A Constraint-Based Formalism for Controlled Text Generation

Lionel Clément
Univ. Bordeaux, CNRS, Bordeaux INP, LaBRI, UMR 5800, F-33400 Talence, France
lionel.clement@u-bordeaux.fr

Abstract

Controlled text generation requires more than selecting words from an input representation: decisions made during realization may constrain later choices. We present a symbolic, constraint-based model of context-sensitive realization in Elvex, a feature-structure-based generation framework. Inherited feature structures encode realization requirements, while synthesized structures record realization results that can be stored in an explicit generation context and reused by other constituents. A chart-based generation algorithm schedules constituents according to information availability rather than surface order and constructs unordered or partially ordered shared forests subject to explicit linearity constraints.

We evaluate synthesized-context reuse on 12 curated predicative-noun support-verb relations covering nine distinct support verbs. FULL and PRE-SPECIFIED each license only the 12 compatible realizations, whereas NO-CONTEXT licenses 96 additional incompatible noun–support combinations. This ablation directly tests whether a lexical constraint produced during realization can restrict a later choice without being pre-specified in the input. Complementary experiments on E2E/GEM and WebNLG assess closed-domain coverage, input preservation, and local RDF realization.

1 Introduction

Controlled text generation aims to produce texts whose content, form, and variation remain inspectable and predictable. This requirement is important in domains where the generated text must express exactly the information provided by the input, without adding unsupported inferences or omitting relevant constraints. Symbolic generation systems are well suited to this setting because their outputs are licensed by explicit grammatical and lexical resources. However, controlled generation raises a specific difficulty: realization decisions are not always independent. A word chosen in one part of the text may determine agreement, anaphora, register, collocations, support-verb selection, or later lexical choices elsewhere in the same generated text.

The difficulty is not merely that generation involves many constraints, but that some of them do not exist before realization begins. They are produced by a realization decision itself. A lexical choice may determine gender, register, a collocational requirement, or an ordering property that another constituent must subsequently observe. A generation architecture must therefore represent both what remains to be realized and what earlier realization decisions have established, without assuming that computation follows surface order.

This positioning differs from general-purpose neural text generation, where surface realization is typically not restricted to forms explicitly licensed by a hand-specified grammar and lexicon. LLMs have strong fluency and broad coverage, but factuality and semantic faithfulness remain active concerns. The goal is not to replace LLMs, but to provide an inspectable alternative when the semantic inventory is controlled and every generated expression must remain traceable to explicit input, grammatical, and lexical constraints. Symbolic realization may also serve as a constrained downstream component of neural or LLM-based planning systems.

This paper addresses this problem through a model of context-sensitive controlled realization in Elvex, which is a symbolic constraint-based generation system described in Clément (2024); its implementation is available as open-source software (Clément, 2026).

The starting point is a constraint-based view of generation in which feature structures encode typed descriptions relevant to realization: semantic predicates and referents, syntactic functions, lexical requirements, morphosyntactic features, discourse-

Proceedings of the 19th International Natural Language Generation Conference: Long Papers, pages 246–256
October 17-21, 2026 ©2026 Association for Computational Linguistics

pragmatic annotations such as speaker viewpoint, presupposition, or polyphony, and register values. The inventory is grammar-dependent, but the formal role of these descriptions is fixed: they license lexical and syntactic choices, record realization results, and constrain later decisions.

The contribution of the paper is to make explicit how information computed during realization can be reused as context for later realization decisions. In this model, a constituent is not only an output fragment. It can also synthesize information that constrains the generation of other constituents.

The distinction between inherited and synthesized information is central. Inherited feature structures describe what a constituent has to realize; synthesized feature structures describe what it has produced once realized. This makes it possible to model dependencies that are difficult to capture in a strictly top-down, left-to-right, or strictly modular generation procedure. The mechanism proposed here lets realization results become explicit constraints on later realization decisions.

For example, realizing a CAR reference as *some beat-up rides* records a particular lexical choice and an informal register, later licensing expressions such as *the old rides* or *the damn things*. In French, choosing *une voiture* 'a car' rather than *un véhicule* 'a vehicle' additionally synthesizes feminine gender, which is determined only once the lexical realization has been selected and can later constrain the choice of *elle* 'it' (feminine), *la voiture* 'the car', or feminine adjectival agreement. The same mechanism applies to support verbs, collocations, and other phraseological constraints.

The generation algorithm operationalizes this view with a chart-based procedure inspired by deductive parsing (Earley, 1970; Knuth, 1965; Shieber et al., 1994). It receives a feature structure and schedules constituents according to inherited-feature availability. Generation order may therefore differ from final surface order. The algorithm builds an unordered or partially ordered shared forest, whose strings are obtained by applying explicit ordering constraints.

The evaluation directly isolates synthesized-context reuse with a lexical-context ablation: FULL reuses a support-verb constraint synthesized by the noun, NO-CONTEXT suppresses that reuse, and PRE-SPECIFIED supplies the same constraint in the input. E2E/GEM and WebNLG provide complementary closed-domain realization tests.

The paper makes two main contributions. First, synthesized information produced during realization can be stored and reused as context for subsequent realization decisions. Second, a dependency-driven chart algorithm schedules daughters by inherited-feature availability rather than surface order and builds unordered or partially ordered shared forests.

