Advances in Cognitive Systems X (2026) 1-19 Submitted 8/2026; published X/2026

A Unified Account of Concepts and Chunks: Extending Cobweb from Categorization to Composition

Karthik Singaravadivelan KSINGARA3@GATECH.EDU
Pat Langley PATRICK.W.LANGLEY@GMAIL.COM
Institute for the Study of Learning and Expertise, 2164 Staunton Court, Palo Alto, CA 94306 USA

Abstract

Cognitive psychology has studied how people encode, use, and learn concepts that describe categories, as well as how they represent, recognize, and acquire chunks for familiar patterns of elements. However, the literatures on these two topics are nearly disjoint and thus pose a challenge for unified theories of cognition. In this paper, we review Cobweb, a computational account of categorization and concept formation, and propose an extended theory that incorporates chunks and their acquisition. We also present TRELLIS, an implementation of this theory, and illustrate its application to learning context-free grammars, which involve both concept-like and chunk-like elements. In addition, we report experimental results on three synthetic grammars that demonstrate the system's ability to represent syntactic knowledge, use it to parse and generate sentences, and learn compositional structures from sample parses. We conclude by discussing related work on concepts and chunks, along with directions for future research in the area.

1. Introduction

A primary aim of science is to develop general theories that explain as broad a range of phenomena as possible. Within the cognitive systems movement, such theories involve postulates about the types of mental structures stored in memory, the performance processes that operate over these elements, and the learning mechanisms that acquire them. Different paradigms take differing positions on these issues, but they share a concern with generality, attempting for broad coverage with as few assumptions as possible.

In this paper, we propose a unified computational account of two facets of cognition that researchers have long treated as distinct and disconnected. One of these revolves around *concepts*, which Langley (2025) defines as:

- *A concept is a cognitive structure that denotes a set of entities or situations and that characterizes them in terms of their regularities*.

The representation, use, and learning of concepts has been a major topic in both cognitive psychology (Rips et al., 2012) and artificial intelligence (Langley, 1996) since the 1960s. Another research area, less popular than concepts, but still receiving substantial attention, concerns *chunks*, which Langley defines as:

- *A chunk is a cognitive structure that denotes a collection of entities, possibly typed, that form a specified configuration or relational pattern*.

Treatments of chunk encoding, recognition, and acquisition have mainly appeared in the psychological literaure (Miller, 1956; Fountain & Doyle, 2012), but there has been some AI work (Campbell & Berliner, 1983). Topics like concept generalization and similarity have been recurring themes in work on concepts, whereas research on chunks has emphasized the composition of higher-level structures from constituents.

Concepts and chunks have many similarities, but work on the former has emphasized generalization, whereas studies of the latter has focused on composition. In the remaining pages, we present a unified theory that addresses both aspects of cognition. Our approach builds on Cobweb (Fisher, 1987), a computational account of concept representation, organization, retrieval, and formation. We start by review Cobweb's main assumptions, which our framework retains, and then introduce new postulates needed to support chunk-related phenomena. After this, we describe TRELLIS, an implementation of the theory, and report experimental results on grammar induction, which requires both generalization and composition. In closing, we examine related ideas from the literature and note avenues for future research.

2. The Cobweb Framework

Fisher's (1987) Cobweb was an innovative model of categorization and concept formation that combined ideas from decision-tree induction, Bayesian classifiers, and nearest-neighbor methods. The system received considerable attention within the maturing machine learning community and made contact with important psychological findings like basic-level and typicality effects. Excitement about the approach led to multiple extensions, many of them documented in Fisher et al. (1991). In this section, we review the framework's main theoretical tenets, which it shares with its descendants.

2.1 Representation in Cobweb

The Cobweb framework's most basic postulates concern how conceptual knowledge is represented in long-term memory, as later assumptions build directly upon them. These include statements that:

R1. Long-term memory contains two types of structures: instances and concepts.

R2. An instance is a set of attribute-value pairs that describe a single observation.

R3. A concept specifies, for each attribute, a probability distribution over that attribute's values.

That is, we can view each concept as a probabilistic generalization for a set of instances in which each attribute's distribution is independent given the concept. Unlike many other theories of categorization, Cobweb retains both types of structures in memory and we can view each instance as a very specific concept with a heavily skewed distribution.

2.2 Organization in Cobweb

Of course, the contents of long-term memory must be organized in some fashion, so the Cobweb framework also takes strong positions about their arrangement. These include postulates that:

O1. Concepts reside in a taxonomic hierarchy with a single root and instances as terminal nodes.

O2. Each nonterminal node is a probabilistic summary of the instances below it in the taxonomy.

O3. Each nonterminal node has two or more children that partition the instances it summarizes.

This organization is similar to that produced by hierarchical clustering systems, except that each nonterminal node (cluster) in the taxonomy has a concept description rather than a set of instances, which are stored at terminal nodes.

The taxonomy is also similar to a decision tree, except that it stores 'tests' on nodes rather than links. Also, the tests are multivariate in that each child describes distributions over a set of attributes rather than just one. Like a decision tree, a Cobweb hierarchy partitions instances into mutually exclusive sets, but each node is a probabilistic generalization of instances rather than a logical characterization. We can also view paths through the taxonomy as *indexing* nodes in long-term memory, which has implications for performance.

2.3 Performance in Cobweb

Claims about representation and organization are not enough for a complete theory. Cobweb also posits performance mechanisms that use its concept, and their arrangement in long-term memory, to produce behavior. We can summarize the framework's assumptions on this front as:

P1. There are two linked performance mechanisms: *classification* and *prediction*.

The first assigns a case to a category, whereas the second infers any values for missing attributes. This is similar to the distinction in the psychological literature between recognition and recall.

P2. Classification involves sorting an instance downward through the taxonomy.

Thus, classification is much like that in decision trees (Quinlan, 1986), except that Cobweb selects the best child at each level. The criterion it uses is *category utility* (Fisher, 1987), a probabilistic measure that favors children with higher intra-category similarity and lower between-category similarity. The process steps greedily down the tree, selecting the best option at each level.[1]

P3. Sorting continues until reaching a terminal node or no child is better than the current node.

This is another important difference from decision trees, which always sort cases to terminal nodes. In contrast, Cobweb halts at a nonterminal node when none of its children have a better score.

P4. At the halting node, prediction imputes the most probable values for unknown attributes.

Again, this scheme is similar to that in decision trees, but rather than predicting a class label, Cobweb infers values for all missing attributes. Fisher (1987) referred to this ability as *flexible prediction*, which is form of pattern completion.

