CCG Revisited: A Multilingual Empirical Study of the Kuhlmann-Satta Algorithm

Paul He and Gerald Penn
Department of Computer Science
University of Toronto
{hepaul, gpenn}@cs.toronto.edu

Abstract

We revisit the polynomial-time CCG parsing algorithm introduced by Kuhlmann and Satta (2014), and provide a publicly available implementation of it. We evaluate its empirical performance against a naive CKY-style parser across the Parallel Meaning Bank (PMB) corpus. While the fast parser is slightly slower on average, relative to the size of the PMB, the trend improves as a function of sentence length, and the PMB is large enough to witness an inversion. Our analysis quantifies this crossover and highlights the importance of derivational context decomposition in practical parsing scenarios.

1 Introduction

Parsing with Combinatory Categorial Grammar (CCG) occupies a crucial space in natural language processing, balancing linguistic expressivity with computational tractability. CCG's position at the bottom of the mildly context-sensitive hierarchy enables the analysis of some long-range dependencies and cross-serial constructions (Kuhlmann et al., 2018; Steedman, 2000), but even an $\mathcal{O}(n^6)$ complexity places it out of reach of several large-scale applications.

Theoretically, CCG parsing was shown to be polynomial-time in sentence length by Vijay-Shanker and Weir (1993), with a worst-case complexity of $\mathcal{O}(n^6)$. Kuhlmann and Satta (2014) much later introduced a simplified algorithm in terms of *derivation contexts* to handle deep stacks of arguments. Through an operational lens, the existence of derivation contexts is the reason that CCG-parsing is polynomial-time.

Yet, CCG's complexity is more subtle than it appears. Kuhlmann et al. (2018) proved that CCG parsing is exponential in the combined size of the grammar and input, contrasting sharply with formalisms like Tree-Adjoining Grammar (TAG), in which parsing remains polynomial in both. This raises important questions about the practical utility of polynomial-time CCG algorithms: how often is an innovation like derivation contexts actually triggered in practice? Do real-world grammars ever blow up? Computational research on CCG, after all, has been driven for two decades by corpora with context-free backbones that use CCG-style notation.

To address this gap, we provide the first empirical evaluation of the algorithm introduced by Kuhlmann and Satta (2014). We compare it against a closely related CKY-style parser that lacks this innovation, widely used in existing CCG systems despite its exponential worst-case runtime. While it is true that traditional parsing algorithms have largely been supplanted by sequential supertagging methods in the research literature, in our experience, a stable and efficient reference algorithm is still important for experimental research in parsing, because it provides all possible parses (not just the most likely as scored by a language model), as well as a filter for derivability that conditions the evaluation statistics of supertagger outputs.

Our contributions here are threefold:

- We make a Python implementation of the Kuhlmann-Satta parser publicly available and evaluate it on the Parallel Meaning Bank (PMB) corpus (Abzianidze et al., 2017), spanning over 12,000 CCG-annotated sentences in English, German, Italian, and Dutch.
- We analyze runtime and rule activation across varying levels of derivational complexity, identifying conditions under which the polynomial parser provides real advantages.
- We show that while the naive parser is faster on average, the Kuhlmann-Satta parser is asymptotically safer, not just on a theoretical basis, but in measurable, practical terms, outperforming its baseline on structurally deep

Proceedings of the 18th International Conference on Parsing Technologies (IWPT, SyntaxFest 2025), pages 20–25

inputs sampled from the PMB and with consistent runtime stability.

This work thus bridges the gap between CCG parsing theory and empirical behavior, providing a grounded assessment of polynomial-time parsing in NLP pipelines.

2 Background

In this section, we introduce some background and terminology that we will refer to across this paper and our implementation.

A CCG Category Categories in Combinatory Categorial Grammar (CCG) are either **atomic elements** or **unary functions** that take a category as input and return another category. More formally, the set of categories $\mathcal{C}$ is defined inductively as follows:

- $A \subset \mathcal{C}$ where A is a set of **atomic categories** (e.g., S, NP, N)
- If $X, Y \in \mathcal{C}$, then $(X/Y) \in \mathcal{C}$ and $(X\backslash Y) \in \mathcal{C}$.

where the slash notation indicates the directionality of functional application. Categories are interpreted as unary functions applied from right to left; for example, S/NP/NP is parsed as (S/NP)/NP, indicating a function that takes two NP arguments in sequence.

