Chemical Science

EDGE ARTICLE

View Article Online
View Journal

Cite this: DOI: 10.1039/d5sc05004a

Grammar-driven SMILES standardization with *TokenSMILES*

Luis Armando Gonzalez-Ortiz,*[a] Lisset Noriega,[a] Filiberto Ortiz-Chi,[b] Gabriela Vidales-Ayala,[a] Emmanuel Soberanis-Cáceres,[a] Amilcar Meneses-Viveros,*[c] Alan Aspuru-Guzik*[d] and Gabriel Merino*[a]

The redundancy of *SMILES* notation, where multiple strings can describe the same molecule, remains a challenge in computational chemistry and cheminformatics. To mitigate this issue, we introduce *TokenSMILES*, a grammatical framework that standardizes *SMILES* into structured sentences composed of context-free words. By applying five syntactic constraints (including branch limitations, balanced parentheses, and aromaticity exclusion), *TokenSMILES* minimizes redundant *SMILES* enumerations for alkanes while maintaining valence and octet compliance through semantic parsing rules. *TokenSMILES* does not replace *SMILES* but rather formalizes its syntax into a standardized, machine-interpretable form. This grammatical structure enables controlled generation and manipulation of valid *SMILES* strings, ensuring syntactic and semantic consistency while substantially reducing redundancy. Implemented into *SmilX*, an open-source tool, *TokenSMILES* generates valid *SMILES* with accuracy comparable to existing computational implementations for molecules with low hydrogen deficiency (HDI ≤ 4). Its applicability extends beyond alkanes through stoichiometric modifications such as bond insertion, cyclization, and heteroatom substitution. Nevertheless, challenges remain for highly unsaturated systems, where canonicalization artifacts highlight the need for dynamic feasibility checks. By integrating linguistic principles with cheminformatics, *TokenSMILES* establishes a scalable framework for systematic chemical space exploration, supporting applications in drug discovery, materials design, and machine learning-driven molecular innovation.

Received 7th July 2025
Accepted 12th November 2025

DOI: 10.1039/d5sc05004a

rsc.li/chemical-science

Introduction

The analysis of chemical space using artificial intelligence relies on comprehensive and standardized molecular representations. The Simplified Molecular Input Line Entry System (*SMILES*), introduced by Weininger and co-workers in the 1980s, encodes molecular structures *via* a context-sensitive grammar using ASCII characters.[1–3] Initially, *SMILES* was designed to represent atoms, bonds, branching, cycles, aromaticity, charges, and hydrogen counts. Over time, its syntax was expanded to include stereochemistry, isotopic labeling, and hybridization. Despite its versatility, *SMILES* is not unique: several distinct strings can describe the same molecule (Fig. 1). This redundancy arises from permissible syntactic variations within the language. Furthermore, grammatical extensions have been proposed to include more complex systems as polymers and crystal structures.[4,5]

Table 1 shows syntactic variations,[6] including Kekulé, aromatic, branching, and ring number/dot bond syntax, using 2-(aminomethyl)benzoic acid ($C_8H_9NO_2$) as an example.

Recent advances have sought to overcome these limitations by introducing alternative representations with improved

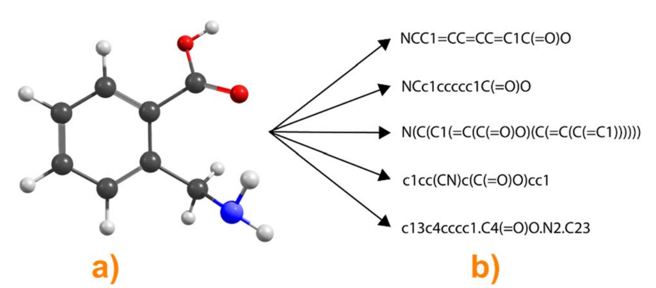

Fig. 1 2-Aminomethylbenzoic acid: (a) molecular model and (b) selected *SMILES* strings representing the same molecule, illustrating the standardization problem addressed by *TokenSMILES*.

[a]Departamento de Física Aplicada, Centro de Investigación y de Estudios Avanzados, Unidad Mérida, km 6 Antigua Carretera a Progreso, Apdo. Postal 73, Cordemex 97310, Mérida, Yucatán, Mexico
[b]Secihti-Departamento de Física Aplicada, Cinvestav-IPN, Antigua Carretera a Progreso km 6, Mérida, Yucatán, 97310, Mexico
[c]Departamento de Computación, Centro de Investigación y de Estudios Avanzados, Unidad Zacatenco, Av. IPN No. 2508, Apdo. Postal 07000, Col. San Pedro Zacatenco, CDMX, Mexico
[d]Department of Chemistry, University of Toronto, DB 421D, Lash Miller Chemical Laboratories, 80 St. George Street, Toronto, ON, M5S 3H6, Canada. E-mail: gmerino@cinvestav.mx; luis.gonzalezo@cinvestav.mx; alan@aspuru.com; amilcar.meneses@cinvestav.mx

Table 1 Syntactic variations in *SMILES* notation

Syntax style	Representative *SMILES*
Kekulé syntax	NCC1=CC=CC=C1C(=O)O
Aromatic syntax	NCc1ccccc1C(=O)O
Branching syntax	N(C(C1(=C(C(=O)O)(C(=C(C(=C1)))))))
Ring numbers and dot bond syntax	c13c4cccc1.C4(=O)O.N2.C23

syntactic control. For instance, *DeepSMILES* simplifies parenthesis handling by adopting postfixed ring-numbering rules,[7] while *t-SMILES* encodes functional groups explicitly, avoiding parentheses and ring indices.[8] *BigSMILES* extends the notation to polymers through the use of braces to denote stochastic binding patterns,[8,9] whereas *CurlySMILES* embeds annotations within braces { } to describe noncovalent or coordinated structures, while preserving the core *SMILES* grammar.[10]

Beyond these structural adaptations, new languages such as *SELFIES*,[11] *Group SELFIES*,[12] and *JAM*[13,14] have emerged. *SELFIES* guarantees that every token sequence corresponds to a chemically valid structures, thereby minimizing parsing and valency errors. *Group SELFIES* refines this idea by representing rings or functional groups as single tokens, simplifying substructure encoding. In contrast, *JAM* adapts *SMILES*-like syntax to describe stacking sequences in crystalline or layered materials, combining chemical and geometric information. Table 2 summarizes these languages using 2-(aminomethyl)benzoic acid as an example, highlighting improvements in grammatical clarity and robustness.

