Machine translation between BigSMILES line notation and chemical structure diagrams

*Michael E. Deagen[1], Bérenger Dalle-Cort[1], Nathan J. Rebello[1], Tzyy-Shyang Lin[1], Dylan J. Walsh[1], and Bradley D. Olsen[1,]**

[1]Department of Chemical Engineering, Massachusetts Institute of Technology, 77 Massachusetts Avenue, Cambridge, Massachusetts 02139, USA

***E-mail: bdolsen@mit.edu**

KEYWORDS

Line notation, structure diagram generation, macromolecular representation, polymer informatics, human–computer interaction

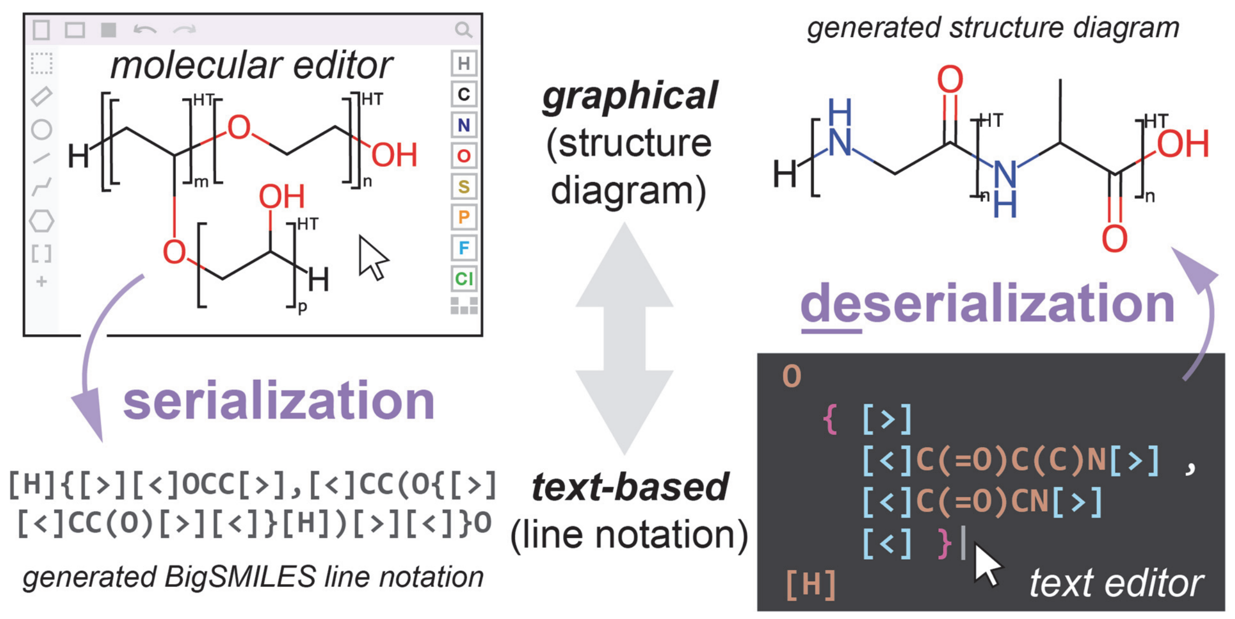
ABSTRACT

The representation of chemical structure forms a core component of polymer science, yet the chemical structure diagrams used to convey such information lack the machine-processability vital for automating analysis, managing abundant data, and harnessing the potential of informatics. On the other hand, the usage of BigSMILES language—a machine-readable representation of polymer chemical structure—requires specialized knowledge of its grammar and syntax. Here, the algorithmic translation between chemical structure diagrams and BigSMILES line notation is demonstrated, providing seamless interconversion to and from the *lingua franca* of polymer chemists across a broad array of polymer architectures (e.g., copolymers, graft and segmented polymers, star polymers, macrocycles, networks, ladder polymers). Serialization from structure diagram into BigSMILES line notation is accomplished by parsing the contents of a connection table and iteratively assembling string representations of the molecular graph and its substructures. Deserialization from BigSMILES line notation into a structure diagram involves parsing the line notation string into a stochastic graph representation, from which a valid graph traversal defines a representative sequence of substructural units comprising the connection table (i.e., structure diagram). These algorithms were validated through round-trip translation on a curated set of 300 polymer structure diagrams, demonstrating semantic preservation of the molecular graph in over 99% of cases and visually equivalent structure diagrams in 38% of cases. The 2-D layout, an isometry of the atomic coordinates generated by the CoordGen library within RDKit, shows the applicability of readily available atomic layout generation algorithms while revealing specific areas in which to improve these layout algorithms for polymers—for example, 60% of test cases could be rectified by orienting backbone atoms in an extended configuration along a horizontal axis. Implemented in JavaScript, this software offers facile integration with web-based resources

and forms an essential interface between informatics and the broader polymer research community. By enabling humans and machines to process vast amounts of polymer chemical structural data, this work aims to democratize access to polymer informatics and foster increasingly interdisciplinary approaches to polymer research.

INTRODUCTION

Despite the increased application of machine learning and artificial intelligence in polymer science, a lack of machine-readable polymer representations and sufficiently detailed datasets have hindered further progress in the field.[1] The urgency for innovation in polymer science and engineering has become increasingly evident as polymeric materials play an irreplaceable role in consumer goods, infrastructure, and public health while simultaneously producing a growing environmental footprint.[2] In the field of polymer informatics, readily available opportunities for expediting research and development exist by improving the interfaces between humans and machines handling polymer data.[3] One key interface involves the discrepancy between the graphical representation preferred by polymer chemists and the sequences of bytes used by machines to store and process information. In this article, a novel methodology and set of algorithms are demonstrated that automate the translation between chemical structure diagrams and BigSMILES line notation, capturing the stochastic structure of polymers in a string-based format for machines and diagrammatic depictions for polymer chemists (**Figure 1**).



```
[H]{[>][<]OCC[>],[<]CC(O{[>]
 [<]CC(O)[>][<]}[H])[>][<]}O
```

```
O
  { [>]
    [<]C(=O)C(C)N[>] ,
    [<]C(=O)CN[>]
    [<] }
[H]
```

Figure 1. Interconversion between macromolecular representations forms an essential interface in polymer informatics, allowing humans and machines to interpret and process chemical structural data for polymers. *Serialization* of structure diagrams into BigSMILES strings enables the graphical input of molecular editor software to be encoded into compact string representations, while *deserialization* of BigSMILES line notation into structure diagrams enables the "programming" of polymers inside a text editor with instantaneous graphical feedback as the structure is decoded from the line notation.

Cheminformatics has stood at the forefront of scientific computing by developing representations and methods for storing and processing vast amounts of chemical data through digital information systems. The Chemical Abstracts Service (CAS) has provided indexing and registration for millions of chemical substances since the 1960s.[4] Structure-based line notations, such as Wiswesser line notation (WLN)[5] and the Simplified Molecular Input Line Entry System (SMILES),[6] capture the connectivity of a molecular graph in a compact format for expedited processing and storage of chemical structural data. Tabular representations such as the connection table (e.g., molfile) incorporate additional details about molecular graphs such as the relative coordinates of atoms within two- or three-dimensional structural depictions.[7] Although verbose, connection tables have persisted as a data structure for visualization of molecules as structure diagrams.[8] When considering the chemical structure of polymers, the locations of constitutional repeating units and their connection points within the macromolecule must also be specified.[9] The substructure group (Sgroup) extension of the molfile format enables the specification of these repeating units as well as hierarchical relationships between them.[10]

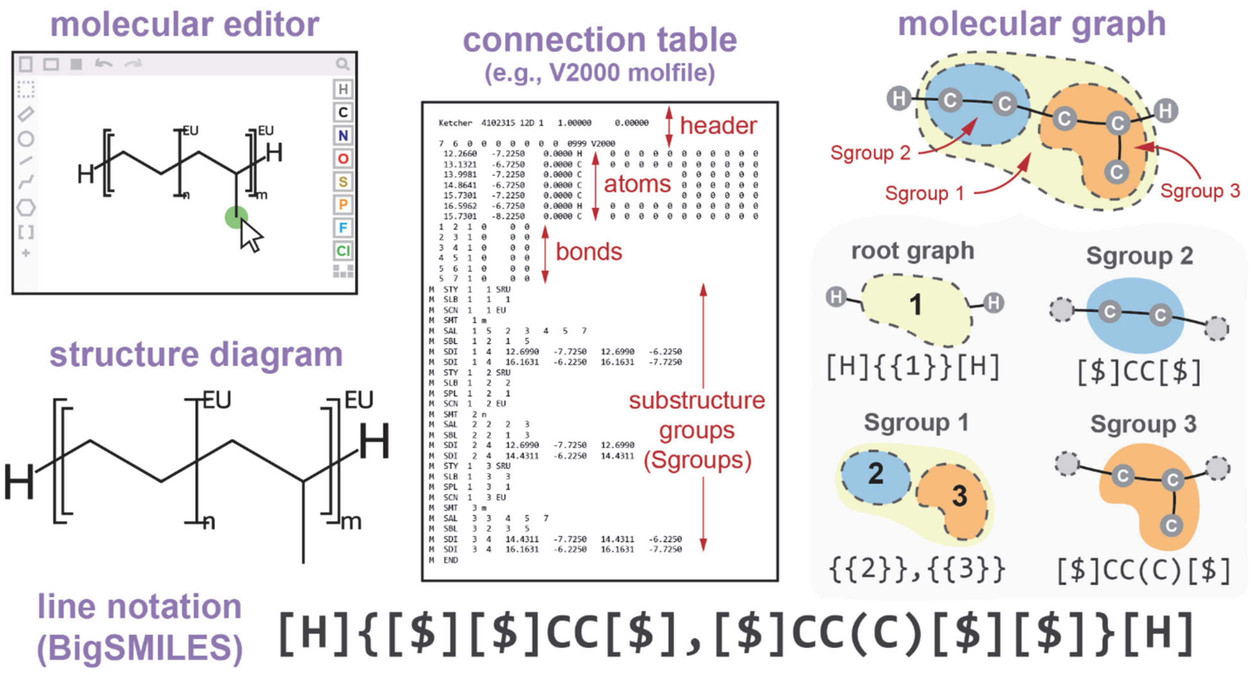
Line notations for polymers have traditionally been limited to lists of small molecule monomers and the use of asterisks to demarcate repeating segments. To overcome the limitations of deterministic chemical representations and express the stochasticity and variety of macromolecular architectures attainable during polymer synthesis, BigSMILES line notation was developed.[11, 12] BigSMILES extends the SMILES language with the notion of stochastic objects and bonding descriptors to represent constitutional repeating units and their available connectivity. The BigSMILES language has been further extended with methods and syntax for canonicalization and non-covalent bonding.[13, 14] While the BigSMILES line notations provide a concise human- and machine-readable representation of polymeric structures for direct entry, the preferred representation and format of chemical information exchange for many polymer chemists remains the chemical structure diagram.

