Toward learned chemical perception of force field typing rules

Camila Zanette[1,a], Caitlin C. Bannan[2,a], Christopher I. Bayly[3], Josh Fass[4,5], Michael K. Gilson[6], Michael R. Shirts[7], John D. Chodera[5], David L. Mobley[1,2*]

[1]Department of Pharmaceutical Sciences, University of California, Irvine; [2]Department of Chemistry, University of California, Irvine; [3]OpenEye Scientific, Santa Fe, NM 87507; [4]Tri-Institutional Training Program in Computational Biology and Medicine, New York, NY 10065; [5]Computational and Systems Biology Program, Sloan Kettering Institute, Memorial Sloan Kettering Cancer Center, New York, NY 10065; [6]Skaggs School of Pharmacy and Pharmaceutical Sciences, University of California, San Diego; [7]Department of Chemical and Biological Engineering, University of Colorado Boulder, Boulder, CO 80309; [a]These authors contributed equally

Abstract Molecular mechanics force fields define how the energy and forces in a molecular system are computed from its atomic positions, thus enabling the study of such systems through computational methods like molecular dynamics and Monte Carlo simulations. Despite progress toward automated force field parameterization, considerable human expertise is required to develop or extend force fields. In particular, human input has long been required to define *atom types*, which encode chemically unique environments that determine which parameters will be assigned. However, relying on humans to establish atom types is suboptimal. Human-created atom types are often developed without statistical justification, leading to over- or under-fitting of data. Human-created types are also difficult to extend in a systematic and consistent manner when new chemistries must be modeled or new data becomes available. Finally, human effort is not scalable when force fields must be generated for new (bio)polymers, compound classes, or materials. To remedy these deficiencies, our long-term goal is to replace human specification of atom types with an automated approach, based on rigorous statistics and driven by experimental and/or quantum chemical reference data. In this work, we describe novel methods that automate the discovery of appropriate chemical perception: *SMARTY* allows for the creation of atom types, while *SMIRKY* goes further by automating the creation of fragment (nonbonded, bonds, angles, and torsions) types. These approaches enable the creation of move sets in atom or fragment type space, which are used within a Monte Carlo optimization approach. We demonstrate the power of these new methods by automating the rediscovery of human defined atom types (SMARTY) or fragment types (SMIRKY) in existing small molecule force fields. We assess these approaches using several molecular datasets, including one which covers a diverse subset of DrugBank.

***For correspondence:**
dmobley@mobleylab.org (David L. Mobley)

1 Introduction

Molecular simulations can provide detailed views of chemical and biological events that involve conformational changes and noncovalent binding, such as allostery, protein folding, and ligand-protein binding.[1–3]

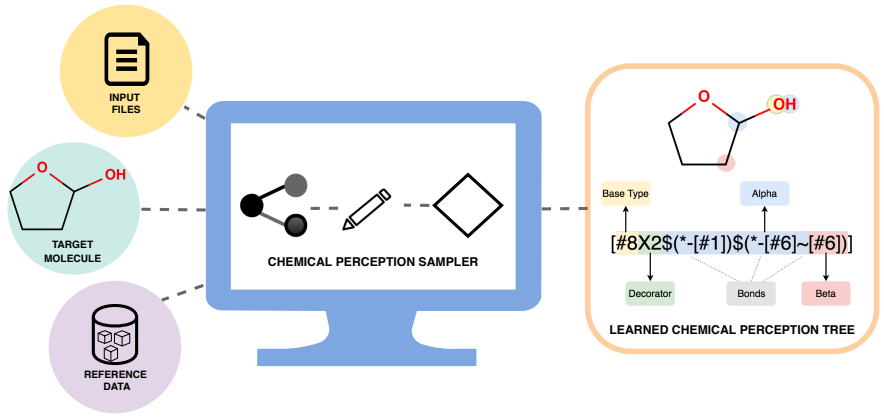


Figure 1. Table of Contents Figure

They also can be used to estimate various experimental observables, including solvation free energies of small molecules ,[4–6] binding free energies of small molecules to proteins and molecular hosts,[7–10] and the on and off rate constants for noncovalent association events.[11,12] Molecular simulation technologies rely on potential functions, or force fields, mathematical functions which estimate the energy of the molecular system and the forces on its atoms as a function of the atomic coordinates. The force field used in a simulation is a critical determinant of the accuracy of the results. The centrality of the force field has motivated decades of pioneering and innovative research and development.[1,13–27] In spite of these efforts, recent studies indicate that force field issues still significantly limit the accuracy of simulations.[9,28–41]

The Open Force Field Initiative seeks a systematic approach to the continuing challenge of improving force fields, reducing the human effort required and generating new force fields that make statistically sound use of appropriate reference data.[42] One major goal of our effort is to automate the development of new force fields given a choice of functional form and reference data. Later, we aim also to automate decisions about what functional forms to use and what reference data to fit. These capabilities will dramatically reduce the human time required to create new force fields, while also producing more accurate force fields with clear dependencies on the underlying data. Such capabilities would also support advances in force field science, such as determining what functional form achieves the best accuracy for particular applications at a specified level of computational cost. For example, a force field that explicitly treats electronic polarizability and includes fixed electric multipoles, such as AMOEBA,[25,43–49] should be able to reach higher accuracy than a force field with fixed, atom-centered point partial charges. But how much additional accuracy do multipoles and polarizability actually provide? We cannot currently answer these questions because the question of the functional form is conflated with other issues, such as the use of different input data, different atom types, different fitting methods, and differences in the chemical intuition brought to the problem. In contrast, an automated approach would allow systematic evaluation of the benefits of an advanced functional form within a given context.

Although some force field parameterization tools already exist,[50–57] the full process of force field development has not been automated. For example, the parameterization software ForceBalance[58–60] advanced the field by automating the adjustment of force field parameters against experimental and theoretical observables with a gradient-based optimizer. However, the researcher must still address not only how to weight the components of the objective function[24,61–63] but also the fundamental question of what atom types to use.

As we seek to automate parameterization, it is important to take note of where human expertise is typically employed, and one key place is in determining which chemical environments (usually treated as atom types) will be treated separately by a force field. This process of distinguishing between different chemical environments in order to assign force fields parameters we call the *chemical perception*. Chemical perception has been a key ingredient in building general purpose small molecule force fields.[35,64–70] Chemical perception in current force fields largely consists of assigning atom types or fragment (bond, angle, torsion)

types, defined using human intuition. Ideally, force field fitting should involve adjusting not only the numerical parameters of a force field, but also the chemical perception. For example, we could start a force field parameterization process with only a single type of carbon-carbon bond, but the automated process might propose adding a second bond type, thus allowing distinct parameters to be assigned to single versus double bonds. If this proposal led to improved accuracy sufficient to justify the increased number of parameters, it would become part of the force field definition. Thus, we need a way to sample not only over the numerical parameters of a force field, but also over the its atom or fragment types. However, we are not aware of any existing algorithm or software tools to carry out this type of automated parameter sampling.

In this paper, we address this fundamental problem with a novel approach to sampling over atom or fragment types typically used in biomolecular force fields, so that these definitions can ultimately be learned automatically without a human expert. In particular, we introduce methods of sampling over hierarchical chemical perception trees. That is, we organize force field atom or fragment types into hierarchical trees, such that child types contain more specific types than parent types. We utilize the `SMARTS` and `SMIRKS` substructure definition languages[71,72] to specify atom types and more general fragment types which may be associated with distinct numerical parameters. Then, we show how a Monte Carlo scheme can be used to sample over chemical perception trees defined using these languages. As a proof of principle for this approach, we compare the chemical complexity of sampled atom and fragment types to those in existing force fields. This sets the stage for future applications in which the scoring function will be based on the agreement of simulations with experimental, quantum, or other reference data.