Arity Bound As per Kuhlmann and Satta (2014), we define c_G as a constant representing the maximum arity permitted in a derivation. It is computed as:

$$c_G \geq \max\{\ell, a + d\} \tag{1}$$

where ℓ is the maximum arity of a lexical entry, a being the maximum arity of any argument type in the grammar, and d being the maximum composition degree allowed. This bound ensures that derivations exceeding a specified complexity are decomposed into reusable subcontexts, maintaining polynomial runtime.

Equivalence in the Limit Let c_G be the arity bound used by the Kuhlmann-Satta parser. Then for any input sentence s, the set of derivable categories produced by the Kuhlmann-Satta parser with $c_G \to \infty$ is equal to that produced by the naive CKY parser.

$$\lim_{c_G \to \infty} \text{Parse}_{\text{KS}}(s; c_G) = \text{Parse}_{\text{CKY}}(s) \tag{2}$$

This follows from the fact that context decomposition is only triggered when the arity of intermediate categories exceeds c_G. In the limit, such decomposition never occurs, and the two parsers are functionally identical.

Chart Items Let w be an input string of length $n \in \mathbb{N}$, and let $0 \leq i < j \leq n$. In CKY-style CCG parsing, chart items are of the form $\langle X, i, j \rangle$ where X is a category that spans the substring $w[i : j]$[1]. While standard application rules suffice for shallow derivation, deeper constructions involving nested argument stacks or crossing dependencies can cause exponential chart growth. To address this, Kuhlmann and Satta (2014) introduce derivation context items

$$\langle /Y, \beta, i, i', j', j \rangle \tag{3}$$

representing a partial derivation, meaning, if a constituent of type X/Y spans $w[i' : j']$, $i', j' < n$ then the entire expression of type $X\beta$ can be derived over $w[i : j]$. Here, β is the residual argument stack, and c_G is the arity bound as defined earlier.

3 Experiments

3.1 Setup and Dataset

Dataset We evaluate the empirical performance of the Kuhlmann-Satta parser on the Parallel Meaning Bank (PMB) corpus (Abzianidze et al., 2017), which provides aligned CCG derivations across English (11,950 sentences), German (3,128), Italian (1,928), and Dutch (1,494). Each sentence is parsed using both the naive CKY-style parser and the polynomial-time parser from Kuhlmann and Satta (2014). We compare runtime, chart size, rule usage, and correctness.

To avoid bias from non-standard derivation rules, we exclude sentences requiring the unary 1x rule as illustrated in Table 1. This is a special case found in both PMB and CCGbank derivation files that introduces a phrasal NP directly from an N (e.g., N $\Rightarrow$ NP), without any overt determiner. This rule is not part of standard CCG combinatorics, and is essentially unsupportable within a purely CCG-based algorithm such as Kuhlmann-Satta.

Parser Architecture and Implementation Details Both the naive and Kuhlmann-Satta parsers are implemented in Python 3. We designed a unified chart data structure to facilitate fair comparison between

[1] $w[i : j]$ returns the substring of w from indices i to $j - 1$.

Language	LX Sentences	Non-LX Sentences	LX %
EN	6219	5731	52.05
DE	1535	1593	49.07
IT	959	969	49.74
NL	710	784	47.52

Table 1: Proportion of sentences using the non-standard 1x rule per language. English and Dutch show the highest occurrence. These were excluded from runtime comparisons.

Language	Context Edges	Other Edges	Context %
EN	6510	87704	6.9
DE	1429	15734	8.3
IT	371	9967	3.6
NL	888	10159	8.0

Table 2: Proportion of **chart edges** using derivation context rules (Kuhlmann edges). While English produces significantly more edges, the relative context usage varies across languages.

Language	With Context	Without Context	Context %
EN	1133	4589	19.8
DE	300	1293	18.8
IT	97	872	10.0
NL	161	623	20.5

Table 3: Percentage of **sentences** using at least one derivation context rule (Kuhlmann item). Most sentences do not trigger context decomposition.

the two systems, ensuring that tokenization, category assignment, and lexical rule application are identical across runs.

The naive parser implements a standard bottom-up CKY strategy with forward, backward, and crossed composition rules. It constructs all derivable categories over input spans using unrestricted functional application, subject only to chart cell boundaries.

