Language Acquisition Device in Large Language Models

Masato Mita **Taiga Someya** **Ryo Yoshida** **Yohei Oseki**
The University of Tokyo
{mita, tsomeya, yoshiryo0617, oseki}@g.ecc.u-tokyo.ac.jp

Abstract

Large Language Models (LLMs) remain substantially less data-efficient than humans. Pre-pretraining (PPT) on synthetic languages has been proposed to close this gap, with prior work emphasizing highly expressive formal languages such as k-Shuffle Dyck. Inspired by the *Language Acquisition Device (LAD)* hypothesis, which posits that innate constraints preemptively restrict the learner's hypothesis space to natural-language-like structure, we propose *LAD-inspired PPT*: pre-pretraining on MP-STRUCT, a formal language whose strings encode hierarchical composition, feature-based dependencies, and long-distance displacement via MERGE, AGREE, and MOVE. A brief 500-step PPT with MP-STRUCT matches strong formal-language baselines in token efficiency while additionally imparting a human-like resistance to structurally implausible languages. Analyzing simplified variants, we find that MP-STRUCT CORE outperforms k-Shuffle Dyck despite not being definable in C-RASP (a formal bound on transformer expressivity), challenging the prior hypothesis that effective PPT languages must be both hierarchically expressive and circuit-theoretically learnable. We show that *functional landmarks*, which reduce dependency resolution ambiguity, are a key driver, suggesting that effective PPT design depends not only on expressivity but also on the accessibility of dependency resolution.

https://github.com/osekilab/LAD-PPT

1 Introduction

Large language models (LLMs) exhibit general linguistic abilities comparable to those of humans; however, their efficiency in language acquisition remains far inferior. While humans acquire language from limited text, LLMs typically require orders of magnitude more data to achieve strong performance (Warstadt et al., 2023). This gap suggests that current LLMs rely on learning from an overly permissive hypothesis space (Yun et al., 2020), leaving substantial room for improving learning efficiency through better inductive biases.

Recent work by Hu et al. (2025) explores *pre-pretraining (PPT)*, where models are first trained on synthetic sequences to acquire useful structural biases before standard pretraining. They show that highly expressive formal languages, such as k-Shuffle Dyck, can improve token efficiency by exercising the model's ability to process hierarchical dependencies. They further interpret these results through the *expressivity hypothesis*, which posits that a formal language conferring a helpful inductive bias should be hierarchically structured in the sense of the *Chomsky hierarchy (Chomsky, 1959)* (either context-free or context-sensitive) and definable in C-RASP (Yang and Chiang, 2024), a formal measure of *circuit complexity* for transformers.

However, such formal languages prioritize abstract structural complexity and often lack key properties characteristic of natural language. This raises the question: *beyond Chomsky-hierarchy complexity and circuit complexity, do natural-language-like properties, such as dependencies conditioned on fixed hierarchical structure, also contribute to effective PPT?* In this work, we approach this question from a complementary perspective. Taking inspiration from the *Language Acquisition Device (LAD)* hypothesis (Chomsky, 1965), which suggests that innate structural constraints can restrict the hypothesis space and favor natural-language-like structure, we ask whether incorporating such constraints into synthetic sequences can further improve learning efficiency. Guided by this idea, we design MP-STRUCT, a synthetic generator taking cues from the *Minimalist Program (MP)*, which produces sequences where dependencies are embedded within a fixed hierarchical structure rather than freely interleaved.

We evaluate LAD-inspired PPT on `Pythia-1B` (Biderman et al., 2023). A brief

Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 19564–19577

500-step PPT phase with MP-STRUCT consistently improves token efficiency over training from scratch and achieves performance comparable to strong baselines based on formal languages such as k-Shuffle Dyck. Moreover, the resulting models exhibit improved structural robustness, including better generalization under lexical perturbation and increased resistance to structurally implausible languages, consistent with the LAD-inspired design goal of proactively restricting the hypothesis space.

To better understand these effects, we analyze simplified variants of the generator. Notably, MP-STRUCT CORE, an idealized abstraction of our generator, achieves higher efficiency than k-Shuffle Dyck despite not being definable in C-RASP, a formal lower bound on the expressivity of future-masked soft attention transformers. This observation is not fully explained by the expressivity hypothesis, which predicts that effective PPT languages should be hierarchically structured and definable in C-RASP. Our analysis instead points to a complementary factor: the *accessibility* of dependency retrieval. While structures that encode dependencies purely through bracketed hierarchy (e.g., k-Shuffle Dyck) define valid dependencies, they may leave multiple plausible antecedents, potentially leading to higher retrieval ambiguity. In contrast, MP-STRUCT and MP-STRUCT CORE introduce explicit structural cues—*functional landmarks*—that can make dependencies easier to locate, thereby reducing the effective search cost for attention-based models. We hypothesize that these differences in accessibility contribute to the observed efficiency gains. More broadly, this suggests that effective PPT design may depend not only on formal expressivity, but also on how structural information is organized to support efficient dependency retrieval.

Algorithm 1 Data Generation Procedure (MP-STRUCT)

Notation: $\mathcal{L}$: lexicon, $V/N/D$: lexical categories, vP: verb phrase, T/C: functional heads, t: trace, $u/iNum$: number features, $wh \in \{+, -\}$.

Input: lexicon $\mathcal{L}$, parameters θ
Output: token sequence S
Step 1: Base Structure via MERGE
Sample lexical items $V, D_1, D_2, N_1, N_2 \sim \mathcal{L}$
$DP_{subj} = \text{MERGE}(D_1, N_1)$, $DP_{obj} = \text{MERGE}(D_2, N_2)$
$V' = \text{MERGE}(V, DP_{obj})$; $vP = \text{MERGE}(DP_{subj}, V')$
// Yields a hierarchical phrase structure with subject and object
Step 2: Functional Structure and AGREE
Assign number feature $iNum \in \{sg, pl\}$ to DP_{subj}
Create $T[uNum]$ and MERGE with vP
Set $uNum \leftarrow iNum$ via $\text{AGREE}(T, DP_{subj})$
Form $TP = [TP\ DP_{subj}\ T[uNum]\ vP]$
// Yields a subject–verb agreement dependency encoded in the structure
Step 3: MOVE (Dependency Encoding)
Copy DP_{subj} to clause-initial position
Replace its original position with a trace: t_{subj}
Form $TP = [TP\ DP_{subj}\ T\ [vP\ t_{subj}\ [V'\ V\ DP_{obj}]]]$
Sample $wh \in \{+, -\}$
MERGE $C[wh]$ with TP
if $wh = +$ **then**
Select a DP in TP and mark it as DP_{wh}
Copy DP_{wh} to clause-initial position
Replace its original position with a trace t_{wh}
Form $CP = [CP\ DP_{wh}\ C\ [TP\ ...\ t_{wh}\ ...]]$
end if
// Yields a long-distance dependency between a moved element and its trace
Step 4: Linearization
Traverse the tree in pre-order
Output brackets, nonterminal labels, features, and traces
Remove lexical terminals $(V, N, D) \rightarrow S$
// Yields a token sequence encoding structural relations without lexical content

