# From Post-Validation to Grammar-Level Pruning: Valence Constraints for Molecule Synthesis

**Midas van Veen**

**Supervisor(s): Dr. Sebastijan Dumančić, Reuben Gardos Reid**

EEMCS, Delft University of Technology, The Netherlands

A Thesis Submitted to EEMCS Faculty Delft University of Technology,
In Partial Fulfilment of the Requirements
For the Bachelor of Computer Science and Engineering
June 21, 2026

Name of the student: Midas van Veen
Final project course: CSE3000 Research Project
Thesis committee: Dr. Sebastijan Dumančić, Reuben Gardos Reid

## Abstract

This report studies valence constraints for molecule synthesis in a staged program-synthesis framework for chemical reaction network discovery. The baseline system already performs limited molecule validation, but it does so as a final filter after candidate SMILES strings have been generated. The change evaluated here is a ringless valence-constrained grammar that encodes atom valence directly in the grammar, so derivations are restricted to locally valence-consistent construction steps. The resulting grammar reproduces the same ringless molecule sets as the legacy baseline on the audited tests and benchmarks, and all generated molecules pass the repository's valence-validity checks. Runtime results are mixed: on the small water benchmark, the new grammar becomes faster from depth $5$ onward and reaches a $2.87\times$ speedup at depth $10$, while on methane and urea it remains slower throughout the measured ringless depth series. A fixed-count methane benchmark shows that this slowdown is not mainly caused by legacy validity checking, but by the added search overhead of the larger valence-aware grammar. The main conclusion is that the new grammar preserves the audited ringless output behavior while shifting pruning earlier in the search, but this does not translate into uniform runtime gains.


## 1 Introduction

Chemical reaction networks (CRNs) describe how chemical species are transformed by reactions over time. In this project, the goal is the inverse problem: given concentration profiles for only some species, recover the missing molecules and the reaction network that explains those observations. This can be posed as a program-synthesis problem, because the unknown CRN must be constructed from smaller pieces rather than recovered only by fitting parameters. The solver generates candidate molecules, reactions, and networks, simulates their behaviour, and checks whether that behaviour matches the observed concentration data. At the same time, it rejects candidates that break basic structural constraints, such as chemically invalid molecule forms [6]. This combination of generating candidate structures and testing them against a specification is why the task can be posed in program-synthesis terms.

One practical way to do that search is not to guess a full reaction network at once, but to build it in several steps. The solver first proposes molecules that could be missing from the observations, then combines those molecules into candidate reactions, and only after that combines reactions into full candidate networks. The active baseline in this report uses exactly this staged top-down synthesis pipeline, moving from problem to molecules to reactions and finally to networks. This decomposition is more tractable than searching for a complete CRN in one step, but it also makes the molecule stage an important bottleneck. If many chemically implausible molecules survive that stage, the later reaction and network stages inherit unnecessary search work. The current implementation of this pipeline is discussed later in the paper when the grammar change itself is introduced.

This report focuses on one way to reduce that upstream cost. The baseline already contains limited valence-like checks, but they are applied after a candidate SMILES string has been generated. This generate-then-filter pattern is common in symbolic virtual-molecule generation, where candidate structures are first enumerated and only afterward checked against valence-based validity rules [3]. The change studied here is to move that knowledge into the grammar itself, so that molecule generation only follows valence-consistent construction steps. The scope is intentionally narrow: the implementation studied here is ringless, so the paper can isolate the effect of grammar-level valence encoding before dealing with the additional non-local constraints required by ring closures.[1] Accordingly, the experiments in this paper isolate the valence-constrained grammar change at the molecule stage and should not be read as a full end-to-end evaluation of CRN recovery quality.

Figure 1 illustrates the synthesis setting on the simple water-formation example used by Wijers [6]. Oxygen and water are treated as observed species, while hydrogen is left unobserved and must be inferred from the same underlying reaction.

**Water Benchmark**

$2\,H_2 + O_2 \rightarrow 2\,H_2O$

Observed: $O_2$, $H_2O$
Missing: $H_2$

**Figure 1.** Water benchmark adapted from Wijers [6]. The solver must recover the missing species from partial observations.