2 Methods

We consider two different approaches for assigning force field parameters to a molecule: *direct* and *indirect* chemical perception.[73] Indirect chemical perception is exemplified by the traditional approach to parameter assignment (such as the AMBER and CHARMM force field families) in which, once a molecule's atoms have been typed, all other information about chemical environments – notably, bond orders – is discarded.[74] All force field parameters, including valence terms, are then assigned using only the atom types and the way they are connected.[75] Thus, in indirect chemical perception, atom types encode the information needed to assign all valence and bonded parameters.[69,76–79] Alternatively, in *direct chemical perception*, parameters are assigned based on the full molecular graph—a full valence representation of the molecule including elements, connectivity and bond orders, rather than just atom types and local connectivity. This approach thus involves direct analysis of the full molecular graph, including bond orders, instead of indirect analysis through the simplified molecular graph, as in indirect chemical perception. As previously detailed, direct chemical perception allows force fields to be fully specified with far fewer numerical parameters than required with indirect chemical perception.[73]

We recently introduced a new force field format, the `SMIRKS` Native Open Force Field (SMIRNOFF)[73] which uses `SMIRKS` patterns to allow direct assignment of force field parameters, thereby implementing direct chemical perception. This is a substantial break from indirect chemical perception which uses a graph labeled with atom types to assign parameters. SMIRNOFF instead uses *direct* chemical perception, using substructure searches via different `SMIRKS` strings to assign parameters when the target molecular substructures are encountered. Thus, the chemical perception and parameters for each force term are separate from those applied to other force terms. For example, a new set of Lennard-Jones parameters could be introduced without needing to introduce additional valence terms, or vise versa.

In this paper, we introduce approaches to sample chemical perception trees with both traditional atom types and SMIRNOFF fragment types. Here, we use the term *chemical perception tree* to describe a hierarchical classification of molecular substructures in order to assign parameters. One of our major interests is to sample over a variety of chemical perception trees to see if we can match the chemical perception used in existing force fields. We further use the term *fragment type* to refer to the more generalized notion of an "atom type" used by direct chemical perception — a particular substructure that would be assigned a particular parameter. The SMIRNOFF format, then, has four categories of fragment types – bond, angle, torsion, and nonbonded fragment types – corresponding to the valence and nonbonded interaction force

field parameters.

In order to automate sampling of chemical perception trees, a language to express atom or fragment types directly was required. We utilize the SMILES Arbitrary Target Specification (SMARTS) and SMILES Reaction Specification (SMIRKS) languages for this purpose.[71] A SMARTS string is a chemical substructure query, where a substructure is a set of atoms connected by bonds, and both atoms and bonds are typically further characterized with "decorators" (Table 1). For example, a bond may be decorated with "@" to indicate that it in a ring, and/or "-" to indicate a single bond, and detailed specifications may be constructed by the use of Boolean operators. Atoms are set apart from bonds with square brackets, for example, "`[#6X4]!@[#6r5]`" describes a tetrahedral carbon atom ("`[#6X4]`") connected by a non-ring bond ("`!@`") to another carbon atom which is in a five-membered ring ("`[#6r5]`"). SMIRKS strings provide a language similar to SMARTS but which also includes atom indexing. SMIRKS were created to allow description of reactions, but here we use only the atom indexing feature. For our purposes, we take advantage of the indexing in SMIRKS to track relevant atoms involved in fragment types such as bond, angle, or torsion types. For example, the SMARTS above could become a SMIRKS string with the addition of "`:`" to identify the atom indices ("`[#6X4:1]!@[#6r5:2]`"). Following the SMIRNOFF notation, a bond parameter involves two indexed atoms, an angle parameter three, and so on.[73]

	Symbol	Definition
Atom	#n	atomic number
	*	any atom
	A	aliphatic
	a	aromatic
	Hn	hydrogen count
	Xn	connectivity
	$\pm n$	charge
Bond	~	any bond
	@	ring bond
	–	single bond
Boolean modifiers	,	logical *or*
	&	high precedence logical *and*
	;	low precedence logical *and*
	!	logical *not*

Table 1. Decorators for elaborating parent SMIRKS and SMARTS strings. A selection of decorators that can be used for atoms (top third) and bonds (middle third) in SMIRKS or SMARTS strings. Decorators on atoms and bonds can be combined using Boolean operators (bottom third), where the high precedence *and* is applied before an *or* operator and the low precedence *and* is applied after. For a complete list of decorators and documentation for SMARTS and SMIRKS see the Daylight Theory Manual.[71]

The atom and fragment type definitions used here are based on the chemical environment of each atom extending up to two bonds away. In both SMARTY and SMIRKY, we only consider the chemical environment around the primary atom and atoms one bond away (alpha) or two bonds away (beta). Moves propose changes at any of these positions. The primary atom is the atom being typed by SMARTS or all of the indexed atoms in a SMIRKS. Atoms beyond this beta position are not currently considered. For example, consider a SMARTS describing the hydroxyl oxygen in an alcohol (Figure 2). Here, the oxygen is the atom to be typed (yellow), the hydrogen and carbon bonded to the oxygen are the alpha atoms (blue), and the carbon two bonds away is a beta atom (pink), as are the ring oxygen and the hydrogen atom bonded to the alpha carbon (not shown). Although future force fields might go further, atom types in most present-day force fields usually depend only on atoms out to the beta position. Therefore, SMARTY and SMIRKY currently do not propose new atom or fragment types that involve atoms past the beta position.

Like existing atom typing schemes,[72,80] both SMARTY and SMIRKY take a hierarchical approach to defining

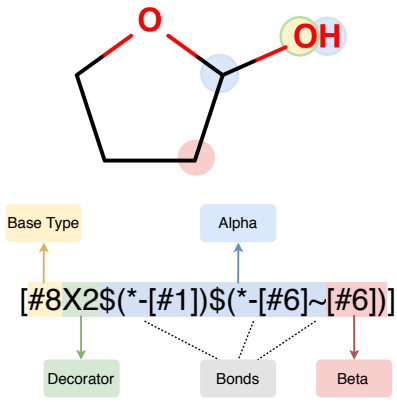


Figure 2. Describing a specific chemical group with SMARTS. An example of how a particular type of hydroxyl group in a molecule (top) can be described using SMARTS strings. In this case, the SMARTS pattern (bottom) is for a hydroxyl connected to a carbon which is itself connected to another carbon. The alpha atoms (light blue) for the oxygen base type ("[#8]") are one hydrogen ("[#1]") and one carbon ("[#6]"), and they are connected to the base type via a *single bond* ("–"). The carbon ("[#6]") atom is connected to another carbon ("[#6]", which is considered a beta atom (pink)) via *any bond* ("~"). The oxygen has a descriptor (light green) ("X2") which means it has two connected atoms.

atom and fragment types. That is, the SMARTS and SMIRKS strings specifying types are listed in a specific order, the last string which matches an atom or fragment is the one assigned ("last one wins"). For example, in Figure 3, HC ("[#1$(*-#6)]") would match all hydrogens bound to carbons, but then the string H1 ("[#1$(*-[#6]~[#8])]") overrides HC on the hydrogens on the alpha carbon. In the SMARTS language the "$" is used to specify neighboring atoms, in "[#1$(*-[#6]~[#8])]" the hydrogen is connected to a carbon bonded to an oxygen. This hierarchy allows a general pattern to catch all hydrogens that are not described by more specific patterns later in the list. We call a complete, hierarchical type specification of this sort a chemical perception tree. The same approach can be used on a hierarchical list of SMIRKS patterns for fragment types. The PATTY algorithm[81] was developed to automatically assign traditional atom types from a hierarchy, where each atom was assigned the last type matched to it. We adapted PATTY for use in SMARTY and SMIRKY. Specifically, the set of SMARTS or SMIRKS is matched to a molecule using OpenEye's OEChem Toolkit[82–84], and then only the last pattern to match a given atom or fragment is stored.