In this work, we introduce *TokenSMILES*, a grammatical and graph-theoretical framework that formalizes the *SMILES* language, together with *SmilX*, its open-source implementation. *SmilX* applies the *TokenSMILES* grammar to generate and validate molecular structures based on explicitly defined syntactic

Table 2 Grammatical representations of 2-(aminomethyl)benzoic acid in different notations

Notation	Representation
SMILES	NCc1ccccc1C(=O)O
DeepSMILES	NCcccccc6C=O)O
t-SMILES	c1([1*])c([2*])cccc1^[1*]C(=O)O^[2*]CN
SELFIES	[N][C][C][=C][C][=C][C][=C][Ring1][=Branch1][C][=Branch1][C][=O][O]
Group SELFIES	[:benzene][Branch][:CH2NH2][pop][Branch][:COOH][pop]

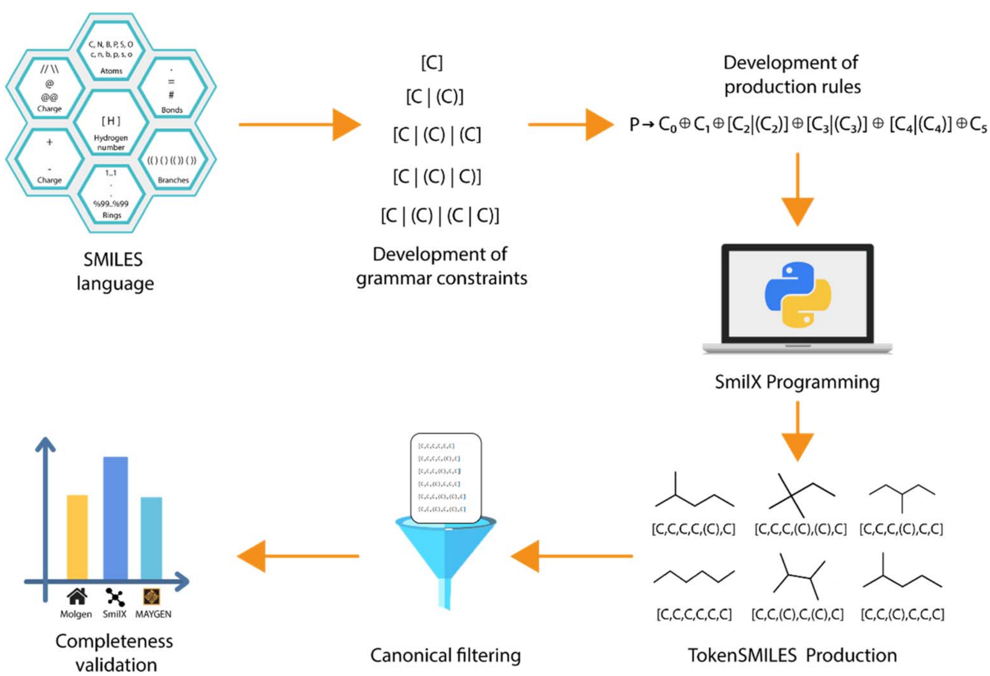

Fig. 2 General workflow diagram for the development of the *TokenSMILES* grammatical framework and the validation of the number of structures.

and semantic rules. Both the conceptual framework and its software implementation are presented here for the first time.

The proposed grammar provides a standardized representation for organic molecules that retains the descriptive capacity of *SMILES* while minimizing ambiguity. By treating molecular strings as complete sentences governed by grammatical rules rather than collections of discrete symbols, *TokenSMILES* enables systematic computational analysis and exhaustive isomer enumeration (Fig. 2). Implemented in *SmilX*, this framework departs from matrix-based approaches (*e.g.*, *MOLGEN*,[15] *MAYGEN*[16]) and block-based methods (*e.g.*, *SMILIB*[17,18]), offering a more structured and linguistically consistent paradigm for molecular representation.

Theoretical development

Kekulé syntax for *SMILES* generation

Let us start by constructing *SMILES* strings for saturated hydrocarbons. The procedure defines transversal paths across all atoms and bonds in a molecule, forming the basis for grammatical constraints in the Kekulé *SMILES* syntax. Using 2,3-dimethylbutane as example (Fig. 3a), the process follows these steps:

(1) Hydrogen removal. Generate a hydrogen-free molecular graph (Fig. 3b).

(2) Atom labeling. Assign unique numerical labels to non-hydrogen atoms (Fig. 3b).

(3) Path definition. Define an ordered set W representing the transversal path. In Fig. 3c, the path $P = \{(C_3, C_2), (C_2, C_4), (C_4, C_6)\}$ corresponds to $W = \{C_3, C_2, C_4, C_6\}$.

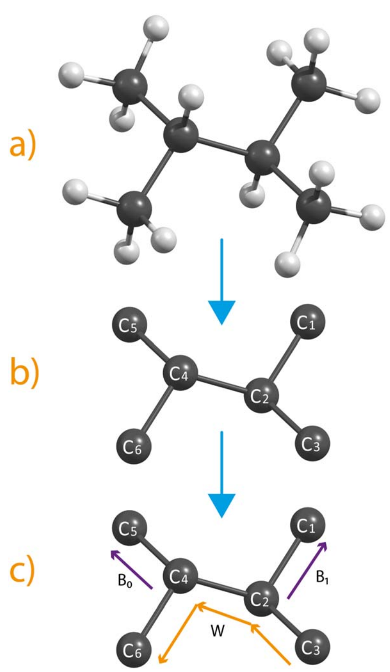

Fig. 3 (a) Molecular model for 2,3-dimethylbutane; (b) hydrogen-free molecular graph with labeled carbon atoms; (c) traversal pathway (orange and purple arrows) covering all atoms.

(4) Branch identification. Identify branches B_m, where each branch contains the maximum possible number of connected atoms. The ordered set of branches is $B = \{B_0, B_1, \ldots, B_m\}$. For the system in Fig. 3c, $B_0 = \{C_4, C_5\}$ and $B_1 = \{C_2, C_1\}$.

(5) Branch insertion. Insert the branches $B = \{B_0, B_1\}$ into W, omitting the first atom of each branch since it already appears in W. Parentheses mark the branch boundaries:

$$\{C_3, C_2, (, C_1,), C_4, (, C_5,), C_6\} \quad (1)$$

(6) Symbol replacement and concatenation. Replace the atomic labels in (1) with the corresponding atomic symbols and concatenate them to obtain the string CC(C)C(C)C.

The resulting *SMILES* string, CC(C)C(C)C, represents 2,3-dimethylbutane with implicit hydrogen atoms. Since *SMILES* grammar is non-commutative, the steps must be performed in the specified order.

Tokenization of *SMILES* into *TokenSMILES*

Our method transforms the Kekulé syntax into a standardized form that equalizes string lengths and isolates chemical information by assigning individual tokens to each atom and symbol. For example, the *SMILES* representation of 2,3-dimethylbutane, CC(C)C(C)C, can be rewritten as [C, C, (C), C, (C), C], where "(" and ")" denote the beginning and end of branches, respectively. So, in this tokenized form, branches occur at positions 1 and 3 (zero-indexed). The resulting sequence of tokens constitutes a standardized-length representation referred to as *TokenSMILES*.

This tokenization follows two sequential rules: First, the original string is parsed into individual characters, each enclosed in square brackets. For CC(C)C(C)C, this yields [C, C, (, C,), C, (, C,), C], maintaining the exact order of symbols in the original notation. In this representation, all symbols are enclosed within square brackets to form an ordered sequence. Although conventional set notation implies unique elements, here repeated tokens are intentionally preserved to retain positional information.