2 Related Work

2.1 LAD/UG and Minimalism

A longstanding challenge in language acquisition is explaining how children converge on rich grammatical competence from comparatively limited and noisy input (often termed the *poverty of the stimulus*) (Chomsky, 1965; Clark and Lappin, 2011). The LAD hypothesis addresses this gap by proposing that learners are endowed with Universal Grammar (UG), a species-specific set of constraints that sharply restricts the hypothesis space of possible grammars. From this perspective, UG serves as a strong innate inductive bias, filtering out "non-human-like" hypotheses *a priori*.

In contemporary generative grammar, the *Minimalist Program* (MP) refines this LAD/UG model by seeking to minimize the computational machinery of language to a small set of operations (Chomsky, 1995, 2000, 2001). A central component of this framework is MERGE, a combinatory operation that builds hierarchical structure. In addition, operations such as MOVE (displacement) and AGREE (feature valuation) are commonly assumed to play roles in dependency formation and morphosyntactic licensing (Chomsky, 2000, 2001).

2.2 Pre-pretraining on Synthetic Structures

Pre-pretraining (PPT) on synthetic structures has emerged as a new learning paradigm for language models. Existing studies have reported that

training models via next-token prediction on data possessing hierarchical structures—such as MIDI music, programming languages, or specific formal languages—can impart useful inductive biases, thereby improving the efficiency of subsequent natural language learning (Papadimitriou and Jurafsky, 2020; Ri and Tsuruoka, 2022; Papadimitriou and Jurafsky, 2023).

More recently, Hu et al. (2025) advanced this paradigm by introducing the *Expressivity Hypothesis*, which posits that a formal language conferring a helpful inductive bias should be hierarchically structured—either context-free or context-sensitive—and definable in C-RASP (Yang and Chiang, 2024). However, while Hu et al.'s approach successfully exercises the model's generic computational capacity, it primarily focuses on abstract structural expressivity rather than properties characteristic of natural language. For instance, k-Shuffle Dyck defines dependencies purely through bracket-matching rules, allowing flexible crossing patterns but lacking the asymmetric, head-driven organization typically observed in natural language. This raises the question of whether incorporating more natural-language-like structural properties into synthetic sequences could lead to more effective inductive biases.

In this work, we take the LAD/UG perspective as motivation: rather than deriving a formal grammar, we operationalize the structural properties that these operations are assumed to encode—hierarchical composition, feature-based dependencies, and long-distance displacement—as explicit sequence-level patterns for use in a PPT setting.

3 Methods

The goal of LAD-inspired PPT is to inject an inductive bias that proactively restricts the model's hypothesis space to natural-language-like structure before standard pretraining begins. To this end, we propose MP-STRUCT, a data generator designed to produce *serialized structural representations*—token sequences that make explicit the hierarchical and dependency structure assumed to underlie natural language. The generator is not intended to model natural language itself, nor does it derive sequences from a formal grammar. Instead, it operationalizes three structural properties—hierarchical composition (MERGE), feature-based agreement (AGREE), and long-distance displacement (MOVE)—as abstract sequence-level patterns, stripped of all lexical content.

MP-STRUCT generates data according to Algorithm 1 in the following steps.

Step 1: Base Structure via MERGE We sample lexical items from a lexicon and construct a vP bottom-up using the MERGE operation. This procedure yields a recursive hierarchical structure—rather than a flat sequence—which forms the structural backbone of the generated data.

Step 2: Functional Structure and AGREE We introduce a functional head T with an uninterpretable number feature ($uNum$), and assign an interpretable number feature ($iNum$) to the subject DP. The value of $uNum$ is then determined via agreement with the subject. This step encodes feature-based dependencies within the hierarchical structure.

Step 3: MOVE (Dependency Encoding) We encode long-distance dependencies by copying elements to higher structural positions and replacing their original occurrences with traces. Specifically, the subject DP is copied to a higher position, forming a dependency between the copied element and its trace. We optionally introduce a complementizer C with a binary wh feature; when $wh = +$, a DP is selected, copied to a higher position, and linked to its original position via a trace. These operations result in structured dependencies spanning multiple hierarchical levels.

Step 4: Linearization We traverse the derived tree and output a sequence consisting of structural brackets, nonterminal labels, features, and traces, while omitting lexical items. This design isolates structural information from lexical content: by removing lexical tokens, the model cannot rely on surface co-occurrence patterns and is instead encouraged to process hierarchical structure and dependency relations directly. In the context of pre-pretraining, this is intended to encourage the model to acquire linguistically motivated inductive biases independently of lexical semantics, which may transfer to subsequent natural language pretraining.

4 Experiments

We test whether introducing linguistically motivated inductive biases through PPT can improve the efficiency of subsequent natural language learning.

4.1 Experimental Setup

Model and training protocol. We follow the blockwise learning paradigm of Hu et al. (2025) and use `Pythia-1B` (Biderman et al., 2023) as the base model. Each run consists of (i) an optional PPT phase and (ii) a natural-language pretraining phase. For pretraining (PT), we train on `C4` (Raffel et al., 2019) for 25,000 optimization steps. For all PPT conditions, we fix the budget for the synthetic PPT phase to 500 steps, and subsequently we transfer the parameters to initialize the standard PT. All experiments are conducted with three random seeds, and we report the mean over seeds. The details of the training hyperparameters are provided in Appendix B.

PPT corpora. For PPT, we pretrain on either (i) unstructured synthetic sequences (**Random**), (ii) formal languages with explicit recursion and/or crossing dependencies (**1-Dyck**, k**-Shuffle Dyck**), or (iii) our LAD-inspired structural representations (**MP-STRUCT**). All synthetic sequences are tokenized with the `Pythia-1B` tokenizer. Detailed generation hyperparameters are provided in Appendix C.