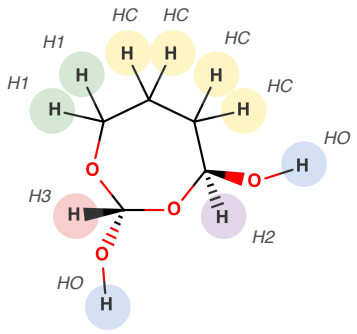


Figure 3. Illustration of the parm99/parm@Frosst hydrogen atom types for AlkEthOH set. This specific set has five different hydrogen types, OH (highlighted in blue), HC (highlighted in yellow), H1 (highlighted in green), H2 (highlighted in lilac), and H3 (highlighted in red) atom types. Reference atom types labels for each hydrogen atom are shown in italics next to each atom. Here are the SMARTY recovered SMARTS for each of these atom types. "[#1]" (HO), "[#1$(*-[#6])]" (HC), "[#1$(*-[#6]~[#8])]" (H1), "[#1$(*-[#6](-[#8])~[#8])]" (H2), "[#1$(*-[#6](-[#8])(-[#8])~[#8])]" (H3).

In the subsections below, we describe methods of varying the SMARTS and SMIRKS strings used to define atom or fragment types, respectively, in order to sample over chemical perception trees. We first provide a general description for our Monte Carlo sampling procedure including how we sample over chemical

perception trees with a scoring function based on the agreement of the sampled atom or fragment types with those of an existing force field .[85–88] Next, we detail two software packages for testing this procedure. The first is SMARTY, which learns chemical perception trees in the setting of traditional indirect chemical perception, such as the atom types found in an AMBER-family force field (Section 2.2). The second is SMIRKY, which learns chemical perception trees using direct chemical perception, such as the fragment types in a SMIRNOFF-format force field. Our description of SMIRKY focuses on those aspects which differ from SMARTY (Section 2.3). Both approaches are diagrammed in Figure 4. We then describe how we evaluate the ability of these methods to sample chemical perception trees of the chemical complexity found in the reference force fields. This analysis mirrors our ultimate goal of using a scoring function to measure the ability of a chemical perception tree, combined with suitable numerical parameters, to replicate a set of reference experimental and/or quantum chemical data. Finally, we describe the molecule sets used and details of simulations run to test both SMARTY and SMIRKY (Section 2.4).

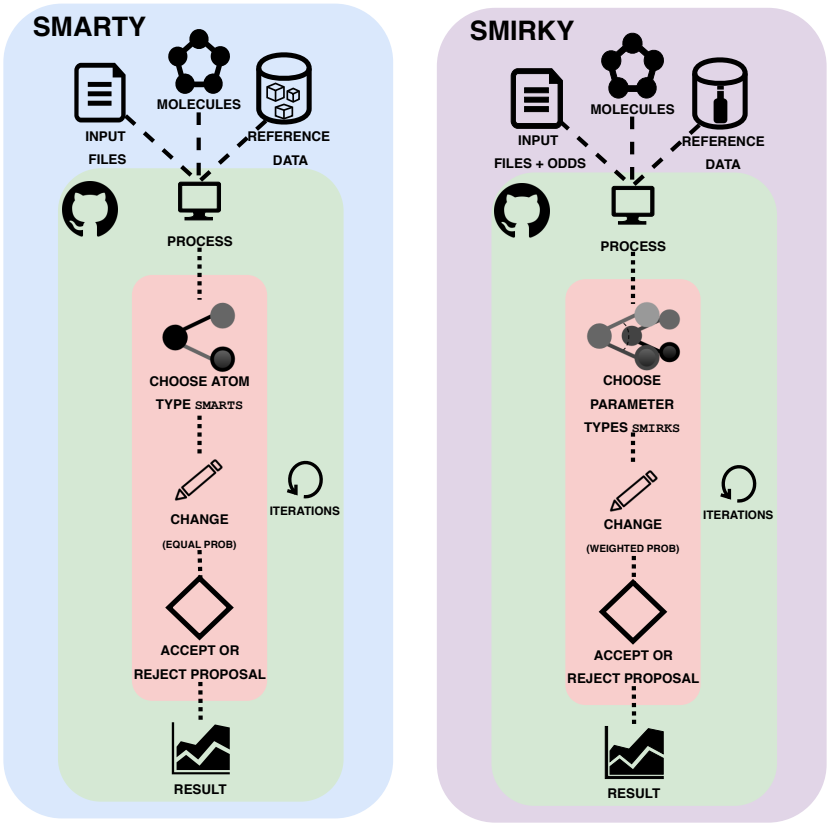


Figure 4. SMARTY and SMIRKY. Workflows for the SMARTY (left) and SMIRKY (right) tools are shown. There are three input data categories for SMARTY and SMIRKY (shown at top). In SMARTY (on the left), there are input files (such as base types, initial atom types, and decorators), molecules (input via a molecule set file), and reference data consisting of typed molecules (parm99/parm@Frosst atom typing was used in this work). SMIRKY has similar inputs, shown in the purple area on the right, and additionally allows the user to set the decorator odds and its reference data is fragment types (here, from smirnoff99Frosst). The algorithms are represented in the green area and are available on GitHub (https://github.com/openforcefield/smarty). Both tools begin by reading and processing the input files (top part of the green area). SMARTY (on the left) then conducts a series of moves (coral area) consisting of choosing one working atom type per iteration (icon with connected atoms), and deleting or modifying (pencil icon) this atom type, making choices made with equal probabilities (Section 2.2). SMIRKY (on the right) is similar, but samples over working fragment types (such as nonbonded types, bonds, angles, and proper and improper torsions; in the figure these are represented by two different icons with connected atoms) and uses weighted probabilities on their choices (Section 2.3). The acceptance criteria (diamond icon) for both tools are the same, using Equation 3 (Section 2.1.1). Both tools run a user-specified number of iterations and iterate over the steps in the coral area (repeat icon), then write out a final results file after all iterations are completed.

2.1 Monte Carlo sampling over chemical perception trees

We use the Metropolis Monte Carlo (MC) algorithm[89,90] to sample over chemical perception trees by making changes to a set atom or fragment types. We call this set of atom or fragment types being changed the "working types" and to be more specific we would say "working atom types" or "working fragment types". A single iteration of the algorithm comprises

1. choosing an atom or fragment type at random from the working set
2. proposing a move in which the type is either deleted (if it is not in the base set) or used as the starting point to create a new, more specific type
3. computing the change in a scoring function due to the proposed move
4. using the Metropolis criterion[89] to either accept or reject the proposed move, where the scoring function plays the role of the energy.

The effective temperature used in the Metropolis accept/reject decision and the desired number of iterations are user-specified inputs. Note that, if the user-defined temperature is zero, SMARTY and SMIRKY act as optimizers, only accepting moves which result in a higher total score, whereas at nonzero temperatures, it is possible for a move with a decreased score to be accepted.

2.1.1 The scoring function measures agreement of a chemical perception tree with that of an existing force field

For this proof of principle study, we developed a scoring function that quantifies the agreement of a chemical perception tree proposed in the course of MC sampling with the chemical perception assignments associated with an existing, operational force field. This scoring function compares how the working types categorize atom and fragments, with reference types from the existing force field. Our basic problem here is to determine which of our working types best corresponds to which reference type when applied to the same set of molecules. Here, in explaining the scoring function, we focus on atom types, but the same approach is also used for other fragment types arising from direct chemical perception, as considered at the end of this subsection.

We use a bipartite graph with a maximum weight matching[91,92] to score the working types (Figure 5). A bipartite graph is a graph with vertices divided into two disjoint sets, X and Y, where each edge only connects a vertex in X with a vertex in Y; that is, there are no X-X or Y-Y edges. Here, set X comprises the working atom types and set Y comprises the reference atom types. To compose a graph for scoring, we first process a set of molecules (Section 2.4), assigning every atom a working atom type and reference atom type. The set of working atom types in the molecules become vertices in set X and the set of reference atom types in the molecules become vertices in set Y (Figure 5(a)). The graph is initialized by construction of an edge of zero weight between each node in X and each node in Y (Figure 5(b)). Then, for each atom in each molecule, we identify its working and reference atom types and increment the weight of the edge joining the corresponding set X and set Y vertices by 1 (Figure 5(c)).