Second, the tokens are categorized according to their syntactic context. Left-context symbols, [, (, =, and #, are placed immediately before atomic symbols, while right-context symbols, @,), %,], and a digit n, are placed immediately after them. Applying these rules to the example yields the standardized *TokenSMILES* form [C, C, (C), C, (C), C].

Grammar constraints for *TokenSMILES*

The production rules used to generate *SMILES* strings from molecular paths allow multiple valid representations for a single structure, making exhaustive *SMILES* enumeration a non-deterministic polynomial-time hard (NP-hard) problem.[19,20] To mitigate this redundancy, and following previous work on grammatical constraints in formal languages,[21] we introduce five grammar rules to reduce the number of equivalent strings representing organic isomers.

Constraint 1. For a molecular model G without ring number and dot bonds, *TokenSMILES* strings are constructed from traversal paths W that include all non-hydrogen atoms and bonds:

$$W = \{a_0, a_1, \ldots, a_{n-1}\} \text{ or } W = [a_0, a_1, \ldots, a_{n-1}],$$

where each $a_i \in W$ denotes a non-hydrogen atom.

Constraint 2. Branch symbols "(" and ")" are not permitted at terminal positions (a_0, a_1, and a_{n-1}) since these locations cannot generate new expressions.

Constraint 3. To control branching, the second and penultimate atoms (a_2 and a_{n-2}) may optionally include branch symbols, *i.e.* $[a_2|(a_2)]$ and $[a_{n-2}|(a_{n-2})]$. These optional insertions produce distinct yet grammatically valid *SMILES* variants.

Constraint 4. Parentheses introduced between a_3 and a_{n-3} must remain balance, with every opening parenthesis "(" matched by a corresponding closing parenthesis ")". Valid expressions include $[a_i|(a_i|a_i)|(a_i)]$, ensuring structural consistency across the entire sequence.

Constraint 5. Aromaticity symbols (c, n, b, p, s, and o) are excluded to retain the grammar to uppercase atomic symbols (C, N, B, P, S, and O).

Production rules for alkane *TokenSMILES*

This section presents a systematic procedure for generating *TokenSMILES* representations of alkanes under the grammatical constraints defined previously. Using the alphabet {C, ( )}, a token dictionary is defined as $\pi = \{\text{C}, (\text{C}), (\text{C}, \text{C})\}$. To construct *SMILES* for all C_nH_{2n+2} isomers, we define a molecular path $W = [a_0, a_1, \ldots, a_{n-1}]$, where each $a_i \in W$ is replaced by tokens from π *via* the production rule P:

$$P \rightarrow a_0 \oplus a_1 \oplus \ldots \oplus a_{n-2} \oplus a_{n-1} \tag{2}$$

Here, the concatenation operator ($\oplus$) joints the strings in W, and the arrow ($\rightarrow$) indicates the production process.

Applying Constraint 2, a_0, a_1 and a_{n-1} are replaced by "C" without branch symbols, resulting in:

$$P \rightarrow \text{C} \oplus \text{C} \oplus a_2 \oplus \ldots \oplus a_{n-2} \oplus \text{C} \tag{3}$$

Next, Constraint 3 specifies that a_2 and a_{n-2} may take the forms [C|(C)], yielding:

$$P \rightarrow \text{C} \oplus \text{C} \oplus [\text{C}|(\text{C})] \oplus a_3 \oplus \ldots \oplus a_{n-3} \oplus [\text{C}|(\text{C})] \oplus \text{C}. \tag{4}$$

For simplify, the inner sequence $a_3 \oplus \ldots \oplus a_{n-3}$ is replaced with a variable Ω, rewriting (4) as:

$$P \rightarrow \text{C} \oplus \text{C} \oplus [\text{C}|(\text{C})] \oplus \Omega \oplus [\text{C}|(\text{C})] \oplus \text{C} \tag{5}$$

To define the instances in Ω, Constraint 4 is applied. Each element in $\{a_3, \ldots, a_{n-3}\}$ has four possible forms: [C|(C|C)|(C)]. Balanced parentheses are maintained through recursive rules.

The first rule, $q_0 \rightarrow \text{C}|\text{C}q_0$, generates chains such as "CC", "CCC", and "CCCC". The second, $q_1 \rightarrow (\text{C})|(\text{C})q_1$, introduces terminal branches represented by parentheses. Combinations of the two are obtained using $q_2 \rightarrow q_0|q_1|q_0q_1|q_1q_0$, allowing permutations of linear and branched fragments. Nested branches are introduced through $q_3 \rightarrow (\text{CC})|(\text{C}q_3\text{C})|(\text{C}q_2\text{C})$, which ensures balanced parentheses within multiple levels of branching. Finally, the general case is described by $q_4 \rightarrow q_2|q_3|q_2q_3|q_3q_2$, encompassing all possible balanced expressions in [C|(C|C)|(C)]. Substituting q_4 into (5) gives:

$$P \rightarrow \text{C} \oplus \text{C} \oplus [\text{C}|(\text{C})] \oplus [q_4] \oplus [\text{C}|(\text{C})] \oplus \text{C} \tag{6}$$

Using (6), the number of *SMILES* representations for C_nH_{2n+2} isomers decreases drastically. For example, Fig. 4 shows the construction of *SMILES* for C_6H_{14} using (6) with $[q_4] = [\text{C}|(\text{C})]$:

$$P \rightarrow \text{C} \oplus \text{C} \oplus [\text{C}|(\text{C})] \oplus [\text{C}|(\text{C})] \oplus [\text{C}|(\text{C})] \oplus \text{C} \tag{7}$$

However, (7) does not account for atomic equivalence or valence restrictions, which may result in redundant (*e.g.*, [C, C, (C), C, C, C] ≡ [C, C, C, C, (C), C]) or chemically invalid strings (*e.g.*, [C, C, (C), (C), (C), C]). To address these limitations, the next section introduces the semantic parsing of *TokenSMILES*, which filters out chemically inconsistent structures and enforces the octet rule.

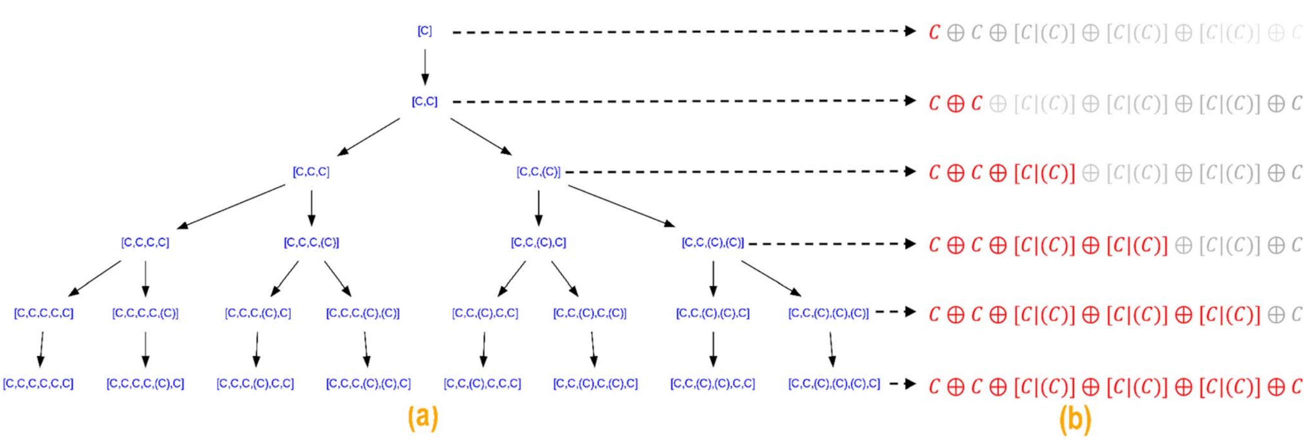

Fig. 4 (a) A generator tree for the *SMILES* strings of the C_6H_{14} system and (b) the progress of the production rule (7) to obtain the atomic symbols in the *SMILES*.

