BigSMARTS: A Topologically-Aware Query Language and Substructure Search Algorithm for Polymer Chemical Structures

Nathan J. Rebello,[1] Tzyy-Shyang Lin,[1] Heeba Nazeer,[2] and Bradley D. Olsen[1]

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

[2]Department of Computer Science, Wellesley College, 106 Central St, Wellesley, Massachusetts, 02481, United States

Corresponding Author: Bradley D. Olsen, E-mail: bdolsen@mit.edu

Abstract

Molecular search is important in chemistry, biology, and informatics for identifying molecular structures within large data sets, improving knowledge discovery and innovation, and making chemical data FAIR (findable, accessible, interoperable, reusable). Search algorithms for polymers are significantly less developed than those for small molecules because polymer search relies on searching by polymer name, which can be challenging because polymer naming is overly broad (i.e. polyethylene), complicated for complex chemical structures, and often does not correspond to official IUPAC conventions. Chemical structure search in polymers is limited to substructures such as monomers without awareness of connectivity or topology. This work introduces a novel query language and graph traversal search algorithm for polymers that provide the first search method able to fully capture all of the chemical structures present in polymers. The BigSMARTS query language, an extension of the small molecule SMARTS language, allows users to write queries that localize monomer and functional group searches to different parts of the polymer, like the middle block of a triblock, the sidechain of a graft, and the backbone of a repeat unit. The substructure search algorithm is based on the traversal of graph representations of the generating functions for the stochastic graphs of polymers. Operationally, the algorithm first identifies cycles

representing the monomers, then the end groups, and finally performs a depth-first search to match entire subgraphs. To validate the algorithm, hundreds of queries were searched against hundreds of target chemistries and topologies from the literature, with approximately 440,000 query-target pairs. This tool provides a detailed algorithm that can be implemented in search engines to provide search results with the full matching of the monomer connectivity and polymer topology.

1. Introduction

Molecular search is one of the most important problems at the intersection of chemistry, biology, and data-driven research impacting the daily lives of researchers as they rapidly find and process information. In cheminformatics, two-dimensional chemical structures are generally treated as molecular graphs with atoms as nodes and bonds as edges.[1-5] Searching functional groups or substructures in molecular graphs is used to search and filter databases, access properties, highlight and count functional groups, and solve a wide range of problems.[6-15] Moreover, search technologies and data mining tools make information easily accessible to researchers in academia, industry, and the general public on databases like PubChem and Reaxys.[16]

Substructure search technologies are well-developed for small molecule chemistries because each molecule may be represented by a deterministic graph, allowing graph traversal algorithms to be used for structure matching. To implement search algorithms, databases index molecules using strings of ASCII characters, or line notations, which are very data compact. The Simplified Molecular Input Line Entry System, or SMILES, is the most popular line notation to index small molecules as deterministic molecular graphs because it is compact and has few grammatical rules.[2, 17] The SMARTS query language[18] encodes molecular patterns or subgraphs to be searched; after initial steps to accelerate the search process on large databases, final matching is performed to SMILES molecular graphs using graph traversal algorithms that have been well-

studied.[3, 4] However, polymers are stochastic graphs, making SMILES and SMARTS technologies unable to fully capture their connectivity and topology within a search algorithm.

Absent full graph traversal search that is available for small molecules, several alternative search technologies are currently used for polymer chemical structures. First, users can search for polymers by name on a tool like Google Scholar or Web of Science. Although IUPAC has developed a standardized naming convention for polymers,[19, 20] common usage continues to involve non-standard naming conventions and abbreviations, including the use of trade names.[20-22] Common polymers like poly(styrene) are referred to by many variations of the same name ("poly(styrene)", "polystyrenes", "styrene polymer"), and for these polymers, a simple name search is not precise enough to effectively filter the search results to desired end groups, topologies, or copolymer structures. Nomenclature gets complicated for complex polymers, and many authors choose not to name polymers, instead referring to them with numbers or proprietary codes that make them effectively unsearchable by name.

Databases like Reaxys and PubChem allow users to search for small molecules by structure, but users cannot search for polymers and classify them by topology. Users can search for polymer structures on some databases like SciFinder, but there are limitations. Searching for polymers by their monomers is valuable, but users may not be familiar with the chemistry. Finding a polymer according to its structural repeating unit (SRU) is beneficial, but users could query variations of the SRU, like frame shifts, that are valid structural identifiers of the polymer but are not recognized by the search algorithm. Moreover, searching for repeat units by molecular formula can be ambiguous because it does not specify how the atoms connect. For example, C_2H_4O can match poly(vinyl alcohol) or poly(ethylene glycol). Alternately, polymers could be queried by string matching to a line notation representation such as BigSMILES.[23] For polymers with defined

backbones a solution for line notation canonicalization has been proposed that makes this a possibility; however, searching substructures in a BigSMILES string can be challenging and this solution excludes many branched and network polymers of technological importance.

Motivated by the impacts of molecular search, this work introduces a novel query language for polymers called BigSMARTS, a finite graph representation for all polymer topologies inspired by state machines, and a substructure search algorithm that extends the depth-first search algorithm to check for a match. First, the query language is introduced that allows users to search chemistry and topology and localize chemistry queries (functional groups and repeat units) to different parts of the polymer topology. Then, a graph generation algorithm is presented for the query and target. Repeat unit graphs are cyclic and periodic in literature,[24-26] and the polymer graphs in this work extend this idea to all topologies. From the graphs, a substructure search algorithm for a match is described. Finally, a validation procedure is presented with hundreds of unit tests crafted to test different types of queries.

2. BigSMARTS Query Language

Table 1. BigSMARTS Summary

Syntax	Syntax Origin	Meaning	Query Example
Regular SMARTS	SMARTS	Searches functional groups anywhere in the target	CC(c1ccccc1)
SMARTS in stochastic objects	BigSMILES	Searches functional groups in the repeat units	{[]CC(c1ccccc1)[]}
SMARTS with bonding	BigSMILES	Forms repeat unit and end	{[][$]CC(c1ccccc1)[$][]}

descriptors attached: {[<],[>],[$]}		group queries; Searches target backbone	
Repeat unit and end group lists	BigSMILES	Searches copolymers	{[][$]CC(c1ccccc1)[$],[$]CC=CC[$][]}
Hydrogens written in repeat units	SMARTS + BigSMILES	Restricts pendant groups	{[][$][CH2][CH](c1ccccc1)[$][]}
Hydrogens written in end groups	SMARTS + BigSMILES	Designates terminal points in target chain	[CH3]CC(C){[$][$]CC(c1ccccc1)[$][]}
Operators before elements in stochastic objects: {[or],[xor],!}	BigSMARTS	Logical operations	{[][or1][$]CC(c1ccccc1)[$],[or1][$]CC=CC[$][]}
!* as element in stochastic objects	BigSMARTS	No other repeat units or end groups	{[][$]CC(c1ccccc1)[$],!*[]}
?* anywhere an atom is allowed	BigSMARTS	Wildcard topological distances	{[][<]C=C?*[>][]}
{[][]}	BigSMARTS	Wildcard stochastic objects	{[][]}?*{[$][$]CC(c1ccccc1)[$][$]}?*{[][]}
!{[][]}	BigSMARTS	No other objects	{[][]}?*{[$][$]CC(c1ccccc1)[$][$]}?*{[][]}!{[][]}

The BigSMARTS query language is a straightforward grammatical extension of the BigSMILES and SMARTS languages, and every BigSMILES[23, 27] and SMARTS[18] string is a valid BigSMARTS string. The key difference between BigSMILES and BigSMARTS is that repeat units and end groups in BigSMARTS are written in SMARTS syntax, whereas those in BigSMILES are written in SMILES syntax. A BigSMILES string represents a set of molecules (a

polymer), whereas a BigSMARTS string represents a set of subgraph patterns (a polymer query). SMARTS and BigSMARTS are more general than SMILES and BigSMILES because labels for a SMARTS atom require properties like the hydrogen count of the atom to be explicitly specified.[18] Unspecified properties in SMARTS or BigSMARTS are not assumed to be a part of the query. Table 1 shows a summary of the query language.