The central research question is: How can atom valence be encoded as a grammar constraint within

[1] In SMILES, a ring closure is written by repeating the same digit at two separate positions, so validity depends on relating distant parts of the string rather than checking one local expansion in isolation; see https://www.daylight.com/dayhtml/doc/theory/theory.smiles.html.

a program synthesis framework for CRNs? More specifically, this report examines two evaluation questions. First, does the new grammar preserve the accepted ringless molecule set while enforcing valid local valence by construction? Second, what runtime trade-off is created by moving valence from a final validity filter into the grammar itself?

The paper follows that progression directly. Section 2 situates the work relative to Wijers and other related approaches. Section 3 presents the ringless valence-constrained grammar, including a worked water derivation and an explanation of why rings require an additional constraint. Section 4 then evaluates output preservation, validity, and runtime on water, methane, and urea. Section 5 interprets the resulting trade-off, and Section 6 closes with the main conclusion and future work.

## 2 Related Work

At a broader level, this report sits within program synthesis: the task is to construct a structured object from partial specifications while enforcing domain-specific constraints. In this case, the structured object is a chemical reaction network, and the constraints come from chemistry as well as from the search formalism itself. That makes program synthesis a natural framing for the work, even though the underlying scientific problem could also be approached in other ways. A nearby formal-methods line treats CRNs as syntax-guided synthesis targets. For example, sketch-like CRN specifications with holes for unknown species, rates, and stoichiometric constants can be solved through SMT-based synthesis [1].

The most relevant prior work for this report is the MSc thesis by Wijers [6], which established the baseline synthesis framework used in this project. That thesis formulates chemical reaction network (CRN) discovery as a program synthesis problem: given partial concentration data over observed species, the solver must recover missing molecules, reactions, and a full network whose simulated behaviour matches the measurements. To make that search explicit, Wijers defines grammars for molecules, reactions, and networks, and combines them with structural constraints and simulation-based evaluation.

A central contribution of that work is the staged synthesis pipeline that remains the active baseline in the current implementation:

$$\text{problem} \rightarrow \text{molecules} \rightarrow \text{reactions} \rightarrow \text{networks}$$

Rather than synthesizing a full CRN in one step, the solver first enumerates candidate molecules, then constructs atom-balanced reactions, and finally assembles candidate networks that are checked against the target dynamics. The thesis shows that this decomposition matters in practice. Across its benchmark problems, staged synthesis is substantially more tractable than more fully combined alternatives, and the added constraints reduce the search space enough for the solver to recover correct networks in several cases. The thesis also introduces a scoring mechanism that often ranks the expected network near the top of the generated candidates. That result remains relevant here as part of the baseline context, but the present report does not extend that ranking direction.

At the same time, Wijers identifies the main limitation that motivates the present report. Even with staged synthesis and existing validity checks, the solver still explores many chemically implausible candidates before they can be rejected. In particular, the baseline molecule-validity logic contains only limited valence-like reasoning, and much of that reasoning is applied only after a candidate molecule has been generated. The present report asks a narrower question than the full thesis: can atom valence be moved into the molecule grammar itself and used as an explicit source of pruning?

## 3 Method

### 3.1 Baseline Molecule Grammar

The active baseline in this project follows the molecule grammar introduced by Wijers [6]. It is a simplified explicit-atom SMILES grammar: atoms are always written explicitly in brackets, bonds are written explicitly, and the available atom terminals are instantiated from the atoms that appear in the problem definition. In compact form, the baseline grammar is:

```
molecule → chain
chain → atom ringbonds
  | structure bond chain
structure → atom ringbonds branches
branch → "(" bond chain ")"
branches → "" | branch branches
ringbond → bond digit
ringbonds → "" | ringbond ringbonds
digit → "1" | … | "9"
bond → "-" | "=" | "#"
atom → "[H]" | "[O]" | "[N]"
  | "[C]" | …
```

This presentation is adapted from the baseline molecule grammar in Wijers [6]. The important point for the present paper is not every terminal choice, but the search discipline it induces: the grammar freely builds SMILES-like candidates, including candidates that may later turn out to be chemically invalid.

### 3.2 Design Change

The baseline molecule synthesizer already contains valence-like checking, but it uses that logic as a final filter. It first derives a candidate SMILES string from the baseline grammar and only then rejects molecules

whose bond pattern is not chemically valid. This matches the broader generate-then-validate pattern used in symbolic virtual-molecule generation, where candidate structures are enumerated before valence-based validity rules are applied [3]. The main change in this work is to move that knowledge into the grammar itself, so that invalid molecules are never generated in the first place.