Chemical context definition

Semantic parsing[22,23] was applied to determine the connectivity encoded in each *TokenSMILES* string. Following the conventions of Weininger *et al.*, each atom (α) is represented by its atomic symbol, and the symbol is linked to a set of chemical properties collectively referred to as the context of the atom, Context(α). For our analysis, the context is defined as:

$$\text{Context}(\alpha) = (p_0, p_1, p_2)$$

where p_0 is the atomic symbol, p_1 corresponds to the valence, and p_2 is the atomic connectivity.

As an example, the *TokenSMILES* [C, C, (C), C, (C), C] can be indexed as [C_0, C_1, (C_2), C_3, (C_4), C_5]. The chemical context for each substring is summarized in Table 3, where the first element identifies the atomic symbol ("C"), the second indicates the valence (4), and the third lists the corresponding connectivity set.

Connectivity extraction from *TokenSMILES*

In *SMILES* notation, connectivity is often implicit and must be derived through semantic parsing to reconstruct chemical relationships. Fig. 5 shows the procedure used to extract connectivity from the *TokenSMILES* sequence [C_0, C_1, (C_2), C_3, (C_4), C_5]:

(a) "C_0" is not concatenated with any other token, so its connectivity set remains empty.

(b) "C_1" is concatenated with "C_0", creating a new bond represented by the tuple (0, 1).

(c) The same procedure is applied to the remaining tokens. Fig. 5c shows how (C_2) is concatenated, using parentheses to indicate the start and end of a branch emerging from "C_1", adding the tuple (1, 2).

(d) "C_3" is then concatenated, forming the tuple (1, 3) rather than (2, 3), since "(C_2)" is a closed branch.

(e) "(C_4)" is concatenated next, adding the tuple (3, 4).

(f) Finally, "C_5" is concatenated, producing the tuple (3, 5).

After processing all tokens, the resulting bond set is {(0, 1), (1, 2), (1, 3), (3, 4), (3, 5)}, corresponding to [C, C, (C), C, (C), C]. Each tuple represents the connectivity in the *TokenSMILES* notation.

As shown in Fig. 5a, the bond contexts initially consist of empty sets. As the strings are concatenated according to the *SMILES* grammar and defined constraints, their contexts expands as new bonds are introduced. From this stage onward, the strings listed in Table 3 are treated as context-free strings or

Table 3 Chemical context of [C_0, C_1, (C_2), C_3, (C_4), C_5]

String	Chemical context
C_0	(C, 4, {(0,1)})
C_1	(C, 4, {(0,1), (1,2), (1,3)})
(C_2)	(C, 4, {(1,2)})
C_3	(C, 4, {(1,3), (3,4), (3,5)})
(C_4)	(C, 4, {(3,4)})
C_5	(C, 4, {(3,5)})

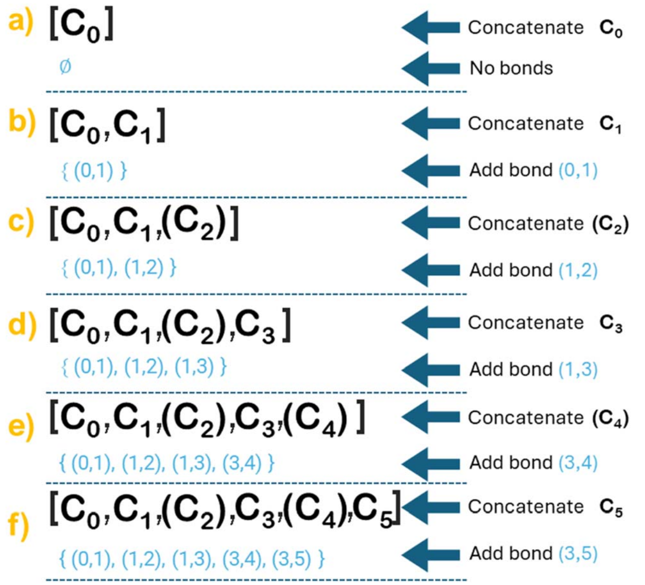

Fig. 5 Procedure for extracting bonds from the *TokenSMILES* of 2,3-dimethylbutane. (a)–(f) Depict the sequential steps of the method.

simply words.[22] These words lack explicit connectivity to other atoms, excluding hydrogen. Consequently, a *TokenSMILES* can be interpreted as a sequence of such words forming a chemically meaningful sentence.

To show this process, we return to the example of the C_6H_{14} isomers. Using production rule (7), connectivity was assigned to each *TokenSMILES* string, and those that violated the octet rule were removed, yielding seven valid *SMILES* candidates (Fig. 6). To eliminate duplicates, each *TokenSMILES* was converted to a canonical *SMILES* form using the canonicalization algorithm implemented in the *RDkit* module.[24] Strings sharing the same canonical form were identified, and only unique *SMILES* were retained. After filtering, five distinct *SMILES* remained, corresponding precisely to the five constitutional isomers of C_6H_{14}.

Integrating grammatical constraints into the Kekulé syntax transforms variable-length isomer strings into standardized, fixed-length representations, reducing redundancy and ensuring syntactic coherence. For example, classical SMILES enumeration of C_6H_{14} yields 125 valid strings, whereas *TokenSMILES* generates only seven normalized candidates: [C, C, C, C, C, C], [C, C, C, C, (C), C], [C, C, C, (C), C, C], [C, C, C, (C), (C), C], [C, C, (C), C, C, C], [C, C, (C), C, (C), C], and [C, C, (C), (C), C, C], each conforming to a consistent six-word structure (Table 4).

Modifying *TokenSMILES* stoichiometry

This section defines the rules for modifying *TokenSMILES* syntax to change stoichiometry and to generate *SMILES* for systems beyond C_nH_{2n+2}. Three operations are introduced: two insertion operations, which add new symbols into a word, and one replacement operation, which alters atomic symbols. Each operation produces a copy of the entire sentence while recording the applied modifications. This procedure enables the systematic generation of isomers with stoichiometries different from those of alkanes.