4.2 Baselines

We compare MP-STRUCT with the following baselines to isolate specific sources of efficiency gains:

- **Non-PPT:** Standard pretraining from random initialization. This serves as the baseline to quantify the absolute benefit of introducing any PPT phase.
- **Random:** An unstructured PPT control trained on i.i.d. uniformly sampled tokens. This verifies that gains are not merely due to additional gradient updates or data exposure, but specifically stem from *structural* inductive bias.
- **1-Dyck:** A minimal recursion baseline representing pure context-free structure (definable in C-RASP). This tests the sufficiency of pure recursive nesting without the complexity of crossing dependencies.
- k**-Shuffle Dyck:** The current state-of-the-art formal bias (Hu et al., 2025). We specifically adopt the configuration with $k = 64$, following the base configuration of Hu et al. (2025). It is a context-sensitive language yet remains definable in C-RASP, theoretically necessitating both Stack- and Queue-like memory operations.

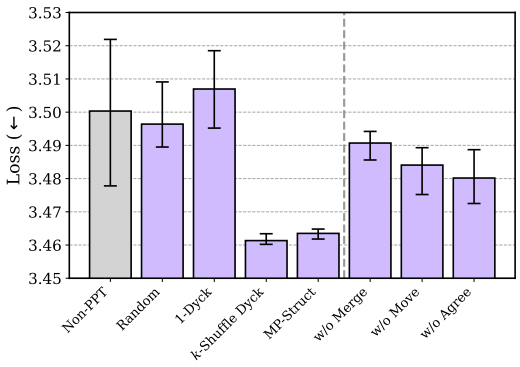


Figure 1: Comparison of C4 validation loss at 25,000 steps across different pre-pretraining conditions. The left section compares MP-STRUCT against baselines, while the right section (separated by the dashed line) presents an ablation study removing specific linguistic components (MERGE, MOVE, AGREE).

Model	BLiMP	MRS	Efficiency
Non-PPT	0.758	–	–
Random	0.761	–	–
1-Dyck	0.759	–	–
k-Shuffle Dyck	**0.764**†	15.6 $\pm$3.31	0.29 $\pm$0.068
MP-STRUCT	0.755	15.3 $\pm$2.86	0.29 $\pm$0.057
MP-STRUCT Core	**0.764**†	**16.2** $\pm$2.97	**0.31** $\pm$0.055

Table 1: BLiMP, MRS, and Efficiency Gain (Efficiency). † indicates a significant difference from Non-PPT at the 5% level. "–" indicates that the condition did not improve upon Non-PPT (i.e., yielded negative MRS and Efficiency values), and is therefore excluded from the efficiency comparison.

4.3 Evaluation Metrics

Following prior work on PPT (Hu et al., 2025), we evaluate whether improvements in learning efficiency are achieved without degrading the quality of the acquired grammar.

- **Learning Efficiency:** We use two metrics from Hu et al. (2025). Let y_1 be the number of PT steps for the Non-PPT baseline, x the number of PPT steps, and y_2 the number of PT steps at which the PPT model first matches the loss of Non-PPT at y_1. **Marginal Rate of Substitution (MRS)** measures how many PT steps are saved per PPT step:

$$\text{MRS} = \frac{y_1 - y_2}{x} \tag{1}$$

Efficiency Gain measures the reduction in total training steps required to reach matched

performance:

$$\text{Efficiency Gain} = 1 - \frac{y_2 + x}{y_1} \quad (2)$$

A concrete calculation example is provided in Appendix E.

- **Grammatical Generalization:** We use BLiMP (Warstadt et al., 2020), which evaluates English grammar using minimal pairs.

4.4 Results

Figure 1 presents the C4 training loss after 25,000 steps, and Table 1 summarizes the grammatical generalization and the learning efficiency.

1. Learning Efficiency Gains from MP-STRUCT. MP-STRUCT outperforms both the Non-PPT baseline and the unstructured Random control in terms of best loss (Figure 1, left). Quantitatively, MP-STRUCT achieves an MRS of 15.3 on average relative to Non-PPT, corresponding to an average efficiency gain of 29% (up to 35%). In contrast, the 1-Dyck baseline does not yield consistent improvements, suggesting that recursive structure alone may not be sufficient to improve learning efficiency.

2. Synergy of Linguistic Operations. To isolate the drivers of this efficiency gain, we conducted an ablation study (Figure 1, right). The results reveal that removing any single component of the generator—MERGE, AGREE, or MOVE—results in a worse final loss compared to the full MP-STRUCT model. This confirms that the efficiency gain is not driven by any isolated feature but by the *synergistic interaction* of hierarchical phrase structure (MERGE) and functional dependencies (AGREE/MOVE), validating the theoretical design of Algorithm 1.

3. Comparable Performance to k-Shuffle Dyck. MP-STRUCT achieves an average efficiency gain of 29%, comparable to the strong baseline k-Shuffle Dyck (29%). While k-Shuffle Dyck shows marginally higher BLiMP scores, MP-STRUCT yields a BLiMP score comparable to the Non-PPT baseline (0.755 vs. 0.758). This suggests that the induced inductive bias primarily facilitates learning efficiency rather than improving final grammatical generalization. These results indicate that linguistically motivated inductive biases can serve as an alternative to high-expressivity baselines for improving token efficiency, and motivate further analysis of the factors underlying these gains (§6).

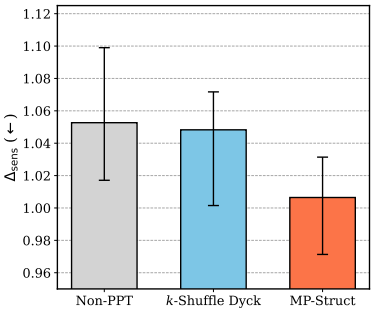


Figure 2: **Robustness against semantic perturbation** ($\Delta_{\text{sens}} = \mathcal{L}_{\text{JW}} - \mathcal{L}_{\text{NL}}$). This metric quantifies the performance gap between semantic-free Jabberwocky inputs and natural language, where lower values indicate less reliance on lexical co-occurrence.

5 Analysis I: Quality of Inductive Bias

Having established the efficiency of MP-STRUCT, we now investigate the *nature* of the acquired biases.