We then determine the set of X-Y edges that maximizes the number of atoms assigned the same working types that are also assigned the same types in the reference set. This corresponds to a maximal weight graph-matching problem.[93,94] A matching set in a graph is a subset of edges that are non-adjacent; i.e., no two edges share a common vertex.[94] Thus, determining the matching set ensures we never have the same working atom type connected by edges to two or more different reference types, and vice versa. For example, in Figure 5(c), we would select only one of the edges for the "`#1`" node. A maximal weight graph matching is one in which the sum of the weights of the retained edges is maximized. Thus, in Figure 5(c), we select only the edge of highest weight, here 224. More generally, a maximal weight graph match would include all possible working and reference types, such as the result for the three working types in Figure 5(d).

Given the resulting maximum weight graph matching, we use partial scores from each remaining edge and a total score for the graph to evaluate the set of working atom types. We define the *partial score* for each reference atom type as

$$S_i = \frac{N_{i,edge}}{N_{i,mol}} \tag{1}$$

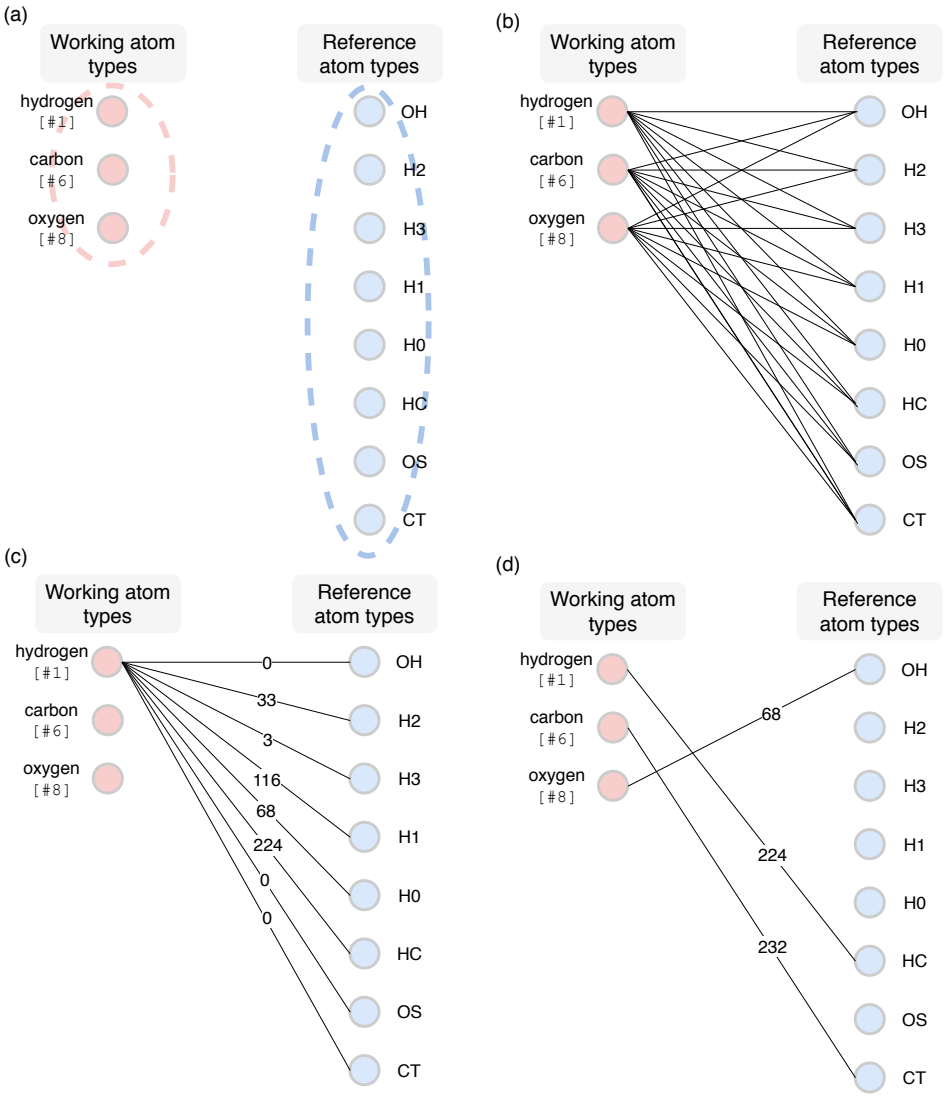


Figure 5. Illustration of the bipartite graph used to calculate the score for a new proposed move. (a) A bipartite graph is created with a node for each working (light red) and reference (light blue) atom type. (b) Edges (represented by lines) are created for all possible connections between the nodes of the two sets. Initially each edge has a weight of zero. (c) For each atom, the weight is incremented by one on the edge connecting that atom's working and reference atom types. In this figure, we illustrate the case for potential graph matches to hydrogen working atom type ("`#1`") using the AlkEthOH molecule set (Section 2.4). However, in practice this process is applied to all working atom types simultaneously. (d) Each node is then restricted to have only a single edge, such that the total weight is maximized.

where $N_{i,edge}$ is the weight of the edge associated with reference atom type i and $N_{i,mol}$ is the number of atoms in the molecule set with reference atom type i. Thus, $S_i = 1$ provides an assessment for how accurately the working atom type captures the chemistry of a specific reference atom type. The total score across types then is

$$S_{Total} = \frac{\sum_{i=1}^{n} N_{i,edge}}{N_T} \tag{2}$$

where n is the number of reference atom types and $N_T = \sum_i^n N_{i,mol}$ is the total number of atoms in the molecule set. Thus the MC acceptance criterion is given by

$$R < e^{\frac{S_{proposed} - S_{previous}}{T}} \tag{3}$$

where R is a random number from 0 to 1, $S_{proposed}$ and $S_{previous}$ are the graph match scores computed for the proposed and previous move, respectively, and T is the temperature assigned by the user.

The atom type scoring function just described pertains to the SMARTY method of sampling indirect chemical perception trees, which define only atom types. For the SMIRKY algorithm, which samples direct chemical perception trees, the atom type scores are supplemented with analogous scores for bond, angle and torsion types.

2.2 SMARTY is an algorithm for learning indirect chemical perception trees

The SMARTY algorithm defines its chemical perception tree using an ordered list of `SMARTS` strings, as described above, and manipulates these strings in to order vary and optimize this chemical perception tree using a selected scoring function. In this work, we use the graph-based scoring function defined in Section 2.1.1 to measure similarity to the atom types in a target force field. Ultimately, our initiative we will use a scoring function that reflects the ability of a force field using the working chemical perception tree to replicate experimental data. In this subsection, we describe the algorithm, including user specified inputs and move sets in `SMARTS` space.

2.2.1 User specified inputs allow for customization

The present SMARTY implementation takes five input files. The first is a file containing a set of `SMARTS` strings defining "base atom types" or base types, which will not be changed. These are typically the most generic type definitions. The second is a file containing "initial atom types," which form a superset of the base types. Typically, the initial types include not only generic base types, but also some more specific types, with the generic types listed before the specific types to define a hierarchical chemical perception tree. Initial types provide an initial configuration for the simulation in the search space of types. The third is a file listing the `SMARTS` atom decorators which will be used during sampling (see examples in Table 1). These three files are provided in a format where the first entry in each line is a complete `SMARTS` string or `SMARTS` decorator, and the second item is an informal name for the string or decorator. For example, "`[#7] nitrogen`" could be an entry in the initial type file, and "`X4 connectivity-4`" could be an entry in the decorator file. The other two input files relate to our specific test in this study, where we seek to determine how well sampled chemical perception trees can capture the typing of an existing force field (Section 2.1.1). Thus, our fourth input file provides the set of molecules to be typed in the process of evaluating a chemical perception tree, and the fifth contains the same molecules labeled with the atom types associated with the target force field.

After processing the user provided input, there is a preparation step before sampling can begin. The first step is to assign the base and initial atom types to the molecules via `SMARTS` chemical matching to substructures and to remove those base and initial types that match no atoms, in order to simplify the sampling problem. The remaining initial atom types become the *working set* of atom types. After the completion of this preparation phase, our atom type sampling works via the MC algorithm detailed above (Section 2.1). We next detail the specific move set used in SMARTY.