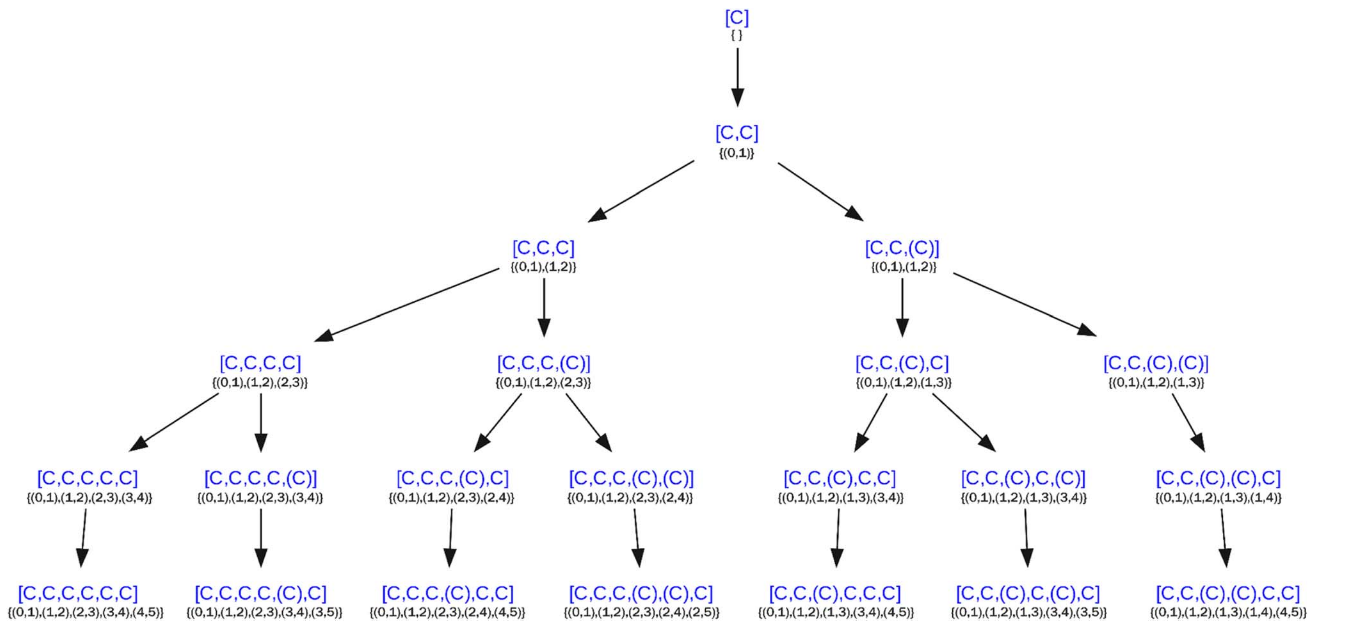

Fig. 6 Production of valid *SMILES* for the C_6H_{14} isomers. Unlike Fig. 5, the invalid *TokenSMILES* [C, C, (C), (C), (C), C] is omitted here because it does not satisfy the octet rule.

Table 4 *TokenSMILES* and *SMILES* for C_6H_{14} isomers

TokenSMILES	*SMILES*
String lengths normalized to 6 words	String length varies from 6 to 14 symbols
[C,C,C,C,C,C]	CCCCCC, CCCCC(C), CCCC(CC), CCC(CCC), C(CC)CCC, C(CC)(CCC), C(CC)CC(C), C(CC)C(CC), C(CCC)(CC), C(CCC)CC, C(CCC)C(C), C(CCCC)C, C(CCCC)(C), C(CCCCC), C(C)(CCCC), C(C)CCC(C), C(C)CC(CC), C(C)CCCC, CC(CCCC), C(C)C(CCC)
[C,C,C,C,(C),C] [C,C,(C),C,C,C]	CCCC(C)C, CCCC(C)(C), CCC(C(C)C), CC(C)CCC, CC(C)CC(C), C(CC)(C(C)C), C(C(C)C)(CC), C(C(CCC)C), C(C(C)CCC), C(C(C)C)C(C), C(C(C)C)CC, C(CCC(C)C), C(CCC)(C)C, C(CCC)(C)(C), C(CC(C)C)C, C(CC(C)C)(C), C(C)(CCC)(C), C(C)(CCC)C, C(C)(C)(CCC), C(C)(C)C(CC), C(C)CC(C)(C), C(C)(C)CC(C), C(C)C(C(C)C), C(C)CC(C)C, C(C)(C)CCC, CC(CCC)C, CC(CC(C)C), CC(CCC)(C), CC(C)(CCC), CC(C)C(CC), C(CC)C(C)C, C(C)(CC(C)C)
[C,C,C,(C),C,C]	CCC(C)CC, CCC(C)C(C), CCC(C)(CC), CCC(CC)C, CCC(CC)(C), C(CC)(C)CC, C(CC)(CC)C, C(CC)(CC)(C), C(C(CC)CC), C(CC)(C)(CC), C(C(CC)C)C, C(C(C)CC)C, C(C(C)CC)(C), C(CC(C)CC), C(CC(CC)C), C(C)(CC)(CC), C(C)(C(CC)C), C(C)(CC)CC, C(C)(CC)C(C), C(C)(C(C)CC), CC(CC)CC, CC(CC)C(C), C(C)C(C)(CC), C(C)C(C)CC, C(C)C(C)C(C), C(C)C(CC)C, C(C)C(CC)(C), CC(C(C)CC), CC(C(CC)C), CC(CC)(CC), CC(C(C)(C)C), C(CC)(C)C(C)
[C,C,(C),C,(C),C]	C(C(C(C)C)C), C(C(C)C)(C)C, C(C(C)C)(C)(C), C(C(C)C(C)C), C(C)(C(C)C)(C), C(C)(C(C)C)C, C(C)(C)(C(C)C), C(C)(C)C(C)(C),C(C)(C)C(C)C, CC(C(C)C)(C), CC(C(C)C)C, CC(C)C(C)C, CC(C)(C(C)C), CC(C)C(C)(C)
[C,C,(C),(C),C,C] [C,C,C,(C),(C),C]	CCC(C)(C)C, CCC(C)(C)(C), C(CC)(C)(C)C, C(C(CC)(C)C), C(CC)(C)(C)(C), C(C(C)(C)CC), C(C(C)(C)C)C, CC(C)(C)C(C), C(C(C)(CC)C), C(CC(C)(C)C), C(C(C)(C)C)(C), C(C)(CC)(C)(C), C(C)(CC)(C)C, C(C)(C(C)(C)C), C(C)(C)(CC)(C), CC(C)(C)CC, C(C)(C)(C)(CC), C(C)(C)(CC)C, C(C)(C)(C)CC, C(C)(C)(C)C(C), CC(CC)(C)C, C(C)C(C)(C)C, C(C)C(C)(C)(C), CC(CC)(C)(C), CC(C)(CC)C, CC(C)(CC)(C), CC(C)(C)(CC)

The modification of *TokenSMILES* stoichiometry is achieved through the analysis of nesting levels and adjacency sets, which describe the implicit connectivities within a string. Each word in a *TokenSMILES* carries a grammar-based index that defines its nesting depth and adjacency relations. By editing these relations while preserving valence consistency, the algorithm modifies molecular topology in a rule-based manner. This approach allows direct structural transformations, such as the insertion of rings or double bonds, from the *TokenSMILES* representation without relying on IUPAC nomenclature or predefined templates (see SI for details).