This work presents algorithms and their implementation in JavaScript for the machine translation between BigSMILES line notation and chemical structure diagrams, supporting graphical and text-based representation of polymers. Here, *serialization* refers to the encoding of a connection table—a data format specifying a structure diagram—into BigSMILES line notation, and *deserialization* refers to the inverse process of decoding a BigSMILES string into a representative connection table (i.e., structure diagram). Serialization is achieved by processing a connection table (i.e., V2000 or V3000 molfile), a standard output of molecular editor software, through iterative traversal of the molecular graph and its specified substructures (i.e., Sgroups). Deserialization involves the parsing of BigSMILES line notation into a stochastic graph representation, followed by the identification of a representative graph traversal—a set of Eulerian paths spanning the stochastic graph. Computation of 2-D atomic coordinates is performed using the CoordGen library available within RDKit, followed by simple coordinate transformations to orient Sgroups and brackets. Round-trip translation on a set of 300 manually curated chemical structure diagrams of polymers enabled evaluation of these algorithms, and the limitations of Sgroup-agnostic atomic coordinate generation are discussed. The implementation of these algorithms in JavaScript facilitates applications such as the interactive binding between BigSMILES strings and chemical structure diagrams as well as the text-based specification of polymers with instantaneous graphical feedback. By automating the translation between line notation and the graphical representation of polymer chemical structure, this work aims to serve the broader polymer chemistry community through the improvement of interfaces for polymer informatics systems.

METHODS

Serialization from Chemical Structure Diagram to BigSMILES Line Notation

In this context, serialization is defined as the translation of a molecular graph from a graphical representation (i.e., chemical structure diagram) into a string of ASCII characters representing the same molecular graph in BigSMILES line notation. The input data structure is a connection table (e.g., V2000 or V3000 molfile) of the molecular graph—an output from molecular editor software—containing: a) a set of atoms and their properties, b) a set of bonds and their properties, and c) an optional set of Sgroups each containing information about a subgraph within the molecular graph such as a repeating unit (**Figure 2**). In addition to a list of atoms in the subgraph, each Sgroup also contains the type of connectivity (e.g., head-to-tail, either/unknown), the location of brackets, the Sgroup label (e.g., "n") typically displayed as a subscript, any hierarchical relationships with other Sgroups, and a list of the bond indices that connect the subgraph to the rest of the molecular graph (i.e., crossing bonds). The molfile connection table format, originally developed by Molecular Design Limited, Inc. and whose specification is currently maintained by Biovia,[15] is commonly found in V2000 format. However, V3000 format (available since 1995) is preferred for its improved machine readability and its ability to specify more than two crossing bonds in a given Sgroup, the latter feature a necessity for representing network polymers. In the following paragraphs, the algorithm for translating a connection table into BigSMILES notation (i.e., serialization) is described in further detail.



```
  Ketcher  4102315 12D 1   1.00000     0.00000

  7  6  0  0  0  0  0  0  0  0999 V2000
   12.2660   -7.2250    0.0000 H    0  0  0  0  0  0  0  0  0  0  0
   13.1321   -6.7250    0.0000 C    0  0  0  0  0  0  0  0  0  0  0
   13.9981   -7.2250    0.0000 C                   0  0  0  0  0  0
   14.8641   -6.7250    0.0000 C                   0  0  0  0  0  0
   15.7301   -7.2250    0.0000 C                   0  0  0  0  0  0
   16.5962   -6.7250    0.0000 H    0  0  0  0  0  0  0  0  0  0  0
   15.7301   -8.2250    0.0000 C    0  0  0  0  0  0  0  0  0  0  0
  1  2  1  0     0  0
  2  3  1  0     0  0
  3  4  1  0     0  0
  4  5  1  0     0  0
  5  6  1  0     0  0
  5  7  1  0     0  0
M  STY  1   1 SRU
M  SLB  1   1   1
M  SCN  1   1 EU
M  SMT   1 m
M  SAL   1  5   2   3   4   5   7
M  SBL   1  2   1   5
M  SDI   1  4   12.6990   -7.7250   12.6990   -6.2250
M  SDI   1  4   16.1631   -6.2250   16.1631   -7.7250
M  STY  1   2 SRU
M  SLB  1   2   2
M  SPL  1   2   1
M  SCN  1   2 EU
M  SMT   2 n
M  SAL   2  2   2   3
M  SBL   2  2   1   3
M  SDI   2  4   12.6990   -7.7250   12.6990
M  SDI   2  4   14.4311   -6.2250   14.4311
M  STY  1   3 SRU
M  SLB  1   3   3
M  SPL  1   3   1
M  SCN  1   3 EU
M  SMT   3 m
M  SAL   3  3   4   5   7
M  SBL   3  2   3   5
M  SDI   3  4   14.4311   -7.7250   14.4311   -6.2250
M  SDI   3  4   16.1631   -6.2250   16.1631   -7.7250
M  END
```

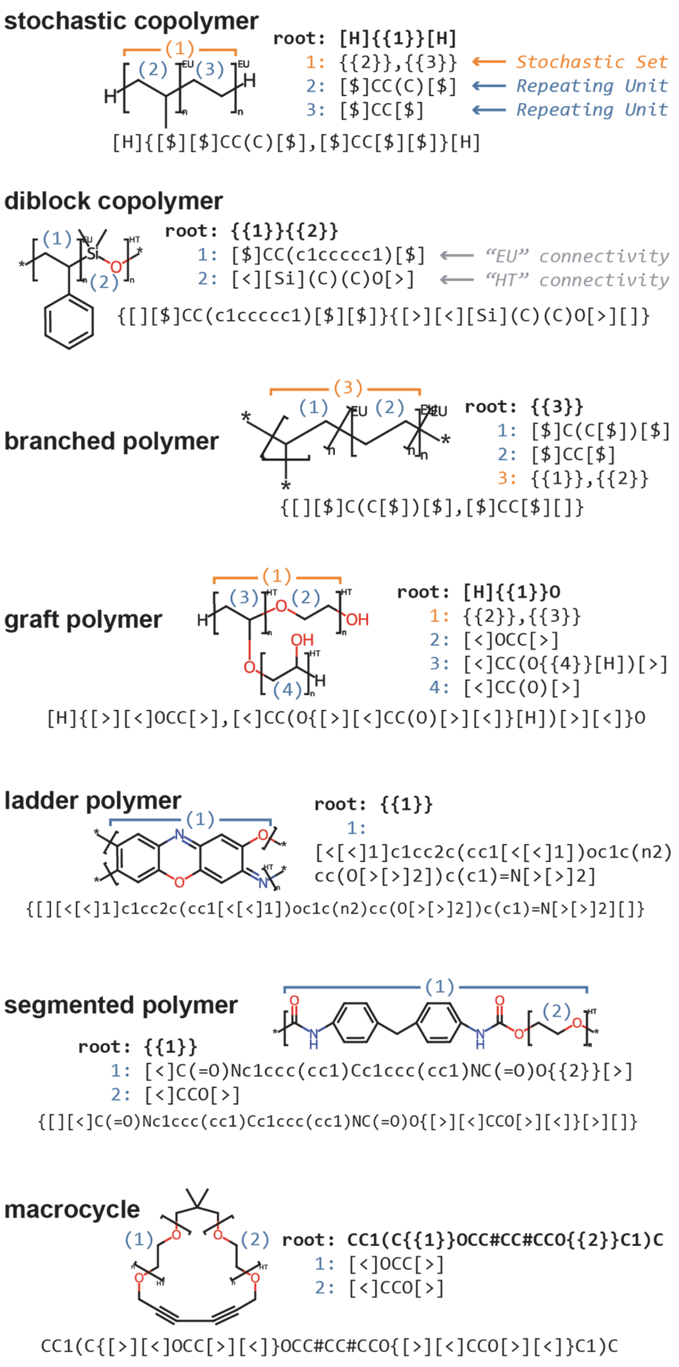
Figure 2. Overview of key concepts and representations for the serialization of a connection table—a text-based representation of a structure diagram generated by a molecular editor—into BigSMILES line notation. At a high level, the molecular graph represented in the connection table is decomposed into subgraphs representing the root graph and each substructure group (Sgroup), and substrings for each subgraph are serialized and then combined to form the final BigSMILES string.

The serialization process uses the information contained within a connection table to generate a string for the root level (i.e., end-group atoms as well as placeholder tokens for root-level Sgroups) as well as a string for each Sgroup in the connection table, followed by the iterative substitution of Sgroup tokens with their line notation string replacements (**Figure 3**). The same algorithm produces a SMILES string in the event that no Sgroups are provided (i.e., small molecule). Both V2000 and V3000 molfile formats are supported by first parsing the contents of either connection table format into a shared JavaScript Object Notation (JSON) data structure (see Supporting Information, Section SII). Although connection tables do not explicitly include the notion of stochasticity, the hierarchical property of Sgroups allows the specification of stochastic copolymers using an encapsulatory Sgroup around neighboring Sgroups in the structure diagram. If the atoms within a given Sgroup are all contained inside child Sgroups, then the Sgroup is classified as a "Stochastic Set" represented by a comma-delimited list of Sgroup tokens, each containing the integer index of the Sgroup surrounded by a double pair of curly braces. Otherwise, the Sgroup is classified as a "Repeating Unit" and its line notation string is generated by traversal of the molecular subgraph corresponding to the given Sgroup (**Figure 3**, Box A).

```
Parse connection table (V2000/V3000 molblock) contents
    into JSON object: { atoms: [ ] , bonds: [ ] , Sgroups: [ ] }
Generate root line notation string (see Box A) with explicit
end-group atoms and tokens for top-level Sgroups
For each Sgroup in the connection table:
    Classify as Repeating Unit or Stochastic Set
        If Repeating Unit, generate replacement line
        notation string according to Box A
        If Stochastic Set, replacement string is
        comma-delimited list of child Sgroup tokens
Set initial value of BigSMILES as root string
While BigSMILES contains Sgroup tokens:
    Select first Sgroup token for replacement and check its
    parent Sgroup type
        If parent is a Repeating Unit Sgroup (or the root)
            Include stochastic object curly braces and
            terminal descriptors around replacement string
        If parent is a Stochastic Set Sgroup
            Replace Sgroup token with replacement string
Return BigSMILES
```

Box A - Serializing line notation from molecular subgraph

```
Create replacement nodes for child Sgroups and assign
replacement tokens using the format '{{' + index + '}}'
Create adjacency list for undirected graph comprising only
top-level atoms, child Sgroups, and (if Sgroup) external
atoms associated with crossing bonds
    Find a spanning tree using depth-first search
If processing an Sgroup (i.e., not the root string):
    If crossing bond count is >2 and even:
        Perform ladder classification procedure
    Map external atoms to replacement tokens representing
    bonding descriptors ( [$], [<], [>], [$[$]1], [<[<]2], etc.)
Identify "backbone" path between start and end nodes
(external atoms for Sgroups) using breadth-first search
Identify ring bonds (edges not captured by spanning tree)
Add start node to stack
While stack is not empty:
    Set current node by removing last node from stack
    If current node in branches set:
        Add '(' to string, add current node as anchor to
        branch stack, and increment depth counter
    Add bond marker or E/Z marker to string if relevant
    If current node is external atom or child Sgroup:
        Add replacement token to string
    Else format atom marker and add to string
    Add ring bond marker if at a ring bond
    Get next neighbors of current node from spanning tree
    If no neighbors and inside a branch:
        Add ')' to string, get prior node anchor from branch
        stack, and decrement depth counter
    Add neighbors to stack
    If number of neighbors >1:
        Add non-backbone nodes to branches set
Return string with wildcard (*) atoms removed
```



```
stochastic copolymer
root: [H]{{1}}[H]
  1: {{2}},{{3}}  ← Stochastic Set
  2: [$]CC(C)[$]  ← Repeating Unit
  3: [$]CC[$]     ← Repeating Unit
[H]{[$][$]CC(C)[$],[$]CC[$][$]}[H]

diblock copolymer
root: {{1}}{{2}}
  1: [$]CC(c1ccccc1)[$]  ← "EU" connectivity
  2: [<][Si](C)(C)O[>]   ← "HT" connectivity
{[][$]CC(c1ccccc1)[$][$]}{[>][<][Si](C)(C)O[>][]}

branched polymer
root: {{3}}
  1: [$]C(C[$])[$]
  2: [$]CC[$]
  3: {{1}},{{2}}
{[][$]C(C[$])[$],[$]CC[$][]}

graft polymer
root: [H]{{1}}O
  1: {{2}},{{3}}
  2: [<]OCC[>]
  3: [<]CC(O{{4}}[H])[>]
  4: [<]CC(O)[>]
[H]{[>][<]OCC[>],[<]CC(O{[>][<]CC(O)[>][<]}[H])[>][<]}O

ladder polymer
root: {{1}}
  1:
[<[<]1]c1cc2c(cc1[<[<]1])oc1c(n2)
cc(O[>[>]2])c(c1)=N[>[>]2]
{[][<[<]1]c1cc2c(cc1[<[<]1])oc1c(n2)cc(O[>[>]2])c(c1)=N[>[>]2][]}

segmented polymer
root: {{1}}
  1: [<]C(=O)Nc1ccc(cc1)Cc1ccc(cc1)NC(=O)O{{2}}[>]
  2: [<]CCO[>]
{[][<]C(=O)Nc1ccc(cc1)Cc1ccc(cc1)NC(=O)O{[>][<]CCO[>][<]}[>][]}

macrocycle
root: CC1(C{{1}}OCC#CC#CCO{{2}}C1)C
  1: [<]OCC[>]
  2: [<]CCO[>]
CC1(C{[>][<]OCC[>][<]}OCC#CC#CCO{[>][<]CCO[>][<]}C1)C
```