5.1 Validation of Structural Robustness

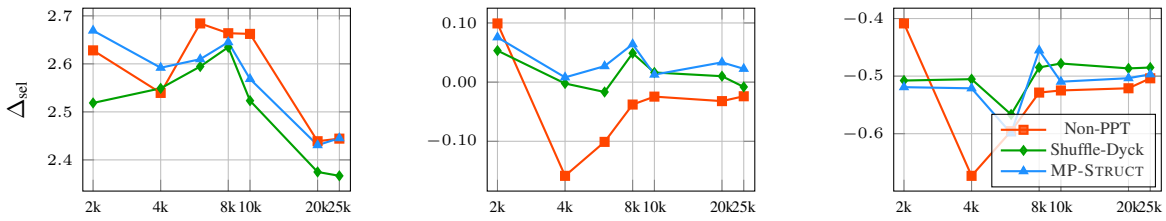
To examine whether the observed efficiency gains are associated with improved structural processing, we perform a *Jabberwocky (JW)* meaning attenuation analysis (Carroll, 1871). This analysis is intended to probe the extent to which models rely on structural information independently of lexical semantics. We construct JW variants of both the training (C4) and evaluation (Wikitext (Merity et al., 2016)) corpora by randomly replacing content words while preserving function words, punctuation, and word order, conditioned on fine-grained POS tags to preserve morphology.[1]

The engineering significance of this analysis lies in quantifying the *disentanglement* of syntactic processing from semantic correlation. Current LLMs often rely on lexical co-occurrence statistics to minimize loss (Dziri et al., 2023; Berglund et al., 2024). However, a robust language learner is expected to maintain predictive performance even when semantic cues are reduced, relying instead on structural regularities (Gulordava et al., 2018).

To assess this, we train separate models under two conditions: natural language (NL) and its Jabberwocky (JW) counterpart, and evaluate each model on the corresponding data (i.e., NL→NL and JW→JW). We then compare their losses to quantify sensitivity to semantic information.

We define sensitivity as $\Delta_{\text{sens}} = \mathcal{L}_{\text{JW}} - \mathcal{L}_{\text{NL}}$,

[1]We provide the details of data construction in Appendix F



(a) SHUFFLE (b) REVERSE (c) HOP

Figure 3: **Structural selectivity** ($\Delta_{\text{sel}} = \mathcal{L}_{\text{Imp}} - \mathcal{L}_{\text{NL}}$) across three impossible language conditions. Positive values indicate a human-like preference for natural linguistic constraints over impossible distortions.

where $\mathcal{L}_{\text{NL}}$ and $\mathcal{L}_{\text{JW}}$ denote the losses obtained under the NL→NL and JW→JW conditions, respectively. A smaller Δ_{sens} indicates that performance degrades less when semantic information is attenuated, which may suggest greater reliance on structural information.

Results. Figure 2 shows that while Non-PPT relies heavily on semantic co-occurrence, MP-STRUCT achieves a lower Δ_{sens} than the strong baseline k-Shuffle Dyck.

While k-Shuffle Dyck theoretically requires memory operations capable of tracking long-distance dependencies, its symbol types lack explicit cues that differentiate their structural roles, potentially leaving multiple plausible antecedents and leading to higher dependency retrieval ambiguity. In contrast, MP-STRUCT introduces distinct structural markers (e.g., functional categories such as T and C), which may help reduce ambiguity in identifying relevant dependencies.

One possible interpretation is that such explicit cues make it easier for the model to rely on structural information when semantic content is attenuated. Accordingly, the improved robustness of MP-STRUCT under meaning attenuation may reflect a greater reliance on structural regularities, rather than lexical co-occurrence alone. To further investigate this hypothesis, we analyze the factors underlying these efficiency gains in §6.

5.2 Resistance to Impossible Languages

We next examine whether the initialization induced by MP-STRUCT tends to align with constraints characteristic of human language (UG) by probing learning behavior on *impossible languages*. Following the protocol of Kallini et al. (2024), we construct synthetically perturbed corpora by applying deterministic transformations to the NL training data. Specifically, we adopt the following implementations from their framework to violate specific linguistic universals while retaining statistical regularities:[2]

- **SHUFFLE**: We use DETERMINISTICSHUFFLE with a window size of $s = 21$. This operation permutes tokens deterministically within a fixed local window, destroying local n-gram statistics and syntactic constituency while preserving the global bag-of-words distribution.
- **REVERSE**: We use FULLREVERSE, which reverses the token order of the entire sequence. While computationally deterministic (requiring a stack), this transformation violates the incremental, left-to-right processing constraint fundamental to human language performance.
- **HOP**: We use WORDHOP (specifically with a 4-word shift). This transformation introduces a dependency based on linear counting: a functional marker is placed at a fixed linear distance (four words) after its associated verb. This mimics "impossible" grammatical rules that rely on counting word positions rather than structural configurations.

We quantify the model's preference for natural constraints using *structural selectivity*: $\Delta_{\text{sel}} = L_{\text{Imp}} - L_{\text{NL}}$. *A larger (positive)* Δ_{sel} indicates that the model finds natural language significantly easier to learn than impossible languages, implying a bias toward human-like structures.

Results and Discussion. Figure 3 presents the structural selectivity scores (Δ_{sel}). Although absolute values vary across pretraining steps, the relative ordering among conditions is largely stable beyond approximately 4k steps, suggesting that the following patterns reflect robust tendencies rather than transient artifacts.

In the SHUFFLE condition (Fig. 3a), all models exhibit high positive Δ_{sel}, indicating a strong pref-

[2]We provide the details of data construction in Appendix G.

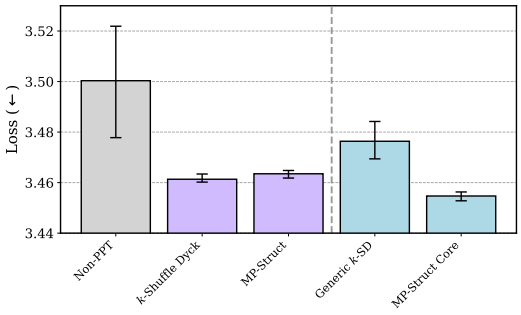


Figure 4: Comparison of C4 loss at 25,000 steps across abstract conditions.

erence for local structural coherence. Conversely, in the HOP condition (Fig. 3c), Δ_{sel} values tend to be negative, reflecting the Transformer's inherent capacity to capture long-distance dependencies.

The most critical divergence appears in the REVERSE condition (Fig. 3b). The most expressive formal bias, k-Shuffle Dyck, adapts remarkably well to reversed sequences, yielding $\Delta_{\text{sel}} \approx 0$. This suggests that k-Shuffle Dyck incentivizes the acquisition of generic processing strategies capable of handling sequences in any direction—an excessive flexibility that lacks linguistic structural constraints. In contrast, MP-STRUCT maintains a positive Δ_{sel}. This resistance is not incidental: the sequences generated by MP-STRUCT encode a fundamentally directional structure: displacement consistently targets structurally higher positions to the left, imposing a directional asymmetry on the linearized sequence. Reversing the input string directly violates these directional biases, making reversed sequences genuinely harder to process for a model that has internalized them. Consequently, the model perceives the hierarchy-violating REVERSE rule as conflicting with its internalized constraints, resulting in a resistance characteristic of human-like structural selectivity.