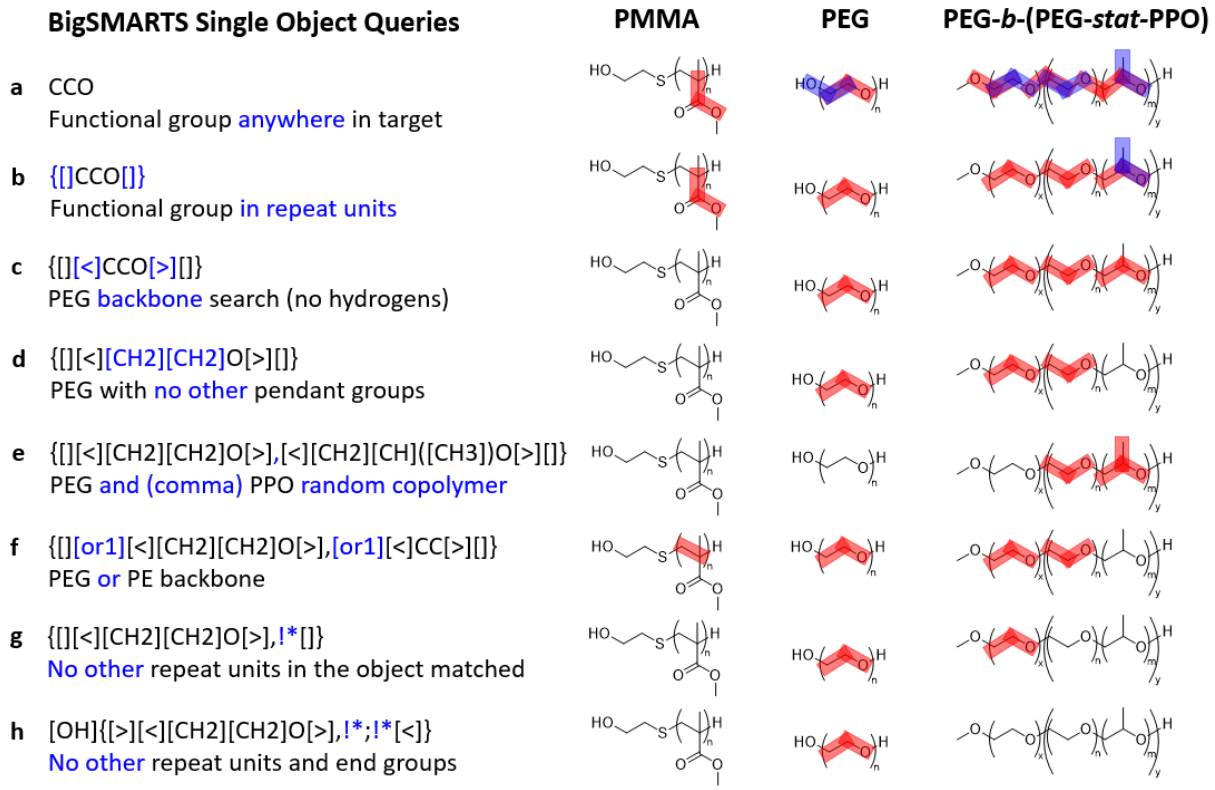


Figure 1. BigSMARTS searches and substructure hits for a variety of single stochastic object queries. Each box highlights an atom pair and bond that is matched, and a second color (transparent blue) is added to show overlapping hits. (a)-(e) have grammatical elements that combine SMARTS and BigSMILES: (a) functional group searched anywhere in the target; (b) localized to the repeat units; (c) descriptors added to form poly(ethylene glycol) backbone; (d) poly(ethylene glycol); (e) copolymer search; (f)-(h) have grammatical elements unique to BigSMARTS: (f) logical search;

(g) poly(ethylene glycol) with no other repeat units in the target object matched; (h) poly(ethylene glycol) with no other repeat units or end groups other than what was written.

2.1. Functional Group Local and Global Queries

When a user writes a BigSMARTS query using only grammatical elements from SMARTS, the query will search for that subgraph anywhere in the target graph. Queries using specific BigSMARTS grammatical elements allow the user to localize the query to specific topological regions of the polymer. For example, the query CCO in Figure 1a searches anywhere in the polymer, while {[]CCO[]}, without bonding descriptors searches for the specified subgraph anywhere within a monomer (Figure 1b).

2.2. Repeat Unit Queries

When bonding descriptors are specified (Figure 1c), these additionally enforce localization of specific atoms to the backbone of the polymer chain, as in the query

{[][<]C(=O)CN[>][]}

which would match any poly(amino acid) backbone because hydrogens in a SMARTS or BigSMARTS query are not specified. It is important to note that only atoms and bonds of the repeat units and end groups match during the search, not the bonding descriptors. Therefore, a query with one set of bonding descriptors may match to a target with another as long as the monomers match according to the matching rules in the search algorithm (*vide infra*). The addition of hydrogens (Figure 1d) yields a query that would only match poly(glycine):

{[][<]C(=O)[CH2][NH][>][]}

2.3. Copolymer Queries

Users can write queries with multiple repeat units in the stochastic object, and all listed repeat units must be found in the target. Therefore, the standard comma-delimited list element from BigSMILES is interpreted as an "and" logical operator in BigSMARTS. The query string poly(alanine-*co*-glycine):

{[][<]C(=O)[CH]([CH3])[NH][>],[<]C(=O)[CH2][NH][>][]}

will only match to targets with both monomers present in the same polymer. Figure 1e shows another example. In this example, if the user were not to specify hydrogens on poly(ethylene glycol) or [<]CCO[>]:

{[][<]CCO[>],[<]CC(C)O[>][]}

The repeat unit queries for poly(ethylene glycol) and poly(propylene glycol) could both match to the repeat unit for poly(propylene glycol) in the target, illustrated in Figure 1e.

2.4. Joint Functional Group and Repeat Unit Queries

The user can remove bonding descriptors from one or more repeat units to search for chemical substructures without regard to the polymer backbone, and both the functional group and repeat unit must match to the same stochastic object, and it is possible that multiple elements in the stochastic object can match to the same repeat unit if hydrogens are not specified:

{[][<]C(=O)[CH]([CH3])[NH][>],C(=O)[CH2][NH][]}

2.5. End Group Queries

So far, all of these examples have not specified end groups. Users can easily search including end groups in BigSMARTS using the same syntax for representing end groups in

BigSMILES; however, unless hydrogens are fully specified, the end groups may not be terminal points in the graph. For example, the query for PEG with end groups:

C{[<][>]OCC[<][>]}OC(=O)

would match the target:

C{[<][>]OCC[<][>]}OC(=O)C=C

However, adding hydrogens to the query:

C{[<][>]OCC[<][>]}O[CH](=O)

would prevent a match.

2.6. Logical Queries

In addition to these grammatical elements that are already present in BigSMILES and SMARTS, BigSMARTS adds several additional grammatical elements to fully capture polymer connectivity and topology. First, because the monomer list within each stochastic object effectively represents a joined search, users can use logical operators “or”, “xor”, and “not” in order to specify more complex queries. As described earlier, by default, no operator means “and”. The not operator is represented by an exclamation point and need only be included on repeat units to be specifically excluded from search, as in this search for polystyrene polymers specifically excluding poly(4-chloromethylstyrene):

{[][$]CC(c1ccccc1)[$],![$]CC(c1ccc(CCl)cc1)[$][]}

Queries that include “or” and “xor” apply the symbols [or] or [xor] before sets of different repeat units that are governed by the desired search logic, and the symbols are numbered in order to allow multiple simultaneous “or” and “xor” logical elements to be included in a search. Illustrating this on the poly(alanine-*co*-glycine) query,

{[][or1][<]C(=O)[CH]([CH3])[NH][>],[or1][<]C(=O)[CH2][NH][>][]}

would match to poly(alanine) homopolymer, poly(glycine) homopolymer, or poly(alanine-*co*-glycine). In contrast,

{[][xor1][<]C(=O)[CH]([CH3])[NH][>],[xor1][<]C(=O)[CH2][NH][>][]}

would match either to poly(alanine) or to poly(glycine) but not to the copolymer. Figure 1f shows an additional example.

2.7. No Other Repeat Units and End Groups

The search logic of BigSMARTS functions such that the query is searched as a subgraph within BigSMILES targets, similarly to SMARTS searches within SMILES strings. Therefore, any search specification for a given monomer set also hits to copolymers that include this monomer set as a subset. To allow exact matches to be specified, the not symbol can be combined with a wild card, yielding !* which has the meaning "nothing more." For example, the query

{[][<]C(=O)[CH]([CH3])[NH][>],!*[]}

matches only to poly(alanine) homopolymer and not to any copolymers with any other monomers. The use of !* is further illustrated in Figures 1g and h.

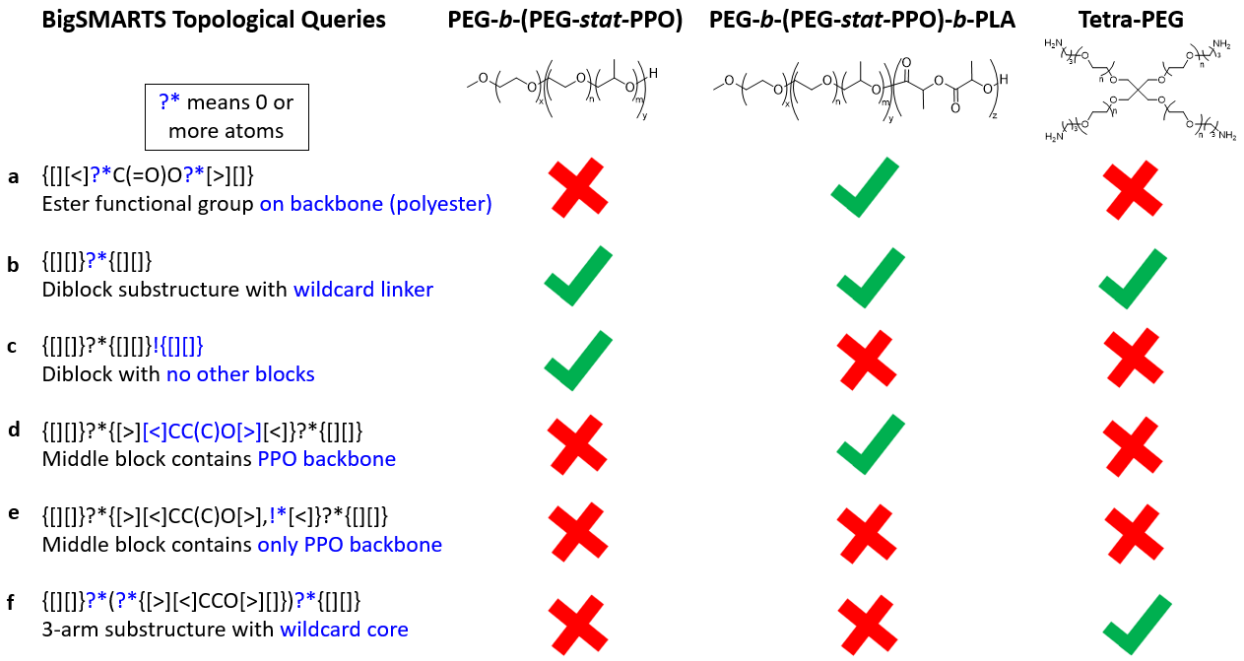


Figure 2. BigSMARTS searches and substructure hits for a variety of topological queries: (a) functional group backbone search; (b) topology search with wildcard stochastic objects and linkers; (c) there should be no other objects other than what is specified in the query; (d)-(e) repeat unit chemistry localization search, with the same grammatical elements in Figure 1; (f) wildcard branch point.