The Kuhlmann-Satta parser extends this by introducing derivation context items as described in Section 2. These items are constructed and recombined according to Rules (1) through (6) from the original paper (Kuhlmann and Satta, 2014), including nested context recombination and contextual substitution. Arity bounds c_G are computed dynamically per input, using the maximum arity observed in the lexicon and a fixed composition degree.

Runtime was measured using Python's built-in timing module, with warm-up excluded. The code for both parsers and all experimental scripts will be made available as open-source code.

3.2 How Often is Context Decomposition Needed?

Table 2 shows the number of chart edges generated using context-based rules ($\langle /Y, \beta, \cdots \rangle$) across languages. English produces an order of magnitude more chart items and context applications than other languages, likely reflecting greater average syntactic depth in this corpus slice.

At the sentence level (Table 3), we find that roughly 20% of English examples require at least one derivation context item. In German, Italian, and Dutch, fewer than 10% of sentences trigger decomposition. These results suggest that derivation context rules are sparsely activated overall but concentrated in a meaningful subset of complex examples.

3.3 Runtime Behavior and Inversion

Figure 2 presents raw runtimes over 5,000 sentences, sorted by sentence index. The fast parser (blue) maintains a steady runtime across the corpus, while the naive parser (orange) exhibits increasing variance and high-runtime spikes—suggesting exponential blowups on structurally deep derivations.

Figure 3 shows moving averages of runtime. We observe a "runtime inversion" trend: initially, the naive parser is faster, but as derivation complexity increases, its runtime begins to match or exceed the fast parser. The fast parser exhibits behavior consistent with a polynomial-time bound.

3.4 Speedup Distribution

Figure 4 shows the distribution of relative speedups $T_{\text{naive}}/T_{\text{fast}}$. The average speedup is $0.81\times$, suggesting the fast parser is slower overall—but the long tail on the right includes examples with speedups exceeding $10\times$.

This suggests that while derivation contexts are overhead on average, they yield significant efficiency gains in high-complexity cases, where redundant recursive expansion is avoided.

4 Analysis and Discussion

Further analysis of our results Let $T_{\text{naive}}(n)$ denote the runtime of the naive CKY-style parser for a sentence of length n, and let $T_{\text{fast}}(n)$ be the runtime of the Kuhlmann-Satta parser on the same input. Let $|C_n|$ represent the number of chart items produced during parsing.