2 Related Work and Positioning

Elvex belongs to the tradition of symbolic and linguistically informed NLG, including KPML (Bateman, 1997), SURGE (Elhadad and Robin, 2000), RealPro (Lavoie and Rainbow, 1997), EasyText (Danlos et al., 2011), and recent symbolic systems such as pyrealb (Lapalme, 2024). It shares with these systems the assumption that generation can be driven by explicit linguistic resources. Elvex itself was introduced in prior work as both a formalism and an implemented program for symbolic text generation (Clément, 2024).

A standard view of NLG architectures separates document planning, microplanning, and surface realization (Reiter and Dale, 2000). In that architecture, microplanning covers tasks such as aggregation, referring-expression generation, and aspects of lexicalization, while surface realization turns phrase specifications into linguistic form. Elvex is not an alternative to this whole architecture. Its specific role is to provide a compact formal environment in which realization choices can produce feature-structure information that constrains later lexical, syntactic, and linearization decisions. Meteer motivates feedback from linguistic realization to planning by showing that abstract plans may fail to be linguistically expressible (Meteer, 1991). Our problem is narrower: we address cases in which information becomes available only during realization and must constrain subsequent realization choices.

The formalism is also related to feature-structure grammars, such as Functional Unification Grammar (Kay, 1984) and Lexical-Functional Grammar (Bresnan, 2016), and to broader constraint-based approaches such as attribute grammars (Knuth, 1968) and Property Grammars (Blache, 2004). We do not claim the inherited/synthesized distinction itself as novel. The contribution studied here is its operational use in controlled generation: realization results enter an explicit context that later inherited structures can retrieve, while chart scheduling remains decoupled from linearization.

The point of comparison with attribute grammars is therefore not the inherited/synthesized terminology itself, but its operational role in generation. Here, dependencies are induced from rule statements as realization proceeds, and synthesized results can make later constituents operationally available while surface order remains independently constrained.

Recent NLG research has renewed interest in controllability and faithfulness. GEM provides shared settings for evaluating NLG systems (Mille et al., 2024), while surveys of NLG faithfulness and LLM factuality document continuing challenges in generated text (Li et al., 2022; Wang et al., 2024). Work on controllable text generation for LLMs imposes constraints through instructions, training objectives, or decoding procedures (Liang et al., 2024). Elvex addresses a related problem differently: constraints are encoded directly in grammar rules and lexical entries, and generated forms remain traceable to these resources.

Data-to-text benchmarks such as E2E, GEM, and WebNLG are useful here because they provide controlled meaning representations with explicit semantic units (Novikova et al., 2017; Dušek et al., 2020; Gardent et al., 2017; Gehrmann et al., 2021). We use E2E/GEM as a closed-domain coverage and input-preservation test. WebNLG usually evaluates full RDF-to-text pipelines, including ordering, aggregation, referring expressions, lexicalization, realization, and sentence segmentation. Our use is narrower: we focus on explicit realization resources, testing local RDF configurations once predicate frames are available, in a way closer to symbolic realization work than to end-to-end RDF-to-text generation.

3 The Elvex Formalism

3.1 Feature and Constituent Structures

The input to Elvex is a feature structure describing a realization request: the semantic predicates, referents, arguments, and optional control features that specify what has to be expressed. It need not specify the syntactic or morphological form of the output. These choices are computed by grammar rules and lexical entries, which may introduce syntactic functions, agreement features, lexical requirements, register values, discourse-pragmatic annotations, and linearization constraints during realization.

In the examples below, `HEAD` is used as a lexical index: it identifies the abstract lexical or conceptual head used to retrieve candidate lexical entries. It is distinct from an imposed surface form, which directly constrains the word form to be realized.

Constituent structures define possible configurations of daughters, but they do not impose a fixed left-to-right realization order. A rule defines a local configuration; realization proceeds by satisfying constraints over the mother structure, inherited and synthesized daughter structures, lexical properties, morphosyntactic agreement, and linearization. Word order is therefore not simply the yield of a derivation tree. It is computed from explicit linearization constraints over constituents that may have been generated in a different order.

Feature-structure compatibility is tested by subsumption and unification. Subsumption lets a more general resource description, such as a lexical entry or guard, match a more specific inherited structure and bind variables from it. Unification combines compatible partial descriptions and merges their bindings; incompatible values reject the candidate. These operations underlie guards, rule equations, and lexical selection.

3.2 Inherited and Synthesized Information

In Elvex, a grammar rule relates a mother constituent to a sequence of daughter constituents. For a rule $X \rightarrow Y_1 \dots Y_k$, each Y_i denotes the i-th daughter of X. Each daughter is associated with two types of feature structures:

- Inherited structures ($\downarrow_i$) express requirements. They specify what daughter Y_i must realize, using information from the input, the mother node, or the current generation context.
- Synthesized structures ($\Downarrow_i$) express results. They record what daughter Y_i has produced after realization, including lexical, morphosyntactic, semantic, discourse, register, or variation information that may be reused later in the generation process.

We write $\uparrow$ for the mother's inherited structure and $\Uparrow$ for the synthesized structure returned by the mother.

The crucial point is that synthesized information is not only local output: it enters the *generation context*, an explicit symbolic store of feature structures produced by realized constituents. This store—not a surface-text prefix or implicit model state—is consulted to compute inherited structures for later constituents, allowing one daughter's realization to constrain another.