2.2.2 Move set for Monte Carlo sampling over atom type chemical perception trees

As illustrated in Figure 6, an MC move proposal either removes a non-base atom type from the working set of atom types or creates a more specific child type from an existing atom type, **A**. For creation of a new

child, **A'**, if the definition of **A** comprises only the primary atom, a decorator may be added to it; or an alpha substituent may be added. When new substituent is added, the type of bond connecting it, single, double, triple, or aromatic, is also chosen. If **A** already has an alpha substituent, then **A'** may be created by adding a decorator to **A**; by adding a second alpha substituent; or by adding a beta substituent any non-hydrogen alpha substituent. Finally, if **A** already has an alpha and a beta substituent, then a new decorator may be added to **A**, or a new alpha or beta substituent may be added. It is possible that in the future, move types might be added to allow the addition of decorations to alpha and beta substituents, but those moves are not included in our current SMARTY tests.

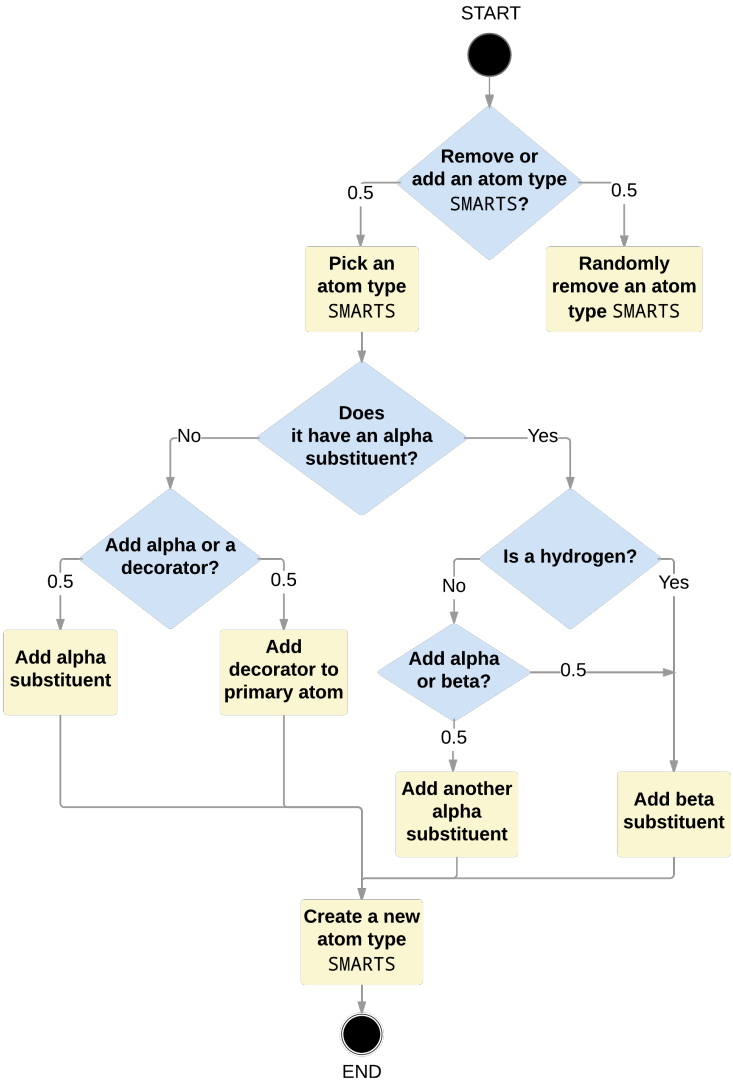


Figure 6. Flowchart illustrating the decision tree used for SMARTY move proposals. The numbers attached to certain arrows correspond to the probability of following that arrow; all moves are made with equal probability in SMARTY. The diamonds represent decisions and the rectangles are processes. We start the SMARTY move proposals with the working atom types and decide randomly with equal probability if we want to remove or add an atom type to the set. If we decide to remove, we randomly choose a working type to be removed and re-score the new set. If we decide to add, we pick a working atom type from the set and check if it has an alpha substituent. If the answer is no, we either add an alpha substituent or a decorator. If the answer is yes, we first check if the working atom type describes a hydrogen, and if yes, we add a beta substituent, if not, we either add an alpha or a beta substituent. The end result (black circle on the bottom) is a move proposal which is then evaluated via scoring function (Section 2.1.1).

Before the chemical perception tree resulting from the move is scored, it is subject to several validity

checks:

1. All atoms in the molecule set must be assigned an atom type from the proposed list.
2. A new child, **A'**, must match at least one atom in the molecule set
3. A new child **A'** must not be a duplicate any other atom type.
4. The parent atom type **A** of a new child must still match at least one atom in the molecule set, unless it is a base type.

Note that criterion 2 eliminates any atom type specifications that violate valence restrictions, such as a carbon with more than four bonds. If these criteria are met, the proposed atom type set is valid and can be scored (Section 2.1.1).

The SMARTY implementation also offers the option of allowing proposed moves only for atom types involving a single element. For example, one might allow deletion and child creation for only carbon atoms. This elemental SMARTY sampler, SMARTY$_{elem}$, avoids the combinatorial complexity that results when changes are allowed for all elements and thus can speed convergence to the global optimum.

2.3 SMIRKY is an algorithm for learning direct chemical perception trees

We recently argued that direct chemical perception[73] has major advantages over indirect chemical perception and illustrated the application of direct chemical perception via a prototype smirnoff99Frosst force field in new SMIRNOFF format.[95] The SMARTY algorithm (Section 2.2) is not adequate to sample over a direct chemical perception trees, because it samples only over atom types, and not the independent bond, angle, and torsion types used in direct chemical perception. Therefore, we have developed a second algorithm, SMIRKY, to sample over chemical perception trees corresponding to these fragment types. SMIRKY uses the same Metropolis MC method to sample over a chemical perception tree, and also uses an analogous scoring function (Section 2.1.1). The differences are the use of more general fragment types instead of atom types, and the resulting differences in the move set. SMIRKY also differs in that it uses of unequal probabilities for move proposals. We made this choice based on the unequal distribution of different `SMIRKS` decorators in SMIRNOFF fragment types. These unequal probabilities come from two sources — odds specified by user in the input files (Section 2.3.2) and odds specified in the SMIRKY move set (Section 2.3.3). Unlike SMARTY's elemental approach, we do not currently have a mechanism for reducing the combinatorial problem associated with sampling `SMIRKS` patterns by limiting the considered chemical space. We now present a new tool developed specifically to manipulate `SMIRKS` strings and then detail the SMIRKY input files and move set.

2.3.1 `ChemicalEnvironment` objects are used to parse complex `SMIRKS` patterns

We created the `ChemicalEnvironment` Python object which divides a `SMIRKS` string into atoms connected by bonds, and stores the decorators associated with each (Figure 7). This object also uses the atom indexing in the `SMIRKS` string to allow extraction of any indexed atom, any atom alpha or beta to an indexed atom, or any associated bond. The user can then easily modify the `ChemicalEnvironment` by adding neighboring atoms, removing neighboring atoms, or changing the list of decorators for a given atom or bond. Each `ChemicalEnvironment` is loaded and output as a `SMIRKS` string, so this object integrates with any other tool that can generate and interpret `SMIRKS`.

The `ChemicalEnvironment` object categorizes decorators on atoms and bonds based on the type of Boolean operator used to combine them; that is, the decorators are separated into OR and AND types. For example, consider `"[#6X3H2,#7X2H1;A+0:1]-[#1:2]"` in Figure 7, which is a `SMIRKS` pattern for a bond type, and hence has two indexed atoms. For atoms, OR types are composed of a base atom type (typically an atomic number) and a list of decorators to be combined via a logical OR ("`,`"). In this example, atom 1 has two OR types ("`#6X3H2`" and "`#7X2H1`") which are divided into OR bases ("`#6`" and "`#7`") and their corresponding decorators. For atoms, AND types are decorators that will be combined via a logical AND ("`;`"). Our example atom 1 has two AND decorators ("`A`" and "`+0`") which apply to both the carbon and nitrogen OR types. Bonds connecting atoms can also have OR and AND decorators. When no Boolean operator is used then the decorator is considered an OR type, so the bond in the present example, "–", would be considered an OR.