6 Analysis II: Drivers of Efficiency

The results in §5.1 suggest that the observed efficiency gains may be related to differences in how structural information is represented, particularly the availability of explicit cues for dependency resolution. This raises a more specific question: *are these gains driven solely by structural expressivity (i.e., C-RASP definability), or by how dependencies are organized and made accessible to the model?*

Algorithm 2 Data Generation Procedure (MP-STRUCT CORE)

Notation: $[\,\cdot\,]/(\,\cdot\,)$: structural/dependency bracket pair, T/C: functional heads, t: trace, AGR $\in \{\text{AGR}_A, \text{AGR}_B\}$: agreement dependency (subject–verb number agreement), SEL: selectional dependency (head selects its complement, e.g., D selects N), MOVE: movement dependency, H_X: head token for layer $X \in \{CP, TP, VP\}$, $wh \in \{+, -\}$.

Input: sequence length L
Output: token sequence S
Step 1: Base Structure via MERGE
Sample $wh \in \{+, -\}$ and AGR $\in \{\text{AGR}_A, \text{AGR}_B\}$
Construct vP containing head H_{VP}, a subject slot (trace t if $wh{=}+$, else empty), and one object linked via SEL dependency
Shuffle vP-internal elements to vary surface order
// Yields an abstract verb phrase with a selectional dependency between head and object
Step 2: Functional Structure and AGREE
Assign AGR feature to DP_{subj}; create $T[H_{TP},\ \text{AGR}]$ and MERGE with vP
if $wh = +$ **then** leave Spec-TP empty **else** place DP_{subj} at Spec-TP **end if**
Form $TP = [TP\ \text{Spec-TP}\ T\ vP]$
// Yields a subject–verb agreement dependency marked by landmark H_{TP}
Step 3: MOVE (Dependency Encoding)
if $wh = +$ **then**
Copy DP_{subj} to Spec-CP; leave trace t_{subj} at its vP-internal position
Attach licensor MOVE to C; form $CP = [CP\ DP_{subj}\ C[H_{CP},\ \text{MOVE}]\ TP]$
else
Form $CP = [CP\ (\text{empty})\ C[H_{CP}]\ TP]$
end if
// Yields a long-distance movement dependency anchored by landmark H_{CP}
Step 4: Linearization
Traverse the tree in pre-order
Output brackets, dependency markers, and traces $\rightarrow S$
Repeat Steps 1–4 and concatenate until $|S| \geq L$
// Yields a token sequence where each dependency site is marked by an unambiguous landmark

6.1 Decomposition and Abstract Conditions

To better isolate the factors contributing to learning efficiency, we construct a controlled experimental setting in which the set of structural operations is held constant across conditions. Specifically, both conditions share the same primitive operations as MP-STRUCT: one type of recursive structure and four types of functional dependencies (see Appendix H for details). The only factor varied is how these components are organized, allowing us to attribute any difference in efficiency directly to the organization of dependencies rather than to their number or type. Within this controlled setting, we analyze how the *organization* of dependencies affects learning, focusing on what we term *dependency identification ambiguity*.

Dependency identification ambiguity refers to

Condition	Example Sequence
1. Generic k-SD *(Random Mix)*	`[0 (1 (2 (4 ]0 )4 )1 [0 )2 ]0`
2. MP-STRUCT CORE *(Constrained Mix)*	`[0 H_C (1 [0 H_T (2 [0 (4 ]0 )4 ]0 )2 ]0 )1 [0 H_V ]0 ]0`

Table 2: **Decomposition of Abstract Generative Conditions.** We contrast **Generic k-SD** with **MP-STRUCT CORE**, which introduces diverse functional heads as explicit landmarks. Both conditions share the same set of dependency types; they differ in whether dependencies are randomly interleaved (Generic k-SD) or organized within a fixed hierarchical topology with landmark tokens adjacent to each dependency site (MP-STRUCT CORE). Colors: **Hierarchy [ ]**, **Dependency ( )**, and **Heads**.

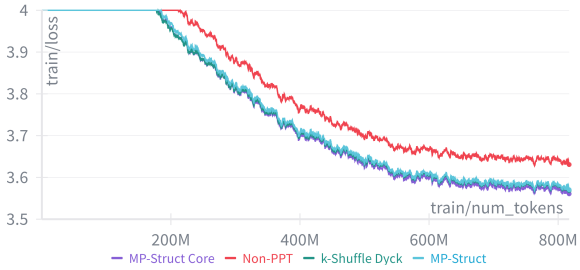


Figure 5: The trajectory of the training loss.

the extent to which a dependency endpoint (e.g.,)1) provides sufficient cues to uniquely identify its corresponding start point (e.g., (1). When such cues are limited, multiple candidate antecedents remain plausible, leading to high ambiguity. Conversely, when structural markers are present near relevant positions, the set of candidates can be sharply constrained, resulting in lower ambiguity.

Based on this perspective, we define two contrasting conditions:

- **1. Generic k-SD:** A condition where the same primitive components are randomly interleaved. As illustrated in Table 2, multiple dependency start points of the same type (e.g., two instances of (1) may appear in an unstructured manner, providing weak cues for identifying which one corresponds to a given endpoint (e.g.,)1). As a result, multiple candidates remain plausible, leading to **high dependency identification ambiguity**.
- **2. MP-STRUCT CORE:** A distillation of MP-STRUCT into a pure abstract formal language, designed to preserve its two key structural properties while eliminating lexical content (see Algorithm 2 for the full generation procedure):
  - **Fixed Topology.** The derivation strictly follows the hierarchical path $CP \rightarrow TP \rightarrow vP$, enforcing the same recursive subordination as MP-STRUCT. Unlike Generic k-SD, where dependencies are randomly interleaved, every dependency in MP-STRUCT CORE is generated within its designated structural domain.
  - **Abstract Landmarks.** The functional heads of MP-STRUCT (C, T, v) are replaced by distinct abstract tokens H_{CP}, H_{TP}, H_{VP}, which are systematically placed adjacent to their associated dependency brackets. This ensures that each dependency site is marked by an unambiguous, consistent cue, mirroring the role of functional heads without introducing lexical noise.

  Together, these properties result in **low dependency identification ambiguity**: the head tokens serve as explicit landmarks that sharply localize the relevant search space for each dependency.

6.2 Analysis of Efficiency Factors

Figure 4 summarizes the results of our abstraction study.[3] Among the abstract conditions, MP-STRUCT CORE outperforms Generic k-SD in terms of token efficiency. Quantitatively, MP-STRUCT CORE achieves an MRS of 16.2 and an average Efficiency Gain of 31% (up to 37%) as shown in Table 1. These values are higher than those of k-Shuffle Dyck (MRS: 15.6, Efficiency: 29%). These results suggest that differences in how structural components are organized may have a substantial impact on learning efficiency, even when the underlying set of operations—one type of recursive structure and four types of functional dependencies—is held constant across conditions.