2.4 Learning in Cobweb

Finally, the theory make statements about how conceptual knowledge is acquired through learning. These postulates include:

L1. Learning is an incremental, online process that is fully interleaved with performance.

1. Some variants, like Cobweb/4V (Barari et al., 2024), replace this greedy descent with best-first search, but we will not adopt that scheme here.

L2. Learning is unsupervised in that instances do not come with class labels.

That is, the very act of sorting each new instance leads to changes in the content of existing concepts and to their organization in long-term memory. And the learner must invent its own categories rather than obtain them from an outside source.[2]

Furthermore, the Cobweb theory posits two varieties of incremental learning, one statistical in character and the other structural in nature:

L3. Sorting an instance through a node in the taxonomy updates its associated probabilities.

This mechanism simply alter counts that reflect the prior probability of a concept given its parent and the probability of each attribute value given that concept, much as in naive Bayesian classifiers. These changes are determined entirely by the instance's content and which path it traverses.

L4. At each node, sorting considers four structural options: adding the instance to an existing child, creating a new child, splitting a child, or merging two children.

The first option does revise taxonomic structure; the probabilities for the child are simply updated. The second alternative creates a new child based on the instance. The latter two actions lead to local restructuring before sorting continues. The same evaluation criterion used in routing determines which of these steps to take. Because Cobweb learns incrementally, the order in which it encounters instances can influence the structure of the acquired taxonomy. Merging and splitting nodes can mitigate these order effects, resulting in hierarchies with better organizations.

The framework also invokes a deterministic restructuring action when an instance reaches a terminal node N. In these cases, the learning process creates two new terminal nodes, one based on the stored instance and the other based on the new case. The existing node N becomes their parents, with its probabilities updated based on values in the recent instance. Together, these operations support the incremental and unsupervised acquisition of a taxonomy of probabilistic concepts.

3. A Unified Theory of Chunking and Categorization

Cobweb unifies ideas from decision trees, naive Bayesian classifiers, and nearest neighbor in an elegant framework, and it makes contact with the psychology literature on categorization and concept learning. However, it does not explain the representation, organization, use, or acquisition of chunks, which are equally important facets of human and machine cognition. The issue is that Cobweb has nothing to say about *composition*, which is central to chunks and their processing. This section introduces new postulates that, when combined with those just described, provide an extended theory that links concepts and chunks. We use examples from language syntax to clarify the ideas because they require both capabilities, although the framework has much broader implications.

3.1 New Representation Postulates

The extended theory's representation retains Cobweb's assumptions, including the distinction between instances and concepts, but it introduces additional postulates that make it substantially richer. These include:

2. Training cases can include a special 'class' attribute, but this is treated no differently from others.

R4. An *experience* is a set of elements and the local relations among them.

For example, someone reading text encounters a set of words that are arranged in a linear sequence. The words are elements that are related by a *before* or a *left-of* relation.

R5. Every element is either a *primitive* structure or a *composite* structure.

Primitive elements are the basic building blocks of memory, similar to instances and concepts in the Cobweb framework. In contrast, a composite is an element composed from primitive elements or lower-level composites, which are specified in its content description. Composite structures correspond to *chunks*.

R6. Each element can be described by its *content* or its *context*.

The first of these specifies an element's structure in terms of its constituents (e.g., *black cat*). The second specifies other elements in its vicinity (e.g., *the*, *small*, *chased*), along with their relations to the element (e.g., *before* or *after*). We will refer to these as *content instances* and *context instances*.

3.2 New Organization Postulates

The expanded theory also retains Cobweb's commitment to organizing concepts in taxonomies, but it requires further assumptions to handle its treatment of structures that are composed of others. Taken together, the content fields of composite structures impose a *partonomic* hierarchy, but this is implicit. However, the distinction between content and content instances leads to additional organizational postulates. These include:

O4. Long-term memory contains a *content taxonomy* that organizes concepts by their contents.

For primitive elements, this hierarchy corresponds to that in Cobweb, which clusters instances that have similar attribute values. However, for composite elements, it groups cases that have similar sets of constituents. For example, the chunks *black dog* and *black cat* share the element sl black.

O5. Long-term memory contains a *context taxonomy* that organizes concepts by their contexts.

This hierarchy instead organizes concepts that occur in similar contexts. For instance, it might group the words *dog* and *cat* together because they appear in like surroundings. The same applies to composite structures, like different types of noun phrases, which occur both before and after verbs. These two taxonomies are intertwined in that nodes in one point to nodes in the other. However, keeping them separate recognizes that one can organize memory either in terms of content (i.e., the constituents of chunks) or in terms of context (i.e, the situations in which chunks occur).

3.3 New Performance Postulates

Performance in Cobweb focuses on classification and prediction, but the introduction of chunks supports new types of cognitive activities. In language processing, these are referred to *parsing* and *generation*, and the extended theory makes commitments to their operation as well.

P5. Parsing iteratively creates a partonomic tree from the bottom up, adding chunks that elaborate an input experience.

In language processing, this involves creating a parse tree from the words in a sentence by introducing nonterminal symbols like *noun*, *noun phrase*, and *verb phrase*.

P6. Parsing sorts candidates through both taxonomies and selects the best recognized chunk.

This introduces a new notion, not present in Cobweb, of a *recognition threshold* that candidates must exceed with respect to a node in the taxonomy before being considered familiar enough to proceed. From among the candidates that satisfy this test, the best-scoring chunk is added to the growing parse tree.

P7. Parsing halts when no candidate is recognized or when only one composite element remains.

The content score asks whether the composition looks like one used before; the context score asks whether the surroundings look like ones that have hosted similar composites. The termination condition allows graceful failure: when nothing clears the threshold, parsing ends ends rather than asserting spurious chunks at the boundary of competence.

The generation process also moves beyond Cobweb's capabilities. This proceeds in the opposite direction from parsing with the aim of creating a well-formed collection of new elements.

P8. Generation iteratively expands a composite element from the top down, decomposing chunks to produce an experience.

This is the inverse of parsing. Rather than combining elements into larger structures, it starts from a single composite and rewrites it into constituents, then rewrites those in turn until only primitives remain. For language, expanding a *sentence* into a *noun phrase* and *verb phrase*, decomposing them, and so forth yields a grammatical sequence of words.

P9. Generation recalls candidate decompositions from the content taxonomy and conditions them on the context taxonomy.