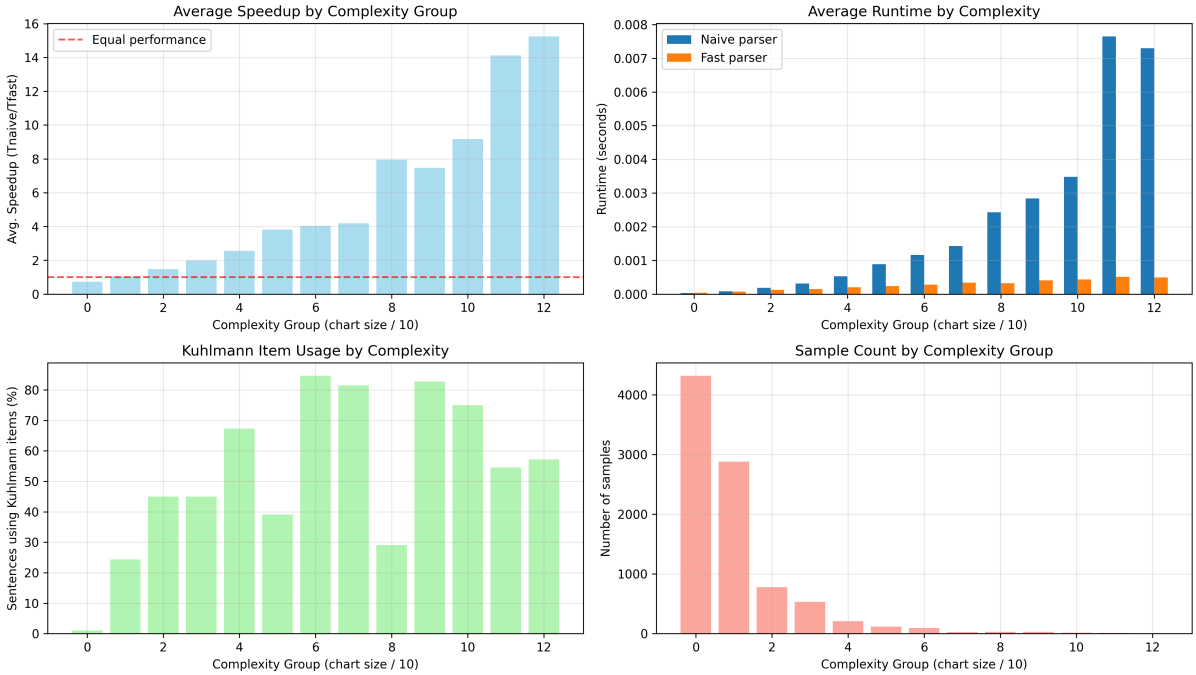

Figure 1: Analysis of parser performance as a function of sentence complexity (measured via chart size in bins of 10). Top-left: Average speedup ($T_{\text{naive}}/T_{\text{fast}}$); a crossover is visible between groups 0 and 1. Top-right: Absolute runtimes per parser. Bottom-left: Percentage of sentences per bin that used Kuhlmann-Satta derivation contexts. Bottom-right: Sample size per bin.

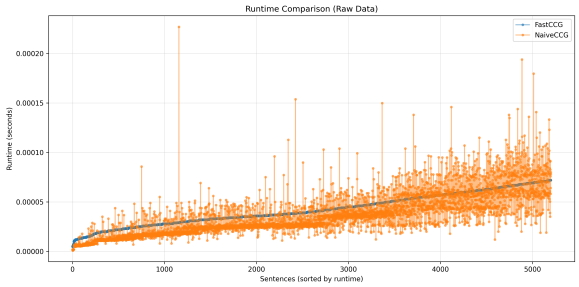

Figure 2: Raw runtime comparison across 5,000 examples. The naive parser is faster on average, but shows instability and many outliers.

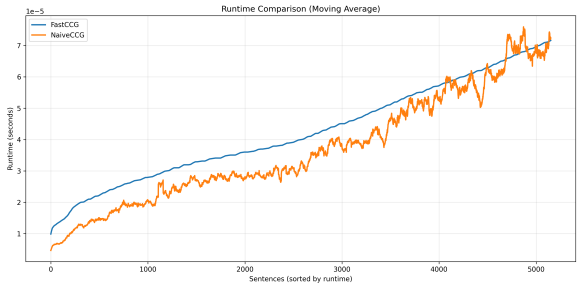

Figure 3: Smoothed runtime comparison. A trend toward inversion is visible: the naive parser is faster early, but the fast parser catches up as complexity rises.

In pathological cases, we empirically observe:

$$T_{\text{naive}}(n) \in \mathcal{O}(2^{|C_n|}), \quad T_{\text{fast}}(n) \in \mathcal{O}(n^6 \cdot g(c_G)) \tag{4}$$

where $g(c_G)$ accounts for grammar-dependent context handling.

To study the crossover empirically, we group sentences by chart size into discrete bins:

$$\mathcal{B}_k = \{n : |C_n| \in [10k, 10(k+1))\} \tag{5}$$

We define average runtimes over each group as:

$$\overline{T}_{\text{naive}}(k) = \frac{1}{|\mathcal{B}_k|} \sum_{n \in \mathcal{B}_k} T_{\text{naive}}(n) \tag{6}$$

$$\overline{T}_{\text{fast}}(k) = \frac{1}{|\mathcal{B}_k|} \sum_{n \in \mathcal{B}_k} T_{\text{fast}}(n) \tag{7}$$

and compute the relative speedup as:

$$S_k = \frac{\overline{T}_{\text{naive}}(k)}{\overline{T}_{\text{fast}}(k)} \tag{8}$$

We observe that $S_k < 1$ for $k = 0$, indicating that the naive parser is faster on simple sentences. However, $S_k > 1$ for all $k \geq 1$, with S_k increasing

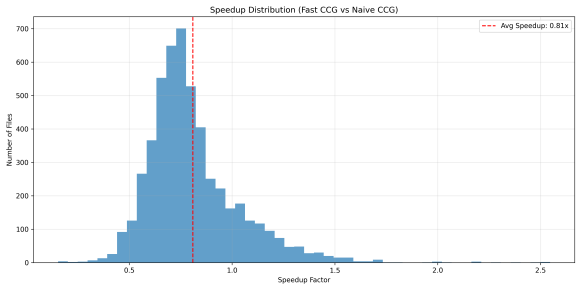

Figure 4: Speedup distribution (fast vs. naive parser). While the fast parser is slower on average, many outliers show large relative gains.

sharply for higher k. This identifies a crossover point near chart size $|C_n| \approx 10$, after which the fast parser becomes consistently preferable.