Figure 3. Detailed on the left, the serialization of BigSMILES line notation from a connection table involves the generation of a root string followed by a substring for each substructure group (Sgroup), which are iteratively assembled into the final BigSMILES string. To generate line notation for each molecular subgraph (Box A), a spanning tree comprising top-level atoms, child Sgroups, and external atoms is

traversed. On the right, examples of various polymer topologies show the resulting root string, Sgroup substrings, and final BigSMILES generated through this method. Sgroups specifying Repeating Units have blue labels, and those specifying a Stochastic Set (i.e., a collection of Repeating Units) have orange labels.

The sequence of characters in the line notation represents the traversal of a molecular graph, where branches are encapsulated within pairs of parentheses and rings are encoded through integer indices. Serialization of line notation for the root BigSMILES string and Repeating Unit Sgroups involves such a traversal, where a reference mapping to child Sgroups and external atoms associated with Sgroup crossing bonds are handled with tokens for Sgroups or bonding descriptors, respectively. To serialize a Repeating Unit such that its bonding descriptors appear at the beginning and end of the corresponding line notation, the shortest path between these bonding descriptors (e.g., set of backbone atoms) is identified using breadth-first search, and branches are always initialized away from this path. Otherwise, the traversal of the molecular subgraph largely follows the SMILES serialization procedure as implemented in the Python library pysmiles.[16]

Prior to graph traversal, nodes for child Sgroups and bonding descriptors are defined and added to a reference map to their respective replacement strings. These nodes and their respective bonds are captured using an adjacency list—a map from each node to a list of its neighboring nodes—to represent the subgraph. The connection points to child Sgroup subgraphs are defined inside the crossing bonds ("xbonds") property of the connection table associated with each child Sgroup. Crossing bonds correspond to the bonds intersected by brackets in the structure diagram and are used for determining where a repeating units subgraph connects to the macromolecular graph. An adjacency list for the subgraph is generated, including external atoms associated with crossing bonds and auxiliary nodes for any child Sgroups. From this adjacency list, a spanning tree is identified using depth-first search, and any broken bonds (i.e., ring bonds) are stored in a reference map. If the Sgroup has an even number of crossing bonds greater than two, a classification procedure is used to determine whether the substructure represents a ladder repeating unit or a branching point of a network polymer (see Supporting Information, Section SIII). If a ladder repeating unit is determined, each bonding descriptor is assigned an inner bonding descriptor and group identifier. In all other cases, the type of bonding descriptor is determined from the connection property of the Sgroup, where "either/unknown" (EU) connectivity is represented by the [$] bonding descriptor token and "head-to-tail" (HT) connectivity is represented by conjugate [<] and [>] bonding descriptor tokens.

To ensure proper directionality of traversal for Sgroups, the starting node index (i.e., bonding descriptor) is selected based on its order of appearance in the root string or other Sgroups. Traversal begins by adding the index of the starting node (e.g., the node representing the first bonding descriptor) to a stack data structure, followed by a while loop that terminates once the stack is empty. Each iteration of the while loop begins with the removal of the last item in the stack, which can represent an atom, a bonding descriptor, or a child Sgroup. If this node forms the base of a branch, an opening parenthesis is concatenated to the line notation string. Characters representing higher-order covalent bonds or E/Z geometric isomer information are also added if applicable (for more information about the handling of E/Z information, see Supporting Information Section SVI). If the node index exists within the replacement map, then the corresponding string representing a bonding descriptor (e.g., "[$]") or child Sgroup (e.g., "{{2}}") is concatenated to the line notation string. Otherwise, the node representing an atom is formatted according to SMILES syntax and concatenated to the line notation string. If the node is part of a bond excluded from the spanning tree (i.e., ring bond), then the integer index of the corresponding ring is concatenated to the line notation.

Once the current node has been represented in the line notation string, the traversal continues by identifying the neighboring node(s) in the spanning tree and adding their node indices to the stack. If no neighboring nodes exist in the spanning tree, a closing parenthesis is concatenated to the line notation string if the traversal has reached the end of a branch. If more than one neighboring node exists, then the graph contains a branch point and the node indices representing the beginning of new branches must be stored. To ensure that branches are initialized away from the backbone, the atom contained in the backbone is placed at the front of the list concatenated to the stack, ensuring that the other node(s) are retrieved from the end of the stack first. The indices of the non-backbone neighboring atoms are added to a reference set used for determining whether an open parenthesis indicating the start of a branch should be added to the line notation when the node index is retrieved from the stack. With this approach, the last node visited for a Repeating Unit Sgroup corresponds to a bonding descriptor. The serialized line notation corresponding to the root string or Repeating Unit Sgroup is returned after the removal of any wildcard tokens (i.e., "*" atoms), as these wildcards represent unspecified end groups omitted from BigSMILES line notation.

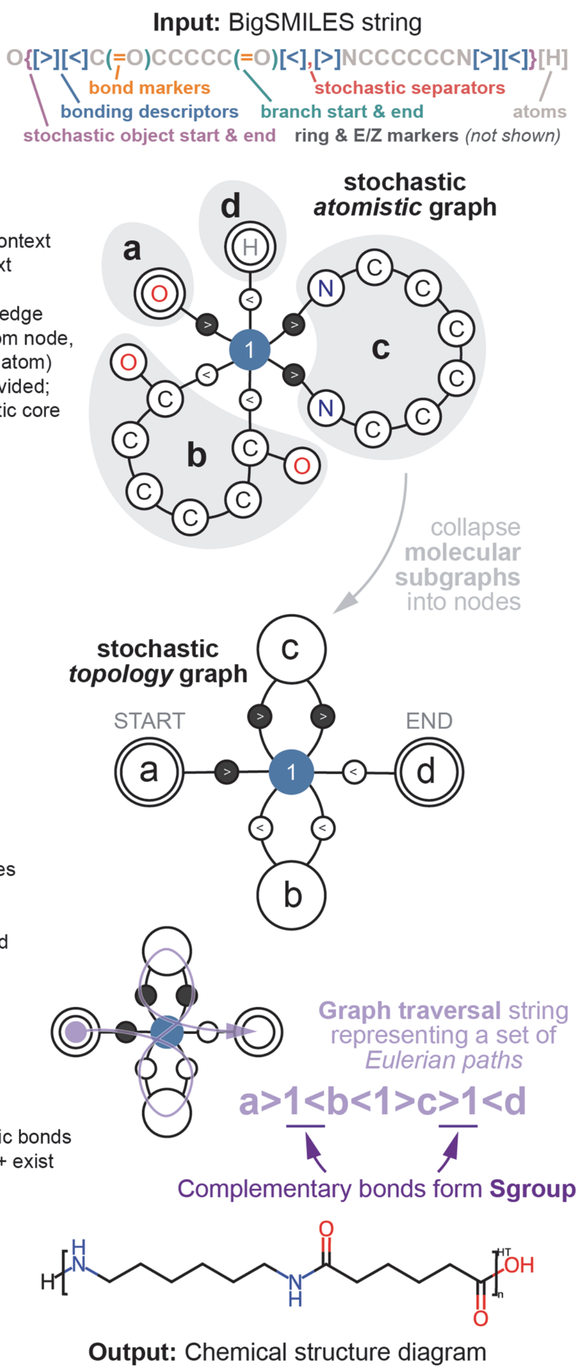
With the line notation strings for the root string and all Sgroups serialized, the BigSMILES line notation is initialized with the root string. A while loop is then executed until the BigSMILES string no longer contains Sgroup tokens. Each iteration of the loop seeks the first available Sgroup token (e.g., "{{1}}") and substitutes this token with the line notation string of the corresponding Sgroup. If the parent of this Sgroup is a Stochastic Set, then a simple string replacement is performed. Otherwise, a pair of curly braces and terminal bonding descriptors are added around the replacement string to signify the beginning and end of the stochastic object. When no Sgroup tokens remain in the line notation string, the conversion to BigSMILES has completed. If no explicit end groups exist at the beginning or end of the BigSMILES, the respective terminal bonding descriptors are replaced with the empty terminal bonding descriptor symbol ("[]") using regular expressions. At this point, the resulting BigSMILES line notation string is returned.

Deserialization from BigSMILES Line Notation to Chemical Structure Diagram

As the inverse of the serialization process, deserialization begins with a BigSMILES line notation string as input and concludes with the assembly of a connection table representing the macromolecular graph as output. At a high-level, this process entails the parsing of the BigSMILES string into a graph representing the stochastic macromolecule, upon which a graph traversal procedure identifies a representative path—or set of paths, in the case of molecules containing branch points—that defines the sequence of molecular fragments to assemble into a connection table for display as a structure diagram (**Figure 4**). Here, the open-source cheminformatics toolkit RDKit was used for converting the connection table into a web-based graphic for the structure diagram,[17] and the CoordGen library within RDKit was used to generate the base set of 2-D coordinates for the atoms in that connection table.[18] Although a structure diagram is more visually expressive than line notation, BigSMILES line notation is more semantically expressive for describing stochastic polymer structure. Therefore, the conversion between a BigSMILES string and a structure diagram is not always guaranteed to capture the stochastic polymer structure without loss of contextual information.