2.8. Wildcard Topological Distances

The presence of large connecting groups in polymers also demands grammatical additions that can include wildcard segments in addition to wildcard atoms traditionally found in SMARTS. These are shown in Figure 2. Among many applications, this enables users to produce more nuanced searches for polymer topology. Wildcard linear sequences of atoms are represented by ?* which specifies a connection between the two adjoining strings without specifying the connection length. A null string is allowed as a possible linker with ?*. For example,

{[][$]CC(c1ccccc1)[$][$]}?*{[$][$]CC(C)(C(=O)OC)[$][]}

specifies a query for a poly(styrene-*b*-methyl methacrylate) diblock copolymer without specifying the type of chemistry used to form the junction between the two monomers. This type of query can be extremely valuable when one wishes to search across different synthetic methods (i.e. anionic polymerization vs. living free radical polymerization) that would use different linker lengths.

The "?*" symbol is also very useful for finding all polymers with a given functional group along the backbone but not in side chains. For example,

{[][<]?*C(=O)O?*[>][]}

would match all polyesters but not polyacrylates or polymethacrylates. This compares favorably to standard substructure search for esters which is unable to differentiate even these basic polymer families. Grammatically, "?*" can be embedded anywhere an atom is allowed in the repeat units and can be used to match entire end groups.

2.9. Wildcard Stochastic Objects

To compose queries for polymer topologies, the user may write empty stochastic objects, {[][]}, which query for the presence of a polymeric segment without specifying its specific chemistry. For example, to search for an unfunctionalized polystyrene homopolymer as the central block between two other polymer blocks, a user would write

{[][]}?*{[$][$][CH2][CH]([c]1[cH][cH][cH][cH][cH]1)[$][$]}?*{[][]}

Alternately, the query

{[][]}?*{[>][<][CH2][CH]([CH3])O[>][<]}?*{[][]}

specifies a search for a middle block of poly(propylene oxide) and would match a poloxamer. While the wildcard sequence element ?* need not be included between blocks in either of these two examples, leaving it out would convey the meaning that there are no atoms between the

stochastic objects, restricting the matches to the query in a way that does not reflect the most likely meaning of the user.

2.10. Complex Topological Queries

The user may write more complex topological queries using the ?* operator. For example, to search for a 4-arm star polymer

{[][]}?*(?*{[][]})(?*{[][]})(?*{[][]})

in which each arm contains the poly(ethylene glycol) repeat unit backbone, the user can write

{[][<]CCO[>][<]}?*(?*{[>][<]CCO[>][]})(?*{[>][<]CCO[>][]})(?*{[>][<]CCO[>][]})

These repeat units can be frame-shifted:

{[][<]COC[>][<]}?*(?*{[>][<]COC[>][]})(?*{[>][<]COC[>][]})(?*{[>][<]COC[>][]})

To search for an H-polymer, the user can write:

{[][]}?*(?*{[][]})?*{[][]}?*(?*{[][]})?*{[][]}

in which the center block can contain the PEG repeat unit:

{[][]}?*(?*{[][]})?*{[>][<]CCO[>][<]}?*(?*{[][]})?*{[][]}

2.11. No Other Objects

Because SMARTS and BigSMARTS specify query subgraphs within a larger target graph, there is a possibility that the target graph contains stochastic objects not in the query. A final additional grammatical element added to the query language allows the user to specify that the query cannot contain stochastic objects that are not explicitly specified. To make this specification, the user combines the not operator and the empty stochastic object, !{[][]}, as the

final two grammatical elements in a query. For example, this query can be used to specify a search for all block copolymers with exactly two blocks:

{[][]}?*{[][]}!{[][]}

or for star polymers with exactly three arms:

{[][]}?*(?*{[][]})?*{[][]}!{[][]}

3. Atomistic and Topological Graphs

A polymer is an extremely large ensemble of molecules with distinct chemical structures. Therefore, they are represented by stochastic graphs. This complicates the generalization of graph traversal algorithms used for small molecules to perform substructure search for polymers because different members of the ensemble may give different results for any query, and a query of a polymer against another polymer has an impossibly large number of binary pairs of molecular graphs to compare. Therefore, building on previous work that has demonstrated that polymers are directly analogous to automata,[28] BigSMARTS converts polymers into a deterministic graph representation based on the generating function that is used to construct the molecules, fully capturing the molecular topology, connectivity, and chemical rules from molecular synthesis. In the previously published automata representation, nodes are states, edges are the building blocks, cycles in the graph represent effective repeat units, and these graphs generate sequences of building blocks from the start to the stop state. State machine minimization algorithms can be used to find a canonicalized or unique representation for a polymer, an equivalent machine with redundant states removed (an entirely separate problem). In this work, the graphs for substructure search have atoms as nodes and bonds (closely analogous to states of automata) as edges, a transformation of the original automata concept that facilitates graph traversal searching. These graphs do not show

probabilities of repeat units and end groups connecting, but rather the set of possible connections that generate the ensemble.

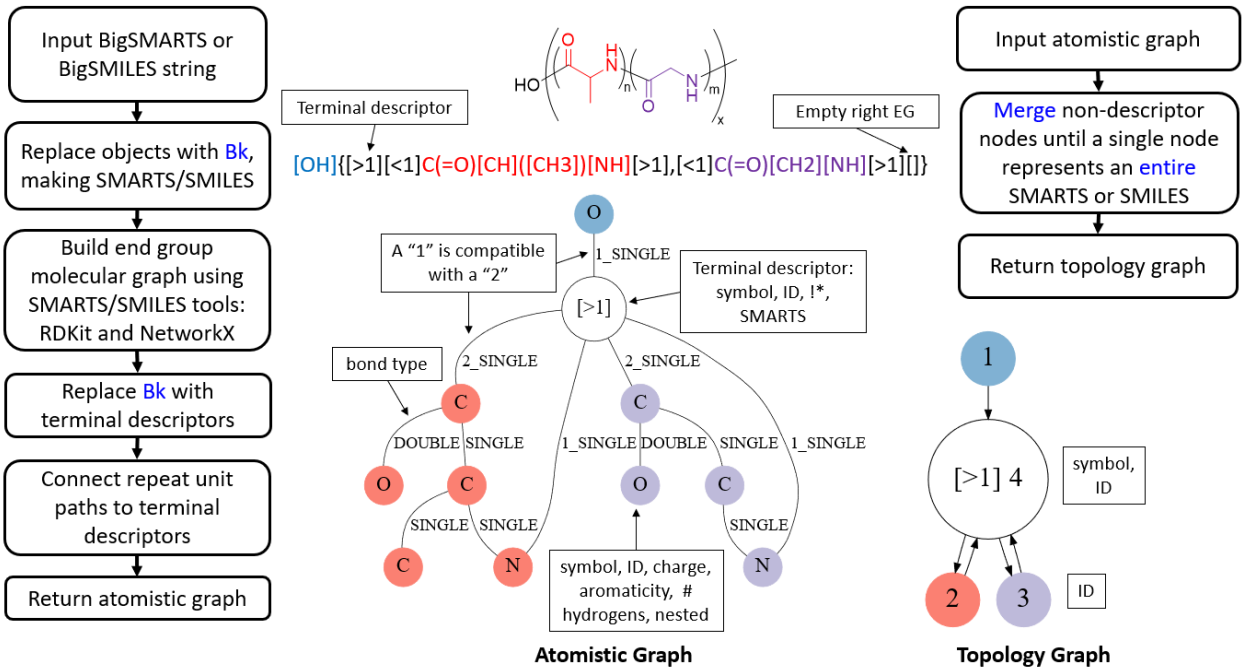
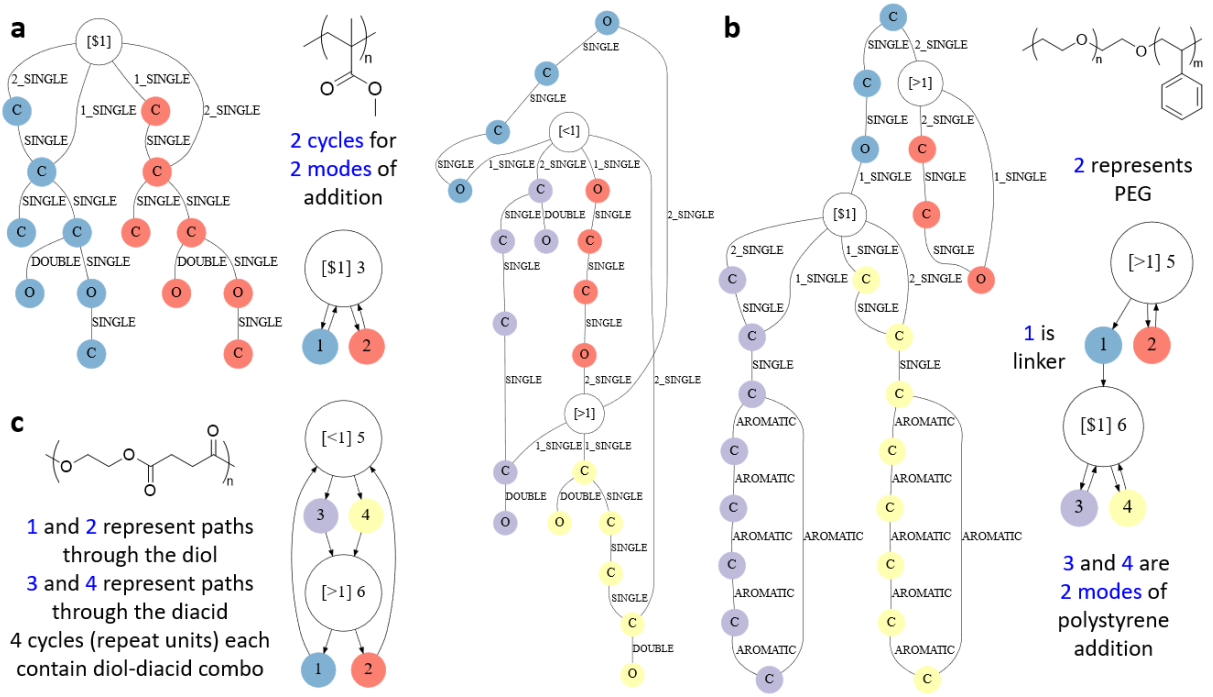


Figure 3. Every BigSMARTS or BigSMILES string is converted into an atomistic and topology graph. First, a BigSMARTS or BigSMILES is converted into a SMARTS or SMILES by replacing stochastic objects, "?*", and "{[][]}" with SMILES atoms. In this example, the input string can be replaced with [OH][Bk][Es] ("Bk" for stochastic object and "Es" for empty end group), which is a valid SMILES. RDKit converts SMILES into a molecular graph, and NetworkX is the graph package used to create polymer graphs and execute substructure search. Each cycle in the graph represents a repeat unit path. An edge labeled "1" connects an atom to its descriptor neighbor in the string and "2" connects an atom to the compatible version of its descriptor neighbor. Thus, oxygen ("O") connects to the terminal descriptor "[>1]" with a "1". Because of string replacement, topological strings in BigSMARTS like "?*" and "{[][]}" are embedded in the graph. Non-repeat unit strings in the stochastic object like "!*" and SMARTS are stored as attributes in the descriptor

nodes. The "!{[][]}" is saved by the algorithm. Attributes in each graph are shown: the integer topology ID links the atomistic graph to the topology graph. In the atomistic graph, only the colored nodes, atoms and bonds, are searched during traversal, not the descriptors.