The implementation studied here focuses on the ringless case. In this setting, the grammar generates only acyclic molecules, so valence can be tracked locally through the type of nonterminal being expanded. The start symbol is `molecule`, with the first rule `molecule` $\rightarrow$ `chain_0`. The subscript records how many bond units are already consumed by the bond coming into the current atom. Thus, `chain_0` means "build a chain whose first atom has no incoming bond yet", while `chain_1` means "build a chain whose first atom already uses one bond unit".

### 3.3 Ringless Valence Grammar

The grammar then encodes valence by construction, following the broader idea of grammar-based molecular generation systems that build hard valency constraints directly into the generative representation [4]. In the current implementation, an atom of valence $v$ may expand from `chain_i` either as a terminal atom, if all valence is already consumed, or as a larger structure with an outgoing bond of order $o$. In the latter case, the grammar creates a rule of the form `chain_i` $\rightarrow$ `structure_`$i$`_`$o$ `bond_`$o$ `chain_`$o$ only when

$$v - i - o \geq 0.$$

The remaining budget $v - i - o$ is assigned to side branches through `branches_`$v - i - o$. In the ringless grammar, `structure_`$i$`_`$o$ expands to `atom_`$v$ `branches_`$v - i - o$, and each branch consumes budget through rules such as `branch_1` $\rightarrow$ `"(" "-" chain_1 ")"`. This directly encodes the atom valences used by the current repository: carbon 4, oxygen 2, nitrogen 3, and hydrogen 1.[2]

The result is a different search discipline. Instead of generating a broad candidate set and validating complete molecules afterward, the solver only follows derivation steps that still fit within the local valence budget. The next subsection illustrates that change on the smallest nontrivial example.

### 3.4 Worked Example: Water

Suppose the available atoms are hydrogen and oxygen, and the grammar is used to generate water. One valid derivation in the current ringless grammar is:

```
molecule ⇒ chain_0
⇒ structure_0_1 "-" chain_1
⇒ atom_2 branches_1 "-" chain_1
⇒ "[O]" branches_1 "-" chain_1
⇒ "[O]" branch_1 branches_0 "-" chain_1
⇒ "[O](" "-" chain_1 ")" "" "-" chain_1
⇒ "[O](" "-" atom_1 ")" "-" atom_1
⇒ [O](-[H])-[H]
```

This derivation shows where the pruning happens. Oxygen has valence 2, so from `chain_0` it may take one outgoing single bond and one single-bond branch, but not three single bonds. The important point is that `chain_1` does not mean "hydrogen"; it means that the next atom already has one incoming bond. In the `H/O` grammar, `chain_1` can still expand through oxygen, but when the derivation chooses `chain_1` $\Rightarrow$ `atom_1`, it has selected hydrogen. Because hydrogen has valence 1, that choice is terminal and cannot be extended by any further bond. The valid water structure is therefore reachable, while rules that would overfill oxygen or continue from a chosen hydrogen atom simply do not exist.

In the ringless setting, that is enough: once the local budgets are encoded in the grammar, no extra post-generation constraint is needed to maintain valence.

### 3.5 Evaluation Benchmarks

The runtime evaluation uses three ringless molecule-stage benchmarks that stress progressively larger atom sets. The water benchmark uses the H/O search space and serves as the smallest nontrivial case. The methane benchmark uses the repository's H/C/O atom set and therefore introduces carbon branching while remaining ringless. To expose a larger mixed heteroatom space, the evaluation also includes a urea-style benchmark over H/C/N/O.

This urea benchmark is still a molecule-stage grammar test, not a full reaction-recovery benchmark. The synthesizer is run only on the H/C/N/O atom set at increasing depth bounds, and the resulting runtime reflects the cost of bounded ringless molecule generation alone. Its role in the paper is to probe whether the same valence-constrained grammar remains competitive once nitrogen and a much larger ringless search space are added.

### 3.6 Expanding The Ringless Grammar To Support Ringbonds

The ring-enabled version extends the same budgeted design rather than replacing it. In the current implementation, a ring closure is represented locally by a typed ringbond helper. For each bond order $o \in \{1, 2, 3\}$, the grammar introduces a rule of the form `ringbond_`$o$ $\rightarrow$ `bond_`$o$ `digit`, so that each ring endpoint carries both a bond order and a ring

[2] https://www.daylight.com/dayhtml/doc/theory/theory.smiles.html

digit. In the current code, the available digits are `"1"` and `"2"`.