In line with the design described in §6.1, the two conditions differ primarily in the degree of de-

[3]The trajectory of the LM loss is provided in Figure 5.

pendency identification ambiguity. Generic k-SD exhibits higher ambiguity due to the lack of explicit cues for identifying dependency relations, whereas MP-STRUCT CORE provides more localized structural markers that help constrain the set of plausible antecedents. From this perspective, reduced dependency identification ambiguity may contribute to more efficient learning, consistent with the hypothesis that structural accessibility—beyond expressivity alone—plays a role in effective PPT design.

6.3 Theoretical Implication

A key implication of this analysis concerns the relationship to the expressivity hypothesis (Hu et al., 2025). This hypothesis posits that a formal language conferring a helpful inductive bias should be hierarchically structured and definable in C-RASP (Yang and Chiang, 2024), the latter serving as a formal lower bound on what future-masked soft attention transformers can express.

However, MP-STRUCT CORE is not definable in C-RASP. The generator enforces a strict adjacency constraint: functional head tokens (e.g., H_C) are systematically placed immediately before their associated dependency brackets, requiring a predicate that jointly references both the head position and the dependency position. C-RASP, being a restriction of FO(M) that permits only single-index predicates per quantifier (Yang and Chiang, 2024), cannot express such a two-position constraint. While Hu et al. (2025) suggest that C-RASP-definability is a desirable property for effective PPT languages, this is framed as a tendency rather than a strict requirement—their results show only that C-RASP-definable languages *generally* achieve equal or better performance, not that non-C-RASP-definable languages necessarily fail. Consistent with this, MP-STRUCT CORE achieves higher efficiency than k-Shuffle Dyck despite not being C-RASP-definable.

Taken together, these results suggest that C-RASP-definability is neither necessary nor sufficient for effective PPT, and that the organization of dependencies—specifically, the availability of explicit structural cues that reduce retrieval ambiguity—is a key complementary factor.

7 Conclusion

We proposed *LAD-inspired PPT*, a pre-pretraining framework in which MP-STRUCT—a formal language taking cues from the Minimalist Program—injects linguistically motivated inductive biases before standard pretraining. Our results show that such biases improve learning efficiency comparably to strong formal baselines, while additionally imparting human-like structural robustness, including resistance to structurally implausible languages. Through controlled analyses, we find that the expressivity hypothesis alone does not fully account for these gains: MP-STRUCT CORE outperforms k-Shuffle Dyck despite not being definable in C-RASP. Instead, our analysis points to *functional landmarks*—explicit structural cues that reduce dependency identification ambiguity—as a key complementary factor, suggesting that effective PPT design depends not only on formal expressivity but also on how structural information is organized to support efficient dependency retrieval.

Limitations

While our results provide compelling evidence for the efficacy of linguistically motivated PPT, several limitations remain.

Scale and Architecture. Our experiments were conducted primarily on the Pythia-1B model (Biderman et al., 2023). While we observed consistent trends across seed runs and smaller scales, it remains to be verified whether the efficiency gains of MP-STRUCT scale linearly to significantly larger models (e.g., 7B or 70B parameters) or alternative architectures (e.g., State-Space Models).

Monolingual Evaluation. We evaluated grammatical generalization using BLiMP (Warstadt et al., 2020), which is limited to English. Although MP-STRUCT is designed based on Universal Grammar principles (e.g., MERGE and MOVE) assumed to be language-universal, our current validation does not explicitly confirm improved acquisition efficiency for typologically distinct languages (e.g., head-final languages like Japanese or morphologically rich languages).

Operationalization of Dependency Identification Ambiguity. While we use dependency identification ambiguity as an explanatory construct, it currently lacks a formal, corpus-independent definition that would allow quantitative comparison across arbitrary languages. For instance, it remains unclear whether k-Shuffle Dyck, C4, or other corpora exhibit higher or lower ambiguity than the conditions studied in §6, limiting the generalizability of our claims. Furthermore, the comparison

between Generic k-SD and MP-STRUCT CORE may not isolate ambiguity as cleanly as intended: introducing landmark tokens not only reduces retrieval ambiguity but also increases vocabulary size, which may alter other properties of the language such as entropy. Disentangling these confounds—for example, by controlling for unigram entropy or vocabulary size—remains an important direction for future work.

Acknowledgments

We thank the anonymous reviewers for their helpful comments and suggestions. This work was supported by JSPS KAKENHI Grant Number JP24H00087, Grant-in-Aid for JSPS Fellows JP24KJ0800, JST BOOST Grant Number JPMJBY24B2, JST CREST Grant Number JPMJCR2565, JST PRESTO Grant Number JPMJPR21C2, JST ACT-X Grant Number JPMJAX25CS, and JST SPRING Grant Number JPMJSP2108.

References

Lukas Berglund, Meg Tong, Maximilian Kaufmann, Mikita Balesni, Asa Stickland, Tomek Korbak, and Owain Evans. 2024. The reversal curse: Llms trained on "a is b" fail to learn "b is a". In *International Conference on Representation Learning*, volume 2024, pages 18623–18642.

Stella Biderman, Hailey Schoelkopf, Quentin Gregory Anthony, Herbie Bradley, Kyle O'Brien, Eric Hallahan, Mohammad Aflah Khan, Shivanshu Purohit, USVSN Sai Prashanth, Edward Raff, and 1 others. 2023. Pythia: A suite for analyzing large language models across training and scaling. In *International Conference on Machine Learning*, pages 2397–2430. PMLR.

Lewis Carroll. 1871. *Through the Looking-Glass, and What Alice Found There*. Macmillan.

N. Chomsky. 1995. *The Minimalist Program*. Current studies in linguistics series. MIT Press.

Noam Chomsky. 1959. On Certain Formal Properties of Grammars. *Information and Control*, 2(2):137–167.

Noam Chomsky. 1965. *Aspects of the Theory of Syntax*. The MIT Press, Cambridge.

Noam Chomsky. 2000. Minimalist inquiries: The framework. In Roger Martin, David Michaels, and Juan Uriagereka, editors, *Step by Step: Essays on Minimalist Syntax in Honor of Howard Lasnik*, pages 89–155. MIT Press, Cambridge, MA.

Noam Chomsky. 2001. Derivation by phase. In *Ken Hale: A Life in Language*. The MIT Press.

Alexander Clark and Shalom Lappin. 2011. *Linguistic Nativism and the Poverty of the Stimulus*. Wiley-Blackwell.