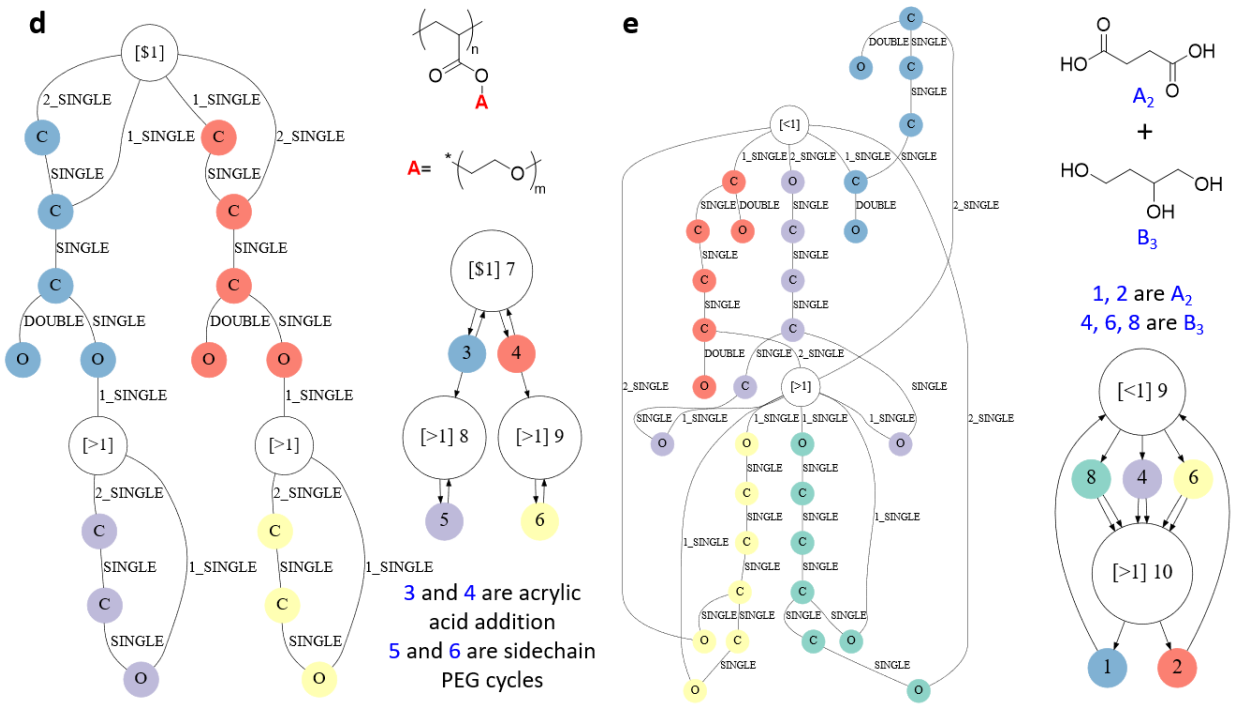


Figure 4. Atomistic and topology graphs for five polymers: (a) poly(methyl methacrylate) or {[][$]CC(C)(C(=O)OC)[$][]}, in which the same repeat unit is represented twice in the graph to illustrate head-to-head tail-to-tail addition; (b) poly(ethylene glycol-*b*-styrene) or {[][<]CCO[>][<]}CCO{[$][$]CC(c1ccccc1)[$][]}; (c) Poly(ethylene succinate) or {[][<]OCCO[<],[>]C(=O)CCC(=O)[>][]}; (d) poly(styrene) with poly(ethylene glycol) sidechain or {[][$]CC(C(=O)O{[>][<]CCO[>][]})[$][]}; (e) branched polyester or {[][<]C(=O)CCC(=O)[<],[>]OCCC(O[>])CO[>][]}.

To convert a polymer into a molecular graph inspired by automata, the algorithm determines all possible paths through the repeat units and end groups. Figure 3 shows a flowchart and an example, and Figure 4 shows examples of a chain-growth polymer, block, AA-BB copolymer, graft, and branched polymer. Figure 5 illustrates these graphs for BigSMARTS

grammatical extensions mentioned in the previous section. Supporting Information Section 1 shows a comprehensive flowchart, and Supporting Information 2 shows examples of different topologies.

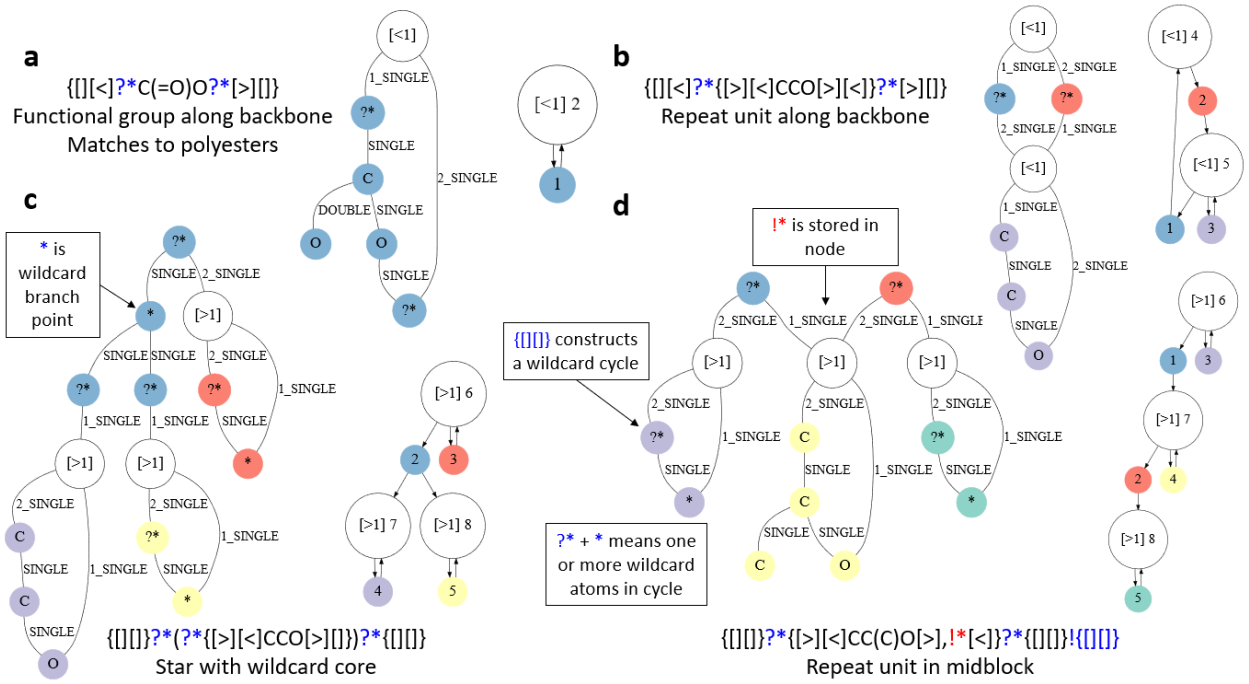


Figure 5. Graphs with BigSMARTS grammatical extensions for queries in Figure 2 except for segmented polymer (b) and "!{[][]}" added to (d). Specifically, "?*" and "{[][]}" are embedded into the graph and are checked during traversal. The string "{[][]}" constructs a wildcard cycle in the query graph, but during search, it matches to an entire stochastic object or set of repeat units; stored in the descriptor node is a Boolean value that this cycle represents a wildcard object. Moreover, "!{[][]}", which means no other objects in the target, is checked before graph traversal begins (does the number of query objects equal the number of target objects?). Non-repeat unit elements, including functional groups (SMARTS) localized to the target repeat units and "!* in the descriptors, are checked during or after substructure search completes.

By way of example, consider the poly(alanine-*co*-glycine) query with hydrogens mentioned earlier in Figure 3, encoded in BigSMARTS as:

[OH]{[>][<]C(=O)[CH]([CH3])[NH][>],[<]C(=O)[CH2][NH][>][]}

The first step is converting the BigSMARTS or BigSMILES into a valid SMARTS or SMILES by replacing the stochastic objects with heavy atoms ([Bk]), empty end groups with [Es], and saving the repeat units. The SMARTS or SMILES "[OH][Bk][Es]" is converted into an RDKit molecular graph with labels for each node, including the element symbol, formal charge, aromaticity, and hydrogen count, and labels for each edge, including the bond type (see Supporting Information Section 1 for full list). For each stochastic object, the algorithm determines all possible paths through the repeat units and inserts them in place of [Bk]: since each repeat unit is head-to-tail addition, there is a single path for each denoted with the list:

[[<], C(=O)[CH]([CH3])[NH], [>]]

[[<], C(=O)[CH2][NH], [>]]

These lists can be connected to form a graph by combining compatible bonding descriptors ([>*n*] is compatible with [<*n*] and [$*n*] is compatible with itself), and a graph with two cycles for two repeat units can be generated. As shown in Figures 3-5, state machine-inspired polymer graphs are directed, and the direction is determined based on how the string is written, parsing the repeat units and end groups from left to right, which means there can be many ways of encoding the same polymer. Nevertheless, if the ensembles generated by the string are the same, then the search algorithm ensures a match. This algorithm does not explicitly construct lists, but at a high level, this is how the state-machine-inspired graphs are generated.