Ring endpoints are then collected through a budget-indexed list nonterminal `ringbonds_`$b$. This helper records that the current atom still has $b$ units of valence available that may be spent on ring closures. Its rules have the same recursive shape as the branch helper:

- `ringbonds_0` $\rightarrow$ `""`;
- for each positive residual budget $b$ and each allowed bond order $o$ not exceeding $b$, `ringbonds_b` $\rightarrow$ `ringbond_o ringbonds_(b-o)`.

In other words, a single-bond ring endpoint consumes one unit of residual valence, a double-bond endpoint consumes two, and so on.

This budget is integrated directly into the atom-expansion rules. If an atom of valence $v$ is reached with incoming bond usage $i$, then a terminal chain may expand as `chain_i` $\rightarrow$ `atom_v ringbonds_(v-i)`. If the atom also takes an outgoing bond of order $o$, then the residual budget $v - i - o$ is split between ring endpoints and side branches: `structure_i_o` $\rightarrow$ `atom_v ringbonds_r branches_(v-i-o-r)`, for each admissible ring usage $r$ with $0 \leq r \leq v - i - o$.

The important point is that ring endpoints are treated as another way of spending local valence budget. The grammar therefore prevents an atom from carrying more ringbond order, branch order, and chain continuation than its valence permits. What it does not guarantee is that the chosen ring digits can be paired later into a globally consistent closure. That remaining problem is why the ring-enabled grammar still needs an additional constraint.

### 3.7 Why Rings Need An Extra Constraint

Rings are harder because a ringbond is not a local decision. Writing a symbol such as `-1` does not complete a bond by itself; it creates a promise that somewhere else in the molecule there will later be a second occurrence of the same digit, and that these two occurrences together form one chemically valid closure.[3] The grammar can express the local shape "bond + digit", but it cannot, by local production rules alone, enforce the global relation between two possibly distant positions in the derivation tree. In particular, the grammar would have to remember which ring digits have already been opened, ensure that each one is closed exactly once, and compare information carried by the two endpoints. That kind of bookkeeping is not naturally local to a context-free or locally valence-budgeted grammar.

For that reason, the ring-enabled version needs a separate constraint. In the current implementation, the grammar may generate ringbond symbols, but a constraint performs the remaining checks on the completed molecule. The constraint must verify that:

- the total number of ringbond endpoints is even;
- each ring digit occurs exactly twice, so every opened ring is closed once;
- the two occurrences of the same digit use the same bond order;
- the same digit is not reused in incompatible local groups of ringbonds;
- a ring closure does not connect an atom to itself;
- two different ring closures do not connect the same pair of atoms, since the resulting molecule must remain a simple graph.

The current valence-constrained implementation therefore has two layers in the ring-enabled case: the grammar still enforces local valence budgets, but a separate constraint is needed to validate the non-local consistency of ring closures. For that reason, the experiments in this paper focus on the ringless grammar, where the effect of grammar-level valence encoding can be isolated cleanly before introducing ring-specific bookkeeping.

## 4 Results

### 4.1 Experimental Setup

This section evaluates the ringless valence-constrained grammar at the molecule stage, because that is where the grammar change was introduced. The comparison is between two implementations:

- the legacy ringless baseline, which uses the old SMILES grammar together with a final `ValidSMILES` check;
- the new ringless valence-constrained grammar, which encodes valid valence during derivation and therefore emits only valence-consistent molecules.

This setup isolates the molecule-generator change studied in this report.

Ring bonds are disabled in both systems. This keeps the comparison focused on a single algorithmic change: moving valence from a final validity check into the grammar itself.

The benchmark problems are the water and methane molecule-stage tasks from the repository, plus a urea-style molecule benchmark over the H/C/N/O atom set. For each problem, the molecule synthesizer is run on the relevant atom set at increasing depth bounds. Because the water benchmark runs in the microsecond-to-millisecond range, it is especially sensitive to timer noise. For that reason, the water comparison uses a repeated ringless benchmark with 30 raw timing shots per

[3] https://www.daylight.com/dayhtml/doc/theory/theory.smiles.html

depth and IQR-based outlier filtering,[4] reporting the filtered mean runtime. The methane comparison uses a fresh ringless benchmark on the current codebase with one warmup run and four timed runs per depth, reporting the mean runtime. The urea comparison uses one warmup run and five timed runs per depth, reporting the median runtime. In all three cases, the timed quantity is wall-clock time for the molecule synthesis call alone.