This assumes a counterpart to the recognition threshold, which determines whether a candidate is familiar enough to use in parsing. Here we need a threshold that determines whether to use a content concept to recall an expansion (Gupta et al., 2025). If this is too low, then generation will draw on very specific nodes supported by only a few cases; if too hight, it will favor general nodes that will mix constituents which do not belong together. Recall samples from the most specific chunks that exceed the threshold to generate candidate expansions, then uses the context hierarchy to rank them.

P10. Generation terminates when every expanded element is a primitive.

The two taxonomies play complementary roles throughout. The content hierarchy specifies candidate chunk expansions, whereas the context hierarchy specifies how well they fit into their surroundings. For grammatical expertise, the first is analogous to individual rewrite rules, such as *VP => Verb NP*, and the second to knowledge about which rule to choose. Because generation recombines familiar parts in new ways, it can produce experiences the system has never seen while staying consistent with what it has learned, which for language means grammatical sentences beyond those in the training corpus.

3.4 New Learning Postulates

Learning in the extended theory does not require any new postulates about mechanisms, as they remain unchanged from the Cobweb framework. However, it does make claims about the inputs to these already established processes. These include:

- Learning incorporates content and context instances into the content and context hierarchies, respectively.
- Learning incorporates instances for composite elements into both taxonomies, but those for primitive elements only into the context taxonomy.

Neither statement should be surprising, and they follow from two other commitments: (a) the distinction between content and context instances, and (a) interleaving learning with performance. For example, there is no separate learning mechanism for creating new chunks, as parsing already produces candidate chunks during its operation. Learning simply treats these new structures as data for use in updating the taxonomies.

This does not reflect a limitation of Cobweb's account of learning. If anything, it provides evidence for substantial generality. Cobweb was designed to learn from independent and identically distributed data, which sequential domains like language clearly violate. Yet combining its original learning methods with the distinction between content and context instances, it can cover types of phenomena for which its creator did not intend.

4. TRELLIS: An Implementation of the Theory

The extended theory offers an account that unifies concepts and chunks, but it is abstract. To make it operational, we have developed TRELLIS, which embeds the postulates in an implemented system that invokes Cobweb as a subroutine. In this section, we describe how TRELLIS elaborates the theory and illustrate its choices with examples from language processing. We delay covering details like formulas for the recognition score until Appendix B.

4.1 Representation in TRELLIS

Every element in TRELLIS includes two fields: *content* specifies the parts that compose the element, whereas *context* names its neighbors (MacLellan et al., 2022). Primitives have only a context field, while composite have both. Figure 1 illustrates one primitive and one composite drawn from the parse of a small sentence. A composite's content field encodes its left and right children with two attributes: an identifier that points to the context taxonomy (referring to the child itself) and a complexity tag (a small integer denoting its nesting depth). Both reside in the context hierarchy's identifier space rather than the surface vocabulary, so the content hierarchy never sees a surface word. Every value encountering during a content sort was created by a context sort of the corresponding child. The complexity tag lets the content taxonomy identify composites whose children have similar contexts but differ in structural depth, such as NP=DET+N vs. S=NP+VP.

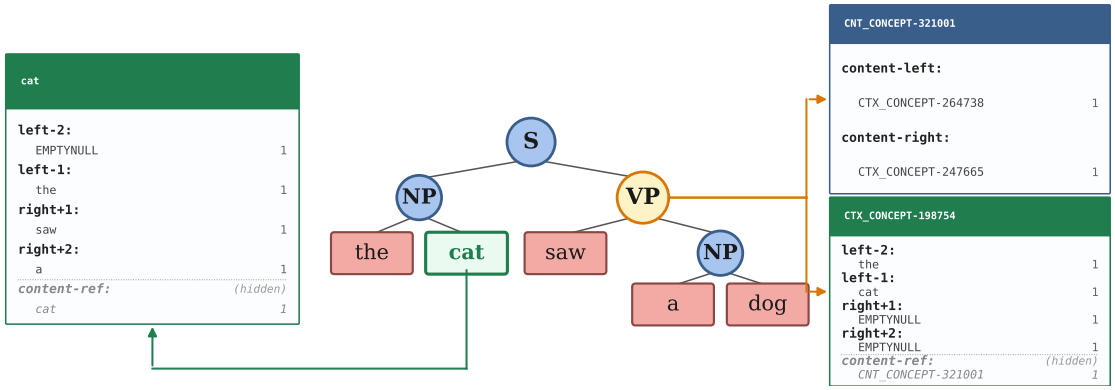

Figure 1. Illustrative representation of one primitive ("cat", green) and one composite ("saw a dog", amber) from a parse of "the cat saw a dog". Each box header is the identifier in long-term memory to which the instance was sorted. The composite's content field points into the context hierarchy, keeping the two descriptions in complementary formats. Window width, distance weighting, and content-bag width are run-time values explained in Appendix B.

4.2 Organization in TRELLIS

Following the extended theory, TRELLIS organizes long-term memory into two Cobweb taxonomies. The content hierarchy stores the content descriptions of composites and the context hierarchy stores the contexts in which they occur. They are linked in that a composite's content refers to identifiers in the context hierarchy, so every description it encounters is produced by an earlier sort through the context hierarchy. Neither must trade off conceptual facets for compositional facets because both data structures carry each of them. Composites with structurally similar children appear near each other in the content tree, reflecting a shared role (say, as an NP-filling constituent), while a noun and an adjective phrase appear near each other in the context tree if they appear in comparable neighborhoods. Figure 2 shows leaf-level subtrees of each hierarchy and the cross-hierarchy references that link them.

4.3 Parsing and Generation in TRELLIS

The TRELLIS system includes two modules that expand on the theory's performance postulates. A parsing mechanism constructs parses from the bottom up by iteratively creating chunks that elaborate on the input experience. In addition, a generation mechanism constructs output experiences from the top down by iteratively decomposing chunks until reaching primitives.

Parsing carries out greedy search through the space of possible parses. The module starts with a frontier of primitive elements, such as the words in a sentene, and repeatedly proposes candidate chunks over adjacent pairs. It sorts each candidate's content and context descriptions through their respective hierarchies, scoring them by how well the candidate matches the concept at which sorting halts relative to its competitors (Appendix B). TRELLIS sums the two scores to produce a combined recognition score. For "the cat saw a dog", the first step would consider four adjacent word pairs.