This tool is robust to equivalent transformations of the stochastic object string, including repeat unit inversions, changes to the SMARTS or SMILES string, repeat unit order, and bonding descriptors. For example, consider poly(alanine-*co*-glycine) query:

[OH]{[>][<]C(=O)[CH]([CH3])[NH][>],[<]C(=O)[CH2][NH][>][]}

The following poly(alanine) strings all produce a graph with a single equivalent cycle:

O=C([<])C(C)N[>] (changes to the SMILES)

[>]C(=O)C(C)N[<] (changes to the bonding descriptors)

[<]NC(C)C(=O)[>] (inversion)

This is because in all three cases, the repeat unit is the same, and the nitrogen must connect with the carbonyl carbon, forming a cycle.

The mapping from repeat units and end groups in the strings to the cycles in the graph is not necessarily one-to-one. If there are multiple modes of propagation, a single repeat unit maps to multiple cycles. For example, consider a chain-growth repeat polymer poly(methyl methacrylate) with the BigSMILES in Figure 4:

{[][$]CC(C)(C(=O)OC)[$][]}

The algorithm parses the repeat units and bonding descriptors and determines that there are two paths through the repeat unit for H-H and H-T addition, represented by two lists with the repeat unit inverted:

[[$], CC(C)(C(=O)OC), [$]]

[[$], C(C)(C(=O)OC)C, [$]]

This forms a graph with two cycles. Conversely, for AA-BB polycondensation reactions, multiple repeat units can form a single cycle of the graph. The BigSMILES string for poly(ethylene succinate):

{[][<]OCCO[<],[>]C(=O)CCC(=O)[>][]}

There are two paths through each repeat unit, but the effective repeat unit for this polymer is the diacid-diamine units joined together, and this would form four linked cycles, shown in Figure 4. Supporting Information Section 2.8 shows a more advanced example.

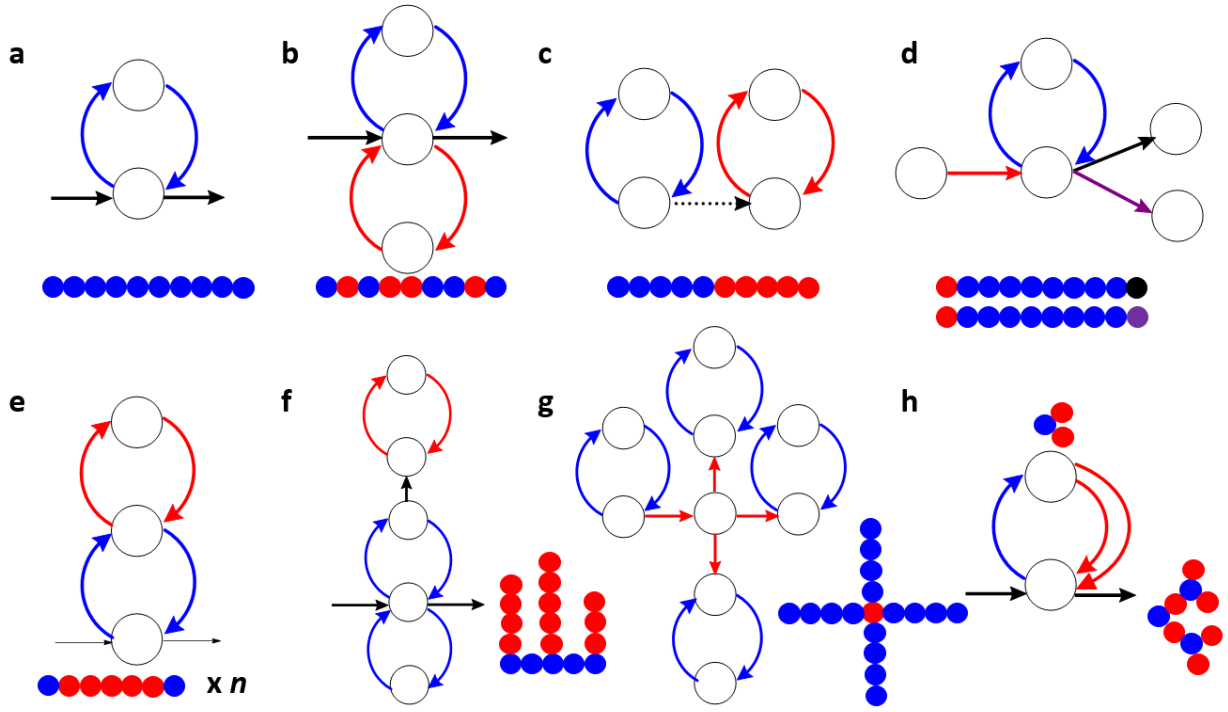


Figure 6. Topological graphs that resemble finite automaton for a: (a) homopolymer, (b) statistical copolymer, (c) block copolymer, (d) polymer with multiple modes of termination, (e) segmented, (f) graft, (g) star, and (h) dendrimer. The nodes represent repeat units, end groups, or bonding descriptors. The only attribute in these graphs is an integer ID (see Figure 3) that connects nodes in this graph to sets of nodes in the atomistic graph.

In order to detect cycles or repeat units for substructure search, the algorithm builds a second graph, the topology graph, which is a directed graph. In order to do this, every SMARTS and SMILES in the atomistic graph is contracted into a single node using NetworkX's

contracted_edge function, which inputs a pair of nodes to merge, and the descriptors become separate nodes in the topological graph. Figures 4 and 5 show several examples of the resulting topological graphs, and Figure 6 shows drawings of polymer graphs for different topologies produced from this algorithm, from block copolymers that contain cycles in series to star polymers that contain cycles connected to a central core.

4. Substructure Search Algorithm

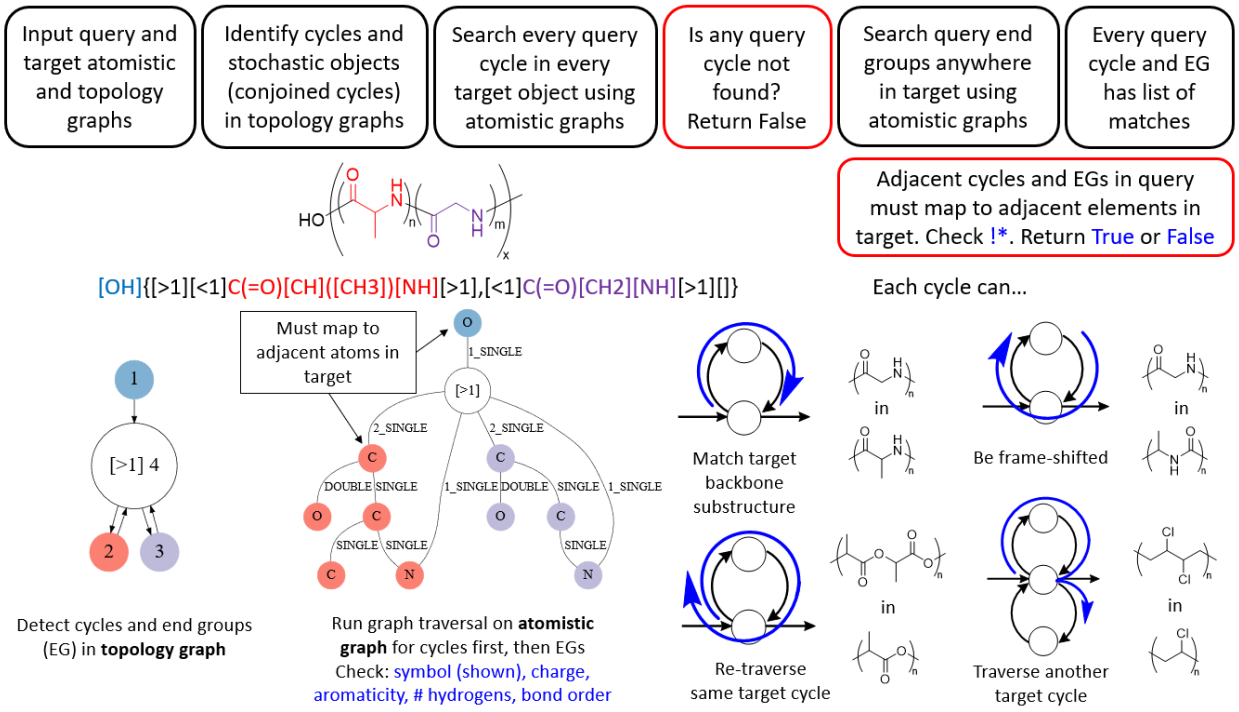


Figure 7. Polymer search requires the atomistic graph for substructure search and topology graph for cycle detection. During graph traversal, descriptors are skipped.

A match between a BigSMARTS query and a BigSMILES target is defined on the basis of sets. Each BigSMARTS query represents a set of SMARTS objects, while each BigSMILES string represents a set of SMILES objects. For a query to match a target, every SMARTS in the set defined by the BigSMARTS must be found as a substructure in at least one of the SMILES in the

set defined by the BigSMILES. Because a pairwise comparison of the two sets would be extremely computationally intensive, the proposed algorithm instead performs substructure search through graph traversal of the state-machine graphs in the previous section, yielding equivalent results. Figure 7 outlines the algorithm.