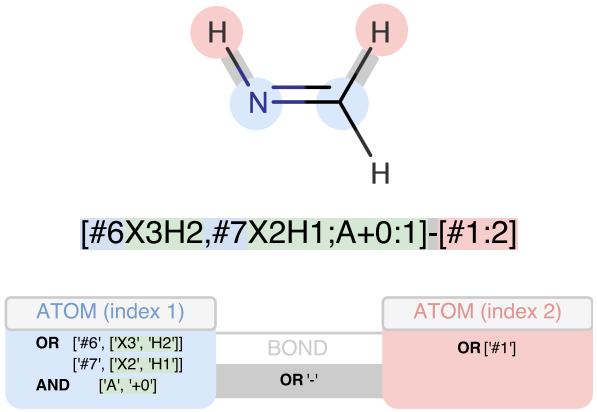


Figure 7. Mapping between `ChemicalEnvironment` and corresponding `SMIRKS` pattern. This illustrates a `SMIRKS` pattern "`[#6X3H2,#7X2H1;A+0:1]-[#1:2]`" (middle) for a substructure (top) being partitioned and how it would be stored in a `ChemicalEnvironment` object (bottom). The `SMIRKS` pattern (middle) represents a bond parameter "–" (gray) between two labeled atoms – a carbon (light blue) *or* nitrogen (light blue) atom and a hydrogen atom (light red). In this figure, we show two matches for this `SMIRKS` pattern on the top structure. The matches are highlighted in light blue (atom 1) and light red (atom 2) circles on the substructure (top). In the `ChemicalEnvironment` representation, at bottom, atom 1 (light blue rectangle), there are two ORtypes "`#6X3H2`" with a carbon atom base "`#6`" and OR decorators "`['X3, 'H2']`", corresponding to a connectivity of three and a total hydrogen count of one and "`#7X2H1`" with nitrogen atom base "`#7`" and OR decorators "`['X2, 'H1']`" (light green). The AND decorators for the atom 1 are "`['A', '+0']`" (light green) corresponding to an aliphatic atom with a zero charge. In `SMIRKS` strings, the high precedence "and" operator is given by a semi-colon "`;`", so the bond described by this pattern is one between a carbon atom with connectivity three and hydrogen count two or a nitrogen atom with connectivity two and hydrogen count one that is also aliphatic with zero charge.

2.3.2 SMIRKY inputs are like those for SMARTY but add user-selected move probabilities

The present SMIRKY implementation takes ten input files which allow users more customized control of probabilities while sampling. Three of these files are similar to SMARTY inputs: a file with initial fragment types (that is `SMRIKS` strings) defining a hierarchical chemical perception tree; a file containing molecules to be typed in the course of scoring the direct chemical perception tree; and a SMIRNOFF file which provides the reference fragment types for use in scoring. As in SMARTY, any of the initial fragment types that do not match any atoms are removed, creating a *working set* of fragment types for the MC algorithm. The next five files provide the atom, bond, and decorator types: OR atom types; OR and AND decorators for atoms; OR bond types; and AND decorators for bonds (Section 2.3.1). Each decorator file includes a column for the odds of proposing a given decorator in the course of the MC sampling. The final category of allow allows the user to specify the odds of picking a certain atom or bond in a fragment that will be changed during the move. There are two files in this category — one for atoms and one for bonds. There are two columns in this format, one specifying if the move is to an indexed, alpha or beta atom or bond using the `SMIRKS` index or the key words "alpha" or "beta." Then the second column is used to set the odds. For example, in a torsion, a user might want to make it more likely to make changes to the outside atoms (1 and 4) than to the inside atoms (2 and 3).

2.3.3 Move set for Monte Carlo sampling over fragment type chemical perception trees

SMIRKY uses the same general move set as SMARTY; that is, a fragment type **A** is chosen and then either deleted or used as the starting point for a child fragment type **A'** (Figure 8). The input odds files for SMIRKY allow the user to customize sampling for a specific fragment type, but the flows and probabilities governing internal decisions are independent of which fragment type is currently being sampled. When creating a child fragment type, the first choice made is whether to change an atom or a bond. If an atom is chosen, then decorators on the atom can be added, swapped, or removed; a new connected atom can be added as a neighbor; or if the atom selected is not an indexed atom it can be removed. (If it is indexed, then it must be retained in order for the fragment to be fully defined.) An atom can be removed from a fragment definition only if it is connected to just one other atom; the associated bond is removed at the same time. If a bond is

chosen instead of an atom, either type of decorator can be added, swapped, or removed. As with SMARTY, we only consider substructures that extend to the beta position of any indexed atom. Because there are more SMIRKS decorators available for atoms than bonds, the move set is weighted to choose atom moves more often than bond moves.

To allow efficient construction of complex SMIRKS patterns that make chemical sense, symmetric moves are also sometimes proposed. In symmetric moves, equivalent modifications are proposed simultaneously to *both* outer atoms in a bond, angle, or torsion. The probability of making a symmetric move is fixed inside SMIRKY. Specifically, the frequency of symmetric bond, angle, and torsion types in smirnoff99Frosst was used to assign the probability of making symmetric moves for each fragment type by category.

2.4 SMARTY and SMIRKY were evaluated using multiple molecule sets compared to reference force fields

We sought to determine whether our machinery could discover SMARTS or SMIRKS patterns that replicate the chemical complexity of types in an existing force field, thus providing a proof of principle for automating a step in force field development that has in the past been done only by hand. In order to measure chemical complexity we defined success based on the total and partial scores for the sample types compared to reference types (Section 2.1.1). A total score of 100% would indicate complete success for a set of working atom or fragment types. However, it is not necessary that we exactly reproduce the atom typing or fragment typing of existing force fields to succeed, especially in view of the combinatorial challenge we face; it is sufficient that we sample comparable chemical complexity. Thus, achieving a high total score on average would also be acceptable, especially if we can achieve at the same time a partial score of 100% for most atom or fragment types. That is, for both SMARTY and SMIRKY, we seek to recover SMARTS or SMIRKS which will mirror the indirect or direct chemical perception from the reference force field, by separating atoms or fragments into similar types. However, the automatically generated trees do not necessarily need to encode the exact same chemical perception trees as those from the reference force field.

To understand the evaluation of results, we consider the concrete example of sampling atom types for 1,3-dioxepane-2,4-diol, which has five different hydrogen atom types when typed with parm99/parm@Frosst:

- hydrogen bonded to oxygen (HO),
- hydrogen bonded to an aliphatic carbon (HC),
- hydrogen bonded to an aliphatic carbon with one electron withdrawing substituent (H1)),
- hydrogen bonded to an aliphatic carbon with two electron withdrawing substituent (H2)), and
- hydrogen bonded to an aliphatic carbon with three electron withdrawing substituent (H3)).

If SMARTY is successful, it will discover SMARTS strings that identify these types (Figure 3), starting from a chemically undistinguished initial hydrogen base type. Finding all five reference types simultaneously would give a 100% total score, but we would also consider discovering SMARTS patterns that receive a 100% partial score for all of these types during different parts of a simulation a success. Partial success might come from distinguishing four for the five types instead.

The chemical perception trees sampled in both SMARTY and SMIRKY were scored against the typing in existing force fields (Section 2.1.1). For SMARTY, we used a traditional indirect chemical perception force field specification as the reference, namely, AMBER's parm99[96] with the parm@Frosst[97] extension. For SMIRKY, we used the direct chemical perception force field smirnoff99Frosst,[95] which was generated by hand to closely follow the parameters in the parm99/parm@Frosst force field.

We developed three molecule sets to test SMARTY and SMIRKY. All three are included as images and molecule files in the Supporting Information.