Figure 1 illustrates how synthesized information

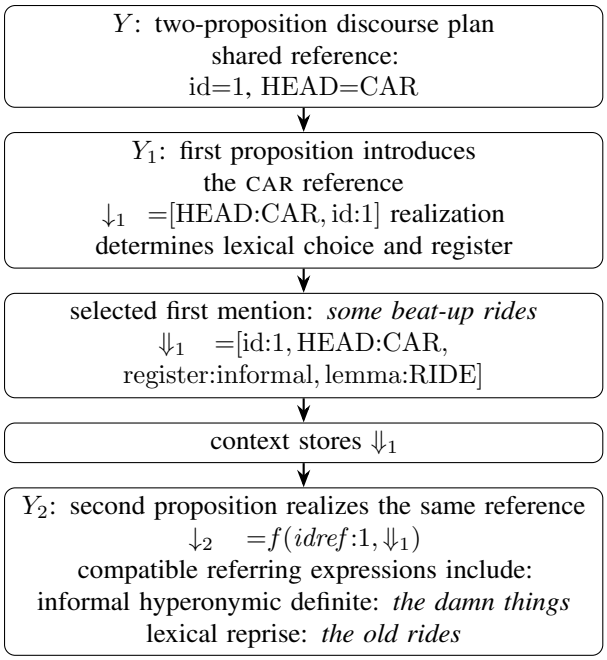


Figure 1: Context reuse across a two-proposition discourse plan. The first realization of the CAR reference synthesizes register and lexical orientation; this information is stored and later constrains compatible referring expressions for the same discourse referent.

can be used to maintain coherence between two propositions in the same discourse. The two propositions Y_1 and Y_2 contain referring expressions that realize the same discourse referent. The first proposition Y_1 introduces this referent; the second proposition Y_2 refers back to it later in the generated text.

The inherited structure $\downarrow_1$ specifies what the referring expression in Y_1 has to realize, including its discourse identifier and semantic class. Other properties, however, only become available once a particular lexical realization has been selected. When the expression is realized as *some beat-up rides*, it synthesizes a structure $\Downarrow_1$ recording the lexical choice RIDE and its informal register. These properties are results of the realization rather than specifications supplied by the input.

This synthesized structure is then stored in the generation context. When the generator later realizes the referring expression in Y_2, it retrieves the information associated with the same discourse identifier and uses it to compute the inherited structure $\downarrow_2$. In other words, $\Downarrow_1$ is not only a local output of the first realization; it becomes part of the context from which $\downarrow_2$ is derived. This licenses later referring expressions compatible with the lexical and register choices made for the first mention, such as *Some beat-up rides pulled up. The damn things still run.* or *Some beat-up rides pulled up. The old rides still work.*, rather than continuations incompatible with those realization choices.

The same mechanism also covers phraseological dependencies. Lexical entries may synthesize support-verb or modifier requirements that constrain later constituents. Thus lexical choice contributes not only words, but also constraints on subsequent realization.

3.3 Ordering with Synthesized Keys

Linearity in Elvex is expressed by constraints on the daughters that are present in a rule application. The default rule order can be left unchanged, but explicit constraints can impose partial or boundary orders: `order 1 < 3 < 2;` states that the first, third, and second daughters must appear in that relative order, while unmentioned daughters remain distributable; `order ≪ 4;` places the fourth daughter at the left boundary; and `order ≫ 1;` places the first daughter at the right boundary. These constraints specify only what is constrained, leaving other daughters free subject to the remaining grammar.

A second form orders selected daughters by a synthesized key. The syntax `order 2, 3 by ⇓.weight;` means that the same relative path is evaluated on each mentioned daughter: here $\Downarrow_2$`.weight` and $\Downarrow_3$`.weight`. The statement does not directly sort a string. It adds precedence arcs only when the compared keys impose a strict order. If the keys are equal, no precedence arc is added and the corresponding daughters remain free to permute.

The key-based order mechanism can model French lightness contrasts of the kind analyzed by (Abeillé and Godard, 2006). In support-verb constructions, a bare noun may have to remain closer to the verb than a following PP, whereas a determined or modified NP may be mobile. The contrast is illustrated below.

(1a) Paul rend [visite] [à sa voisine].
(1b) *Paul rend [à sa voisine] [visite].
'Paul visits his neighbour.'

(2a) Paul rendra [une longue visite] [à sa voisine].
(2b) Paul rendra [à sa voisine] [une longue visite].
'Paul will pay his neighbour a long visit.'

(3a) Marie a porté [plainte] [contre l'entreprise].
(3b) *Marie a porté [contre l'entreprise] [plainte].
'Marie filed a complaint against the company.'

```
VP → VP NP PP { // light-verb case
  [arg2:$arg2, $rest];
  ↓1 = $rest;
  [obj:$objSynt, prep_obj:$prep_objSynt] ⊂ ⇓1;
  ↓2 = $objSynt;
  ↓3 = $arg2 U $prep_objSynt;
  order 1 < 2; order 1 < 3;
  order 2, 3 by ⇓.weight;
}

// Shorthand, not full Elvex  syntax:
N without modifier  => weight:1
N with adjective    => weight:2
N with relative     => weight:3
PP                  => weight:2
```

Figure 2: Limited Elvex fragment for lightness ordering in a light-verb VP, showing only object and secondary-argument constraints.

(4a) ?Marie a porté [une plainte qui expose dans le détail tous les faits] [contre l'entreprise].

(4b) Marie a porté [contre l'entreprise] [une plainte qui expose dans le détail tous les faits].
'Marie filed a complaint against the company, setting out all the facts in detail.'

Abeillé and Godard analyze French *lightness* with a syntactic feature POIDS 'weight' for words and phrases, and formulate order constraints over head-complement branches.