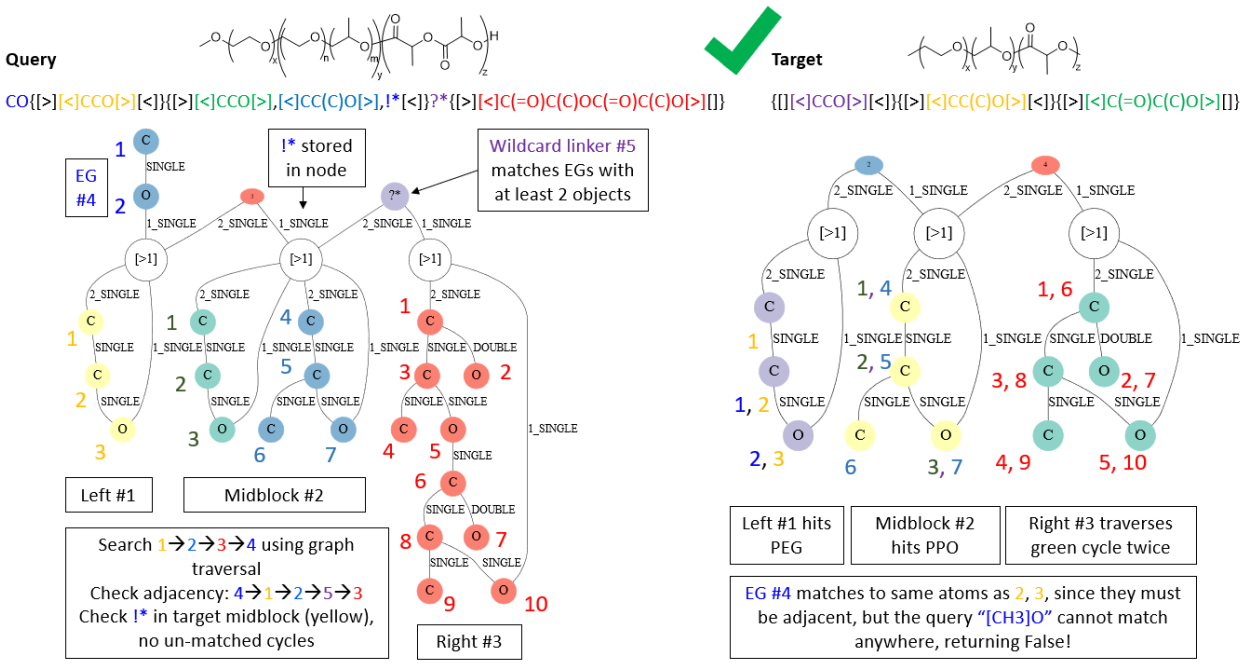


Figure 8. A triblock query with a wildcard linker and no other repeat units in the middle block (!*) is searched in a triblock target. The cycles are searched first #1-3, followed by the end groups (EG) #4-5. PEG (poly(ethylene) glycol) in the left-most block #1 matches to the left-most block (labeled 1-3 in yellow). PEG and PPO (poly(propylene) oxide) in the query midblock #2 match to the same cycle in the target midblock because hydrogens were not added (labeled 1-3 in green and 4-7 in blue). Multiple query objects cannot match to the same target object, so the left two blocks cannot both match the middle block. The lactide repeat unit in the right block #3 re-traverses the target lactic acid repeat unit, such that multiple query atoms map to the same target atom (labeled 1-10 in red). The EG #4 in blue matches to all target repeat units because hydrogens were not added, but in the final solution matches to the left object due to adjacency. Wildcard EG #5 can

match to either target EG, but in the final solution matches the red one due to adjacency. Enforcing adjacency is important: if triblock ABC was reversed to CBA, the match would still return True, but for BCA would return False.

A necessary condition for a match between a query and a target is that both contain the same polymeric segments, represented by sets of conjoined cycles and end groups in the atomistic and topological graphs. Therefore, the search algorithm first determines all query and target cycles using the topological graphs (NetworkX's function *simple_cycles*), determines sets of conjoined cycles or stochastic objects (NetworkX's function *connected_components*), and a substructure search is executed on the cycles in the atomistic graphs to match different sets of conjoined cycles. Searching cycles first allows rapid down-selection of molecules, reducing computational cost later in the algorithm. When searching a query cycle in a target cycle, the nodes and edges in the molecular graph are traversed using a depth-first search (DFS) algorithm (Supporting Information Section 3 provides a flowchart), in which the algorithm starts from a start node and traverses as far as it can down a given branch before backtracking, checking whether the node and edge information matches, shown in Figures 8 and 9. For the purpose of graph traversal, bonding descriptors are treated as null symbols, so atoms connected through bonding descriptors are treated as equivalent to directly connected.

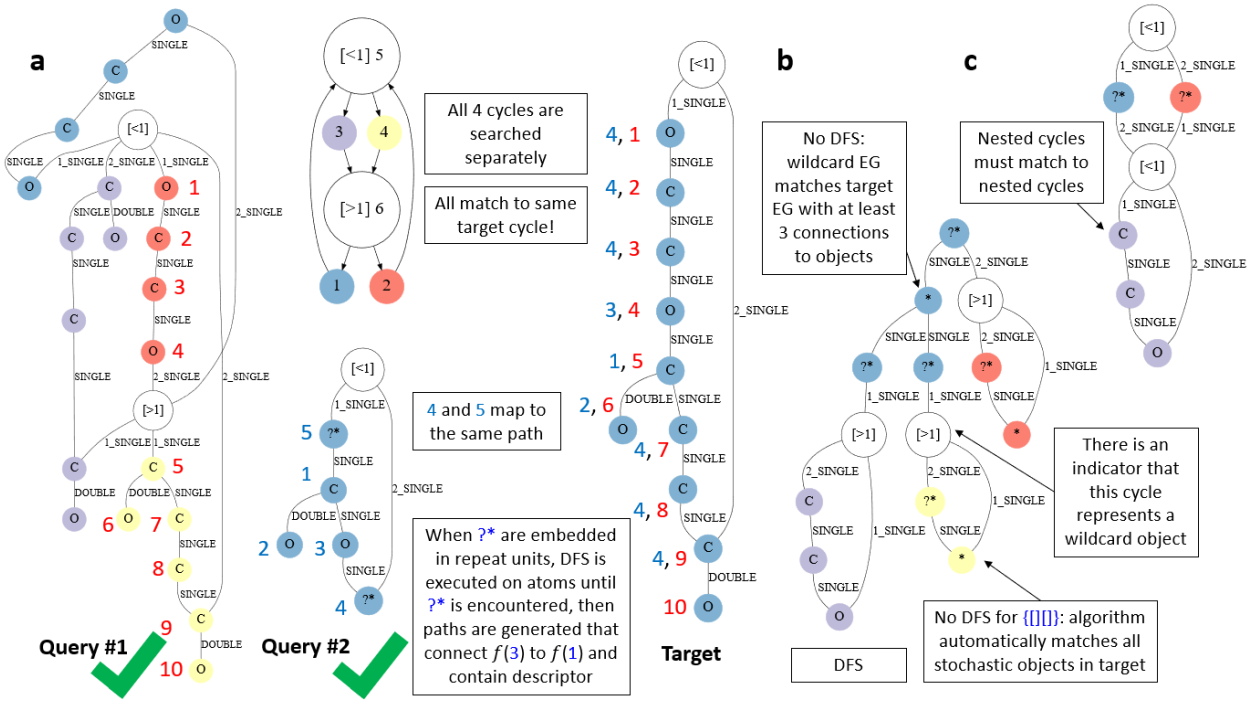


Figure 9. (a) Poly(ethylene succinate) or {[][<]OCCO[<],[>]C(=O)CCC(=O)[>][]} has two repeat units in the AA-BB representation in Figure 4(c) and is searched in a target with a single repeat unit {[][<]OCCOC(=O)CCC(=O)[>][]} in the AB representation. Every query cycle can be found in the target and is connected, yielding a match. The query from Figure 5(a) or {[][<]?*C(=O)O?*[>][]} is searched in the same target. (b) The algorithm detects wildcard end groups and matches them to entire target end groups depending on the number of stochastic objects the wildcard end group connects. The wildcard end group in this figure, the same star polymer query in Figure 5(c) or {[][]}?*(?*{[>][<]CCO[>][]})?*{[][]}, is a substructure of any target end group that connects at least 3 objects and would match to Tetra-PEG, shown in Figure 2(f). (c) When querying polymers with nested repeat units, like grafts and segmented copolymers, nested repeat units must match to nested repeat units.

Then, the query end groups are searched in the target using the DFS, but as previously mentioned, end groups with atoms written can traverse cycles and match exclusively to repeat units if hydrogens are not written in the query. This is because end groups in a query represent deterministic patterns, which can match to substructures of repeat units but may not necessarily match to terminal points in the target chain. An example is shown in Figure 8. For each cycle and end group, a list of matches is stored, and the final match is True if there exists one match for each RU and EG such that connected cycles and EGs in the query map to connected segments in the target. After adjacency between cycles and end groups is enforced, a set of final conditions are checked: the not RU element "!*", multiple query stochastic objects cannot match to the same target object, and nested objects must match to nested objects.

The key differences between polymer search and small molecule search is cycles and end groups are searched separately, query cycles can traverse target cycles multiple times, and the grammatical extensions in Section 2 and Figure 9 ("{[][]}", "?*") enable chemistry and topological classification. The string "?*" embedded as an atom allows for functional group searches along the backbone. When a "?*" node is encountered during DFS traversal, NetworkX's function *simple_paths* is used to generate paths in the target that connect $f(\text{prev})$ to $f(\text{next})$, where prev and next are atoms in the query before and after "?*", f is the mapping function, and "?*" maps to a set of atoms. Moreover, end groups can be entirely wildcard with "?*"; in this case, no graph traversal is executed, the algorithm will simply match wildcard end groups connecting *n* stochastic objects to entire target end groups that connect at least that many stochastic objects.

5. Database Validation

To validate the algorithm, a matrix of BigSMARTS and BigSMILES strings was constructed, and the match state was recorded as true or false for each pair. Overall, there are approximately 900

BigSMARTS queries searched over the entire target dataset of 500 BigSMILES for polymer chemistries from literature. The authors used domain knowledge and made a concerted effort to add diverse chemistries and topologies to the BigSMILES database from Google Scholar, Web of Science, open-source polymer datasets,[24] and textbooks.[29] Nevertheless, in order to grow this polymer database in size and scope, users can deposit structures and references to Community Resource for Innovation in Polymer Technology (CRIPT),[30] a data ecosystem to organize polymer data. Different rules are tested for regular SMARTS, single objects like homopolymers, statistical copolymers, alternating copolymers, block copolymers, segmented, grafts, stars, networks, chiral polymers, macrocycles, polyelectrolytes, dendrimers, and H-polymers.