For the first set, our goal was to have a small set of molecules with minimal complexity to test whether these tools could work as intended. Thus, this set was limited to molecules composed of carbon, oxygen, and hydrogen atoms, with only single bonds; that is, to alkanes, ethers, and alcohols and the compounds were drawn from the AlkEthOH compound set.[73] The specific set of compounds used here comprises 42 molecules, with a total of 803 atoms. We refer to this minimal set as AlkEthOH in our results.

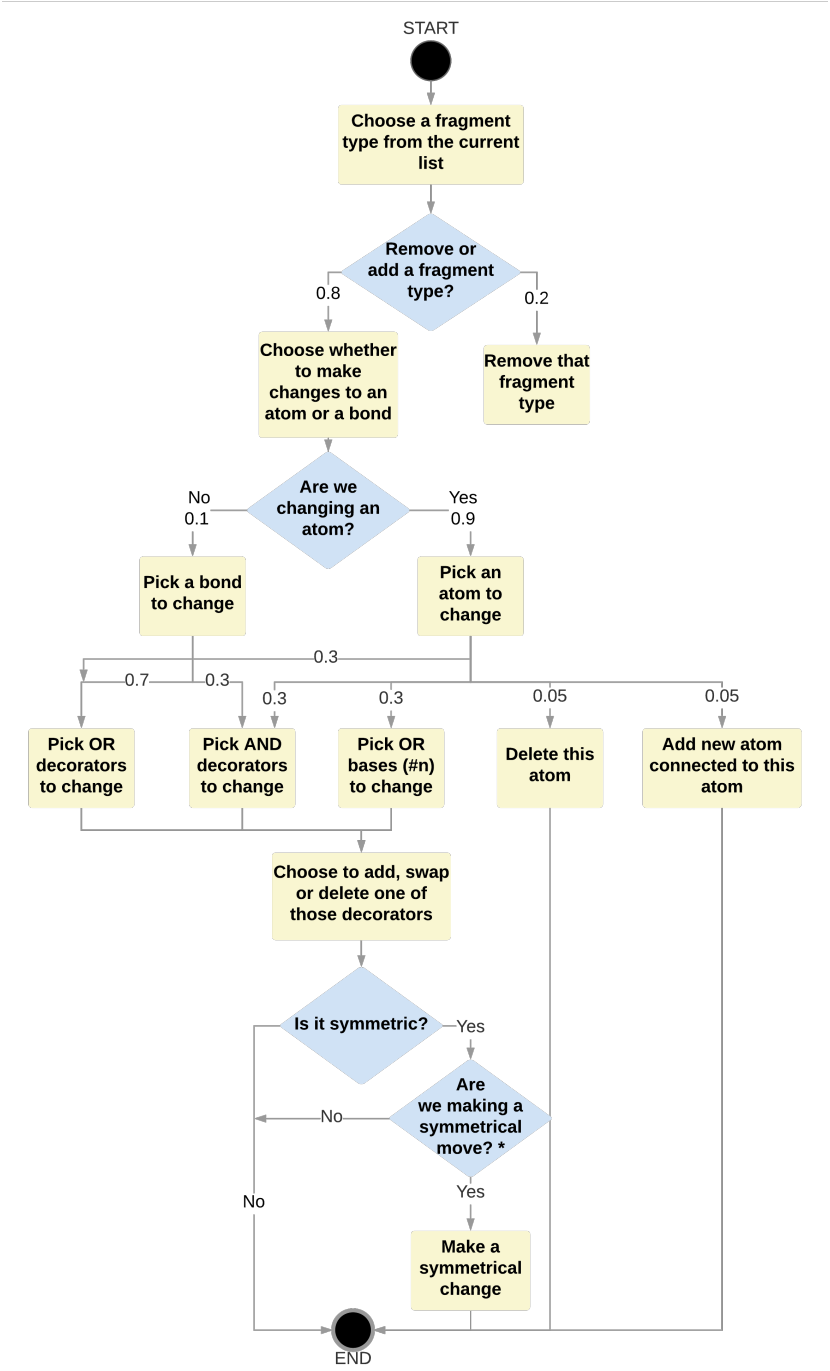


Figure 8. Flowchart to illustrate the decision tree used for SMIRKY move proposals. The numbers attached to certain arrows correspond to the probability of following that arrow, as currently implemented inside SMIRKY. The diamonds represent decisions and the rectangles are processes. We start the SMIRKY move proposals by choosing an initial fragment type `SMIRKS` from the set. SMIRKY then decides randomly with specified probabilities whether it will remove or add a fragment type `SMIRKS`. To add, the algorithm chooses between making a change to an atom or a bond. If the change is in a bond, SMIRKY picks OR or AND decorators to change, but if the change is in an atom, SMIRKY can pick OR decorator or OR base to change; and then choose between deleting, adding or swapping one of those decorators, always checking if it is symmetric and modifying it accordingly. Also, if SMIRKY chooses to change an atom, it can delete or add a new atom to this atom. The end result (black circle) is a move proposal which is then evaluated via the scoring algorithm described in Section 2.1.1.

The second set, PhEthOH, is the first set supplemented with aromatic compounds, and comprises 200 molecules with a total of 7185 atoms. Chemically, the difference in these sets is that PhEthOH includes aromatic rings, along with alkanes, ethers, and alcohols. AlkEthOH has one atom type not found in PhEthOH, namely `H3`, corresponding to a hydrogen bound to a carbon which is connected to three electron withdrawing groups. The aromatic groups in PhEthOH introduce two atom types not found in AlkEthOH — specifically, `CA` for aromatic carbons and `HA` for the hydrogens connected to those carbons. However, PhEthOH has more bond, angle, and fragment types than AlkEthOH.

The third set of molecules was obtained by filtering DrugBank, a free database of drug and drug-like molecules.[98–101] Molecules with fewer than three or more than 100 heavy atoms were removed, as were any molecules with metals or metalloids. Next, the molecules were assigned parm99/parm@Frosst atom types, and any molecule that could not be typed was removed. The molecules were then also typed with smirnoff99Frosst version 1.0.5 and any molecules assigned a generic parameter, for example *any bond* ("`[*:1]~[*:2]`"), were also removed. Finally, we selected a reduced set of compounds that include all atom and fragment types in the reference force fields used in DrugBank after this filtering. The final set, termed MiniDrugBank, has 371 molecules containing 15678 atoms, and is available separately on GitHub (https://github.com/openforcefield/MiniDrugBank).

2.4.1 Evaluation procedures

A series of tests were performed to evaluate success for each tool described above — the original SMARTY sampler (SMARTY_{orig}), the elemental SMARTY sampler (SMARTY_{elem}), and SMIRKY — in discovering chemical perception of the complexity in existing force fields. Each test was run on each of the three molecule sets described above, AlkEthOH, PhEthOH, and MiniDrugBank. For SMARTY_{elem}, tests were performed for each element with more than one atom type in the molecule set. Using the elemental sampler on elements with only a single atom type (such as halogens) would not provide useful insight since even a generic `SMARTS` pattern could match the relevant chemical perception. SMIRKY tests were performed for bond, angle, proper torsions, and non-bonded (vdW) fragment types. While the SMIRNOFF format and SMIRKY support improper torsions, there are a very limited number in smirnoff99Frosst so those were not tested.

MC sampling was run at eight effective temperatures: 0, 10^{-6}, 10^{-5}, 0.0001, 0.001, 0.01, 0.1, and 1. Ten initial tests of 10,000 iterations were performed for each combination of molecule set, temperature, and tool. These were followed by three additional tests with first 50,000 and then 100,000 iterations for SMIRKY, since the search space for fragment types is so large. Output for both tools was saved in two ways: a human-readable log of the run, and a comma-separated trajectory file which stores the partial score for each reference atom or fragment type and the total score at each iteration.

Output files and plots of score versus time for all simulations along with analysis scripts used for the results shown here can be found in the Supporting Information. Source code and usage examples for SMARTY and SMIRKY can also be found on our GitHub repository (https://github.com/open-forcefield-group/SMARTY).