Figure 2 gives the corresponding minimal Elvex fragment. The fixed orders make the verbal head precede both complements, while the key-based order compares the synthesized `weight` of the object and PP daughters. Bare objects synthesize `weight:1`, PPs and modified NPs synthesize `weight:2`, and NPs with relative clauses synthesize `weight:3`. A strict comparison between object and PP adds a precedence arc; equality adds no arc. Thus light objects must precede the PP; modified objects of equal weight may appear on either side, whereas heavier objects are ordered after the PP.

Operationally, the ordering statement is evaluated lazily. If $\Downarrow_2$`.weight` or $\Downarrow_3$`.weight` is not yet available when the rule is first processed, the statement remains pending and is retried after the relevant daughters have been realized. This is essential because the ordering key is not a property supplied by the input alone; it is a realization result synthesized by the daughters. The resulting local precedence graph is then linearized by enumerating its topological orders. For a bare object the graph contains `1 < 2`, `1 < 3`, and `2 < 3`; for a modified object it contains only `1 < 2` and `1 < 3`, leaving the two complements unordered with respect to each other.

The same key-based mechanism also applies beyond lightness: lexical entries may synthesize ranks or weights that later rules use for controlled linearization.

4 Generation Algorithm

The generation procedure implemented in Elvex is a chart-based deduction algorithm inspired by Earley analysis (Earley, 1970). It represents partial generation states as items organized into successive chart states. Unlike parsing, however, the input is not a fixed string but an inherited feature structure for the start symbol. The algorithm incrementally constructs a shared derivation forest together with the linearization constraints that determine its possible surface realizations.

Rules and items. A grammar rule relates a mother category X to a finite sequence of daughters $(Y_1, \ldots, Y_k)$ and a set of rule statements. The right-hand side lists the daughters licensed by the rule; it does not by itself impose a surface order. When a daughter Y_i is operationally available, the corresponding rule application is represented by an enriched active item:

$$[X \to \alpha \bullet Y_i\beta, \rho, \uparrow, (\downarrow_j), (\Downarrow_j), \mathcal{S}, \mathcal{E}, \mathcal{N}]$$

Here ρ records the chart-state boundaries covered by the item. A boundary is the chart index reached after a given number of terminal SHIFT operations. The first is the state in which the rule application was introduced; later boundaries are added by shifts or reductions and delimit the forest fragment associated with the item.

The remaining fields store the information needed to continue the rule application. The structure $\uparrow$ is the inherited feature structure of the mother X. For each daughter Y_j, $\downarrow_j$ is the inherited structure passed down to that daughter, and $\Downarrow_j$ is the synthesized structure returned after the daughter has been realized. The set $\mathcal{S}$ contains rule statements that cannot yet be evaluated because some of their operands are still unavailable.

The environment E records variable bindings introduced by guards, equations, or built-in operations. Variables are written with a leading `$`; for example, `$rest` denotes the variable *rest*.

Finally, $\mathcal{N}$ records the forest identifiers for the fragments already built by lexical shifts and reductions.

In short, an item records both a partial rule application and the semantic, constraint, and forest information needed to complete it.

The bullet identifies the daughter currently available for an operation; it does not encode surface order. If the marked daughter Y_i is terminal and $\downarrow_i$ is known, SHIFT may choose a lexical realization compatible with $\downarrow_i$. If Y_i is non-terminal, a complete item for Y_i may be used by REDUCE.

Daughters whose inherited structures are not yet known are left unmarked. A complete item is written

$$[X \to Y_1 \dots Y_k\bullet, \rho, \uparrow, (\downarrow_j), (\Downarrow_j), \mathcal{S}, \mathcal{E}, \mathcal{N}].$$

Pending rule statements determine when daughters become operationally available. During rule evaluation, they induce a local dependency graph over inherited structures, synthesized structures, and variables. This graph is computed from the operands required and produced by the statements, rather than being written explicitly in the grammar. A rule application can complete only when the dependencies provide the inherited daughter structures needed for realization or reduction and the synthesized structures needed by the mother. Statements whose operands are not yet available remain pending and are reconsidered whenever closure or reduction makes new information available. A genuine dependency cycle for which no operand can be supplied reaches a fixed point with the corresponding statements still pending; it does not by itself cause saturation to loop.

These dependencies define an order of computation, not an order of words. Figure 3 illustrates the distinction with English adjective ordering.

The example shows why operational availability and surface order must be kept separate: ranks are synthesized by lexical entries and propagated through recursive adjective groups before the relative `order` statement can be evaluated.

Init, Shift, Close, Reduce, and Saturate. The algorithm constructs chart states $S_0, S_1, \dots, S_k$, where i counts the number of terminal realizations admitted so far. INIT builds S_0 from the start symbol and the input inherited structure. SHIFT builds S_{i+1} from shiftable active items in S_i.

Between two shifts, SATURATE computes the local fixed point of the current chart state by repeatedly applying CLOSE and REDUCE. CLOSE performs non-lexical inference within the state: it evaluates rule statements whose operands are available, updates bindings and feature structures, removes inconsistent items, records ordering constraints, and opens rule applications for newly available non-terminal daughters.