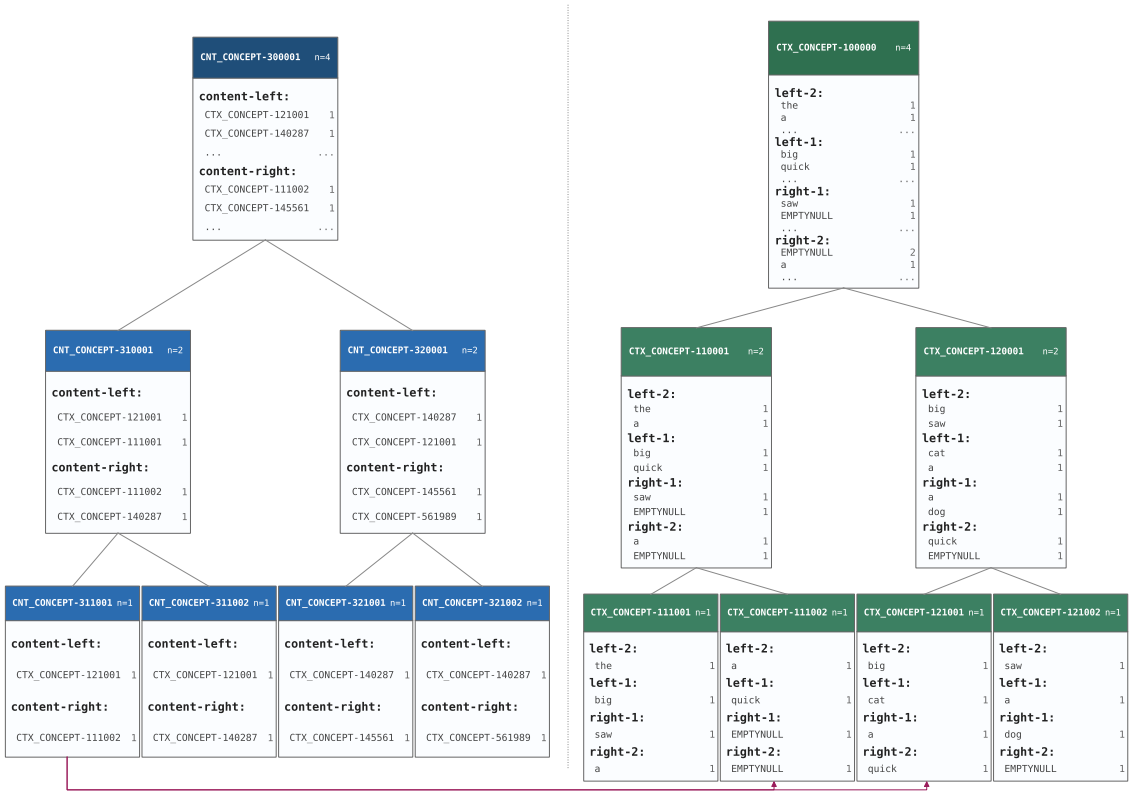

Figure 2. Leaf-level subtrees of the content (left) and context (right) hierarchies. Arrows show the cross-hierarchy link: a content leaf's child specify the context concepts that describe its children.

TRELLIS admits a candidate only when a concept through which it was sorted has been visited enough times, with primitives gated on the context hierarchy and composites on the content hierarchy. The module walks upward from the node at which sorting halted; the candidate is recognized if the count of some nonroot ancestor exceeds the recognition threshold τ. This gating scheme suppresses selection early in learning but has little effect later. After this, TRELLIS scores candidates by their posterior probabilities based on nodes in the two hierarchies and replaces the selected candidate's constituents with the composite element on the frontier. Parsing terminates if the module recognizes none of the candidate chunks, in which case it halts with a partial parse.

Generation proceeds in the opposite direction to produce a grammatical sentence. This process begins from a seed node that is an unexpanded composite with an identifier in the content hierarchy. The module samples from a *recombination pool* associated with the seed, decomposing the result into two children. To create the pool, TRELLIS walks up from the seed's leaf until it reaches an ancestor whose count exceeds τ, a parameter similar to the recognition threshold. At this point, it considers the constituent pairs stored at this node, each of which has an associated probability. Thus, the system draws from a set of decompositions it has seen often enough to treat as acceptable,

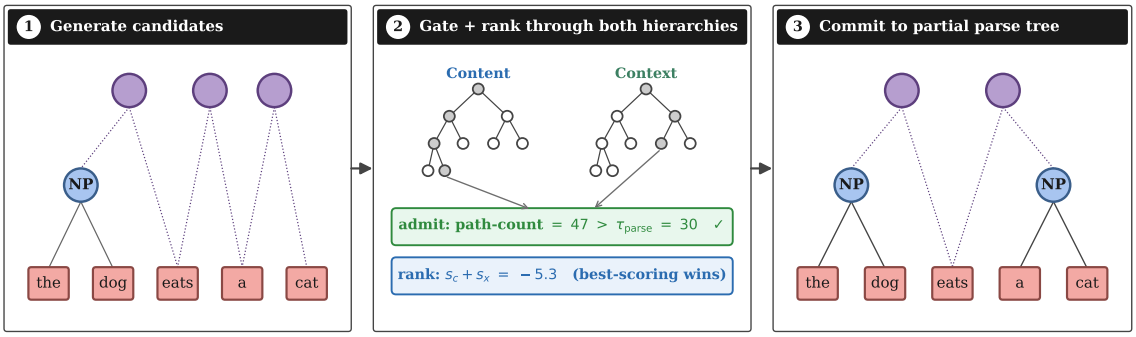

Figure 3. One parsing step: (1) forms candidates over adjacent frontier pairs; (2) sorts each candidate through both hierarchies, admitting it iff some nonroot ancestor on the sort path has instance count greater than τ_{parse}, and ranking the admitted candidates by $s_c + s_x$; (3) commits the top-ranked one to the parse tree. The numeric values in panel (2) are illustrative.

from which it then samples, rather than ones with little empirical support. The sampling process can produce new combinations that no single training experience has contained.

After selecting a decomposition, TRELLIS uses the context hierarchy to filter candidates. If a candidate's parent does not expect one of them, then the system eliminates it from competition to ensure it cannot emit an incorrect constituent. If the filter eliminates all candidates, say because the context concept has not yet acquired any compatible chunks, the system falls back to a wider, less-filtered pool that incurs a small risk for errors of commission in return for broader coverage. Generation halts when every leaf of the parse tree is a primitive structure.