This setup measures the trade-off most directly relevant to the new grammar: is it cheaper to pay for valence-aware grammar expansion, or to generate a broader set of candidates and validate them afterward?

### 4.2 Regression and Validity Evidence

The first question is whether the grammar rewrite preserves valid ringless output on the audited cases. In this paper, that claim rests on three different kinds of empirical evidence, and it is useful to keep them separate.

First, there is agreement with the legacy implementation. On audited small search spaces, we compared the molecules produced by the new grammar with those produced by the legacy ringless generator, and they matched exactly. In the repository regression tests, the ringless valence-constrained grammar and the ringless legacy `ValidSMILES` path produced the same SMILES set on the checked search spaces. This does not by itself prove general equivalence, because it only shows that the new generator agrees with the old accepted output on those audited cases. However, it is still important regression evidence: the grammar rewrite did not change the accepted molecule set in the audited comparisons.

The benchmark problems support the same regression claim at a larger scale. For every completed compared depth, the old and new ringless implementations produced the same molecule counts:

- water, depths $1:10$: $0, 0, 0, 2, 3, 4, 9, 16, 25, 36$;
- methane, depths $1:9$: $0, 0, 0, 2, 4, 7, 26, 258, 5063$.
- urea, depths $1:8$: $0, 0, 0, 3, 9, 23, 196, 5618$.

For urea, depth $9$ is not part of this completed-depth agreement claim, because both implementations were not available as completed depth-series runs.

Second, there is direct validity under the repository predicates. The generated molecules were checked with `molecule_has_valid_valence(...)` and, where relevant, `molecule_has_simple_graph(...)`. In the repository tests and Step 2 validation runs, all generated ringless molecules passed these checks. This is stronger than agreement with the legacy generator alone, because it evaluates the output against the current project criteria for valid molecules rather than only against past behavior. In particular, all generated ringless molecules passed `molecule_has_valid_valence(...)`, including the ringless hydrogen, oxygen, and mixed H/C/O/N test cases.

Third, the grammar enforces local valence consistency by construction. The ringless valence-constrained grammar only creates productions when the residual budget $v - i - o$ is nonnegative, so no rule can extend a partial molecule in a way that already exceeds local valence. This does not replace the empirical checks above, and it is not a formal proof of full equivalence, but it explains why the generator is expected to remain within the valid part of the search space.

### 4.3 Runtime Measurements

The second question concerns runtime. Tables 1 and 2 report the measured molecule synthesis times for the two ringless implementations in milliseconds. In these tables, speedup is old runtime divided by new runtime, so values above $1$ favor the new grammar.

On water, the new grammar is slower at shallow depths but becomes faster from depth $5$ onward. On methane, the legacy baseline remains faster at every measured depth, even though both implementations return the same molecule counts. Urea extends the methane pattern further: through depth $8$, the implementations still return the same molecules, but the new grammar is already much slower. At depth $9$, the old baseline still completes, while the new grammar is reported as `DNF (>180)`. The 6 section interprets this three-benchmark contrast.

### 4.4 UntilFound Behavior Under BFS

A separate ringless `UntilFound` benchmark shows that the new grammar can also perform worse in candidates-to-goal, not only in runtime.

For methane, the fair ringless baseline reaches $O_2 + H_2O$ after three molecules, while the new ringless valence-constrained grammar needs five. For ringless estherification, the old baseline reaches the full goal after $61$ molecules, while the new grammar needs $370$. The discussion section explains why BFS makes this ordering effect visible.

## 5 Responsible Research

Research on automated chemical reasoning has an obvious dual-use dimension [5]. Methods that help generate or prioritize chemical structures could, in principle, also be used by bad actors to accelerate harmful chemical or pharmaceutical development [5]. That general risk applies to the broader field in which this work sits, and it should not be ignored.