To illustrate the process, we begin with 2,3-dimethylbutane (C_6H_{14}) and modify it to generate 1-cyclopropylideneethanol (C_5H_8O, Fig. 7). First, the connectivity set is extracted as {(0, 1), (1, 2), (1, 3), (3, 4), (3, 5)}. Using these bonds, the algorithm identifies adjacent words representing bonds in *TokenSMILES*. Words at positions 1 and 3 (Fig. 7a) are selected for double-bond insertion. To evaluate feasibility, the valence excess (Δ) is computed for each atom, defined as the difference between its valence and its degree (the number of incident bonds). When $\Delta \geq 1$, a " = " symbol is inserted into the word at the higher position in the sentences (Fig. 7b).

To prevent excessive " = " insertions, the hydrogen deficiency index (HDI) of the initial system (C_6H_{14}) is used as a control variable. Each additional double bond increases the HDI by one, ensuring that the connectivity modifications remain chemically valid and stoichiometrically consistent.

Next, to insert a cycle, two non-adjacent words are selected based on the same connectivity {(0, 1), (1, 2), (1, 3), (3, 4), (3, 5)}. If no bond exists between the chosen positions, the algorithm proceeds with cycle insertion. For the example in Fig. 7b, words at positions 0 and 2 are chosen, and the absence of bond (0, 2) is confirmed. The valence excess for both atoms in then evaluated; if $\Delta > 0$ for each, cycle symbols are inserted (Fig. 7c). A random ring number between 1 and 9 is assigned and placed to the right of the atomic symbol (*e.g.*, "C1"). For numbers 10–99, the cycle symbol is prefixed with "%" (*e.g.*, "C%10"). Each cycle insertion increases the HDI by one (Fig. 7c).

Finally, to substitute a carbon atom with oxygen, the condition degree(C) $\leq$ valence(O) must be satisfied before replacement. The operation reduces the carbon and hydrogen count by one and two, respectively (Fig. 7d). For heteroatoms, the allowed are B, Br, Cl, F, I, N, O, P, and S. This approach generates all structural isomers of C_5H_8O (Table SI1). Once all transformations are complete, equivalent *SMILES* are filtered using the canonicalization algorithm in *RDKit*.

SmilX software

To automate the generation of *SMILES* under grammatical constraints, the *SmilX* program was developed as an open-source Python tool, available at **https://github.com/LuisOrz/SmilX.git**. *SmilX* constructs *SMILES* representations of isomers while maintaining compliance with the specified stoichiometry. A user-oriented web interface was also implemented using *Streamlit*, which provides both interactive functionality and server infrastructure. The interface is included with the package and accessible at **https://smilx-isogenerator.streamlit.app/**.

The workflow begins with a molecular formula provided as a string, such as C_nH_{2n+2}. Based on this input, *SmilX* generates all possible *TokenSMILES* corresponding to the defined stoichiometry and adjusts their syntax accordingly. The resulting words in each *TokenSMILES* are concatenated to form complete *SMILES* strings, which are then processed using a canonicalization algorithm to eliminate duplicates and retain unique structures. Finally, the software uses the *RDKit* module to generate stereoisomeric variants, returning a curated list of *SMILES* strings that satisfy the input molecular formula.

Results and discussion

Two experiments were performed to evaluate the isomer-generation capabilities of *SmilX* and to validate the *TokenSMILES* framework. The first reproduced the data reported by Elyashberg *et al.*[25] for C–H systems (Fig. 7), which serve as reference structures for more complex compositions. The resulting isomer counts from *SmilX* were compared with those obtained using *MOLGEN* (a matrix-based generator) and *MAYGEN* (an open-source, *Java*-based tool). The second experiment assessed *SmilX*'s performance in systems containing heavier elements (O, N, Cl) across a range of HDI and atom counts.

First experiment

Isomer enumeration followed the molecular formulas reported by Elyashberg *et al.*,[25] to allow direct comparison between *SmilX* and *MAYGEN*. Systems containing at least two hydrogen atoms were prioritized, while hydrogen-free cases (C_6 or C_{10}) were excluded by omitting the "C = *n*" notation.

SmilX reproduced Elyashberg's isomer counts for most systems (Fig. 8). Minor deviations occurred when the HDI approached the total heavy-atom count: *SmilX* occasionally yielded one missing structure (false negative, *e.g.*, C_8H_4) or 1–7

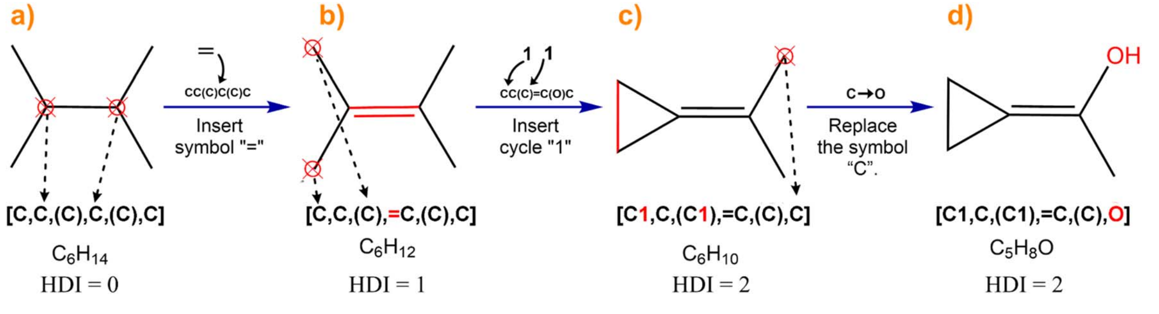

Fig. 7 Syntax transformation from 2,3-dimethylbutane *TokenSMILES* to 1-cyclopropylideneethanol *TokenSMILES*. (a)–(d) Show the sequential operations: double-bond insertion, cycle formation, and heteroatom substitution. Changes are highlighted in red, and dotted arrows indicate the modified word.