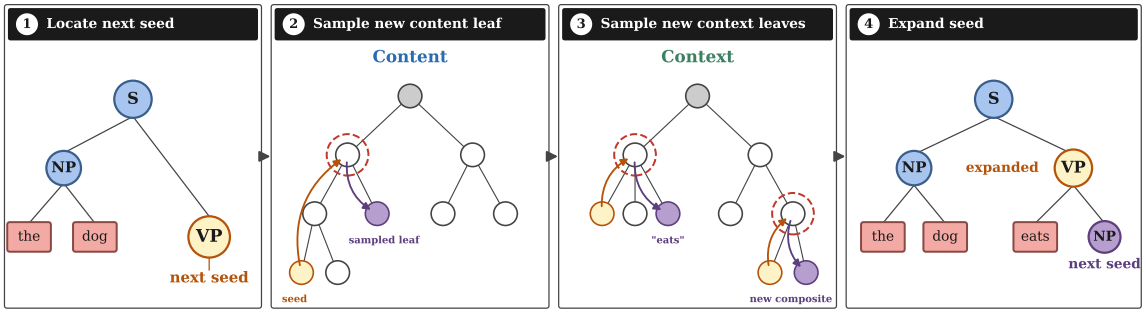

Figure 4. One generation step: (1) select the next seed; (2) sample a fresh content leaf from a well-supported neighborhood above it; (3) resolve each of the sampled leaf's two child references against the class the parent expects; (4) expand the seed with these results.

4.4 Learning in TRELLIS

Learning in TRELLIS does not require any new mechanisms beyond those in Cobweb, but it operates over what we have called 'experiences', which are are collections of related elements. The system is

provided not only with training experiences, but also with desired parse trees for them. For example, the sentence *the big dog chased the black cat* would come with a parse into phrases at various levels. This constitutes a form of supervised learning, although nonterminal nodes in the parse trees are not themselves labeled.

As with parsing, TRELLIS learns iteratively from the bottom up. This involves selecting an element on the frontier (initially the set of primitive structures) and sorting it through taxonomies. The system only sends primitive elements through the context hierarchy, as they have no internal structure. Composite elements go through both taxonomies, which incorporates them based on either their content or content descriptions. TRELLIS sorts composite elements only after their constituents have been processed.

Once the system has handled all elements in a given parsed experience, it turns to the next one, continuing until processing all available training data. Thus, learning occurs both incrementally within each experience and across a succession of experiences. TRELLIS has no separate routine for creating chunks: every composite structure becomes a new node in the hierarchies. This node corresponds to what later parsing uses as the category label when the system encounters a structurally similar chunk.

5. Evaluation of TRELLIS

We maintain that TRELLIS is a faithful implementation of our unified theory of concepts and chunks, but whether it operates as intended is another question. To answer it, designed and carried out experiments on grammar acquisition that examined its ability to represent, organize, use, and learn linked content and context taxonomies. In this section, we describe the experimental design and test domans, after which we present the results we obtained.

5.1 Experimental Design

The acquisition of grammatical expertise is more difficult to study than learning for classification tasks. We chose to follow Langley and Stromsten's (2000) methodology, which recorded two dependent variables: *errors of omission* and *errors of commission*. The first measures the degree to which a learned grammar is overly specific, in that it either fails to parse or generate some legal sentences with correct parses. The second measures the degree of overgeneralization, in that the grammar parses or generates illegal constructions.

For our initial studies, we devised synthetic context-free grammars to generate training data, rather than a corpus of natural sentences like the Penn Treebank. To compute the two metrics, we used the target grammar to generate training sentences and associated parse trees, then let TRELLIS to learn concepts and chunks from these experiences. To measure errors of omission, we used the target grammar to generate novel sentences, then used TRELLIS' parsing module to attempt parsing them. Similarly, to measure errors of commission, we called the generation module to produce sentences with the learned taxonomies, then used the target grammar to attempt parsing them. Because the system can acquire incomplete grammars, especially early in training, we did not use an all-or-none measure of success. Instead, we gave partial credit for each substructure

of the target parse. In particular, we compared the *span* of words covered by each subtree,[3] as TRELLIS' memory has no class labels on nonterminals. For each test sentence, we took the fraction of substructures matched between the target and learned sentences.

The main purpose of our experiments was to reveal whether TRELLIS induces accurate grammatical knowledge, or whether it forms overly specific or overly general structures. We were also interested in the rate of improvement as measured by learning curves. In addition, we also wanted to know how behavior scaled in response to two types of grammatical complexity. One involved the number of nonterminal categories, such as *NP* and *VP*, in the target grammar. Raising this count increases the number of chunk types the content hierarchy must encode and acquire. The other concerned the words per part-of-speech class. Increasing this number requires the context hierarchy to generalize across more surface forms.

In one experiment, we varied the number of nonterminal symbols, using three grammars with 3, 6, and 8 nonterminals, which we will refer to as small, medium, and large, while holding number of words per class constant. In a second study, we varied the words per part of speech class from 11 (low) to 22 (medium) to high (39), while holding the number of nonterminals at 6. Appendix A presents the rewrite rules and lexical items for each grammar, most of which involve recursion, while Appendix B provides the system parameters, which we did not vary. Each experimental condition used 400 sentence-parse pairs generated from the target grammar. To measure errors of omission, we used five-fold cross validation, holding out a different subset of 40 pairs for each fold for testing and the remaining 320 pairs for training. To measure errors of commission, we used the same five learned taxonomies to generate 40 sentence-parse pairs. Thus, we averaged each dependent variable over five runs based on different training data.

5.2 Experimental Results

Figure 5 shows the results of the first study, plotting one minus the omission rate on the left and one minus the commission rate on the right as a function of the number of training sentences. Each graph shows three learning curves, one for each of the grammatical complexity levels. The most important result is that TRELLIS is able to acquire grammatical expertise well, with both errors of omission and commission dropping to low percentages even for the most complex grammar. Moreover, learning occurs rapidly, with both errors dropping to 20 percent almost immediately and then improving gradually with more training sentences. Increasing the number of nonterminals slows the rate of learning, as the system requires examples of every chunk type to master the grammar, but the degradation is graceful. Inspection of the content hierarchy showed reasonable concepts for each nonterminal in the target grammar, as desired.

The plots in Figure 6 reveal similar results from the experiment that varied the lexicon size. As before, TRELLIS comes close to mastering the grammars from a moderate number of training sentences, with low errors of omission and commission for all complexity levels. Increasing the number of words per part of speech class slows the rate of learning, but improvement is rapid in all conditions and the reduction with more complex grammars is modest. Inspection of the context hierarchy showed the system grouped nouns with nouns, verbs with verbs, and so forth, as desired.

3. The grammar induction literature typically calls such spans *brackets*; we use *substructure* to avoid the Penn Treebank's labelled-bracket convention, which combines structural span agreement with part-of-speech or phrasal labels.