Unlike the well-defined molecular graphs specified by SMILES, polymers represented by BigSMILES contain the ensemble of all macromolecules allowed under a set of molecular fragments and constraints around their connectivity. Identification of a representative structure diagram for the polymer involves the selection of a well-defined member of this ensemble, recognizing that brackets will be used to denote the possibility of repetition. For example, in a stochastic copolymer, the ordering of repeating units is arbitrary but must be specified for purposes of displaying the structure diagram. To capture whether a repeating unit is reversible, superscripts outside the brackets (e.g., “HT” for head-to-tail, “EU” for either/unknown) can be used, but a polymer defined solely by a structure diagram is often subject to ambiguity. Despite the fundamental limitations of structure diagrams, however, the following deserialization procedure is capable of capturing the variety of polymer architectures (e.g., block and stochastic copolymers, graft polymers, segmented polymers, network and hyperbranched polymers, star polymers, ring polymers, ladder polymers, etc.) available through the more expressive BigSMILES line notation.

```
1) Tokenize and parse BigSMILES string
    Replace fragments in BigSMILES with their string substitutes
    For each character in BigSMILES string:
        Identify token character(s), classify token, and push to tokens array
        If token length >1, skip forward to next unvisited character
    For each token in tokens array:
        Case BRANCH_START: Add anchor to branches stack
        Case BRANCH_END: Get prior anchor from branches stack
        Case BOND: Save bond order for next atom; check E/Z
        Case STO_OBJ_START: Check for macrocycles; intialize stochastic context
        Case STO_OBJ_END: Check for macrocycles; close stochastic context
        Case STO_UNIT_SEPARATOR: Remove anchors
        Case RING_NUMBER: Add ring to map, or close open ring by adding edge
        Case ATOM: Parse atom info (element, charge, parity, etc.); create atom node,
        add edge between atom and relevant anchor node (stochastic core or atom)
        Case STO_BOND: Parse descriptor info; assign integer ID if none provided;
        create stochastic core node if none exists; add edge between stochastic core
        and relevant anchor node (stochastic core or atom)
    Return stochastic atomistic graph { nodes: [ ], edges: [ ] }
2) Generate array of molecular subgraphs
    Initialize set of visited atom nodes
    For each atom node:
        If atom already visited, skip this iteration
        Create adjacency list, but do not add atoms to adjacency lists of
        stochastic cores to prevent extension beyond subgraph
        Breadth-first search to identify all atoms in subgraph
        Extract subgraph { nodes: [ ], edges: [ ] } and push to array
    Return array of subgraph objects
3) Join subgraphs into stochastic topology graph
    For each molecular subgraph in array:
        Create node for molecular fragment with lower-case letter ID
        Create new stochastic core node if none exists
        Create edge between stochastic cores and molecular fragment
    Return stochastic topology graph { nodes: [ ], edges: [ ] }
4) Identify set of Eulerian paths that traverse topology graph
    Create adjacency list for topology graph including bonding descriptor types
    Select start node and add to queue
    While queue is not empty:
        Extract current node from queue; if molecular fragment, mark as visited
        Add bonding descriptor and node ID to path string
        Prioritize next valid neighbor node from adjacency list
            Create branch if at a molecular fragment with 2+ neighbors
        Add to queue any neighbors not currently in queue
    Return path string
5) Parse path string and assemble subgraphs into connection table
    Add wildcard (*) atoms wherever path terminates at stochastic core
    Create "Repeating Unit" Sgroup for each complementary pair of stochastic bonds
    Create "Stochastic Set" Sgroup for first and last of a stochastic bond, if 3+ exist
    Return connection table { atoms: [ ], bonds: [ ], Sgroups: [ ] }
6) Retrieve atom coordinates and compute final layout
    Create placeholder molblock, where any atoms with specified E/Z
    configuration are assigned dummy x- and y-coordinates
    Send placeholder molblock to RDKit get_new_coords() function
    Perform global rotation to align crossing bonds to horizontal axis
    Compute bracket positions and rotations
    Return molblock for depiction
```



Input: BigSMILES string
O{[>][<]C(=O)CCCCC(=O)[<],[>]NCCCCCCN[>][<]}[H]
bond markers
stochastic separators
bonding descriptors
branch start & end
atoms
stochastic object start & end
ring & E/Z markers (not shown)
stochastic atomistic graph
a
b
c
d
collapse molecular subgraphs into nodes
stochastic topology graph
START
END
Graph traversal string representing a set of Eulerian paths
a>1<b<1>c>1<d
Complementary bonds form Sgroup
Output: Chemical structure diagram

Figure 4. To deserialize BigSMILES line notation into a chemical structure diagram, the BigSMILES string is tokenized and parsed into a stochastic graph representing the ensemble of polymers valid under the BigSMILES string. A representative member of the ensemble is selected by finding the set of Eulerian paths that visit each molecular fragment exactly once. Atomic coordinates for the connection table are

computed using RDKit, followed by a global coordinate transformation and bracket placement procedure. A high-level overview of the algorithmic steps is shown on the left, with a corresponding illustrative example of Nylon-6,6 on the right.

The deserialization of a BigSMILES string begins with the decomposition of the string into a list of individual tokens, each represented by a single character or short string of characters (e.g., atom, bond, bonding descriptor, branch start or end marker, stochastic object start or end marker, ring index, etc.). If the BigSMILES string contains fragment notation syntax, these fragments are replaced with their string equivalents prior to tokenization. The tokenization and parsing procedure roughly follows the approach used within the pysmiles library for reading SMILES strings,[16] extending the approach with BigSMILES-specific syntax and a stochastic graph representation as opposed to the deterministic graph for small molecules. A macromolecule is only considered a valid member of the ensemble represented by a BigSMILES string if it can be represented by the traversal of this stochastic graph. The stochastic graph is assembled by iterating across the array of tokens, where each token is evaluated according to its type using a switch/case pattern. At its essence, the stochastic graph contains *nodes* representing atoms or “stochastic cores” (i.e., a set of compatible bonding descriptors) connected by *edges* representing covalent (i.e., atom–atom) or stochastic bonds (i.e., with a stochastic core). The JSON data structure for the stochastic graph is described further in the Section SIX in the Supporting Information. Each stochastic core is assigned an integer identifier if none is provided in the BigSMILES bonding descriptor.

The stochastic *atomistic* graph formed by parsing the BigSMILES string is further decomposed into a stochastic *topology* graph by collapsing all deterministic molecular subgraphs into individual nodes. The identification of each molecular subgraph, or fragment, is performed using breadth-first search on atom nodes from an adjacency list wherein stochastic cores contain no neighboring nodes other than adjacent stochastic cores (e.g., a diblock copolymer). In this implementation, each molecular fragment node is identified by a lower-case alphabetical identifier to distinguish from the integer identifiers of nodes representing stochastic cores.

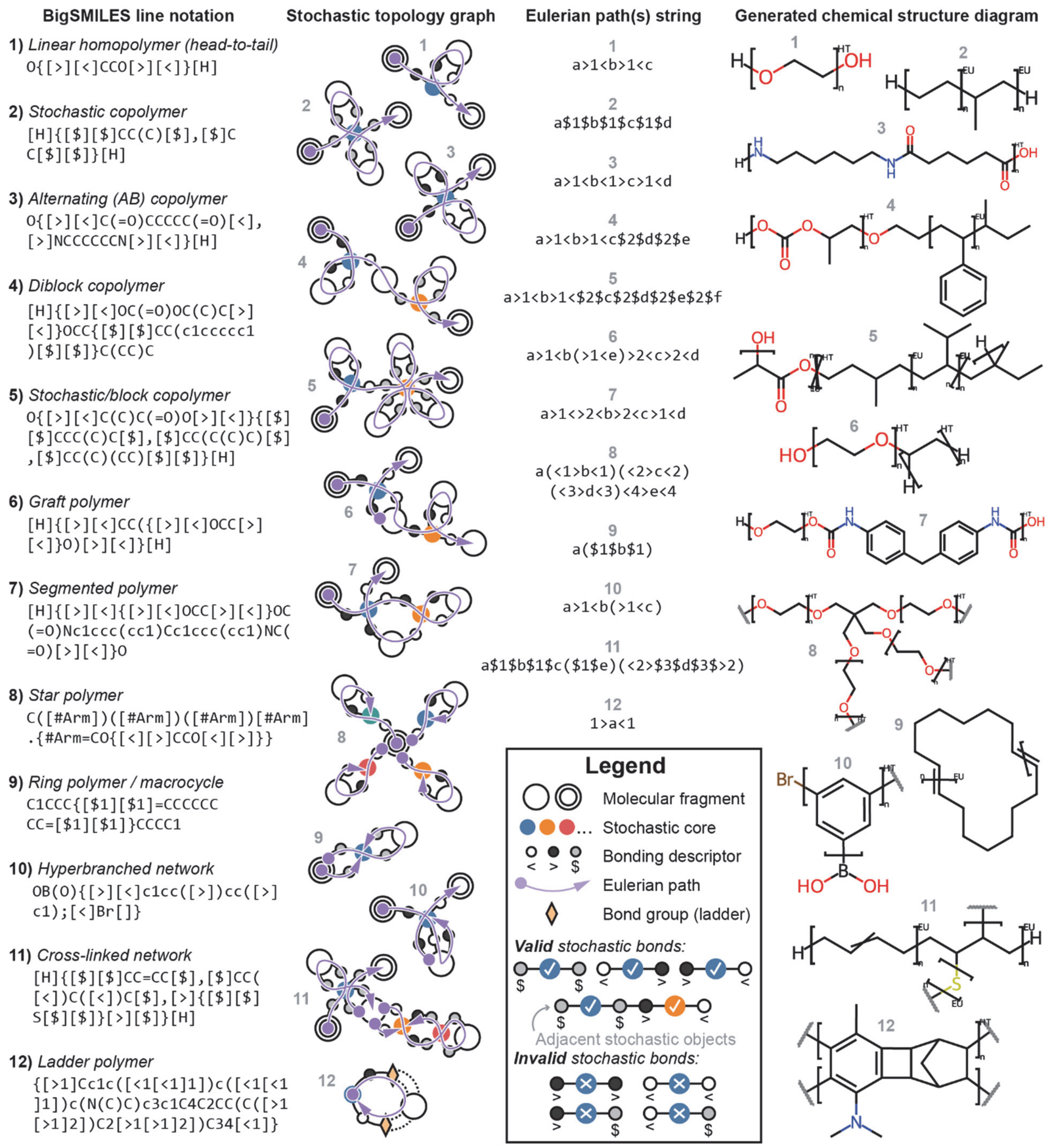
A BigSMILES string represents an ensemble of all possible connectivity patterns for a polymer. In a structure diagram for a polymer, brackets indicate repetition of a given molecular fragment, and a superscript indicates the connectivity (“HT” for head-to-tail, or “EU” for either/unknown). The representative member of the BigSMILES ensemble for depiction is defined as that which: a) traverses the entire stochastic graph, b) follows all bonding constraints set by bonding descriptors, and c) visits each molecular fragment exactly once. Such a traversal can be described as a set of Eulerian paths, where stochastic core nodes may be revisited but each edge must be visited exactly once. Additional Eulerian paths are initialized at branching molecular fragments (e.g., graft polymers, network polymers, star polymers). The traversal of the stochastic graph is represented by a string generated through a procedure similar to the serialization of BigSMILES, where branching follows the same parenthetical syntax of SMILES and stochastic bonds between molecular fragments are represented by bonding descriptor symbols and the integer identifier of their corresponding stochastic core. The starting node for traversal is selected using the following rules:

1. If the BigSMILES string begins with an atom token, the molecular fragment containing that atom is selected, otherwise:
2. A terminal molecular fragment node connected to stochastic core with index 1, otherwise:
3. The lowest-index stochastic core.

During traversal of the graph, neighboring nodes are identified from an adjacency list and filtered to those with compatible bonding descriptors (if at a stochastic core), removing any visited molecular fragments. The next node to visit is chosen from this set based on the following selection criteria, where priority is given to the node whose identifier occurs first when sorted alphabetically:

1. If starting at a stochastic core with an odd number of outgoing edges and conjugate bonding descriptors (e.g., [<], [>], [<]), select a path with the more frequent descriptor (e.g., [<]), otherwise:
2. Visit nodes that loop back to the current node before visiting outgoing nodes, otherwise:
3. Visit a terminal node (i.e., node with only one neighbor), otherwise:
4. Visit the first node in the alphabetically sorted list of candidates.