Algorithm 1: Generation control loop

Require: start symbol S and inherited feature structure $\uparrow$
Ensure: ordered surface strings generated from the root forest

$S_0 \leftarrow \textsc{Init}(S, \uparrow)$
$S_0 \leftarrow \textsc{Saturate}(S_0, 0)$
$i \leftarrow 0$
while S_i contains no complete root item **do**
  if S_i contains no shiftable active item **then**
    return $\emptyset$
  end if
  $S_{i+1} \leftarrow \textsc{Shift}(S_i, i+1)$
  if $S_{i+1} = \emptyset$ **then**
    return $\emptyset$ ▷ generation failure
  end if
  $S_{i+1} \leftarrow \textsc{Saturate}(S_{i+1}, i+1)$
  $i \leftarrow i+1$
end while
return linearizations of the complete root items licensed by the order graph

Algorithm 2: State saturation

Require: chart state S_i and state index i
Ensure: S_i is closed under CLOSE and REDUCE

repeat
  $S_{\text{old}} \leftarrow S_i$
  $S_i \leftarrow \textsc{Close}(S_i, i)$
  $S_i \leftarrow \textsc{Reduce}(S_i, i)$
until $S_i \equiv S_{\text{old}}$
return S_i

Here $S_i \equiv S_{\text{old}}$ means that the two states contain the same logical information relevant to further generation. Chart states are sets of items: rederiving an equivalent item does not create a new item, although information associated with an existing item may be enriched. Saturation stops when neither CLOSE nor REDUCE can add such information.

Pending statements do not prevent this fixed point from being reached. If a later reduction supplies one of their missing operands, they become evaluable during the next closure step. If a genuine cyclic dependency supplies no new information, the statements simply remain pending at the fixed point.

REDUCE combines a complete item

$$[X \to Y_1 \dots Y_k\bullet, \dots]$$

with an active parent item

$$[X' \to \alpha \bullet X\beta, \dots].$$

If the complete item is compatible with the marked daughter X, the parent item is advanced past X

```
Input
S [mod:<[HEAD:cooking],[HEAD:round],
        [HEAD:big],[HEAD:iron],
        [HEAD:french],[HEAD:red],
        [HEAD:old]>]

Grammar fragment
S → ADJs {
  ↓1 = ↑;
}

ADJs → ADJ {
  [mod:<$Uniq>];
  ↓1 = $Uniq;
  ⇑ = [rank:⇓1.rank];
}

ADJs → ADJs ADJ {
  [mod:<$Head :: $Tail>];
  ↓1 = [mod:$Tail];
  ↓2 = $Head;
  order 1, 2 by ⇓.rank;
```

```
Grammar fragment (continued)

  $minRank = min(⇓1.rank, ⇓2.rank);
  ⇑ = [rank:$minRank];
}

ADJ → adj {
  ↓1 = ↑;
  ⇑ = ⇓1;
}

Lexicon fragment
big adj [HEAD:big, rank:1];
old adj [HEAD:old, rank:2];
round adj [HEAD:round, rank:3];
red adj [HEAD:red, rank:4];
French adj [HEAD:french, rank:5];
iron adj [HEAD:iron, rank:6];
cooking adj [HEAD:cooking, rank:7];
```

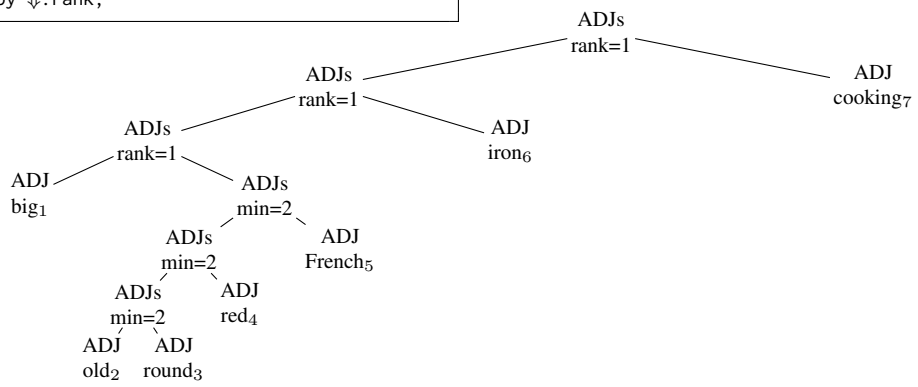


Figure 3: Compact adjective ordering. The input modifier list is not in surface order. Lexical entries synthesize ranks according to adjective type, recursive adjective rules expose a group-level ordering key (`rank`), and a relative `order` statement applies after the relevant ranks are available. Internal nodes propagate the minimum rank of their subtree. The tree sketches the ordered forest realization, yielding *a big old round red French iron cooking pot*, not the input-order realization **a cooking round big iron French red old pot*.

and inserted into the current state:

$$[X' \to \alpha X \bullet \beta, \ldots].$$

The ellipses stand for the item's feature structures, pending statements, bindings, state boundaries, and forest identifiers. During reduction, the forest identifier carried by the complete X-item is attached to the corresponding daughter position of the parent rule application. This operation creates, or reuses, a forest fragment whose root is the parent rule application $X' \to \alpha X \beta$ and whose daughter at the X-position is the completed X-forest. Thus reduction is both an item-combination step and a forest-construction step.

SHIFT plays the role of Earley's scanner, except that it does not consume a fixed input word: it admits a terminal realization licensed by the lexicon and by the inherited feature structure. The lexicon is searched using the inherited `head` value when available, or an imposed surface form. Each accepted entry creates a forest leaf and inserts the advanced item into the next chart state.

Surface strings are generated from the complete root forests subject to the ordering constraints introduced by grammar and lexical entries.