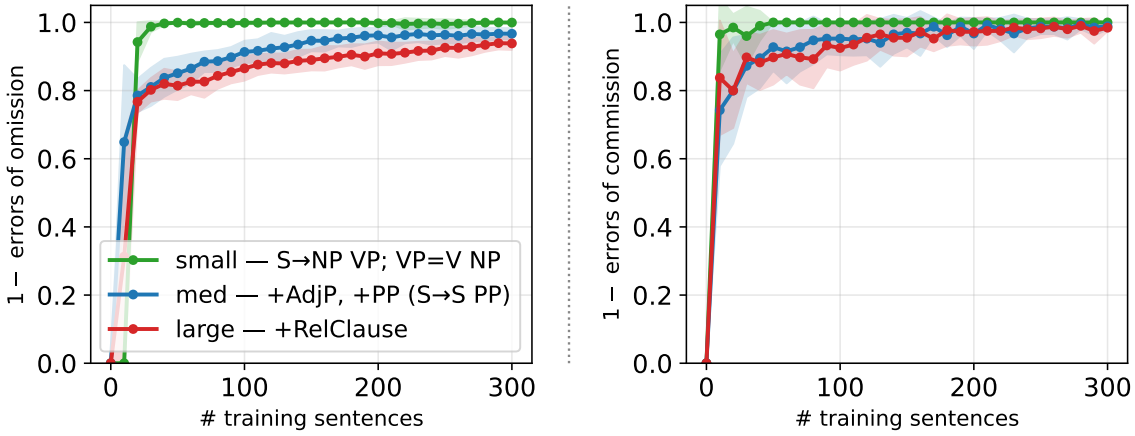

Figure 5. $1-\text{omissionrate}$ (parse, left) and $1-\text{commissionrate}$ (generation, right) across the three grammars (five data sets, $\pm 1\sigma$ b and).

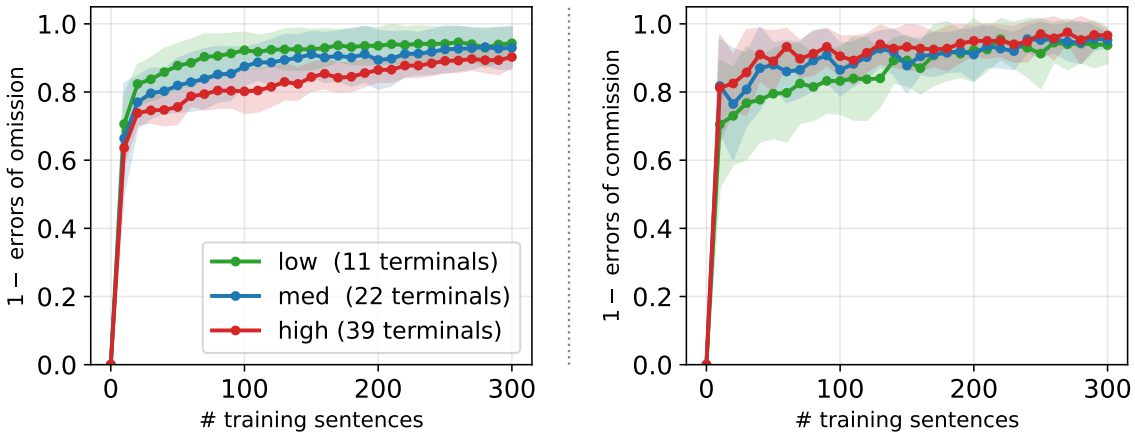

Figure 6. $1-\text{omissionrate}$ (parse, left) and $1-\text{commissionrate}$ (generation, right) across the three lexicon sizes on the MED grammar (five data sets, $\pm 1\sigma$ band).

In general, both experiments produced results consistent with our expectations. They showed that TRELLIS can acquire accurate grammatical expertise, with low errors of omission and commission, from a moderate number of sentence-parse pairs. The rate of learning was rapid, but increasing the number of nonterminal symbols and lexical items led to modest slowing, suggesting that the system scales reasonably in response to both forms of grammatical complexity. Examination of the content hierarchy showed concepts that mapped well onto nonterminals in the target grammar and inspection of the context hierarchy showed categories for target word classes like determiners, adjectives, nouns, verbs, and prepositions.

Overall, these results indicate that TRELLIS' extension of Cobweb to support linked content and context hierarchies support the rapid acquisition of both chunks and concepts, both of which are important in grammar induction. We considered briefly the idea of a lesion study that compared TRELLIS' behavior to that of Cobweb, but rejected it because the earlier system only handles individual items and not collections of related items like words in sentences. The experiments demonstrate clearly that the new system's ability to represent, use, and learn both concepts and chunks give it this ability. Certainly many other systems can acquire grammatical knowledge, but few of them offer a unified account of chunks and concepts.

6. Related Research on Concepts and Chunks

Our unified theory offers a promising account of how contents and chunks are two sides of the same coin, and the TRELLIS results suggest its viability. Yet some other frameworks have addressed the problem and they merit discussion. The closest examples involve other variants of Cobweb, starting with LABYRINTH (Thompson & Langley, 1991), which encoded, used, and learned both primitive and composite concepts, the latter similar to chunks. However, it lacked the notion of a contextual taxonomy and suffered from substantial order effects. Another descendant, Trestle (MacLellan et al., 2016), operated over instances organized in a partonomic hierarchy, but flattened the representation during processing. A more recent convolutional version of Cobweb (MacLellan & Thakur, 2021) learns over multiple levels of image descriptions, giving some effects of chunks but not treating them explicitly.

Research on discrimination networks shares many features with the Cobweb family. EPAM (Feigenbaum, 1961; Richman, 1991) organizes knowledge in a univariate taxonomy that it learns with incremental, unsupervised methods, and its operations for extending the network are the same as Cobweb's but predated them by 25 years. The CHREST architecture (Gobet & Lane, 2012) extends EPAM on multiple fronts, including the addition of lateral links among nodes, similar to the interleaving in TRELLIS. Moreover, the framework has always supported use of terminal nodes (e.g., letters) as tests for recognizing higher-level discriminants (e.g., words). But despite similarities in the theories treatment of categorization, their accounts of chunking follow different paths.

Although seldom recognized, neural networks (Rumelhart et al., 1986) also offer a unified narrative about concepts and chunks. Nodes in such a network have very different semantics from that in Cobweb and EPAM, but we can still view them as encoding graded concepts that can match to a greater or lesser degree. But we can also view them as chunks that combine simpler elements, as demonstrated by visualizations of learned 'features' as weight maps. Taken together, this framework would provide a compelling unification of concepts and chunks if only it included operations for creating new structures, rather than merely updating weights, and it support more rapid, human-like learning.