Nouha Dziri, Ximing Lu, Melanie Sclar, Xiang Lorraine Li, Liwei Jiang, Bill Yuchen Lin, Peter West, Chandra Bhagavatula, Ronan Le Bras, Jena D. Hwang, Soumya Sanyal, Sean Welleck, Xiang Ren, Allyson Ettinger, Zaid Harchaoui, and Yejin Choi. 2023. Faith and fate: limits of transformers on compositionality. In *Proceedings of the 37th International Conference on Neural Information Processing Systems*, NIPS '23, Red Hook, NY, USA. Curran Associates Inc.

Kristina Gulordava, Piotr Bojanowski, Edouard Grave, Tal Linzen, and Marco Baroni. 2018. Colorless green recurrent networks dream hierarchically. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1195–1205, New Orleans, Louisiana. Association for Computational Linguistics.

Michael Y. Hu, Jackson Petty, Chuan Shi, William Merrill, and Tal Linzen. 2025. Between circuits and Chomsky: Pre-pretraining on formal languages imparts linguistic biases. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 9691–9709, Vienna, Austria. Association for Computational Linguistics.

Julie Kallini, Isabel Papadimitriou, Richard Futrell, Kyle Mahowald, and Christopher Potts. 2024. Mission: Impossible language models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 14691–14714, Bangkok, Thailand. Association for Computational Linguistics.

Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. 2016. Pointer sentinel mixture models. *Preprint*, arXiv:1609.07843.

Isabel Papadimitriou and Dan Jurafsky. 2020. Learning Music Helps You Read: Using transfer to study linguistic structure in language models. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 6829–6839, Online. Association for Computational Linguistics.

Isabel Papadimitriou and Dan Jurafsky. 2023. Injecting structural hints: Using language models to study inductive biases in language learning. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 8402–8413, Singapore. Association for Computational Linguistics.

Colin Raffel, Noam M. Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2019. Exploring

the Limits of Transfer Learning with a Unified Text-to-Text Transformer. *Journal of Machine Learning Research*, 21:140:1–140:67.

Ryokan Ri and Yoshimasa Tsuruoka. 2022. Pretraining with artificial language: Studying transferable knowledge in language models. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 7302–7315, Dublin, Ireland. Association for Computational Linguistics.

Alex Warstadt, Aaron Mueller, Leshem Choshen, Ethan Wilcox, Chengxu Zhuang, Juan Ciro, Rafael Mosquera, Bhargavi Paranjabe, Adina Williams, Tal Linzen, and Ryan Cotterell. 2023. Findings of the BabyLM challenge: Sample-efficient pretraining on developmentally plausible corpora. In *Proceedings of the BabyLM Challenge at the 27th Conference on Computational Natural Language Learning*, pages 1–34. Association for Computational Linguistics.

Alex Warstadt, Alicia Parrish, Haokun Liu, Anhad Mohananey, Wei Peng, Sheng-Fu Wang, and Samuel R. Bowman. 2020. BLiMP: The benchmark of linguistic minimal pairs for English. *Transactions of the Association for Computational Linguistics*, 8:377–392.

Andy Yang and David Chiang. 2024. Counting like transformers: Compiling temporal counting logic into softmax transformers. In *First Conference on Language Modeling*.

Chulhee Yun, Srinadh Bhojanapalli, Ankit Singh Rawat, Sashank J Reddi, and Sanjiv Kumar. 2020. Are transformers universal approximators of sequence-to-sequence functions? In *International Conference on Learning Representations*.

A Examples of Derivation Tree.

Figure 6 shows the example of a derivation tree up to Step 3 in Algorithm 1.

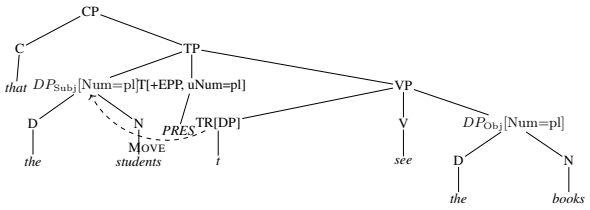


Figure 6: Derivation tree up to Step 3. The subject DP moves from vP to Spec-TP (EPP), establishing a long-distance dependency linked by a trace.

B Training Configurations

Table 3 shows the shared training hyperparameters for PPT and PT, following (Hu et al., 2025). For the experiment, a single NVIDIA RTX 6000 Ada (48GB) GPU was used, and the training time for each run was approximately 20 hours.

Hyperparameter	Value
Batch size	16
Gradient accumulation	2
Effective batch size	32
Max sequence length	1024
Learning rate	5×10^{-4}
LR schedule	Cosine with warmup
Min. LR	5×10^{-5}
Warmup steps	1000
Weight decay	0.1
Gradient clipping	1.0
Optimizer	AdamW
β_1, β_2	0.9, 0.999
ϵ	10^{-6}
Mixed precision	bf16

Table 3: Shared training hyperparameters for PPT and PT (identical across all models), following (Hu et al., 2025) except for max sequence length.

C Hyperparameters for MP-STRUCT

Table 4 shows the hyperparameters for MP-STRUCT.

D Examples used in pre-pretraining

Table 5 shows the examples used in pre-pretraining.

E Calculation Examples of Learning Efficiency Metrics

We present an actual calculation example using the results from one trial (Seed=0) in our experiment.

Hyperparameter	Value
# sequences (n)	100,000
Max length	1024
Strip lexical items	True
EPP on T	True
$P(C[+wh])$	0.2
$P(DP[-wh])$	0.2
Number prior ($P(\text{sg})$)	0.5
Agreement-match ratio	1.0

Table 4: MP-STRUCT corpus generation hyperparameters.

Language	Example
1-Dyck	((( )))
k-Shuffle Dyck	([{]) }
MP-STRUCT	[CP [C] [[TP [[DP[Num:pl] [D] [N]]] [T(+EPP,uNum:pl)] [[VP V [[DP[Num:pl] [D] [N]]] [TR[DP]]]]]]]
W/O MERGE	C [D N DP[Num:pl] [-wh]] T(+EPP,uNum:pl) V D N DP[Num:pl] TR[-wh]
W/O MOVE	[CP [C] [[TP [T(+EPP,uNum:pl)] [[VP V [[DP[Num:pl] [D] [N]] [-wh]] [[DP[Num:pl] [D] [N]]]]]]]]
W/O AGREE	[CP [C] [[TP [[DP[Num:pl] [D] [N]] [-wh]] [T(+EPP,uNum:u)] [[VP V [[DP[Num:pl] [D] [N]]] [TR[-wh]]]]]]]

Table 5: Examples used in pre-pretraining.