5 Controlling the Search Space

The expressiveness of constraint-based generation raises a practical issue: many lexical, syntactic, discourse, and linearization alternatives may be licensed by the grammar. Without control, generation can grow exponentially. Elvex addresses this by combining early filtering, availability-driven evaluation, local linearity constraints, and shared-forest factorization. Bounded or stochastic search strategies may shuffle or prune alternatives, but they do not change the logical structure of initialization,

saturation, shift, and forest construction.

First, guards filter rules before chart insertion. A guard is a bracketed feature-structure pattern matched against the mother's inherited structure ↑: constants must match and variables are bound. A mismatch discards the rule.
For example, a passive rule can be guarded by a passive diathesis:

```
S → NP VP {
  [diathesis:passive, pred:$Predicate, agent:$Agent,
     patient:$Patient, $Rest];
  ↓1 = [role:patient] U $Patient;
  ↓2 = [pred:$Predicate, voice:passive,
                        byComp:$Agent] U $Rest;
  ⇑ = ⇓2; }
```

Here `diathesis:passive` is the discriminating constant; the variables bind matching values from ↑, and `$Rest` binds the remainder. Hence an incompatible diathesis rejects the rule before prediction.

Second, closure evaluates rule statements only when their operands are available. A daughter can be generated only after its inherited structure has been computed; a statement that reads a synthesized value remains pending until that value has been produced. This is both a correctness condition and a search-control device: lexical choices can synthesize constraints that later daughters use, instead of forcing the generator to explore unrelated alternatives in advance.

For instance, a lexical entry may synthesize a collocational modifier, a support-verb requirement, or an ordering key. The lightness and adjective examples in Section 3.3 show how such keys may become available only after the relevant daughters have been realized. The same principle applies to phraseological constraints introduced by nouns or predicative expressions, which can restrict a later modifier or verbal projection.

Third, linearity constraints contribute to search control without imposing a global derivation order. They restrict only local linearization: mentioned daughters are ordered or placed at a boundary, while unmentioned daughters remain distributable subject to the remaining constraints. Equal ordering keys add no precedence relation, so both orders remain licensed.

Finally, compatible partial results are stored in a shared forest. Paraphrases, synonyms, optional modifiers, and alternative linearizations may combine multiplicatively as strings, but the forest stores shared material once and represents only local differences as alternatives. The generator can thus preserve many licensed realizations without immediately enumerating them. Search control is distributed across guards, availability-driven closure, local linearity constraints, and forest factorization.

The same mechanisms also support bounded variation. Numerical values from lexical entries, rule statements, or built-ins such as `rand()` may be stored and reused as ranks, costs, weights, or register choices. Randomness, when used, selects only among realizations already licensed by the grammar; with a fixed seed, such choices remain reproducible.

6 Evaluation

We separate direct mechanism validation from broader closed-domain coverage. The lexical-context ablation tests synthesized-context reuse directly; E2E/GEM and WebNLG then test coverage and realization with explicit symbolic resources. Resources, scripts, and validation instructions for all experiments reported in this section are available at https://github.com/lionelclement/Elvex-paper-reproducibility.

Lexical-context ablation. We created a dedicated English Elvex resource, `en-oper1`, for a reproducible support-verb experiment. It contains 12 curated $Oper_1$ relations spanning seven predicate classes and nine distinct support verbs. In Meaning–Text terminology, $Oper_1$ is a standard lexical function associating a predicative noun with a support verb whose subject realizes the noun's first semantic actant and whose object or complement is the noun itself (Mel'čuk, 1996). The nine verbs are ADOPT, GIVE, HAVE, IMPOSE, MAKE, OFFER, PAY, PROVIDE, and TAKE.

The semantic input specifies the nominal predicate but not its support verb; for example, lexicalizing *warning* synthesizes `support:[HEAD:GIVE]`. The three experimental conditions differ only in access to this specification. FULL reuses the support value synthesized by the noun; NO-CONTEXT ignores it and lets the verb range over the same nine-verb inventory; PRE-SPECIFIED supplies the correct support verb in the input from the start.

Across the 12 cases, FULL and PRE-SPECIFIED license exactly the same 12 resource-compatible realizations, one per $Oper_1$ relation. NO-CONTEXT preserves these 12 realizations but additionally licenses the other eight support verbs for each case, yielding 108 realizations overall, of which 96 are incompatible noun–support combina-

tions. Synthesized-context reuse therefore provides the same lexical restriction as pre-specifying the support verb, while allowing the initial semantic input to remain underspecified.

Across 30 measured runs after two warm-ups, with condition order balanced across runs, the median wall-clock time for the 12-case batch was 18.480 ms for FULL, 18.931 ms for PRE-SPECIFIED, and 26.298 ms for NO-CONTEXT; the corresponding p95 values were 19.920, 19.979, and 27.824 ms. All lexical-context measurements were run on an Apple M3 Pro (11 CPU cores, 36 GB RAM, macOS 14.6.1).

E2E/GEM coverage and input preservation. On the full E2E/GEM test split (Novikova et al., 2017; Dušek et al., 2020; Gehrmann et al., 2021), we evaluate generation coverage and preservation of the eight standard slots: `name`, `eatType`, `food`, `priceRange`, `area`, `familyFriendly`, `near`, and `customerRating`. The lexical inventory is constructed from the training split only; all test-set slot values are observed there. The grammar programmatically instantiates the 255 non-empty slot-set signatures over these eight slots. A slot is counted as preserved when the evaluator detects an accepted surface realization of its input value. Each Elvex input carries its exact slot-set signature, so a rule realizing only a subset cannot license a richer meaning representation.