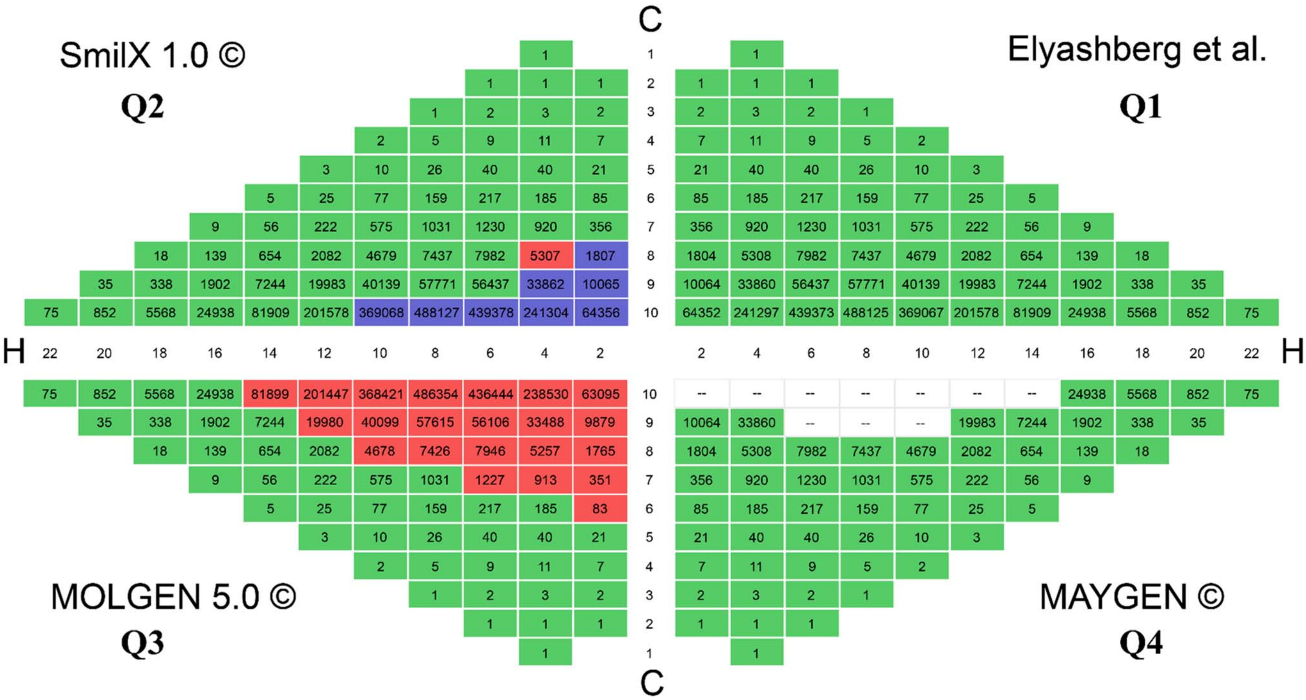

Fig. 8 Isomers composed of carbon and hydrogen as reported by Elyashberg *et al.* (Quadrant 1) compared with those obtained using *SmilX* (Q2), *MOLGEN* (Q3), and *MAYGEN* (Q4). Green cells indicate matching isomers; blue cells represent false positives; red cells correspond to false negatives; and white cells denote missing data due to resource limitations. All results are shown relative to the reference data from Elyashberg *et al.*

additional isomers (false positives) in systems as C_8H_2, C_9H_2, $C_{10}H_2$, C_9H_4, $C_{10}H_4$, $C_{10}H_6$, C_9H_8, and $C_{10}H_{10}$ (Fig. 8).

MOLGEN accurately enumerated isomers at low HDI values but showed decreased accuracy as HDI increased (Fig. 8). In contrast, *MAYGEN*'s online version encountered memory saturation in larger systems (C_9H_6, C_9H_8, C_9H_{10}, $C_{10}H_2$, $C_{10}H_4$, $C_{10}H_6$, $C_{10}H_8$, $C_{10}H_{10}$, $C_{10}H_{12}$, $C_{10}H_{14}$, and $C_{10}H_{16}$) but reproduced Elyashberg's data for smaller, computationally feasible cases.

MAYGEN employs a canonicalization procedure distinct from *RDKit*'s implementation of the Weininger algorithm.[16] Consequently, canonical *SMILES* can struggle to distinguish equivalent atoms in molecules containing multiple nested rings, particularly when HDI is equal to or greater than half the number of heavy atoms. *SmilX*'s reliance on *RDKit* likely accounts for the few observed false positives and negatives.

Second experiment

Building on these results, the second experiment examined *SmilX*'s robustness in systems of increasing compositional complexity: (C, H, O), (C, H, O, N), and (C, H, O, N, Cl). These compositions typically produce more isomers than pure C–H systems. For each composition, six molecular formulas were generated for HDI values ranging from 0 to 5, and the number of heavy atoms was limited to 3–10 to avoid cases where HDI ≥ half the heavy-atom count, previously associated with enumeration errors. Results are summarized in Tables SI1–SI3.

SmilX showed efficient performance, aided by disk-caching optimization. All three tools produced identical isomer counts for HDI ≤ 4, with small discrepancies emerging at higher HDI values. These deviations further support the hypothesis that *RDKit*'s canonicalization algorithm faces difficulties in handling highly nested ring topologies.

The *TokenSMILES* framework effectively reduced both string redundancy and computational overhead through grammatical constraints and caching. While *SmilX* correctly enumerated the majority of organic systems, boundary cases where HDI approached the heavy-atom count remained problematic. These discrepancies are consistent with the theoretical limitations of canonicalization algorithms in systems containing high symmetry or multiple fused rings, rather than with specific software errors. Despite these edge-case issues, *SmilX* maintained low execution times, and the reuse of cached structures enabled scalable exploration of extensive chemical spaces.

Classical *SMILES* representations provide a foundation for cheminformatics but suffer from significant redundancy, as shown in Table 4. To overcome this limitation, the present work redefines *SMILES* not merely as atomic sequences but as grammatically structured sentences, a conceptual framework rooted in formal language theory. The *TokenSMILES* approach implements this through hierarchical syntax (word- and sentence-level organization), enforced grammatical constraints, and standardized string lengths. This reformulation reduces redundancy, ensures systematic chemical-space coverage, and facilitates new computational applications.

Unlike conventional structure generators such as *MOLGEN* or *MAYGEN*, *TokenSMILES* emphasizes formal representation rather than speed or memory efficiency, which justifies the omission of runtime benchmarks. Similarly, unlike cheminformatics toolkits such as *RDKit*, it augments rather than replaces *SMILES* syntax by operating on grammatical constructs. This strategy enables dynamic programming, partial-solution reuse, and exploration beyond the limitations of purely graph-based methods, while remaining compatible with evolutionary and machine-learning algorithms.

Conclusions

This study presents *TokenSMILES* as a grammatical framework that redefines the *SMILES* language through explicit syntactic rules. By interpreting *SMILES* as structured sentences composed of context-free words, *TokenSMILES* minimizes redundancy and enforces grammatical consistency. Constraints on branching, parentheses balance, and aromaticity reduce the number of valid *SMILES* variants for C_nH_{2n+2} isomers. This structured representation facilitates systematic chemical-space exploration while ensuring valence and octet compliance through semantic parsing. Integrated within *SmilX*, *TokenSMILES* performs comparably to *MOLGEN* and *MAYGEN* in generating isomers for systems with low hydrogen deficiency (HDI $\leq$ 4), demonstrating reliable canonical *SMILES* generation.

Beyond alkanes, *TokenSMILES* enables stoichiometric modifications such as bond insertion, cyclization, and heteroatom substitution, extending its applicability to broader organic systems. In high-HDI cases, minor misidentifications arise from *RDKit*'s canonicalization limitations, suggesting the need for improved feasibility checks.