When a molecular fragment contains more than one outgoing path, a branch is initialized and its corresponding Eulerian path is nested in parentheses. Therefore, the symbols within the graph traversal string include: a) lower-case alphabetical identifiers representing molecular fragments, b) integer identifiers representing stochastic cores, c) characters representing BigSMILES bonding descriptor types (i.e. <, >, or $), and d) parentheses indicating branching in the stochastic topology graph. The versatility of this approach is evidenced by the variety of distinct topologies, each representing a different polymer architecture, that can be parsed and traversed (**Figure 5**).


BigSMILES line notation
Stochastic topology graph
Eulerian path(s) string
Generated chemical structure diagram
Legend
Molecular fragment
Stochastic core
Bonding descriptor
Eulerian path
Bond group (ladder)
Valid stochastic bonds:
Adjacent stochastic objects
Invalid stochastic bonds:

1) *Linear homopolymer (head-to-tail)*
```
O{[>][<]CCO[>][<]}[H]
```
2) *Stochastic copolymer*
```
[H]{[$][$]CC(C)[$],[$]C
C[$][$]}[H]
```
3) *Alternating (AB) copolymer*
```
O{[>][<]C(=O)CCCCC(=O)[<],
[>]NCCCCCCN[>][<]}[H]
```
4) *Diblock copolymer*
```
[H]{[>][<]OC(=O)OC(C)C[>]
[<]}OCC{[$][$]CC(c1ccccc1
)[$][$]}C(CC)C
```
5) *Stochastic/block copolymer*
```
O{[>][<]C(C)C(=O)O[>][<]}{[$]
[$]CCC(C)C[$],[$]CC(C(C)C)[$]
,[$]CC(C)(CC)[$][$]}[H]
```
6) *Graft polymer*
```
[H]{[>][<]CC({[>][<]OCC[>]
[<]}O)[>][<]}[H]
```
7) *Segmented polymer*
```
[H]{[>][<]{[>][<]OCC[>][<]}OC
(=O)Nc1ccc(cc1)Cc1ccc(cc1)NC(
=O)[>][<]}O
```
8) *Star polymer*
```
C([#Arm])([#Arm])([#Arm])[#Arm]
.{#Arm=CO{[<][>]CCO[<][>]}}
```
9) *Ring polymer / macrocycle*
```
C1CCC{[$1][$1]=CCCCCC
CC=[$1][$1]}CCCC1
```
10) *Hyperbranched network*
```
OB(O){[>][<]c1cc([>])cc([>]
c1);[<]Br[]}
```
11) *Cross-linked network*
```
[H]{[$][$]CC=CC[$],[$]CC(
[<])C([<])C[$],[>]{[$][$]
S[$][$]}[>][$]}[H]
```
12) *Ladder polymer*
```
{[>1]Cc1c([<1[<1]1])c([<1[<1
]1])c(N(C)C)c3c1C4C2CC(C([>1
[>1]2])C2[>1[>1]2])C34[<1]}
```

```
1
a>1<b>1<c
2
a$1$b$1$c$1$d
3
a>1<b<1>c>1<d
4
a>1<b>1<c$2$d$2$e
5
a>1<b>1<$2$c$2$d$2$e$2$f
6
a>1<b(>1<e)>2<c>2<d
7
a>1<>2<b>2<c>1<d
8
a(<1>b<1)(<2>c<2)
(<3>d<3)<4>e<4
9
a($1$b$1)
10
a>1<b(>1<c)
11
a$1$b$1$c($1$e)(<2>$3$d$3$>2)
12
1>a<1
```

Figure 5. To cover the broad range of polymer architectures expressible through the BigSMILES language, the stochastic graph traversal algorithm must produce a valid set of Eulerian paths across a variety of topologies. Here, illustrative examples of twelve types of polymers are shown, from BigSMILES string to the resulting structure diagram corresponding to the selected graph traversal. In cases where the generated

structure diagram is suboptimal, namely examples 5 and 6, potential improvements to the layout by including Sgroup information are discussed in the Supporting Information, Section SVII.

Following the identification of a valid traversal of the stochastic topology graph, this set of Eulerian paths is converted into a connection table by joining molecular fragments with bonds at their specified stochastic attachment points. Self-repeating substructures in this macromolecular graph are identified where the traversal revisits a given stochastic core through a complementary pair of bonding descriptors (i.e., [$]–[$], [<]–[>], or [>]–[<]), and each substructure is captured by adding a corresponding Sgroup to the connection table. The Sgroup may contain more than one molecular fragment, such as in the Nylon-6,6 example of an alternating (AB) polymer (**Figure 4**). In the event that a stochastic core is visited several times for different repeating units (e.g., stochastic copolymer), an encapsulatory Sgroup containing all molecular fragments between the first and last occurrence of the stochastic core is created to represent the Stochastic Set.

Structure diagram generation (SDG) involves the calculation of a set of Cartesian coordinates specifying the 2-D layout of atoms in a connection table.[19] Significant prior work has gone into the development of SDG algorithms that produce publication-quality structure diagrams for small molecules.[20-26] While the International Union of Pure and Applied Chemistry (IUPAC) has compiled a list of recommendations for the graphical representations of polymers,[9, 27] the procedural generation of such representations has not been reported. Instead of introducing new SDG algorithm for polymers, this work repurposes the high-quality CoordGen library available within RDKit for generating the base set of coordinates,[17, 18] followed by global coordinate transformation and a bracket placement procedure. For any atoms containing geometric isomer information (e.g., *cis* or *trans*), a set of placeholder coordinates is produced in order to convey this information to the SDG algorithm (see Supporting Information, Section SVI). To orient the structure such that brackets are aligned along an approximately horizontal axis, a least-squares linear regression is computed on the midpoints of Sgroup crossing bonds and the coordinates are rotated according to the inclination angle of the regression line (see Supporting Information, Section SIV). Brackets, which are ignored by the SDG algorithms in RDKit, are placed at crossing bond midpoints and oriented such that bracket pairs are parallel while avoiding overlap with the crossing bonds (see Supporting Information, Section SV). The finalized connection table with atom and bracket coordinates is then serialized into a V3000 molfile and passed to RDKit for conversion into a web-based image (e.g., SVG or HTML5 Canvas) of the structure diagram, concluding the BigSMILES deserialization process.

Special Cases of BigSMILES Deserialization

As molecular fragments are assembled into a connection table according to the graph traversal string, additional considerations must be made regarding the specification of bonds at stochastic attachment points (**Figure 6**). Before the graph traversal string is parsed, asterisk characters are placed, through regular expressions, wherever a path begins or ends at a stochastic core, as these configurations represent unspecified end groups. Whenever an end group is unspecified (**Figure 6**a), a wildcard atom (represented by a "*" elemental symbol) is added to the connection table to complete the respective crossing bond for the Sgroup. In RDKit, these wildcard atoms can be configured to render as attachment points, otherwise the default depiction uses the asterisk symbol. In network polymers (**Figure 6**b), the assembly of the connection table leaves at least one bond open, and this open bond must be converted into a crossing bond by adding a wildcard atom to the connection table. In ring polymers (e.g., macrocycles) (**Figure 6**c), macrocycles are captured as a property of an edge of the stochastic graph when the BigSMILES string is parsed, and this information is used to close an available bond instead of adding a new wildcard atom to the connection table. Ladder polymers (**Figure 6**d) follow the same graph traversal form as linear polymers, except the open bonds corresponding to bond groups must be handled concurrently. In the case of adjacent ladder units, the compatibility of inner descriptors is considered when adding bonds to the connection table. Finally, due to the greater number of attachment points, additional wildcard atoms are added to form the extra crossing bonds associated with the Sgroup. Ladder polymers with explicit end groups are not supported.

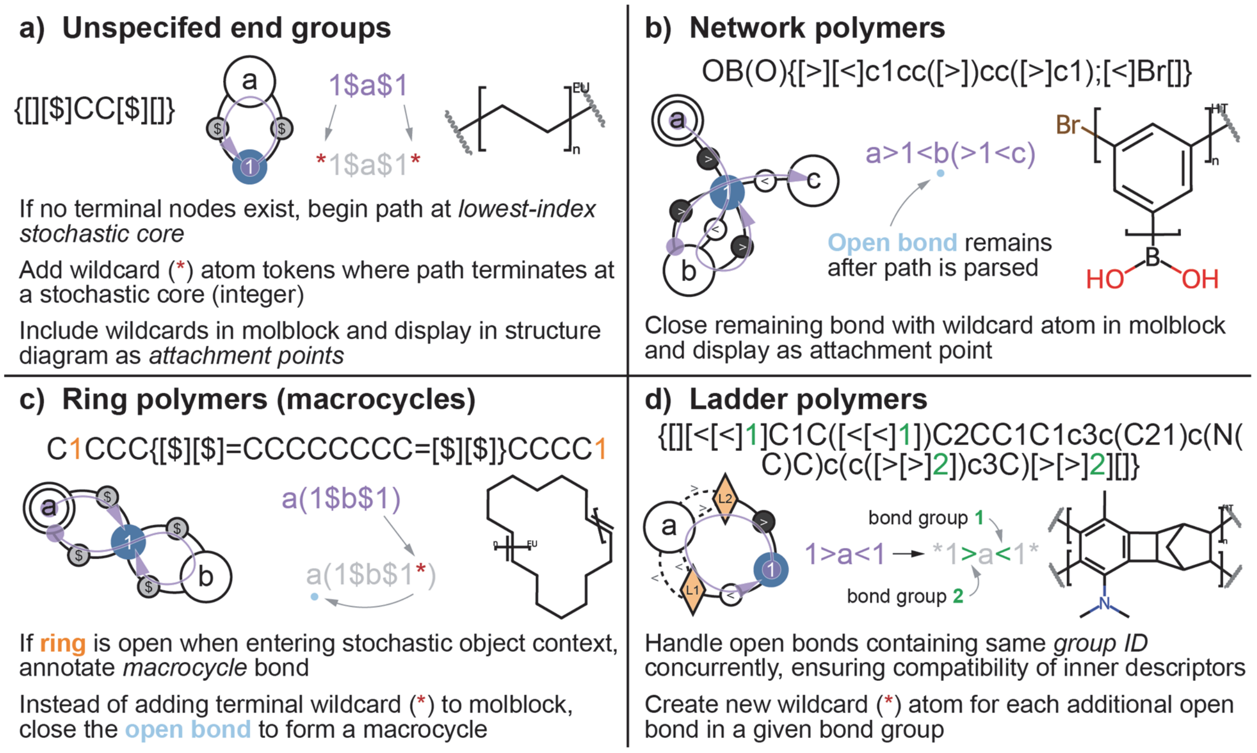

a) Unspecifed end groups
{[][$]CC[$][]}
1a1
1a1
If no terminal nodes exist, begin path at lowest-index stochastic core
Add wildcard (*) atom tokens where path terminates at a stochastic core (integer)
Include wildcards in molblock and display in structure diagram as attachment points
b) Network polymers
OB(O){[>][<]c1cc([>])cc([>]c1);[<]Br[]}
a>1<b(>1<c)
Open bond remains after path is parsed
Close remaining bond with wildcard atom in molblock and display as attachment point
c) Ring polymers (macrocycles)
C1CCC{[$][$]=CCCCCCCC=[$][$]}CCCC1
a(1b1)
a(1b1*)
If ring is open when entering stochastic object context, annotate macrocycle bond
Instead of adding terminal wildcard (*) to molblock, close the open bond to form a macrocycle
d) Ladder polymers
{[][<[<]1]C1C([<[<]1])C2CC1C1c3c(C21)c(N(C)C)c(c([>[>]2])c3C)[>[>]2][]}
bond group 1
1>a<1
1>a<1
bond group 2
Handle open bonds containing same group ID concurrently, ensuring compatibility of inner descriptors
Create new wildcard (*) atom for each additional open bond in a given bond group

Figure 6. Accommodating the broad range of polymer topologies expressible through BigSMILES involves proper handling of open bonds as the connection table is assembled: a) Unspecified end groups are replaced with wildcard atoms displayed here as attachment points; b) network polymers with open bonds remaining once the path has been parsed are closed with wildcard atoms displayed as attachment points; c) ring polymers (i.e., macrocycles) are closed by forming a bond with the subgraph containing the macrocycle; and d) ladder polymers require the concurrent handling of bond groups, ensuring compatibility of the specified inner bonding descriptors.