At the same time, the specific contribution of this report is narrow. The work presented here

[4] IQR stands for interquartile range, i.e. $Q_3 - Q_1$. Tukey-fence filtering removes timings below $Q_1 - 1.5$ IQR or above $Q_3 + 1.5$ IQR.

**Water benchmark**

| Depth | Count | Old (ms) | New (ms) | Speedup |
|---|---|---|---|---|
| 1 | 0 | 0.12 | 0.25 | 0.48× |
| 2 | 0 | 0.14 | 0.27 | 0.53× |
| 3 | 0 | 0.16 | 0.28 | 0.58× |
| 4 | 2 | 0.30 | 0.36 | 0.83× |
| 5 | 3 | 0.39 | 0.38 | 1.03× |
| 6 | 4 | 0.54 | 0.44 | 1.22× |
| 7 | 9 | 1.16 | 0.65 | 1.78× |
| 8 | 16 | 2.35 | 1.02 | 2.30× |
| 9 | 25 | 4.18 | 1.59 | 2.63× |
| 10 | 36 | 6.93 | 2.41 | 2.87× |

**Methane benchmark**

| Depth | Count | Old (ms) | New (ms) | Speedup |
|---|---|---|---|---|
| 1 | 0 | 0.16 | 0.81 | 0.20× |
| 2 | 0 | 0.18 | 0.87 | 0.20× |
| 3 | 0 | 0.20 | 0.87 | 0.23× |
| 4 | 2 | 0.41 | 1.03 | 0.40× |
| 5 | 4 | 0.50 | 1.21 | 0.41× |
| 6 | 7 | 0.69 | 1.60 | 0.43× |
| 7 | 26 | 2.00 | 4.07 | 0.49× |
| 8 | 258 | 17.11 | 44.27 | 0.39× |
| 9 | 5063 | 457.19 | 1290.02 | 0.35× |

Table 1: Water and methane benchmarks.

| Depth | Count | Old (ms) | New (ms) | Speedup |
|---|---|---|---|---|
| 1 | 0 | 0.21 | 3.70 | 0.06× |
| 2 | 0 | 0.24 | 3.74 | 0.06× |
| 3 | 0 | 0.31 | 3.93 | 0.08× |
| 4 | 3 | 0.58 | 4.35 | 0.13× |
| 5 | 9 | 0.82 | 5.66 | 0.14× |
| 6 | 23 | 2.25 | 20.71 | 0.11× |
| 7 | 196 | 21.52 | 313.93 | 0.07× |
| 8 | 5618 | 505.83 | 14011.38 | 0.04× |
| 9 | 830787 / DNF | 59572.77 | `DNF (>180000)` | `DNF` |

Table 2: Urea benchmark.

is a molecule-stage grammar reformulation inside a research prototype for chemical reaction network synthesis. It focuses on valence constraints, ringless molecule generation, and small benchmark problems such as water, methane, and estherification. It does not provide a realistic workflow for end-to-end drug design, synthesis planning, toxicity prediction, or laboratory optimization. For that reason, the direct practical impact of this particular contribution on the viability of harmful drug development is likely to be limited.

The main responsible-research concern in this report is therefore not immediate deployment risk, but careful scientific framing. Encoding valid valence in the grammar improves one aspect of structural plausibility, but narrow validity constraints alone do not guarantee chemically realistic or practically useful molecules in a broader sense [2]. The current method does not model energetics, stereochemistry, charge, aromaticity, or full ring chemistry. So the paper should avoid implying that grammar-level valence checking is a sufficient test of chemical usefulness or safety.

This work is also intended to be reproducible. The report is tied to a concrete codebase, benchmark artifacts are stored in the repository, and the paper states the measurement settings used for the reported comparisons. That makes it possible to inspect the exact grammar construction, rerun the ringless benchmarks, and verify the reported trade-offs directly. In that sense, transparency and limited claims are the main responsible-research commitments of this project.

# 6 Discussion

## 6.1 What The Benchmarks Show

The main result of this paper is not a uniform speedup, but a clearer picture of the trade-off created by grammar-level valence encoding. In the legacy ringless baseline, the solver builds molecules with a broader grammar and rejects invalid ones at the end. In the new ringless grammar, more work is paid earlier: the solver only follows locally valence-consistent steps, but it must navigate a larger typed grammar while doing so.