Moreover, the proportion of sentences that utilize Kuhlmann-Satta derivation contexts (items of the form $\langle /Y, \beta, i, i', j', j \rangle$) increases with k, from 1.0% in group 0 to over 80% by group 6. This strongly supports the theoretical intuition that derivation contexts are unnecessary for simple inputs, but crucial for more complex constructions.

Figure 1 summarizes these trends: the top-left panel reveals a clear crossover in speedup at $k = 1$, the top-right panel shows that naive parsing time grows rapidly with complexity, while the fast parser remains stable, the bottom-left panel shows the increasing use of derivation contexts with complexity, and the bottom-right panel reflects sample distribution, affirming the statistical strength of early bins.

These results validate the theoretical runtime guarantees of Kuhlmann and Satta (2014) and demonstrate that their parser provides real-world benefits for high-complexity sentences.

Context Use and Syntactic Depth Our results suggest a strong relationship between derivational complexity and the activation of derivation contexts. Let $D(s)$ denote the syntactic depth of a sentence s, and let $\Pr[\text{context} \mid s]$ denote the probability that at least one derivation context (i.e., Kuhlmann item) is used in parsing s. Then we conjecture:

$$\Pr[\text{context} \mid s] \propto D(s) \tag{9}$$

While we do not measure $D(s)$ directly, we use chart size $|C_s|$ as a proxy for derivational complexity. Empirically, Figure 1 (bottom-left) shows that the proportion of sentences using derivation contexts increases with $|C_s|$ on average, though not strictly monotonically. We attribute deviations at high complexity to data sparsity (cf. bottom-right), and to the fact that high chart complexity may also arise from breadth (many constituents) rather than depth (nested arguments).

5 Conclusion

We have presented the first empirical evaluation of the Kuhlmann-Satta polynomial-time CCG parser, comparing it to a naive CKY-style baseline across multiple languages in the Parallel Meaning Bank corpus. While the naive parser is faster on average, we find that the Kuhlmann-Satta algorithm offers clear advantages on structurally complex inputs, where it avoids exponential blowup within a reasonable overhead and maintains stable runtime behavior.

Our results validate the theoretical strengths of context decomposition in CCG parsing, and clarify when polynomial-time methods are practically beneficial. This work bridges the gap between parsing theory and real-world efficiency, demonstrating that asymptotic safety is achievable without sacrificing empirical utility.

Looking ahead, the Kuhlmann-Satta parser could play a key role in scaling symbolic parsers for tasks like grammar induction, real-time parsing, and multilingual NLP. Further research might explore adaptive strategies for setting arity bounds, tighter memory constraints, and hybrid approaches that combine polynomial safety with CKY-style heuristics in simpler cases.

6 Limitations and Future Work

While our results validate the theoretical runtime bounds, future work can explore: 1. grammar-aware heuristics for setting c_G dynamically, 2. hybrid parsers that fall back to CKY-style rules when contexts are overkill, 3. optimizing the memory footprint of context items. 4. extending this parser to freer-word-order languages.

References

Lasha Abzianidze, Johannes Bjerva, Kilian Evang, Hessel Haagsma, Rik van Noord, Pierre Ludmann, Duc-Duy Nguyen, and Johan Bos. 2017. The Parallel Meaning Bank: Towards a multilingual corpus of translations annotated with compositional meaning representations. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Pa-*

pers, pages 242–247, Valencia, Spain. Association for Computational Linguistics.

Marco Kuhlmann and Giorgio Satta. 2014. A new parsing algorithm for Combinatory Categorial Grammar. *Transactions of the Association for Computational Linguistics*, 2:405–418.

Marco Kuhlmann, Giorgio Satta, and Peter Jonsson. 2018. On the complexity of CCG parsing. *Computational Linguistics*, 44(3):447–482.

Mark Steedman. 2000. *The Syntactic Process*. MIT Press, Cambridge, MA.

K Vijay-Shanker and David J. Weir. 1993. Parsing some constrained grammar formalisms. *Computational Linguistics*, 19(4):591–636.