All 1 847 test inputs are generated (100% coverage), and each generated realization is slot-complete. Across the full test set, all 11 428 input slots are preserved (100% slot accuracy, SER 0), with 1.00 generated realization per input.

WebNLG local RDF realization. We evaluate the WebNLG v2.1 test split, restricted to entries containing one to three modified RDF triples (Gardent et al., 2017; Shimorina et al., 2018; Gehrmann et al., 2021). The grammar contains explicit predicate frames together with local combination, repeated-entity, typed-relative, and controlled fallback rules. For example, `X location Y` and `X canton Y` may be realized as *X is located in Y, which is its canton*; when such a combination is not licensed, the facts can remain independently realized rather than forcing aggregation.

Table 1 reports generation coverage and, for generated inputs, oracle-style best-of-forest reference scores rather than single-output WebNLG metrics.

Tr.	Inp.	Gen.	Cov.	Out./Gen.	Ex.	NEx.	BLEU	chrF
1	388	378	97.4	2.44	29.9	31.2	66.58	83.44
2	298	286	96.0	10.37	0.0	0.0	43.60	71.56
3	331	317	95.8	41.23	0.0	0.0	38.44	67.60

Table 1: WebNLG v2.1 test-set realization for one-to-three-triple inputs. Tr.: number of RDF triples; Inp.: input meaning representations; Gen.: inputs with at least one generated realization; Cov. (%): generation coverage; Out./Gen.: mean number of compatible realizations per generated input; Ex./NEx. (%): exact and normalized exact match with a reference, using the best realization in the generated forest; BLEU/chrF: mean sentence-level best-of-forest reference scores over generated inputs.

This is a local realization evaluation rather than a leaderboard comparison: the reference scores use oracle-style selection over the generated forest, and the experiment does not measure the individual structural phenomena separately. Coverage remains high and stable (95.8–97.4%), indicating broad local realization coverage within the tested one-to-three-triple setting.

7 Discussion and Conclusion

Elvex handles dependencies in which one realization decision produces information needed by another: inherited structures encode requirements, synthesized structures record reusable results, and availability-driven generation separates computation from surface order.

The lexical-context ablation provides a direct empirical test of this mechanism. FULL and PRE-SPECIFIED coincide on all 12 curated $Oper_1$ relations, whereas NO-CONTEXT retains the compatible realizations but licenses 96 additional incompatible noun–support combinations. Thus a lexical restriction produced during realization can constrain a later choice without being encoded in the initial semantic input.

E2E/GEM complements this targeted mechanism test with complete closed-domain coverage and preservation of all 11 428 input slots. WebNLG obtains 95.8–97.4% local realization coverage over one-to-three-triple inputs, with reference-overlap scores used only as oracle-style diagnostics. Together, these experiments support a narrower claim than general-purpose NLG benchmarking: explicit symbolic resources can license controlled realizations while realization-dependent information remains available to constrain later decisions. Preferred-output selection remains separate.

Limitations

Elvex depends on the coverage and quality of its grammar and lexicon. Missing constructions cannot be inferred at runtime, and overly permissive rules may license unwanted outputs. The reported guarantees therefore hold only relative to the supplied symbolic resources.

High closed-domain coverage does not imply low porting cost: new domains require new lexical and constructional resources.

The primary lexical ablation isolates one contextual phenomenon—support-verb selection—across 12 curated $Oper_1$ relations. It therefore provides direct empirical evidence for phraseological context reuse, not for every motivating phenomenon in the paper: gender, register, anaphora, and ordering remain supported by formal examples; the underlying deferred-context and synthesized-ordering mechanisms are additionally covered by regression tests. The lexical resource was constructed specifically for this controlled experiment, so the evaluation validates context propagation over explicitly encoded support-verb relations; it is not a benchmark of lexical acquisition or out-of-domain generalization.

Lexical wall-clock values are reported as distributions over 30 measured runs on one machine. E2E/GEM establishes closed-domain coverage and input preservation but does not measure grammar portability or lexical acquisition. WebNLG is evaluated for coverage rather than with phenomenon-specific correctness metrics.

Finally, the WebNLG best-of-forest reference scores are oracle-style diagnostics rather than a runtime selection algorithm. The experiments also do not include a direct comparison with pyrealb, published E2E/WebNLG benchmark systems, or neural models.

References

Anne Abeillé and Danièle Godard. La légèreté en français comme déficience de mobilité. *Lingvisticæ Investigationes*, 29(1):11–24, 2006. doi: 10.1075/li.29.1.03abe.

John A. Bateman. Enabling technology for multilingual natural language generation: The KPML development environment. *Natural Language Engineering*, 3(1):15–55, 1997.

Philippe Blache. Property Grammars: a Fully Constraint-Based Theory. In *Constraint Solving and Language Processing*, pages 1–16. Springer, 2004. URL https://hal.science/hal-00134202.

Joan Bresnan. *Lexical-functional syntax*. Blackwell textbooks in linguistics. John Wiley & Sons, Ltd, Chichester, West Sussex, second edition, 2016.

Lionel Clément. Linguistic resources for the automatic generation of texts in natural language: The Elvex formalism. In Max Silberztein, editor, *Linguistic Resources for Natural Language Processing*, pages 27–48. Springer, 2024.

Lionel Clément. Elvex: A symbolic constraint-based generation system. https://github.com/lionelclement/Elvex, 2026. Software repository.

Laurence Danlos, Frédéric Meunier, and Vanessa Combet. EasyText: an Operational NLG System. In *ENLG 2011 - 13th European Workshop on Natural Language Generation*, Nancy, France, 09 2011. URL https://inria.hal.science/inria-00614760.