Tables 2-5 shows four case studies, and a database validation sheet is provided with all other case studies. The first case study in Table 2 explores mutations to the query string that do not change its chemical meaning and demonstrates that the algorithm is robust such that these mutations do not affect the search results. The second case study in Table 3 illustrates classifying polymers according to their backbone. The third case study in Table 4 provides examples of queries to search functional groups along the backbone, illustrating the ability to perform substructure search across a wide range of functionalities. The fourth case study in Table 5 provides examples of queries that classify topology.

A substructure match is perfect: either true or false. In all of these case studies, every query-target pair that is expected to match, either as a substructure match or a full match, is marked with a true in the spreadsheet, and the algorithm always returns true. Every query-target pair that is not expected to match is marked with a false in the spreadsheet (that is, at least one query atom or element cannot be found in the target), and the algorithm always returns false. Across all chemistries and topologies, it takes approximately 15-20 seconds for a single query to search all

of the targets. The authors are building algorithms to execute filtering steps to speed up search, as discussed in the next section.

Table 2. Database validation examples for repeat unit mutations with increasing restriction on the target ensemble.

BigSMARTS Queries	Meaning	# BigSMILES Hits	Example
CCO	Functional group search anywhere in target	207	C#CCOC(=O)C(C)(C){[$][$]CC(c1ccccc1)[$][$]}SC(=S)c1ccccc1 Poly(styrene) with end group match
{[]CCO[]}	Localized to RUs	198	{[][$]CC(C(=O)OCCCC)[$][]} Poly(n-butyl acrylate) with side chain match
{[][<]CCO[>][]}	PEG	68	O{[>][<]C(=O)C(C)O[>][<]} Poly(lactic acid)
{[][<]COC[>][]}	Frameshift	68	Same matches
{[][<]CC[<2],[>2]O[>][]}	Split, but same RU	68	Same matches
{[][<]OCC[>][]}	Inversion	68	Same matches
{[][>3]CCO[<3][]}	Changes to descriptor IDs	68	Same matches
{[]O([<])CC[>][]}	Bonding descriptors not at ends	68	Same matches

{[][<]CCO[>],[<]CCO[>][]}	Duplication (both match to same target unit)	68	Same matches
{[][<]OCC[>][<]}OCC	Repeat unit in end group	68	Same matches
{[][<][CH2][CH2]O[>][]}	No other pendant groups with hydrogens	57	C(c1ccccc1)O{[>][<]CCO[>],[<]CC(COCCN(C(C)C)(C(C)C))O[>][<]}[H] Poly(ethylene oxide-*co*-DEGE)
{[][<][CH2][CH2]O[>],!*[]}	No other pendant groups and no other RU	45	{[][$]CC(c1ccccc1)[$][$]}{[>][<]CCO[>][]} Poly(styrene-*b*-ethylene oxide)
{[][<]CCO[>];!{[][]}[]}	Eliminates multi-blocks	34	{[][$]CC(C(=O)O)[$],[$]CC(C(=O)O{[>][<]CCO[>][<]}C)[$][]} Poly(acrylic acid-*graft*-ethylene oxide)
{[][<]CCO[>],!*;!{[][]}[]}	Matches to PEG with no other repeat units and deterministic end groups	17	{[>][<]OCC[>][<]}O Hydroxy-terminated PEG
{[][<][CH2][CH2]O[>],!*;!*[]}	No other RU and EG	1	{[][<]CCO[>][]} PEG

Table 3. Database validation examples for repeat unit backbone classification.

BigSMARTS Queries	Meaning	# BigSMILES Hits	Example
{[][$]CC[$][]}	Poly(ethylene) backbone	257	{[][$]CC(C)(C)[$][]} Poly(isobutylene)
{[][<][Si]O[>][]}	Poly(siloxane) backbone	23	{[][<][Si](C)(C)O[>][]} Poly(dimethyl siloxane)
{[][$]CC(C(=O)O)[$][]}	Poly(acrylate) backbone	64	{[][$]CC(C(=O)OCC)[$][]} Poly(ethyl acrylate)
{[][$]CC(OC(=O))[$][]}	Poly(acetate) backbone	6	{[][$]CC(OC(=O)CC)[$][]} Poly(vinyl propionate)
{[][$]CC(c1ccccc1)[$][]}	Poly(styrene) backbone	115	{[][$]CC(c1ccc(OC)cc1)[$][]} Poly(4-methoxystyrene)
{[][$]CC=CC[$][]}	Poly(diene) backbone	29	{[][$]CC=C(C)C[$][]} Poly(isoprene)

Table 4. Database validation examples for repeat unit chemistry classification according to functional groups along backbone.

BigSMARTS Queries	Meaning	# BigSMILES Hits	Example
{[][<]C(=O)O?*[>][]}	Polyester	75	{[][<]OCCO[<],[>]C(=O)c1ccc(cc1)C(=O)[>][]} Poly(ethylene terephthalate)
{[][<]OC(=O)O?*[>][]}	Polycarbonate	29	{[][<]Oc1ccc(cc1)C(C)(C)c2ccc(cc2)O[<],[>]C(=O)[>][]} Bisphenol A polycarbonate
{[][<]NC(=O)O?*[>][]}	Polyurethane	1	{[][>]C(=O)Nc1ccc(C)c(c1)NC(=O)[>],[<]OCC{[>][<]OCC[>][<]}O[<],[<]OCCCO[<][]} Poly(urethane) with nested PEG
{[][<]C=C?*[>][]}	Polydiene	31	{[][$]CC=C(Cl)C[$][]} Poly(chloroprene)
{[][<]NC(=O)N?*[>][]}	Polyurea	6	{[][<]NCCC{[>][<][Si](C)(C)O[>][<]}[Si](C)(C)CCCN[<],[>]C(=O)NC(CC1)CCC1CC(CC2)CCC2NC(=O)[>][]} Segmented Poly(urea)

Table 5. Database validation examples for topology searches.

BigSMARTS Queries	Meaning	# BigSMILES Hits	Example
{[][]}	Matches to all polymers (not small molecules in database)	489	{[][$]CC(c1ccccc1)[$][$]}{[>][<]CCO[>][]} Poly(styrene-*b*-ethylene oxide)
{[][]}!{[][]}	Prohibits matches to multiblocks	382	{[][$]CC(F)(F)[$][]} Poly(vinylidene fluoride)
{[][]}?*{[][]}	Diblock substructure	107	{[][<]OCC[>][<]}{[<][>]OC(C)C[<][>]}{[>][<]OCC[>][]} Pluronic
{[][$]CC(c1ccccc1)[$][$]}?*{[>][<]CCO[>][]}	Unspecified midblock between styrene and PEG	14	CCC(C){[$][$]CC(c1ccccc1)[$][$]}{[$][$]C\C=C(C)/C[$],[$]C\C=C(C)\C[$],[$]CC(C(C)=C)[$],[$]CC(C)(C=C)[$][$]}{[$][$]CC(c1ccccc1)[$][$]}{[>][<]CCO[>][<]}[H] SISO tetrablock
{[][]}?*{[][]}!{[][]}	No other blocks	78	CCC(C){[$][$]CC(c1ccccc1)[$][$]}{[$][$]CC(c1ncccc1)[$][$]}[H] Poly(styrene-*b*-2-vinyl pyridine)

{[][]}?*{[][]}?*{[][]}	Triblock substructure	15	CCC(C){[$][$]CC(c1ccccc1)[$][$]}{[$][$]C\C=C(C)/C[$],[$]C\C=C(C)\C[$],[$]CC(C(C)=C)[$],[$]CC(C)(C=C)[$][$]}{[$][$]CC(c1ccccc1)[$][$]}{[>][<]CCO[>][<]}[H] SISO tetrablock
{[][<]?*{[>][<]?*[>][<]}?*[>][]}	Segmented copolymer with repeat unit along backbone	10	{[][<]NCCC{[>][<][Si](C)(C)O[>][<]}[Si](C)(C)CCCN[<],[>]C(=O)NC(CC1)CCC1CC(CC2)CCC2NC(=O)[>][]} Segmented Poly(urea)
{[][<]?*{[>][<]CCO[>][<]}?*[>][]}	Segmented copolymer with nested PEG	5	{[][>]C(=O)Nc1cccc(C)c(c1)NC(=O)[>],[<]OCC{[>][<]OCC[>][<]}O[<],[<]OCCCO[<][]} Poly(urethane) with nested PEG
{[][<]?*(?*{[>][<]?*[>][]})?*[>][]}	Graft copolymer with repeat unit in sidechain	11	{[][$]CC(c(cc1)ccc1CCCC{[>][<]OCC[>][<]}S)[$],[$]CC(c1ccccc1)[$][]} Poly(styrene)-*graft*-Poly(ethylene oxide)
{[][]}?*(?*{[][]})?*{[][]}	3-arm star polymer substructure	21	C(c1ccccc1)OCC{[>][<]OCC[>][<]}OC(COC(=O)C(C)(C){[$][$]CC(C(=O)OC(C)(C)C)[$][$]}Br)CC(c1ccccc1)C

			{[$][$]CC(c1ccccc1)[$][$]}CCCC 3-Miktoarm Star Polymer: PS-PEO-(P*t*BA-Br)

6. Discussion

Current polymer chemical structure search capability is limited. Users can search for polymers by name on tools like Google Scholar or Web of Science, but multiple strings could match to the same polymer (PEG vs PEO vs poly(ethylene glycol)), and polymer nomenclature is non-standardized and complex. Users could search for monomers or structural repeat units on tools like SciFinder, but searches for complex topologies can be difficult, limiting our ability to discover data. This algorithm allows the user to write queries of polymer structures, including a single repeat unit substructure without knowledge of the monomer chemistry, repeat unit sets in a stochastic object forming copolymers, chemistries with desired end functionalities, and advanced chemistries and topologies with branched repeat units and end groups. For those unfamiliar with the grammar of line notations, a drawing tool is also available so that the user can draw any query topology and use copy the generated BigSMARTS query string to feed to this algorithm.[31] A key feature of the algorithm is its ability to search across all of these rich features while still being robust to variations in the way that users may have represented the chemical structure of a target or query, overcoming limitations in current canonicalization technology. Several of the features of this algorithm, including classifying polymers by localizing functional groups to specific parts of the target topology, including the backbone, or searches for arbitrary topological structures, have not previously been achievable.