When the reference point is set to $y_1 = 25{,}000$, the loss of the baseline (Non-PPT) was approximately 3.633. The proposed method (MP-STRUCT) first reached this loss at $y_2 \approx 15{,}755$ steps. Since the number of formal language training steps is $x = 500$, the metrics are calculated as follows:

$$\begin{aligned} \text{MRS} &= \frac{25{,}000 - 15{,}755}{500} \\ &= \frac{9{,}245}{500} = 18.49 \end{aligned} \tag{3}$$

$$\begin{aligned} \text{Efficiency Gain} &= 1 - \frac{15{,}755 + 500}{25{,}000} \\ &= 1 - 0.65 = 0.35 \end{aligned} \tag{4}$$

F Jabberwocky Dataset Construction

To evaluate the structural robustness of the models, we constructed a Jabberwocky (JW) variant of the C4 dataset. This process aims to eliminate

semantic correlations from lexical co-occurrence while strictly preserving the syntactic structure and morphological consistency of the original text.

Implementation Details We implemented the generation pipeline using the `spaCy` library with the `en_core_web_sm` model. The transformation process operates as follows:

- **Fine-grained POS Tagging:** We first tokenize the input text and assign fine-grained Part-of-Speech (POS) tags. Unlike coarse tags (e.g., `NOUN`), fine-grained tags (e.g., `NNS` for plural nouns, `VBD` for past tense verbs) allow us to distinguish morphological forms strictly.
- **Content Word Identification:** We identify content words defined as tokens belonging to the set of coarse categories: {`NOUN`, `VERB`, `ADJ`, `ADV`}. Function words (e.g., determiners, prepositions) and punctuation are preserved to maintain the grammatical structures.
- **Tag-wise Shuffling (Document Level):** To preserve morphological agreement (e.g., subject-verb number agreement), we strictly shuffle words within the same fine-grained tag category. Specifically, we group all content words within a processing batch by their fine-grained tags and shuffle these buckets randomly.
- **Lexical Replacement with Casing Constraints:** Each content word in the original sequence is replaced by a different word drawn from the corresponding shuffled tag bucket. Crucially, we apply casing constraints: if the original token was capitalized (e.g., sentence initial), the replaced token is capitalized to maintain sentence boundaries.

By using fine-grained tags rather than coarse categories, our method ensures that, for instance, a singular noun is always replaced by another singular noun, and a past-tense verb by another past-tense verb. This guarantees that the resulting sequences preserve syntactic well-formedness despite the removal of semantic information.

G Impossible Language Datasets Construction

To evaluate whether the model's inductive bias aligns with human-like linguistic constraints, we constructed three "Impossible Language" datasets. We adapted the perturbation logic from the official implementation of Kallini et al. (2024)[4] and integrated it into our preprocessing pipeline.

While the Jabberwocky dataset operates at the document level to maintain vocabulary pools, the following transformations were applied at the **sentence level** after segmentation using `spaCy`. We generated the datasets using the following configurations:

- **SHUFFLE:**
  Generated with the argument `shuffle_deterministic21`. This transformation permutes tokens deterministically within a fixed local window of size $s = 21$. By destroying local word order (n-grams) while preserving global bag-of-words statistics, this condition tests the model's reliance on local syntactic constituency.
- **REVERSE:**
  Generated with the argument `reverse_full`. This operation reverses the token order of the entire sentence string ($w_1, w_2, \ldots, w_n \rightarrow w_n, \ldots, w_2, w_1$). While computationally deterministic (requiring a stack), this transformation violates the incremental, left-to-right processing constraint fundamental to human language performance.
- **HOP:**
  Generated with the argument `hop_words4`. This transformation introduces a dependency based on linear counting rather than structural configuration. Specifically, a functional marker is placed at a fixed linear distance of $k = 4$ words after its associated verb. This mimics "impossible" grammatical rules that rely on counting word positions in the linear string, violating the structure-dependence principle of Universal Grammar.

All transformations were applied to the same subset of the C4 training data as the other conditions, ensuring comparable data volume and lexical coverage.

H Complexity Calibration of Abstract Conditions

In §6, we parameterize our abstract formal languages by two values: k_{struct}, the number of bracket

[4] https://github.com/jkallini/mission-impossible-language-models/

types used for hierarchical structure, and k_{dep}, the number of distinct dependency types. We set $k_{\text{struct}} = 1$ (corresponding to 1-Dyck) and $k_{\text{dep}} = 4$ (corresponding to 4-Shuffle Dyck). This choice is not arbitrary but is derived from an analysis of the dependency types inherent in the full MP-STRUCT generator. MP-STRUCT generates sequences based on five distinct structural operations:

1. **Structure (MERGE):** The fundamental recursive skeleton formed by brackets (e.g., $[\dots]$). This corresponds to the **1-Dyck** component.
2. **Dependencies (MOVE/AGREE/SELECT):** Within this skeleton, four distinct types of long-distance dependencies are established. These correspond to the **4-Shuffle Dyck** component ($k = 4$):
    - **Type 1: Agreement (Plural).** The dependency between T_{pl} and DP_{pl}.
    - **Type 2: Agreement (Singular).** The dependency between T_{sg} and DP_{sg}.
    - **Type 3: Movement.** The dependency between a functional head (e.g., C) and a trace (TR) formed by Wh-movement.
    - **Type 4: Selection.** The local dependency where a determiner (D) selects a noun (N).

By setting $k_{\text{struct}} = 1$ and $k_{\text{dep}} = 4$, we ensure that both **Generic k-SD** and **MP-STRUCT CORE** possess the same "vocabulary size" of dependency types as the original model, allowing us to isolate the effect of topological arrangement and landmarks without confounding factors related to task complexity.

I MP-STRUCT CORE: Design Details

The design of MP-STRUCT CORE and its motivation are described in §6.1. Here we provide supplementary detail on the mapping from MP-STRUCT to its abstract counterpart.

Fixed Topology The derivation in MP-STRUCT follows a recursive path ($CP \to TP \to vP$). MP-STRUCT CORE strictly enforces this same hierarchical subordination, ensuring that every dependency is generated within its designated structural domain.

Abstract Landmarks The functional heads of MP-STRUCT are replaced by distinct abstract tokens with the following correspondence:

- **Complementizer (C) $\to$ H_{CP}:** Marks the clause boundary and movement landing site.
- **Tense (T) $\to$ H_{TP}:** Marks the inflectional domain and agreement trigger.
- **Verb (v) $\to$ H_{VP}:** Marks the thematic domain (argument structure).

Argument structure is fixed to a transitive frame, ensuring that H_{CP}, H_{TP}, and H_{VP} serve as unambiguous, consistent landmarks across all generated sequences.