A fourth paradigm, with fewer links to cognitive science, involves the construction of context-free grammars from training cases. Classic systems in this paradigm (Wolff, 1982; Langley & Stromsten, 2000) include operations for inventing nonterminal symbols (creating chunks) and merging them (forming classes). Although most such systems are strongly committed to symbolic representations, the learned structures are organized in ways similar to neural networks. Research on

predicate invention for inductive logic programming (Muggleton & Buntine, 1988) have championed related ideas. Despite many differences from Cobweb and EPAM, it provides an important source of ideas about concepts and chunks for those working in other paradigms.

7. Concluding Remarks

In this paper, we presented a unified computational theory of concepts and chunks. We reviewed the assumptions behind Cobweb, which forms a taxonomy of probabilistic concepts that it uses for classification and prediction. Next we introduced additional postulates that extend the framework to support both categorization of experience and composition into chunks. The extended framework assumes two linked taxonomies, one based on content descriptions and another on contextual ones. We also covered *Trellis*, an implementation of the theory that invokes Cobweb as a subroutine and tested the system on its ability to learn grammatical expertise from sentence-parse pairs.

Our theory bridges the longstanding chasm between accounts of concepts and chunks, which are usually treated as disjoint phenomena. Moreover, experimental results with TRELLIS suggest not only that it can acquire syntactic knowledge, which involves both categories and chunks, but does so from a modest number of training sentences. Nevertheless, we should follow up on our initial findings with further work that tackles some open issues:

- Can the approach learn syntactic content from sentences alone, without parse trees, like some earlier systems for grammar induction?
- Can TRELLIS represent, use, and acquire not only context-free but also context-sensitive grammars, as suggested by its context taxonomy?
- Can the framework support learning both concepts and chunks in other arenas that involve hierarchical structures, such as vision?
- Can we extend TRELLIS to support a new type of large language model that has interpretable structures and is vastly more sample efficient?

We hope to address each of these challenges in future work by continuing to incorporate ideas from earlier efforts and combining them in novel ways, following the tradition established by other researchers in the cognitive systems movement.

Acknowledgements

The research reported here was supported by Grant No. FA9550-23-1-0580 from the US Air Force Office of Scientific Research, which is not responsible for its contents. We thank Chris MacLellan, Zekun Wang, Xin Lian, Kyle Moore, and friends at the Teachable AI Lab for useful discussions.

References

Barari, N., Lian, X., & MacLellan, C. J. (2024). Avoiding catastrophic forgetting in visual classification using human concept formation. *Proceedings of the Eleventh Annual Conference on Advances in Cognitive Systems*.

Feigenbaum, E. A. (1961). The simulation of verbal learning behavior. *Proceedings of the Western Joint Computer Conference* (pp. 121–132). Los Angeles, CA.

Fisher, D. H. (1987). Knowledge acquisition via incremental conceptual clustering. *Machine Learning*, *2*, 139–172.

Fisher, D. H., Pazzani, M. J., & Langley, P. (Eds.). (1991). *Concept formation: Knowledge and experience in unsupervised learning*. San Francisco, CA: Morgan Kaufmann.

Gobet, F., & Lane, P. C. R. (2012). CHREST: A model of perception, learning, and problem solving. In N. M. Seel (Ed.), *Encyclopedia of the sciences of learning*, 556–559. New York, NY: Springer.

Gupta, A., Singaravadivelan, K., & Wang, Z. (2025). Hierarchical semantic retrieval with cobweb. From `https://arxiv.org/abs/2510.02539`.

Langley, P. (1996). *Elements of machine learning*. San Francisco, CA: Morgan Kaufmann.

Langley, P. (2025). Concepts and chunks in cognitive systems. *Advances in Cognitive Systems*, *11*, 1–12.

Langley, P., & Stromsten, S. (2000). Learning context-free grammars with a simplicity bias. *Proceedings of the Eleventh European Conference on Machine Learning* (pp. 220–228). Barcelona, Spain: Springer.

MacLellan, C. J., Harpstead, E., Aleven, V., & Koedinger, K. R. (2016). TRESTLE: A model of concept formation in structured domains. *Advances in Cognitive Systems* (pp. 131–150).

MacLellan, C. J., Matsakis, P., & Langley, P. (2022). Efficient induction of language models via probabilistic concept formation. *Proceedings of the Tenth Annual Conference on Advances in Cognitive Systems*.

MacLellan, C. J., & Thakur, H. (2021). Convolutional Cobweb: A model of incremental learning from 2D images. *Proceedings of the Ninth Annual Conference on Advances in Cognitive Systems*.

Muggleton, S., & Buntine, W. (1988). Machine invention of first-order predicates by inverting resolution. *Proceedings of the Fifth International Conference on Machine Learning* (pp. 339–352). Ann Arbor, MI: Morgan Kaufmann.

Quinlan, J. R. (1986). Induction of decision trees. *Machine Learning*, *1*, 81–106.

Richman, H. B. (1991). Discrimination net models of concept formation. In D. H. Fisher, M. J. Pazzani, & P. Langley (Eds.), *Concept formation: Knowledge and experience in unsupervised learning*. San Francisco, CA: Morgan Kaufmann.

Rips, L. J., Smith, E. E., & Medin, D. L. (2012). Concepts and categories: Memory, meaning, and metaphysics. In K. J. Holyoak & R. G. Morrison (Eds.), *The oxford handbook of thinking and reasoning*, 177–209. Oxford, UK: Oxford University Press.

Rumelhart, D. E., Hinton, G. E., & Williams, R. J. (1986). Learning internal representations by back-propagating errors. In D. E. Rumelhart & J. L. McClelland (Eds.), *Parallel distributed processing*, volume 1, 318–362. Cambridge, MA: MIT Press.

Thompson, K., & Langley, P. (1991). Concept formation in structured domains. In D. H. Fisher, M. J. Pazzani, & P. Langley (Eds.), *Concept formation: Knowledge and experience in unsupervised learning*, 127–161. San Francisco, CA: Morgan Kaufmann.

Wolff, J. G. (1982). Language acquisition, data compression, and generalization. *Language and Communication* (pp. 57–89).