The water benchmark shows the favorable side of that trade-off. After the shallow noisy regime, the new grammar becomes faster from depth $5$ onward while returning the same molecules. By depth $9$, both implementations emit $25$ molecules, but the new grammar reaches them in $0.00159$ seconds rather than $0.00418$ seconds, a speedup of about $2.63\times$. The same pattern continues at depth $10$, where both emit $36$ molecules and the speedup rises to about $2.87\times$.

The methane benchmark shows the opposite regime. There, the new grammar is slower at every measured depth, even though the output remains identical. At depth $9$, both implementations produce $5063$ molecules, but the valence-constrained grammar takes $1.29$ seconds where the legacy ringless baseline takes $0.46$ seconds. So the new path is about $2.8\times$ slower on the larger search space.

The urea benchmark pushes the same effect further. Through depth $8$, the two implementations

| Problem | Old count | New count | Count ratio |
|---|---|---|---|
| Methane ($O_2 + H_2O$) | 3 | 5 | $1.67\times$ |
| Estherification (full molecule goal) | 61 | 370 | $6.07\times$ |

Table 3: Ringless `UntilFound` comparison under BFS.

still agree on the completed output, so the comparison remains like-for-like there. But the runtime gap becomes much larger than on methane. At depth $8$, both implementations emit $5618$ molecules, yet the new grammar takes $14.01$ seconds where the legacy baseline takes $0.51$ seconds, so the valence-constrained path is about $27.7\times$ slower. A higher-cap rerun then shows that the depth-$9$ row is not merely a small cap artifact: the old baseline finishes in about $60$ seconds with $830787$ molecules, while the new grammar is still incomplete after $180$ seconds and has reached only about $46$ thousand molecules.

## 6.2 Why BFS Ordering Also Gets Worse

The separate ringless `UntilFound` benchmark shows that the same problem appears in search order, not only in runtime. Under BFS, enumeration order is determined by derivation depth in the grammar rather than by chemical simplicity. If the grammar surfaces many shallow but irrelevant derivations early, the solver must inspect more molecules before it reaches the target set.

The methane case is the smaller example of this effect. The legacy ringless baseline reaches $O_2 + H_2O$ after three molecules, while the new ringless valence-constrained grammar needs five. In the new BFS prefix, `[H]-[H]` and `[O]=[O]` are followed by two acetylene derivations before `[H]-[O]-[H]` appears, whereas the legacy ringless baseline reaches `[H]-[O]-[H]` immediately after the first two molecules.

The same effect is much larger in the ringless estherification benchmark. The legacy baseline reaches the full goal after $61$ molecules, while the new grammar needs $370$. The delay is concentrated in the oxygenated carbon products: the first positions of the three goal molecules are old/new = water $3/5$, formic acid $24/100$, and methanol $61/370$. So the worse `UntilFound` result comes from the BFS prefix spending many more slots on shallow carbon-chain families before it reaches the relevant oxygenated products.

## 6.3 Why Methane And Urea Slow Down

A separate fixed-count methane benchmark strengthens that interpretation. Instead of comparing the two implementations at equal derivation depth, this benchmark asked how long each one needed to emit the same number of molecules and also measured how much time the legacy implementation spent inside its `ValidSMILES` machinery. Across target counts from $100$ to $50000$, the valence-constrained grammar was slower at every point, and the new-old runtime gap was larger than the entire measured legacy constraint cost at every tested target.

At $50000$ molecules, for example, the old path took $6.78$ seconds in total, of which $2.75$ seconds were spent in the measured legacy validity machinery, while the new path took $22.78$ seconds. So even if the legacy implementation could perform its validity checking for free, it would still remain faster on this methane task. The methane slowdown is therefore not mainly explained by the cost of final validity checking. It is dominated by the additional Herb search overhead induced by the expanded valence-constrained grammar.

The urea benchmark is consistent with the same explanation, although the evidence there is indirect rather than decomposed by constraint cost. Up to depth $8$, the old and new implementations still return the same completed ringless molecule sets, so the difference is again not caused by one path finishing a smaller search space. Yet the runtime gap is much larger than on methane, and at depth $9$ the high-cap rerun shows that the old baseline finishes in about $60$ seconds while the new grammar is still far from completion after $180$ seconds. That pattern strongly suggests that the same grammar-induced search overhead becomes even more severe once the ringless H/C/N/O space is used.

