phil. Trans. R. Soc.*
- Boleda & Korhonen (2025), *Annual Review*.
- Kleyko et al. (2023), *ACM CS*.
- Pustejovsky (2026), FGA (arXiv).

Distributed Compositionality

Formal Foundations for Distributional Semantics

James Pustejovsky · Elisabetta Ježek

Brandeis University · University of Pavia

Questions and Discussion

jamesp@brandeis.edu · elisabetta.jezek@unipv.it

Slides, handout, and references available at the tutorial page. Extended technical material: Pustejovsky & Jezek (2027, forthcoming OUP).