RESULTS AND DISCUSSION

Evaluation of Machine Translation Procedures

The serialization and deserialization procedures were evaluated through round-trip translation (RTT) on a set of 300 connection tables of polymers (i.e., V2000 and V3000 molblocks) curated using the Ketcher web-based molecular editor from EPAM Systems.[28] RTT demonstrates the preservation (or loss) of information during interconversion by performing a forward translation followed by a reverse translation. In this procedure, each curated connection table was serialized into a BigSMILES string, followed by the deserialization of the generated BigSMILES string into a connection table. The input structure diagrams, generated BigSMILES strings, and generated structure diagrams are available in Section SXII of the Supporting Information and as an HTML table in a supporting computational notebook on Observable.[29] The re-assembly of the molecular graph itself (without position information) performs reliably in over 99% of cases, failing in one case to re-create a hyperbranched network polymer from the generated BigSMILES string. In the failed case, a valid BigSMILES string was generated but the graph traversal algorithm during deserialization failed to identify a set of Eulerian paths including all explicitly defined end groups. Overall, the machine translation was demonstrably robust in producing semantically valid representations during serialization and deserialization.

BigSMILES line notation does not capture the spatial arrangement of atoms and bonds, so the translation from a connection table (i.e., V2000 or V3000 molfile) into BigSMILES removes all atomic coordinate information from the original structure diagram. The extent to which the generated structure diagram resembles the input structure diagram depends on the layout algorithms employed for SDG. Once the deserialization procedure decodes a BigSMILES string into a representative molecular graph, the layout of the molecular graph is computed using the CoordGen library in RDKit,[17, 18] followed by a global coordinate transformation to orient Sgroups horizontally and the placement of brackets (see Supporting Information Sections SIV and SV). While this approach takes advantage of the significant prior work in structure diagram generation for small molecules,[19-26] structure diagrams for polymers have an additional set of drawing conventions related to the display of repeating units and their respective backbones. Generated structure diagrams were manually inspected for several classes of layout errors (e.g., non-extended backbone, suboptimal layout of hierarchical repeating units, hyperextended side chains). If the generated diagram contained no such errors, it was deemed a "reasonable depiction" even if it did not exactly match the input diagram—for example, the same structure inverted or rotated. In the test set of 300 polymer connection tables, 113 (37.7%) produced a reasonable depiction with the readily available, Sgroup-agnostic layout algorithm in RDKit (**Table I**). The most common form of suboptimal layout (179, or 59.7%) featured backbone atoms not placed in an extended orientation along a horizontal axis, where the incorporation of Sgroup information becomes necessary in order to produce a depiction that follows the drawing conventions for polymers. The limitations of Sgroup-agnostic layout generation and opportunities for Sgroup-aware counterparts are further discussed, along with example structure diagram outputs, in Section SVII of the Supporting Information.

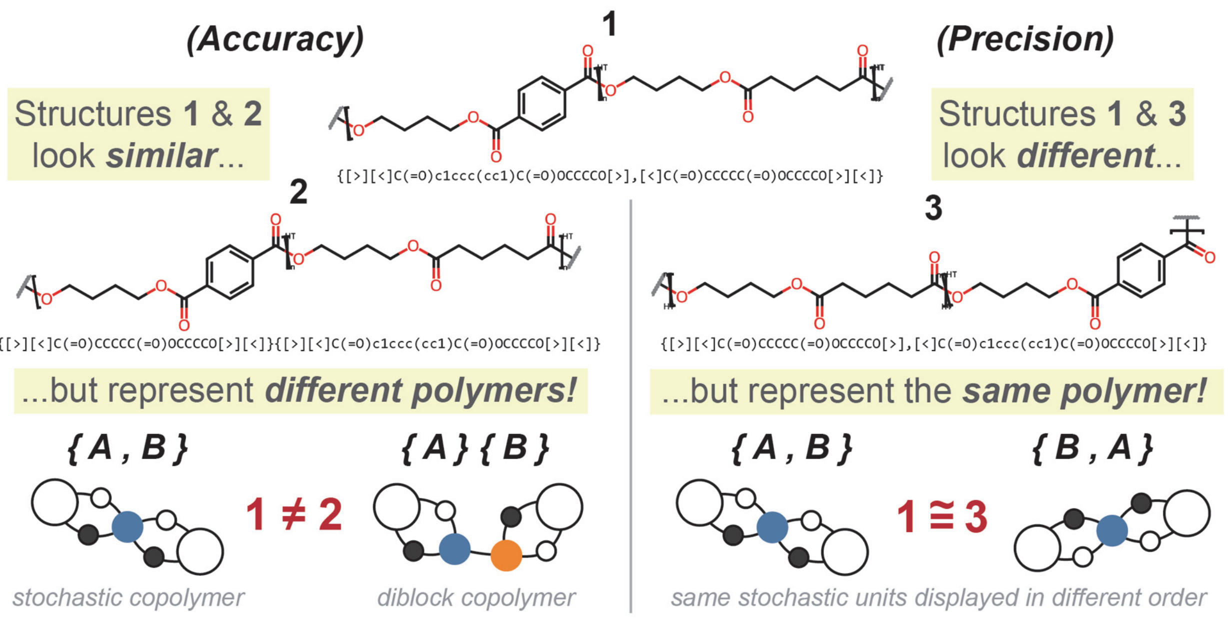
Table I. Summary of the round-trip translation from a structure diagram (manually produced in a molecular editor) into a generated BigSMILES string (i.e., serialization) and back into a structure diagram (i.e., deserialization). Note that a given structure diagram output may have more than one classification, and polymers containing large side chains or hierarchical repeating are infrequent in the test set.

Classification	**Count of occurrences**	**Frequency**
Molecular graph semantically preserved (ignoring visual layout)	**299**	**99.7%**
Visual layout of structure diagram preserved (or reasonable variant)	**113**	**37.7%**
Backbone not extended	179	59.7%
Suboptimal placement of hierarchical repeating units	3	1.0%
Hyperextended side chains	10	3.3%
Unable to generate depiction	1	0.3%

In some cases, the translation from BigSMILES line notation to a structure diagram results in information loss about the original stochastic ensemble. Sgroups capture constitutional repeating units, precluding the distinct representation of structural units that do not self-repeat (e.g., units A and B in an alternating AB copolymer). An example of such a polymer is Nylon-6,6, whose representative BigSMILES and structure diagram were shown in **Figure 4**. The serialization of the structure diagram back into BigSMILES will include a single repeating unit inside the stochastic object. This fundamental limitation, relevant to polymers formed by condensation, is discussed further in Section SVIII of the Supporting Information. Another example of information loss between BigSMILES and structure diagrams involves the specification of end groups within the stochastic object of a BigSMILES string instead of as explicit end groups outside the stochastic object. This feature of BigSMILES enables the representation of polymer ensembles whose members may contain different end groups, such as polymers formed through reversible addition–fragmentation chain transfer (RAFT). Such an ensemble is not representable through a single structure diagram, in which end groups must be specified explicitly. In this case, the polymer ensemble could be represented by multiple structure diagrams, each derived from a unique set of Eulerian paths on the stochastic graph representation. The serialization of multiple structure diagrams into a single BigSMILES, outside the scope of this work, would require additional inference in order to determine the appropriate placement of end groups inside the stochastic object.

Representing Polymer Ensembles as Stochastic Graphs

The validity of a BigSMILES string does not guarantee that the BigSMILES string is equivalent to the polymer intended by the chemist. When designing a new polymer or specifying an existing polymer, graphical feedback offers a mechanism for correcting semantic errors. However, the chemical structure diagram alone may be insufficient for providing semantic validation. For example, the biodegradable copolymer poly(butylene adipate-co-terephthalate) (PBAT), a stochastic copolymer, can easily be mistaken for a diblock copolymer through inspection of the structure diagram alone, and the ordering of its repeating units has a disproportionate effect on the structure diagram (**Figure 7**). The stochastic topology graph derived from BigSMILES line notation, an intermediate representation and byproduct of this work, better captures the essential semantic differences or similarities between a given pair of polymer representations. Similar graph representations for polymer ensembles, akin to state machines, have been presented in earlier work.[13, 30] Compact representations of stochastic polymers, rendered using Graphviz or an equivalent graph layout software,[31] can complement chemical structure diagrams in order to facilitate the semantic validation of structural data. Further examples of these stochastic graphs and their interpretation can be found in Section SXI of the Supporting Information.


(Accuracy)
1
(Precision)
Structures **1** & **2** look ***similar***...
Structures **1** & **3** look ***different***...
2
3
...but represent ***different polymers!***
...but represent the ***same polymer!***
{ A , B }
{ A } { B }
1 ≠ 2
{ A , B }
{ B , A }
1 ≅ 3
stochastic copolymer
diblock copolymer
same stochastic units displayed in different order

```
{[>][<]C(=O)c1ccc(cc1)C(=O)OCCCCO[>],[<]C(=O)CCCCC(=O)OCCCCO[>][<]}
```

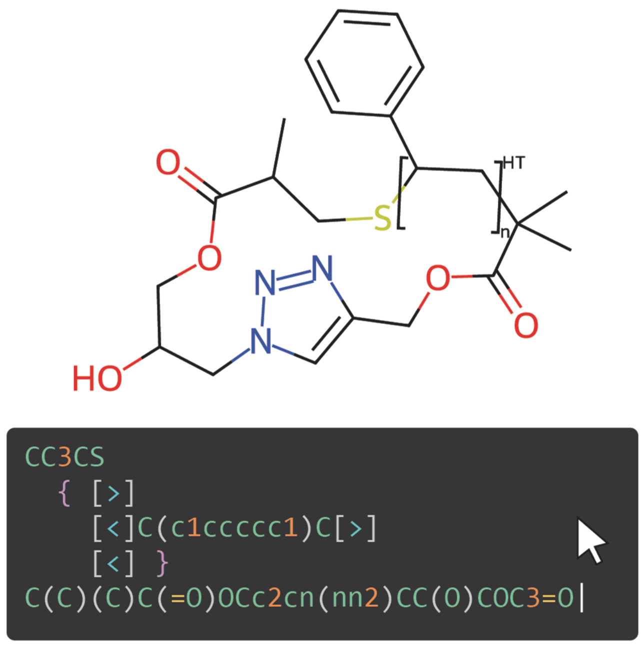
```
{[>][<]C(=O)CCCCC(=O)OCCCCO[>][<]}{[>][<]C(=O)c1ccc(cc1)C(=O)OCCCCO[>][<]}
```

```
{[>][<]C(=O)CCCCC(=O)OCCCCO[>],[<]C(=O)c1ccc(cc1)C(=O)OCCCCO[>][<]}
```

Figure 7. Chemical structure diagrams may fail to distinguish essential differences in polymeric structure and topology (poor accuracy), and the multiplicity of valid structure diagrams for a given stochastic copolymer may lead to superficial differences in the chemical structure diagram that do not impart semantic meaning (poor precision). The stochastic topology graph—derived directly from BigSMILES line notation—provides a visual feedback mechanism for ensuring that a polymer is represented as intended by the chemist. In this example featuring the biodegradable stochastic copolymer poly(butylene adipate-co-terephthalate) (PBAT), structure diagrams 1 and 2 appear nearly identical, but a missing encapsulatory Sgroup in 2 changes the topological interpretation from PBAT to that of a diblock copolymer instead. Furthermore, inverting the positions of stochastic repeating units leads to superficial differences in two structure diagrams that are semantically equivalent.