## 6.4 Implications For Valence Constraints

These results separate empirical validity from performance in a useful way. The regression and validity evidence shows that the ringless grammar rewrite preserves the accepted molecule set on the audited cases, that the generated molecules satisfy the repository's final validity predicates, and that the grammar itself only creates locally valence-consistent productions. That is the primary technical result: valence has been moved from a late filter into the grammar without changing the accepted ringless outputs on the checked tasks.

The runtime evidence then shows that earlier pruning does not automatically translate into lower runtime. On small ringless problems such as water, the more selective grammar can be beneficial once the search becomes nontrivial. On larger ringless problems such as methane and urea, the current implementation pays more for grammar-induced search overhead than it saves by avoiding late validation. The most defensible claim is therefore narrower than "valence constraints make the solver faster": the present grammar is empirically

equivalent to the legacy ringless generator on the audited ringless cases, and its performance depends strongly on problem scale and search structure.

### 6.5 Current Scope

The ringless focus is an important limitation rather than a cosmetic choice. Supporting rings is not only a matter of adding more productions. It introduces non-local consistency conditions that require a separate constraint to check whether ring endpoints are matched correctly. So even though the present grammar establishes a useful valence-constrained baseline, extending it to richer chemistry will require solving a different technical problem than the one addressed in the experiments reported here.

## 7 Conclusions and Future Work

This report asked how atom valence can be encoded as a grammar constraint within the staged CRN synthesis framework used in `CRNSynthesizer`. The answer developed here is a ringless molecule grammar that tracks residual valence through typed nonterminals and only creates productions when the local budget remains nonnegative. On the audited ringless tests and benchmarks, this grammar preserves the same accepted molecule sets as the legacy ringless baseline and all generated molecules satisfy the repository's valence-validity checks. The main contribution is therefore empirical evidence that the new grammar is equivalent to the legacy ringless generator on those audited ringless cases.

The runtime results give a more qualified answer to the performance question. The new grammar can help on small ringless tasks: on the water benchmark it becomes faster from depth $5$ onward and reaches a $2.87\times$ speedup at depth $10$. But this does not generalize automatically. On methane, the new grammar is slower at every measured depth, and the fixed-count benchmark shows that the slowdown is dominated by the added search overhead of the larger grammar rather than by the legacy validity checks alone. The urea benchmark strengthens that conclusion further: the old baseline completes depth $9$ in about $60$ seconds, while the new valence-constrained grammar is still incomplete after $180$ seconds on the same ringless H/C/N/O search space. Valence-constrained generation therefore improves correctness discipline earlier in the search, but it is not by itself a guarantee of better runtime.

That conclusion should also be read at the right scope. The present report evaluates a valence-constrained grammar rewrite at the molecule stage. It does not by itself establish how valence information should be used in every other part of the CRN synthesis pipeline; it only establishes the behavior of this grammar component on the reported ringless benchmarks.

The next steps follow directly from that narrower result. First, ring support should be extended with an explicit constraint for non-local ring-closure consistency, so the valence-constrained approach can move beyond the acyclic case. Second, the grammar-induced overhead on larger search spaces should be reduced, since the methane experiments show that this is currently the main bottleneck of the approach. Third, the grammar itself should be refined so that more of the pruning benefit can be retained without paying as much search overhead on larger ringless spaces.

## References


[1] Luca Cardelli, Milan Češka, Martin Fränzle, Marta Kwiatkowska, Luca Laurenti, Nicola Paoletti, and Max Whitby. Syntax-guided optimal synthesis for chemical reaction networks. In Computer Aided Verification, pages 375–395. Springer, 2017.

[2] Wenhao Gao and Connor W. Coley. The synthesizability of molecules proposed by generative models, 2020.

[3] Rafel Israels, Astrid Maaß, and Jan Hamaekers. The octet rule in chemical space: generating virtual molecules. Molecular Diversity, 21(4):769–778, August 2017.

[4] Hiroshi Kajino. Molecular hypergraph grammar with its application to molecular optimization, 2019.

[5] Fabio Urbina, Filippa Lentzos, Cédric Invernizzi, and Sean Ekins. Dual use of artificial-intelligence-powered drug discovery. Nature Machine Intelligence, 4(3):189–191, March 2022.

[6] Richard Wijers. Automated discovery of chemical reaction networks using program synthesis. Master's thesis, Delft University of Technology, 2025.