Ondřej Dušek, Jekaterina Novikova, and Verena Rieser. Evaluating the state-of-the-art of end-to-end natural language generation: The E2E NLG challenge. *Computer Speech & Language*, 59: 123–156, 2020. doi: 10.1016/j.csl.2019.06.009.

Jay Earley. An efficient context-free parsing algorithm. *Communications of the ACM*, 13(2): 94–102, 1970.

Michael Elhadad and Jacques Robin. An overview of SURGE: a reusable comprehensive syntactic realization component. 03 2000.

Claire Gardent, Anastasia Shimorina, Shashi Narayan, and Laura Perez-Beltrachini. The WebNLG challenge: Generating text from RDF data. In Jose M. Alonso, Alberto Bugarín, and Ehud Reiter, editors, *Proceedings of the 10th International Conference on Natural Language Generation*, pages 124–133, Santiago de Compostela, Spain, September 2017. Association for Computational Linguistics. doi: 10.18653/v1/W17-3518. URL https://aclanthology.org/W17-3518/.

Sebastian Gehrmann et al. The GEM benchmark: Natural language generation, its evaluation and metrics. In *Proceedings of the 1st Workshop*

on Natural Language Generation, Evaluation, and Metrics, pages 96–120. Association for Computational Linguistics, 2021. URL https://aclanthology.org/2021.gem-1.10/.

Martin Kay. Functional unification grammar: A formalism for machine translation. In *10th International Conference on Computational Linguistics and 22nd Annual Meeting of the Association for Computational Linguistics*, pages 75–78, Stanford, California, USA, 07 1984. Association for Computational Linguistics. doi: 10.3115/980491.980509.

Donald E. Knuth. On the translation of languages from left to right. *Information and Control*, 8(6): 607–639, 1965. doi: 10.1016/S0019-9958(65)90426-2.

Donald E. Knuth. Semantics of context-free languages. *Math. Syst. Theory*, 2(2):127–145, 1968. doi: 10.1007/BF01692511.

Guy Lapalme. pyrealb at the GEM'24 data-to-text task: Symbolic English text generation from RDF triples. In *Proceedings of the 17th International Natural Language Generation Conference: Generation Challenges*, pages 54–58, Tokyo, Japan, 09 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.inlg-genchal.5.

Benoit Lavoie and Owen Rainbow. A fast and portable realizer for text generation systems. In *Fifth Conference on Applied Natural Language Processing*, pages 265–268, Washington, DC, USA, 03 1997. Association for Computational Linguistics. doi: 10.3115/974557.974596.

Wei Li, Wenhao Wu, Moye Chen, Jiachen Liu, Xinyan Xiao, and Hua Wu. Faithfulness in natural language generation: A systematic survey of analysis, evaluation and optimization methods, 2022. URL https://arxiv.org/abs/2203.05227.

Xun Liang, Hanyu Wang, Yezhaohui Wang, Shichao Song, Jiawei Yang, Simin Niu, Jie Hu, Dan Liu, Shunyu Yao, Feiyu Xiong, and Zhiyu Li. Controllable text generation for large language models: A survey, 2024. URL https://arxiv.org/abs/2408.12599.

Igor Mel'čuk. Lexical functions: A tool for the description of lexical relations in a lexicon. In Leo Wanner, editor, *Lexical Functions in Lexicography and Natural Language Processing*, pages 37–102. John Benjamins, 1996. doi: 10.1075/slcs.31.08mel.

Marie W. Meteer. Bridging the generation gap between text planning and linguistic realization. *Computational Intelligence*, 7(4):296–304, 1991. doi: 10.1111/j.1467-8640.1991.tb00402.x.

Simon Mille, João Sedoc, Yixin Liu, Elizabeth Clark, Agnes Axelsson, Miruna Clinciu, and Yufang Hou. The 2024 GEM shared task on multilingual data-to-text generation and summarization: Overview and preliminary results. In *Proceedings of the 17th International Natural Language Generation Conference: Generation Challenges*. Association for Computational Linguistics, 2024. doi: 10.18653/v1/2024.inlg-genchal.2.

Jekaterina Novikova, Ondřej Dušek, and Verena Rieser. The E2E dataset: New challenges for end-to-end generation. In *Proceedings of the 18th Annual SIGdial Meeting on Discourse and Dialogue*, pages 201–206, Saarbrücken, Germany, 08 2017. Association for Computational Linguistics. doi: 10.18653/v1/W17-5525.

Ehud Reiter and Robert Dale. *Building Natural Language Generation Systems*. Cambridge University Press, Cambridge, 2000.

Stuart M. Shieber, Yves Schabes, and Fernando C. N. Pereira. Principles and implementation of deductive parsing, 1994. URL https://arxiv.org/abs/cmp-lg/9404008.

Anastasia Shimorina, Claire Gardent, Shashi Narayan, and Laura Perez-Beltrachini. WebNLG Challenge: Human Evaluation Results. Technical report, Loria & Inria Grand Est, January 2018. URL https://hal.science/hal-03007072.

Yuxia Wang, Minghan Wang, Muhammad Arslan Manzoor, Fei Liu, Georgi Nenkov Georgiev, Rocktim Jyoti Das, and Preslav Nakov. Factuality of large language models: A survey. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 19519–19529. Association for Computational Linguistics, 2024. doi: 10.18653/v1/2024.emnlp-main.1088.