Currently, *TokenSMILES* prioritizes grammatical completeness and semantic accuracy over computational efficiency. Although benchmarking was not the focus of this work, future versions could adopt optimization strategies inspired by algebraic isomer generators. The framework is inherently compatible with machine learning due to its discrete syntax, fixed-length representations, and reusable grammatical components, which enable hybrid symbolic–neural modeling and grammatical evolution.

Treating *SMILES* as grammatically structured sentences introduces a new paradigm for cheminformatics, linking linguistic theory with chemical representation. This approach supports machine-learning-based molecular design and systematic chemical-space mapping. Future extensions to polymers, organometallics, and crystalline systems may open new applications in materials and drug discovery.

Conflicts of interest

There are no conflicts to declare.

Data availability

The data supporting this article have been included as part of the supplementary information (SI). Supplementary information: the comparison of the results obtained with *SmilX*, *MOLGEN*, and *MAYGEN*. It also provides an explanation of the nesting concept in a *TokenSMILES* string. See DOI: **https://doi.org/10.1039/d5sc05004a**.

Acknowledgements

This work was supported by Cinvestav. A. G.-O. thank Secihti for their PhD fellowship. L. N. thanks Secihti for the postdoctoral fellowship.

References

1 D. Weininger, *J. Chem. Inf. Comput. Sci.*, 1988, **28**, 31–36, DOI: **10.1021/ci00057a005**.
2 D. Weininger, A. Weininger and J. L. Weininger, *J. Chem. Inf. Comput. Sci.*, 1989, **29**, 97–101, DOI: **10.1021/ci00062a008**.
3 D. Weininger, *J. Chem. Inf. Comput. Sci.*, 1990, **30**, 237–243, DOI: **10.1021/ci00067a005**.
4 H. Kim, J. Na and W. B. Lee, *J. Chem. Inf. Model.*, 2021, **61**, 5804–5814, DOI: **10.1021/acs.jcim.1c01289**.
5 M. Quirós, S. Gražulis, S. Girdzijauskaitė, A. Merkys and A. Vaitkus, *J. Cheminf.*, 2018, **10**, 1–17, DOI: **10.1186/s13321-018-0279-6**.
6 C. A. James, R. Apodaca, N. O'Boyle, A. Dalke, J. van Drie, P. Ertl, G. Hutchison, G. Landrum, C. Morley, E. Willighagen, H. De Winter, T. Vandermeersch and J. May, OpenSMILES specification, Version 1.0, 2016-05-15, **http://opensmiles.org/opensmiles.html**, accessed 14 February 2025.
7 N. O'Boyle and A. Dalke, DeepSMILES: An Adaptation of SMILES for Use in Machine-Learning of Chemical Structures, *ChemRxiv*, 2018, DOI: **10.26434/chemrxiv.7097960.v1**.
8 J.-N. Wu, T. Wang, Y. Chen, L.-J. Tang, H.-L. Wu and R.-Q. Yu, *Nat. Commun.*, 2024, **15**, 4993, DOI: **10.1038/s41467-024-49388-6**.
9 T.-S. Lin, C. W. Coley, H. Mochigase, H. K. Beech, W. Wang, Z. Wang, E. Woods, S. L. Craig, J. A. Johnson, J. A. Kalow, K. F. Jensen and B. D. Olsen, *ACS Cent. Sci.*, 2019, **5**, 1523–1531, DOI: **10.1021/acscentsci.9b00476**.
10 A. Drefahl, *J. Cheminf.*, 2011, **3**, 1–7, DOI: **10.1186/1758-2946-3-1**.
11 M. Krenn, F. Häse, A. Nigam, P. Friederich and A. Aspuru-Guzik, *Mach. Learn. Sci. Technol.*, 2020, **1**, 045024, DOI: **10.1088/2632-2153/aba947**.
12 A. H. Cheng, A. Cai, S. Miret, G. Malkomes, M. Phielipp and A. Aspuru-Guzik, *Digital Discovery*, 2023, **2**, 748–758, DOI: **10.1039/d3dd00012e**.
13 J. Arcudia, F. Ortíz-Chi, A. Sánchez-Valenzuela, A. Aspuru-Guzik and G. Merino, *Matter*, 2023, **6**, 1503–1513, DOI: **10.1016/j.matt.2023.02.014**.
14 J. Arcudia, F. Ortíz-Chi, J. Barroso and G. Merino, *Nanoscale*, 2025, **17**, 2215–2223, DOI: **10.1039/D4NR03696D**.
15 C. Benecke, T. Grüner, A. Kerber, R. Laue and T. Wieland, *Fresenius. J. Anal. Chem.*, 1997, **359**, 23–32, DOI: **10.1007/s002160050530**.
16 M. A. Yirik, M. Sorokina and C. Steinbeck, *J. Cheminf.*, 2021, **13**, 1–14, DOI: **10.1186/s13321-021-00529-9**.

17 A. Schüller, G. Schneider and E. Byvatov, *QSAR Comb. Sci.*, 2003, **22**, 719–721, DOI: 10.1002/qsar.200310008.

18 A. Schüller, V. Hähnke and G. Schneider, *QSAR Comb. Sci.*, 2007, **26**, 407–410, DOI: 10.1002/qsar.200630101.

19 D. S. Hochbaum, *Approximation Algorithms for NP-hard Problems*, ACM Sigact News, 1997, vol. 28, pp. 447–476, ISBN 978-0534949681.

20 R. M. Wharton, *Inf. Contr.*, 1977, **33**, 253–272, DOI: 10.1016/S0019-9958(77)80005-3.

21 S. Kadioglu and M. Sellmann, *Constraints*, 2010, **15**, 117–144, DOI: 10.1007/s10601-009-9073-4.

22 S. L. Graham and M. A. Harrison, in *Advances in Computers*, ed. M. Rubinoff and M. C. Yovits, Elsevier, 1976, vol. 14, pp. 77–185, DOI: 10.1016/S0065-2458(08)60451-9.

23 D.-Q. Zhang, K. Zhang and J. Cao, *Comput. J.*, 2001, **44**, 186–200, DOI: 10.1093/comjnl/44.3.186.

24 G. Landrum, P. Tosco, B. Kelley, R. Rodriguez, D. Cosgrove, R. Vianello, Sriniker, P. Gedeck, G. Jones, N. Schneider, E. Kawashima, D. Nealschneider, A. Dalke, M. Swain, B. Cole, S. Turk, A. Savelev, T. Hurst, A. Vaucher, M. Wójcikowski, I. Take, V. F. Scalfani, R. Walker, K. Ujihara, D. Probst, J. Lehtivarjo, H. Faara, G. Godin, A. Pahl and J. Monat, *rdkit/rdkit: 2024_09_5 (Q3 2024) Release*, Zenodo, 2025, DOI: 10.5281/zenodo.28640.

25 M. Elyashberg, A. Williams and K. Blinov, *Contemporary Computer-assisted Approaches to Molecular Structure Elucidation*, Royal Society of Chemistry, Cambridge, U.K, 2012, vol. 1, pp. 28–30, ISBN 978-1-84973-432-5.