Graphical User Interfaces (GUIs) Combining BigSMILES and Structure Diagrams

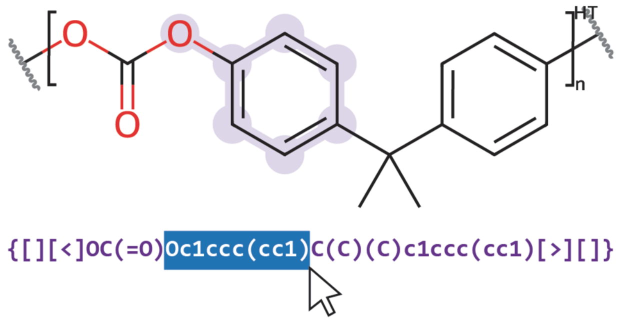
The JavaScript implementation of the serialization and deserialization algorithms enables the facile integration with web-based applications and introduces opportunities for interactive components built around the translation between BigSMILES and structure diagrams. In one such example,[32] a text-based editor for BigSMILES line notation displays the generated structure diagram above a text editor, providing instantaneous visual feedback when specifying a polymer through direct entry of its BigSMILES line notation (**Figure 8**). Such an interface allows one to "program" a polymer structure symbolically, similar to how one might write mathematical equations or format documents in LaTeX.[33] To improve the readability of the line notation inside the text editor, syntax highlighting is implemented using Prism;[34] the automated indentation for stochastic objects aims to further improve readability of the line notation through the addition of whitespace and newline characters, which are removed from the string before processing the BigSMILES.



```
CC3CS
  { [>]
    [<]C(c1ccccc1)C[>]
    [<] }
C(C)(C)C(=O)OCc2cn(nn2)CC(O)COC3=O
```

Figure 8. Deserialization of BigSMILES line notation enables text-based specification of polymer chemical structure diagrams through a text editor interface, akin to interfaces for computer programming or Markdown formatting. Here, formatting of the BigSMILES string (e.g., syntax highlighting, auto-indentation) was implemented in order to improve the readability of the line notation.

The BigSMILES string itself can serve as an input element within a graphical user interface. In another interactive demonstration,[35] highlighting a segment of the BigSMILES string results in the instantaneous highlighting of the corresponding atoms and bonds in the generated structure diagram (**Figure 9**). The binding between characters in the BigSMILES line notation and atoms in the structure diagram is accomplished by preserving the indices of tokens in the BigSMILES string throughout the deserialization process and creating a map of atoms to their token indices during the assembly of the connection table. When the BigSMILES string is highlighted, the corresponding atoms within the bounds of the highlighted segment are identified, and their indices are passed as an input to the structure diagram rendering function in RDKit. If the precise pixel coordinates of the atoms in the structure diagram are known, the reverse mapping to tokens in the BigSMILES string is also possible. Such elements can transform a structure diagram from a static graphic into an interactive canvas for polymer informatics or serve as visual feedback in educational resources for learning the BigSMILES language.



```
{[][<]OC(=O)Oc1ccc(cc1)C(C)(C)c1ccc(cc1)[>][]}
```

Figure 9. Capturing the character index of each token in the BigSMILES string and propagating this information throughout the deserialization process—preserving the link between atoms in the BigSMILES and atoms in the structure diagram—enables binding of BigSMILES to chemical structure diagrams, transforming these depictions from static graphical representations into interactive elements. In this

example, highlighting a segment of the BigSMILES for polycarbonate results in the highlighting of corresponding atoms in the generated chemical structure diagram.

As depicted earlier, these translation algorithms can augment the abilities of existing web-based molecular editors (**Figure 1**). For example, one-way binding between connection table and BigSMILES line notation allows a chemist to create or manipulate the structure directly within the molecular editor while the serialization algorithm updates the BigSMILES string instantaneously. A more advanced interface could feature two-way binding between connection table and BigSMILES line notation, allowing the chemist to dynamically switch between molecular editor and BigSMILES text editor as inputs for updating the structure incrementally, where an update to one input is automatically reflected in the other. Given the quality of modern molecular editor interfaces, such approaches can significantly expedite the generation of BigSMILES—even for chemists already familiar with the BigSMILES language—compared to the keystrokes required for direct entry of the line notation.

Toward Data-Driven Polymer Design

The properties of a polymer depend on the polymer's synthesis (e.g. molecular mass distribution) as well as its processing (e.g., temperature, extrusion rate, ambient conditions). Such attributes are not captured in chemical structure diagrams or BigSMILES line notation. As a result, BigSMILES provides a chemically-resolved, structurally-based identifier for larger data models that address the various other dimensions within the vast design space for polymers. Records of polymer synthesis and process history have been demonstrated using schema- and ontology-based approaches,[36, 37] and BigSMILES is currently used as the default chemical identifier within the data model for the Community Resource for Innovation in Polymer Technology (CRIPT).[38] The application of machine learning techniques to polymers hinges not only on representing polymer structure, but also on the ability to amass large quantities of useful data on the synthesis, characterization, processing, and properties of these polymers. Additional considerations such as the sustainability and criticality of these materials may involve capturing data attributes outside the conventional realm of polymer and materials science.[39, 40] The flexibility of a data model such as that used within CRIPT lends itself to comprehensive capture of broadly relevant information for the development of the next generation of sustainable polymers, but the additional degrees of freedom pose a challenge for creating interfaces for data curation. The ability to rapidly translate between line notations and structure diagrams can reduce or eliminate barriers to data curation that currently throttle data-driven polymer science and engineering by providing streamlined structural annotation as well as visual verification.

The translation between hand-drawn or printed structure diagrams and BigSMILES could further alleviate challenges in historical data curation. In this work, the input representation for serialization to BigSMILES is a connection table (i.e., V2000 or V3000 molfile), a direct output of molecular editor software. Translation from unstructured data—hand-drawn or computer-generated images of molecular structure, where the connection table is no longer available—into a connection table could bridge the gap between this work and the large amount of structural data currently available in the literature. One such approach might employ a transformer-based artificial neural network for optical recognition, such as Image2SMILES.[41] Developing this system would require a large training set of polymer chemical structure diagrams with a variety of bracket styles and structural annotations. An extension of the work presented here could assist in the procedural generation of such a training set.

Forward-looking applications of this work include visual validation for the generative design of polymers and population of databases for polymer informatics. As new methods emerge to create valid molecules *in silico* as candidates for computational modeling or experimentation,[42-45] decision-making on the part of humans monitoring this process can be aided with the proper rendering of structure diagrams. Additionally, whereas the rudimentary representation of polymers as SMILES strings with asterisks at attachment points has served as the basis for generative models,[44, 46] this representational scheme is limited to linear homopolymers without end groups. BigSMILES, on the other hand, captures the much broader range of synthetically available polymer architectures (e.g., copolymers, macrocycles, graft polymers, segmented polymers, star polymers, ladder polymers, etc.). The design space for polymer informatics thus expands significantly with the introduction of BigSMILES line notation, now made more accessible through the machine translation software developed in this work.

CONCLUSIONS

Algorithms for the machine translation between chemical structure diagrams and BigSMILES are reported, making line notations and their benefits more readily available to the broader polymer science community. The implementation of these algorithms in JavaScript enables facile integration with web-based tools and resources for polymer informatics. The translation from a structure diagram to BigSMILES (i.e., serialization) and from BigSMILES to structure diagram (i.e., deserialization) support humans and machines in the processing of large quantities of polymer data. Validation on a set of 300 curated polymer structure diagrams demonstrated the robustness of the translation procedures, and opportunities for improving the layout of generated structure diagrams through Sgroup-aware algorithms are discussed. By bridging an important gap between computers and polymer chemists, these tools offer a reimagination of human–computer interfaces for the data-driven design and discovery of polymer materials.

ASSOCIATED CONTENT

Supporting Information. The following files are available free of charge.

Links to interactive demonstrations, JSON data structure for connection table, ladder classification procedure, global coordinate transformation procedure, bracket placement and fine-tuning, methods for parsing and propagating E/Z geometric isomer information, notable examples of Sgroup-agnostic 2-D coordinate generation, known limitations to interconversion, JSONdata structure for stochastic graph, interpretation of stochastic graphs, round-trip translation between structure diagrams and BigSMILES (PDF)

AUTHOR INFORMATION

Corresponding Author

E-mail: bdolsen@mit.edu

Author Contributions

Conceptualization: M.E.D., T-S.L., B.D.O.; Methodology: M.E.D., B.D.O., N.J.R.; Software: M.E.D. , B.D-C., T-S.L.; Data curation: M.E.D., N.J.R., D.J.W.; Writing – Original Draft: M.E.D.; Writing – Review and Editing: B.D.O., D.J.W., B.D-C., M.E.D.; Validation: M.E.D., D.J.W.; Visualization: M.E.D.

Funding Sources

National Science Foundation Innovation and Technology Ecosystems (NSF 2040636) Convergence Accelerator program.

ACKNOWLEDGMENTS

This work was supported by the National Science Foundation Innovation and Technology Ecosystems (NSF 2040636) Convergence Accelerator program and by the MIT-IBM Watson AI Lab. The authors would also like to thank Kenneth Kroenlein, Eric Muckley, Debra Audus, Jiale Shi, Atsuhisa Miyawaki, Weizhong Zou, Ludwig Schneider, Bruno Salomão Leão, Reid Mello, Haley Beech, Katharina Fransen, and Melody Morris for helpful discussions.

ABBREVIATIONS

SMILES, Simplified Molecular Input Line Entry System; Sgroup, substructure group; SRU, structural repeating unit; SDG, structure diagram generation; JSON, JavaScript Object Notation; RTT, round-trip translation; SVG, Scalable Vector Graphics.

REFERENCES

(1) Martin, T. B.; Audus, D. J. Emerging Trends in Machine Learning: A Polymer Perspective. *ACS Polym. Au* **2023**. DOI: 10.1021/acspolymersau.2c00053

(2) Geyer, R.; Jambeck, J. R.; Law, K. L. Production, use, and fate of all plastics ever made. *Sci. Adv.* **2017**, *3* (7), e1700782. DOI: 10.1126/sciadv.1700782

(3) Deagen, M. E.; Walsh, D. J.; Audus, D. J.; Kroenlein, K.; de Pablo, J. J.; Aou, K.; Chard, K.; Jensen, K. F.; Olsen, B. D. Networks and interfaces as catalysts for polymer materials innovation. *Cell Reports Phys. Sci.* **2022**, *3* (11). DOI: 10.1016/j.xcrp.2022.101126

(4) Weisgerber, D. W. Chemical abstracts service chemical registry system: history, scope, and impacts. *J. Am. Soc. Inform. Sci* **1997**, *48* (4), 349-360.