This tool has been validated on hundreds of thousands of query-target pairs that produce the expected match/no match results. However, the search algorithm has a time complexity that scales O(N), making implementation on large databases challenging. The overall search performance can be accelerated by implementing filtering steps before the substructure search. There are several simple classifiers that can be applied to polymers that are easily indexed and can assist with such filtering. First, the number of stochastic objects in the query should be less than or equal to the number of stochastic objects in the target, which is a simple check. For example, a block copolymer cannot match a homopolymer. Second, a branched polymer cannot match a linear polymer; this can be simply determined by classifying polymers based on the number of bonding descriptors on the repeat units. Finally, every cycle in the graph can be treated as a small molecule cycle and searched using RDKit, which has already been optimized for small molecule search. If the repeat units are properly canonicalized and indexed, then repeat units from the atomistic graphs can be searched using string-based search algorithms that scale well in large data systems.[17, 32]

The finite state machine-inspired graph representations have several advantages over existing polymer graph representations for polymer search and have broad applications in many fields. These graphs are finite, machine-readable, and work for different polymer topologies. Beyond search, these graphs can be converted into vector representations or fingerprints, in which bits contain information on the presence or absence of substructures. These fingerprints can be used to compare molecules (similarity metrics) and can be used in machine learning to connect the structure to the property.[33]

7. Conclusions

This work introduces a query language for polymers, a molecular graph representation inspired by state machines for different topologies, and a substructure search algorithm that together provide

a search functionality for polymer chemical structure that is fully aware of polymer chemical connectivity and topology. The query language allows users to search combinations of linear and branched repeat units and end groups with the connections specified, and query wildcard elements, including wildcard stochastic objects, wildcard end groups, searches targeting the target objects, and searches along the target backbone. The molecular graph representation is finite and accurately represents a set of molecules, and wildcard elements are embedded in the graph. The substructure search algorithm runs a graph traversal of the repeat unit cycles first, followed by the end groups, and then checks whether matches are adjacent in the target graph. The depth-first search algorithm is extended to wildcard elements embedded in the graph.

This tool offers advantages over existing search technologies. Users can search by structure rather than by name, resolving challenges in searching complex structures according to polymer nomenclature. Moreover, users can search any polymer chemistry and topology including wildcard elements, which extends the capabilities in SciFinder and allow users to discover data for complex polymers. This search technology will allow polymer chemists to enjoy the same benefits of molecular search that small molecule chemists have enjoyed and support FAIR data principles by making data much more discoverable, accelerating materials innovation.

Acknowledgements

Funding: This work was funded by the National Science Foundation Convergence Accelerator award number ITE-2134795.

Data and Materials Availability: The unit testing spreadsheet for database validation is provided as an Excel sheet at https://github.com/olsenlabmit/BigSMARTS. The drawing tool for BigSMARTS can be found at https://github.com/olsenlabmit/BigSMILES_builder.

Supporting Information: list of node and edge attributes in graphs; examples of graphs for different polymer chemical structures; substructure search flowcharts

References

(1) Kearnes, S.; McCloskey, K.; Berndl, M.; Pande, V.; Riley, P. Molecular graph convolutions: moving beyond fingerprints. *Journal of computer-aided molecular design* **2016**, *30*, 595-608.
(2) Weininger, D. SMILES, a chemical language and information system. 1. Introduction to methodology and encoding rules. *Journal of chemical information and computer sciences* **1988**, *28* (1), 31-36.
(3) David, L.; Thakkar, A.; Mercado, R.; Engkvist, O. Molecular representations in AI-driven drug discovery: a review and practical guide. *Journal of Cheminformatics* **2020**, *12* (1), 1-22.
(4) Landrum, G. Rdkit documentation. *Release* **2013**, *1*, 1-79.
(5) O'Boyle, N. M.; Banck, M.; James, C. A.; Morley, C.; Vandermeersch, T.; Hutchison, G. R. Open Babel: An open chemical toolbox. *Journal of cheminformatics* **2011**, *3* (1), 33.
(6) Coley, C. W.; Barzilay, R.; Jaakkola, T. S.; Green, W. H.; Jensen, K. F. Prediction of organic reaction outcomes using machine learning. *ACS central science* **2017**, *3* (5), 434-443.
(7) Irwin, J. J.; Sterling, T.; Mysinger, M. M.; Bolstad, E. S.; Coleman, R. G. ZINC: a free tool to discover chemistry for biology. *Journal of chemical information and modeling* **2012**, *52* (7), 1757-1768.
(8) Kenny, P. W.; Sadowski, J. Structure modification in chemical databases. *Chemoinformatics in drug discovery* **2005**, *23*, 271-285.
(9) Jeliazkova, N.; Kochev, N. AMBIT-SMARTS: Efficient Searching of Chemical Structures and Fragments. *Molecular Informatics* **2011**, *30* (8), 707-720.
(10) Brown, N.; Fiscato, M.; Segler, M. H.; Vaucher, A. C. GuacaMol: benchmarking models for de novo molecular design. *Journal of chemical information and modeling* **2019**, *59* (3), 1096-1108.
(11) Sushko, I.; Salmina, E.; Potemkin, V. A.; Poda, G.; Tetko, I. V. ToxAlerts: a web server of structural alerts for toxic chemicals and compounds with potential adverse reactions. ACS Publications: 2012.
(12) Harper, G.; Bravi, G.; Pickett, S. D.; Hussain, J.; Green, D. V. The reduced graph descriptor in virtual screening and data-driven clustering of high-throughput screening data. *Journal of chemical information and computer sciences* **2004**, *44* (6), 2145-2156.
(13) Yang, K.; Swanson, K.; Jin, W.; Coley, C.; Eiden, P.; Gao, H.; Guzman-Perez, A.; Hopper, T.; Kelley, B.; Mathea, M. Analyzing learned molecular representations for property prediction. *Journal of chemical information and modeling* **2019**, *59* (8), 3370-3388.
(14) Coley, C. W. a. B. R. a. G. W. H. a. J. T. S. a. J. K. F. Convolutional Embedding of Attributed Molecular Graphs for Physical Property Prediction. *Journal of Chemical Information and Modeling* **2017**, *57* (8), 1757-1772. DOI: 10.1021/acs.jcim.6b00601.
(15) Coley, C. W.; Green, W. H.; Jensen, K. F. Machine learning in computer-aided synthesis planning. *Accounts of chemical research* **2018**, *51* (5), 1281-1289.
(16) Wilkinson, M. D.; Dumontier, M.; Aalbersberg, I. J.; Appleton, G.; Axton, M.; Baak, A.; Blomberg, N.; Boiten, J.-W.; da Silva Santos, L. B.; Bourne, P. E. The FAIR Guiding Principles for scientific data management and stewardship. *Scientific data* **2016**, *3* (1), 1-9.

(17) Weininger, D.; Weininger, A.; Weininger, J. L. SMILES. 2. Algorithm for generation of unique SMILES notation. *Journal of chemical information and computer sciences* **1989**, *29* (2), 97-101.
(18) *SMARTS - A Language for Describing Molecular Patterns*. 2022. https://www.daylight.com/dayhtml/doc/theory/theory.smarts.html (accessed Dec. 2020).
(19) Kahovec, J. a.; Fox, R.; Hatada, K. Nomenclature of regular single-strand organic polymers (IUPAC Recommendations 2002). *Pure and Applied Chemistry* **2002**, *74* (10), 1921-1956.
(20) Jones, R. G.; Division, I. U. o. P. a. A. C. P.; Wilks, E. S. *Compendium of polymer terminology and nomenclature: IUPAC recommendations, 2008*; RSC Pub., 2009.
(21) Audus, D. J.; de Pablo, J. J. Polymer informatics: opportunities and challenges. ACS Publications: 2017.
(22) Hu, B.; Lin, A.; Brinson, L. C. ChemProps: A RESTful API enabled database for composite polymer name standardization. *Journal of cheminformatics* **2021**, *13* (1), 1-13.
(23) 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. BigSMILES: A Structurally-Based Line Notation for Describing Macromolecules. *ACS central science* **2019**, *5* (9), 1523-1531.
(24) Arora, A.; Lin, T.-S.; Rebello, N. J.; Av-Ron, S. H.; Mochigase, H.; Olsen, B. D. Random forest predictor for diblock copolymer phase behavior. *ACS Macro Letters* **2021**, *10* (11), 1339-1345.
(25) Antoniuk, E. R.; Li, P.; Kailkhura, B.; Hiszpanski, A. M. Representing Polymers as Periodic Graphs with Learned Descriptors for Accurate Polymer Property Predictions. *Journal of Chemical Information and Modeling* **2022**, *62* (22), 5435-5445.
(26) Aldeghi, M.; Coley, C. W. A graph representation of molecular ensembles for polymer property prediction. *Chemical Science* **2022**, *13* (35), 10486-10498.
(27) *The BigSMILES Line Notation*. 2022. https://olsenlabmit.github.io/BigSMILES/docs/line_notation.html#the-bigsmiles-line-notation (accessed.
(28) Lin, T.-S.; Rebello, N. J.; Lee, G.-H.; Morris, M. A.; Olsen, B. D. Canonicalizing BigSMILES for Polymers with Defined Backbones. *ACS Polymers Au* **2022**.
(29) Bicerano, J. *Prediction of polymer properties*; cRc Press, 2002.
(30) 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. Community Resource for Innovation in Polymer Technology (CRIPT): A Scalable Polymer Material Data Structure. ACS Publications: 2023.
(31) Lin, T.-S. BigSMILES Builder.
(32) O'Boyle, N. M. Towards a Universal SMILES representation-A standard method to generate canonical SMILES based on the InChI. *Journal of cheminformatics* **2012**, *4* (1), 22.
(33) Bajusz, D.; Rácz, A.; Héberger, K. Why is Tanimoto index an appropriate choice for fingerprint-based similarity calculations? *Journal of cheminformatics* **2015**, *7* (1), 1-13.

Table of Contents Figure