Appendix A. Synthetic Grammars

This appendix lists the productions and terminal lexicons of the three synthetic context-free grammars used in Section 5. Every non-terminal expansion is strictly binary. Sentences are derived by sampling from the right-hand-side alternatives; most alternatives are equally weighted, with a few exceptions that keep RelClause sentences near 20% of the LARGE corpus and adjective-recursion chains short: MED NP splits 3:2 (bare-noun / AdjP); LARGE NP splits 6:2:1 (bare-noun / AdjP / Nbar); AdjP and Nbar each split 2:1; and the AdjP-recursion continuation decays as $(1/3)^d$ with depth d. The three grammars are strictly nested: every production and terminal of SMALL is retained in MED, and every production and terminal of MED is retained in LARGE.

SMALL. A determiner-noun NP and a transitive VP, sharing MED's full determiner, noun, and verb lexicons (only the Adj and P classes and the modifier productions are absent).

S	→ NP VP
NP	→ Det N
VP	→ V NP
Det	→ the | a
N	→ cat | dog | man | woman | park | telescope
V	→ saw | liked | chased | found | admired

MED. Extends SMALL with adjective recursion through an AdjP head that always carries at least one adjective, prepositional phrases, and a packaged direct object plus PP through the intermediate VPobj; it introduces the Adj and P classes (additions over SMALL in **bold**). Bare-noun NPs use the no-modifier shortcut "NP → Det N" so the grammar never produces a unary syntactic node.

S	→ NP VP
NP	→ Det N | Det **AdjP**
AdjP	→ Adj N | Adj AdjP
VP	→ V NP | V **VPobj**
VPobj	→ NP PP
PP	→ P NP
Det	→ the | a
N	→ cat | dog | man | woman | park | telescope
Adj	→ big | small | red | quick | lazy
V	→ saw | liked | chased | found | admired
P	→ with | in | on | under

LARGE. Adds relative clauses on top of MED: a relative-clause NP shape (Det Nbar), the Nbar and RelClause non-terminals, and the RelPro part-of-speech class, together with a handful of extra terminals (additions over MED in **bold**). Splitting AdjP (adjective-only) from Nbar (head with relative clause) keeps each label semantically faithful; recursion is weighted toward the non-recursive alternatives so RelClause sentences do not dominate the corpus.

S	→ NP VP
NP	→ Det N | Det AdjP | Det **Nbar**
AdjP	→ Adj N | Adj AdjP
Nbar	→ N RelClause | AdjP RelClause
VP	→ V NP | V VPobj
VPobj	→ NP PP
RelClause	→ RelPro VP
PP	→ P NP
Det	→ the | a | **this** | **that**
N	→ cat | dog | man | woman | park | telescope | **boy** | **girl** | **teacher**
Adj	→ big | small | red | quick | lazy | **tall** | **curious**
RelPro	→ who | that | which
V	→ saw | liked | chased | found | admired | **carried** | **read**
P	→ with | in | on | under | **near**

Terminal-experiment lexicons. The terminal-vocabulary experiment (Section **??**) holds the MED productions fixed and varies only the lexicon, drawing the first k words of a shared pool for each part-of-speech class. The three variants are strictly nested (LOW $\subset$ MED $\subset$ HIGH):

Part of speech	LOW (11)	MED (22)	HIGH (39)
Det	the, a	the, a	the, a, this, that
N	cat, dog, man	cat, dog, man, woman, park, telescope	cat, dog, man, woman, park, telescope, robot, apple, book, girl, boy, bird
Adj	big, small	big, small, red, quick, lazy	big, small, red, quick, lazy, ancient, blue, tall, wise
V	saw, liked	saw, liked, chased, found, admired	saw, liked, chased, found, admired, carried, read, knew
P	with, in	with, in, on, under	with, in, on, under, near, behind

Appendix B. Implementation Formulas

Recognition score. The Cobweb log-probability of an instance $\mathbf{i}$ under a concept C is, by R3,

$$\log P(\mathbf{i} \mid C) = \sum_{(a,v)\in\mathbf{i}} \log P(v \mid C, a), \qquad P(v \mid C, a) = \frac{n_{C,a,v} + \alpha_a}{n_{C,a} + V_a \alpha_a},$$

where $n_{C,a,v}$ is the count of value v for attribute a at C, V_a the value-vocabulary size, and α_a the uniform-prior smoothing (R3). Candidates are ranked by the *class posterior* $\log P(C \mid \mathbf{i}) =$

$\log P(\mathbf{i} \mid C) + \log P(C) - \log P(\mathbf{i})$: normalizing out the instance marginal makes the score discriminative, so a non-constituent that fits nowhere in particular scores low. The ranking key sums the content and context posteriors, $s_c + s_x$.

Recognition threshold. The sort path of a candidate is walked upward from its destination leaf until a non-root ancestor's instance count exceeds the recognition threshold τ_{parse}; the candidate is admitted at that point and rejected only if the walk reaches the root without a match. Primitives are gated on their context-hierarchy path and composites on their content-hierarchy path, since those are the paths that identify them. In every reported condition we use $\tau_{\text{parse}} = 30$; the gate suppresses commits during the first handful of training sentences and is essentially transparent thereafter.

Recall threshold. The same climbing-ancestor notion realizes the level P6 samples from: the anchor above a seed leaf is the deepest (most specific) ancestor whose count exceeds the recall threshold $\tau_{\text{gen}} = 50$ - as per the postulates, we identify this as the recall whereas parsing utilizes the . During training every committed chunk is recorded as a tuple $(\text{parent-leaf}, \text{left-child}, \text{right-child})$, indexed by both its parent-leaf identifier and that leaf's anchor. Generation samples a sentence-root chunk as the seed and recurses on each child: primitives are emitted directly; composites are expanded by sampling a chunk from the anchor's replay pool, restricted to those whose own context class matches what the parent expects at that slot (P7 conditioning). The pool makes recombinations no single leaf could produce reachable, while context conditioning keeps the wider pool from emitting a wrong-type constituent.

Parameter values. Across every condition: $\alpha_{\text{content}} = 10^{-4}$, $\alpha_{\text{context}} = 10^{-5}$, $\tau_{\text{parse}} = 30$, $\tau_{\text{gen}} = 50$. Context windows span five tokens with distance-based weighting; each content instance stores a bag of up to three concept identifiers drawn from depth four of the context hierarchy.

Content-instance encoding. Each content attribute value is a small bag over concept identifiers rather than a single symbol, preserving the concept's own uncertainty rather than forcing a commit to its most-likely value. A visible complexity tag on each child lets category utility separate composites whose children share a distributional context but differ in structural depth (NP=DET+N versus S=NP+VP, which share the same right-hand-side signature at the coarsest granularity). Because the context hierarchy restructures as new instances are absorbed, content references can dangle over time; TRELLIS canonicalizes each reference to a recent ancestor of its target leaf and rewrites stored instances whenever the context tree restructures.