(5) Vollmer, J. J. Wiswesser line notation: an introduction. *J. Chem. Educ.* **1983**, *60* (3), 192.

(6) Weininger, D. SMILES, a chemical language and information system. 1. Introduction to methodology and encoding rules. *J. Chem. Inf. Comp. Sci.* **1988**, *28* (1), 31-36.

(7) Dalby, A.; Nourse, J. G.; Hounshell, W. D.; Gushurst, A. K.; Grier, D. L.; Leland, B. A.; Laufer, J. Description of several chemical structure file formats used by computer programs developed at Molecular Design Limited. *J. Chem. Inf. Comp. Sci.* **1992**, *32* (3), 244-255.

(8) Barnard, J. M. Representation of Molecular Structures-Overview. *Handbook of Chemoinformatics: From Data to Knowledge in 4 Volumes* **2003**, 27-50.

(9) Jones, R. G.; Pure, I. U. o.; Division, A. C. P.; Wilks, E. S. *Compendium of polymer terminology and nomenclature: IUPAC recommendations, 2008*; RSC Pub., 2009.

(10) Gushurst, A. J.; Nourse, J. G.; Hounshell, W. D.; Leland, B. A.; Raich, D. G. The substance module: the representation, storage, and searching of complex structures. *J. Chem. Inf. Model.* **1991**, *31* (4), 447-454. DOI: 10.1021/ci00004a003

(11) Lin, T. S.; Coley, C. W.; Mochigase, H.; Beech, H. K.; Wang, W.; Wang, Z.; Woods, E.; Craig, S. L.; Johnson, J. A.; Kalow, J. A.; Jensen, K.F.; and Olsen, B.D. BigSMILES: A Structurally-Based Line Notation for Describing Macromolecules. *ACS Cent. Sci.* **2019**, *5* (9), 1523-1531. DOI: 10.1021/acscentsci.9b00476

(12) Lin, T. S. *The BigSMILES Line Notation*. 2019. https://olsenlabmit.github.io/BigSMILES/docs/line_notation.html (accessed 17 Jan. 2023).

(13) Lin, T.-S.; Rebello, N. J.; Lee, G.-H.; Morris, M. A.; Olsen, B. D. Canonicalizing BigSMILES for Polymers with Defined Backbones. *ACS Polym. Au* **2022**. DOI: 10.1021/acspolymersau.2c00009

(14) Zou, W.; Martell Monterroza, A.; Yao, Y.; Millik, S. C.; Cencer, M. M.; Rebello, N. J.; Beech, H. K.; Morris, M. A.; Lin, T. S.; Castano, C. S.; Kalow, J.A.; Craig, S.L.; Nelson, A.; Moore, J.S.; and Olsen, B.D. Extending BigSMILES to non-covalent bonds in supramolecular polymer assemblies. *Chem. Sci.* **2022**, *13* (41), 12045-12055. DOI: 10.1039/d2sc02257e

(15) Biovia. *CTFile Formats*; Dassault Systèmes, France, 2020.

(16) Kroon, P. C. *pysmiles: The lightweight and pure-python SMILES reader and writer*. 2018. https://github.com/pckroon/pysmiles (accessed 17 January, 2023).

(17) Landrum, G. *RDKit: Open-Source Cheminformatics Software*. 2006. https://rdkit.org/ (accessed 9 March, 2023).

(18) *CoordgenLibs*. Schrödinger, Inc., 2022. https://github.com/schrodinger/coordgenlibs (accessed 17 January, 2023).

(19) Helson, H. E. Structure diagram generation. *Rev. Comp. Ch.* **1999**, 313-398.

(20) Dittmar, P. G.; Mockus, J.; Couvreur, K. M. An algorithmic computer graphics program for generating chemical structure diagrams. *J. Chem. Inf. Comp. Sci.* **1977**, *17* (3), 186-192.

(21) Clark, A. M.; Labute, P.; Santavy, M. 2D structure depiction. *J. Chem. Inf. Model.* **2006**, *46* (3), 1107-1123. DOI: 10.1021/ci050550m

(22) Clark, A. M. 2D depiction of fragment hierarchies. *J. Chem. Inf. Model.* **2010**, *50* (1), 37-46. DOI: 10.1021/ci900350h

(23) Clark, A. M. Rendering Molecular Sketches for Publication Quality Output. *Mol. Inform.* **2013**, *32* (3), 291-301. DOI: 10.1002/minf.201200171

(24) Fricker, P. C.; Gastreich, M.; Rarey, M. Automated drawing of structural molecular formulas under constraints. *J. Chem. Inf. Comp. Sci.* **2004**, *44* (3), 1065-1078. DOI: 10.1021/ci049958u

(25) Weininger, D. SMILES. 3. DEPICT. Graphical depiction of chemical structures. *J. Chem. Inf. Comp. Sci.* **1990**, *30* (3), 237-243.

(26) Willighagen, E. L.; Mayfield, J. W.; Alvarsson, J.; Berg, A.; Carlsson, L.; Jeliazkova, N.; Kuhn, S.; Pluskal, T.; Rojas-Cherto, M.; Spjuth, O.; Torrance, G.; Evelo, C.T.; Guha, R.; and Steinbeck, C. The Chemistry Development Kit (CDK) v2.0: atom typing, depiction, molecular formulas, and substructure searching. *J. Cheminformatics* **2017**, *9* (1), 33. DOI: 10.1186/s13321-017-0220-4

(27) Bareiss, R.; Kahovec, J.; Kratochvil, P. Graphic representations (chemical formulae) of macromolecules (IUPAC Recommendations 1994). *Pure Appl. Chem.* **1994**, *66* (12), 2469-2482.

(28) *Ketcher*. EPAM Systems, Inc., 2022. https://github.com/epam/ketcher/ (accessed 9 March, 2023).

(29) Deagen, M. E. *Round-trip translation of BigSMILES*. Observable, 2023. https://observablehq.com/@mdeagen/bigsmiles-round-trip-translation (accessed 19 January, 2023).

(30) Aldeghi, M.; Coley, C. W. A graph representation of molecular ensembles for polymer property prediction. *Chem. Sci.* **2022**, *13* (35), 10486-10498. DOI: 10.1039/d2sc02839e

(31) Ellson, J.; Gansner, E.; Koutsofios, L.; North, S. C.; Woodhull, G. Graphviz—open source graph drawing tools. In *International Symposium on Graph Drawing*, 2001; Springer: pp 483-484.

(32) Deagen, M. E. *Interactive BigSMILES Editor*. Observable, 2023. https://observablehq.com/@mdeagen/interactive-bigsmiles-editor (accessed 19 January, 2023).

(33) Lamport, L. *I1 (\LaTeX)---A Document*; pub-AW, 1985.

(34) *Prism*. 2023. https://prismjs.com/ (accessed 14 March, 2023).

(35) Deagen, M. E. *Interactive Polymer Structure Search*. Observable, 2023. https://observablehq.com/@mdeagen/interactive-polymer-structure-search (accessed 19 January, 2023).

(36) Lin, T. S.; Rebello, N. J.; Beech, H. K.; Wang, Z.; El-Zaatari, B.; Lundberg, D. J.; Johnson, J. A.; Kalow, J. A.; Craig, S. L.; Olsen, B. D. PolyDAT: A Generic Data Schema for Polymer Characterization. *J. Chem. Inf. Model.* **2021**, *61* (3), 1150-1163. DOI: 10.1021/acs.jcim.1c00028

(37) McCusker, J. P.; Keshan, N.; Rashid, S.; Deagen, M.; Brinson, C.; McGuinness, D. L. NanoMine: A knowledge graph for nanocomposite materials science. In *The Semantic Web–ISWC 2020: 19th International Semantic Web Conference, Athens, Greece, November 2–6, 2020, Proceedings, Part II*, 2020; Springer: pp 144-159.

(38) Walsh, D. J.; Zou, W.; Schneider, L.; Mello, R.; Deagen, M. E.; Mysona, J.; Lin, T. S.; de Pablo, J. J.; Jensen, K. F.; Audus, D. J.; and Olsen, B.D. Community Resource for Innovation in Polymer Technology (CRIPT): A Scalable Polymer Material Data Structure. *ACS Cent. Sci.* **2023**, *9* (3), 330-338. DOI: 10.1021/acscentsci.3c00011

(39) Donahue, C. J. Reimagining the Materials Tetrahedron. *J. Chem. Educ.* **2019**, *96* (12), 2682-2688. DOI: 10.1021/acs.jchemed.9b00016

(40) Deagen, M. E.; Brinson, L. C.; Vaia, R. A.; Schadler, L. S. The materials tetrahedron has a "digital twin". *MRS Bull.* **2022**. DOI: 10.1557/s43577-021-00214-0

(41) Khokhlov, I.; Krasnov, L.; Fedorov, M. V.; Sosnin, S. Image2SMILES: Transformer-Based Molecular Optical Recognition Engine. *Chemistry–Methods* **2022**, *2* (1), e202100069. DOI: 10.1002/cmtd.202100069

(42) Krenn, M.; Häse, F.; Nigam, A.; Friederich, P.; Aspuru-Guzik, A. Self-referencing embedded strings (SELFIES): A 100% robust molecular string representation. *Mach. Learn. Sci. Technol.* **2020**, *1* (4), 045024. DOI: 10.1088/2632-2153/aba947

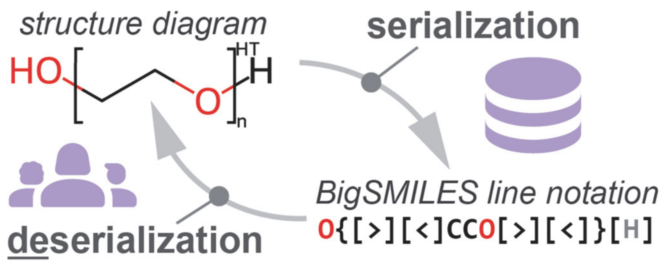
(43) Krenn, M.; Ai, Q.; Barthel, S.; Carson, N.; Frei, A.; Frey, N. C.; Friederich, P.; Gaudin, T.; Gayle, A. A.; Jablonka, K. M.; Lameiro, R.F.; Lemm, D.; Lo, A.; Moosavi, S.M.; Nápoles-Duarte, J.M.; Nigam, A.; Pollice, R.; Rajan, K.; Schatzchneider, U.; Schwaller, P.; and Aspuru-Guzik, A. SELFIES and the future of molecular string representations. *Patterns* **2022**, *3* (10), 100588. DOI: 10.1016/j.patter.2022.100588

(44) Kim, S.; Schroeder, C. M.; Jackson, N. E. Open Macromolecular Genome: Generative Design of Synthetically Accessible Polymers. *ACS Polym. Au* **2023**. DOI: 10.1021/acspolymersau.3c00003

(45) Bilodeau, C.; Jin, W.; Jaakkola, T.; Barzilay, R.; Jensen, K. F. Generative models for molecular discovery: Recent advances and challenges. *WIREs Comp. Mol. Sci.* **2022**, *12* (5), e1608. DOI: 10.1002/wcms.1608

(46) Ma, R.; Luo, T. PI1M: A Benchmark Database for Polymer Informatics. *J. Chem. Inf. Model.* **2020**, *60* (10), 4684-4690. DOI: 10.1021/acs.jcim.0c00726

Table of Contents Graphic:


structure diagram
HO
O
HT
H
n
serialization
BigSMILES line notation
O{[>][<]CCO[>][<]}[H]
deserialization