3 Results and Discussion

A central goal of this study is to test the ability of the SMARTY and SMIRKY sampling methods to discover type definitions closely matching those in existing force fields. As detailed above, if successful, we would recover `SMARTS` and `SMIRKS` patterns matching all atom or fragment types, by achieving total scores of 100% or partial scores of 100% for all reference types (Section 2.4). However, we would consider high total scores along with a 100% partial score for most reference types to also indicate success, showing promise for expanding these algorithms for sampling chemical perception trees beyond the proof of principle in this paper. Both methods were tested on three molecule sets of increasing complexity — AlkEthOH, PhEthOH, and MiniDrugBank (section 2.4).

3.1 SMARTY

We performed 10 SMARTY runs, of 10,000 steps each, for each of the three molecule sets, at effective temperatures of 0, 10^{-6}, 10^{-5}, 0.0001, 0.001, 0.01, 0.1, and 1, using both the original SMARTY sampler

(SMARTY$_{orig}$) and the elemental sampler (SMARTY$_{elem}$). The results of these calculations are analyzed in the following two subsections.

3.1.1 High scores are achieved for the AlkEthOH and PhEthOH molecule sets

We first tested SMARTY on the AlkEthOH and PhEthOH sets. Although these have only eight atom types, some of the five hydrogen types used in these sets require multiple atoms in the beta position, so they still provide a significant challenge. For the AlkEthOH set run at effective temperatures between 10^{-6} and 10^{-2}, the original sampler, SMARTY$_{orig}$, yielded total scores, averaged across the 10 runs, of $> 90\%$ (Table 2), and the lowest average score is still fairly high, at 69%. The degradation of performance at lower and higher temperatures is reasonable for the MC method, as low temperatures can lead to trapping in local maxima, while high temperatures can lead to a lower preference for favorable states. Figure 9 provides sample plots of total score against step number for MC runs at various temperatures. As expected, the higher temperatures lead to greater score fluctuations. Equivalent results were obtained for the PhEthOH set (Table 2), which has a similar number of atom types (Section 2.4). Overall, these results strongly support the ability of SMARTY to sample chemically relevant ensembles of atom types.

	Average Score (%)*		
Temperature	**AlkEthOH**	**PhEthOH**	**MiniDrugBank**
1.0	68.9 ± 0.1	56.9 ± 0.1	43.0 ± 0.1
0.1	75.3 ± 0.1	62.2 ± 0.2	45.7 ± 0.2
0.01	94 ± 1	88 ± 1	67 ± 1
0.001	97.5 ± 0.8	96 ± 1	77 ± 1
0.0001	97.9 ± 0.9	95 ± 1	78 ± 2
1e-05	98.5 ± 0.7	95 ± 2	70 ± 2
1e-06	97.4 ± 0.8	95 ± 1	74 ± 2
0	89 ± 3	73 ± 4	72 ± 2

Table 2. Average of average total scores of 10 SMARTY simulations with AlkEthOH, PhEthOH, and MiniDrugBank. We took the average score at each temperature for each of our 10 simulations (10,000 iterations each) using SMARTY$_{orig}$, then averaged this across all 10 simulations.

* uncertainties were estimated by the standard error over all 10 trials.

Analysis of the partial scores offers a finer-grained view of these overall results. Thus, we made heat maps showing the frequency of generating a partial score of 100% for each reference atom type (Figure 10), at the various simulation temperatures. These show that some reference atom types are easier to discover than others. This can be for multiple reasons, including because they have simpler specifications, and because they occur more frequently. Also, although low to moderate temperatures yield the best results for most atom types, higher temperatures worked better for atom type HC, at least for the PhEthOH molecule set.

Figure 11 shows a sample working atom type hierarchy tree (top panel) that was generated during a SMARTY simulation, with levels of the hierarchy represented by different colors. The bottom panel shows which reference atom types are matched by each `SMARTS` pattern, and the partial score for each match. It is worth noting in this regard that SMARTY's hierarchical approach to sampling dramatically simplifies the complex search problem. For example, in the parm99/parm@Frosst force field for AlkEthOH, there are four atom types for hydrogens bound to carbon, differing in the number of electron withdrawing groups bound to the carbon atom (i.e., in the beta position). As mentioned before, these hydrogens provide a significant challenge to SMARTY. With hierarchical sampling, SMARTY can first discover less specialized `SMARTS` and then proceed to more complex derivative types further down the tree (Figure 11). SMARTY would not be able to find `H1` ("`[#1$(*-[#6]-[#8])]`"), a carbon with one electron withdrawing group, without first discovering `HC` ("`[#1$(*-[#6])]`"), because moves are made one atom at a time. This also highlights the importance of the hierarchy when matching; if the more complex child `SMARTS` patterns were placed above their parents, then

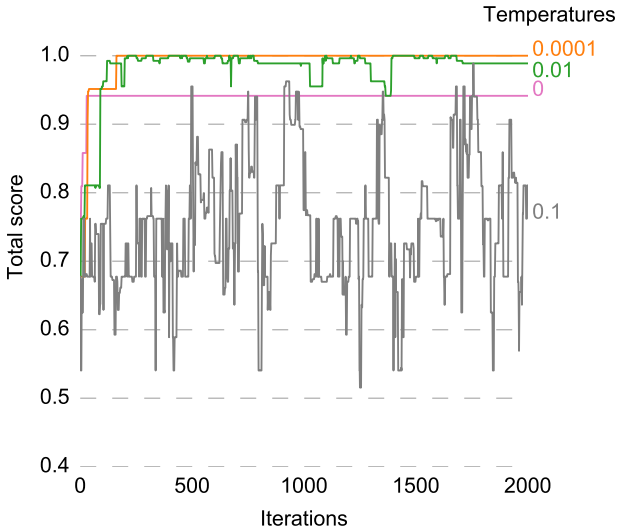


Figure 9. Total score as a function of iteration for SMARTY$_{orig}$ sampling AlkEthOH. The graph shows the total score (fraction of atoms matching the parm99/parm@Frosst reference atom types) as a function of iteration varies with the temperature for SMARTY$_{orig}$ sampling AlkEthOH. The lines in the graph represent the temperatures 0 (pink), 0.0001 (orange), 0.01 (green) and 0.1 (gray). We can see a high variation throughout the simulation and a predominance of lower scores for extreme temperatures such as 0 and 1.0. At intermediate temperatures, such as 0.01 and 0.0001, SMARTY scored higher. The Supporting Information includes plots for simulations at all temperatures.

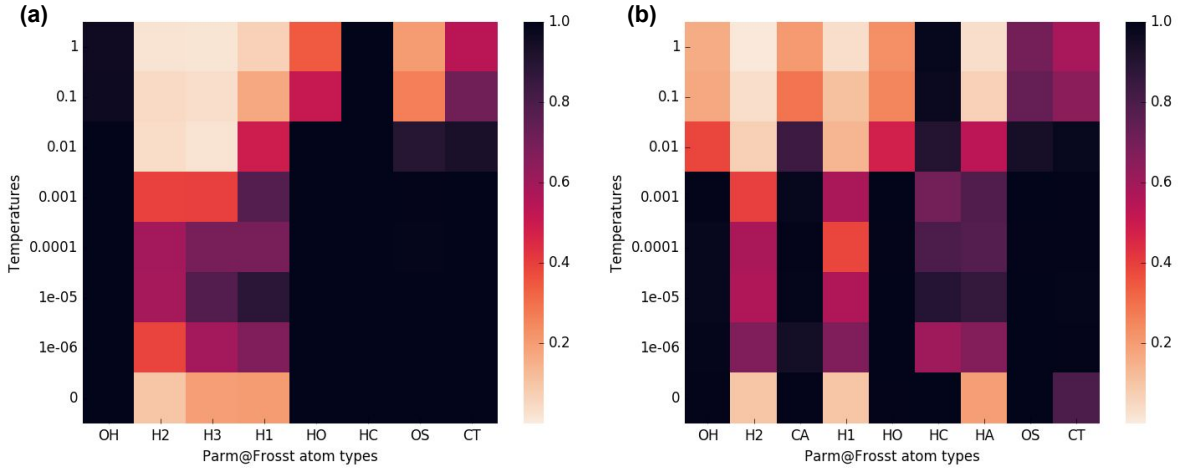
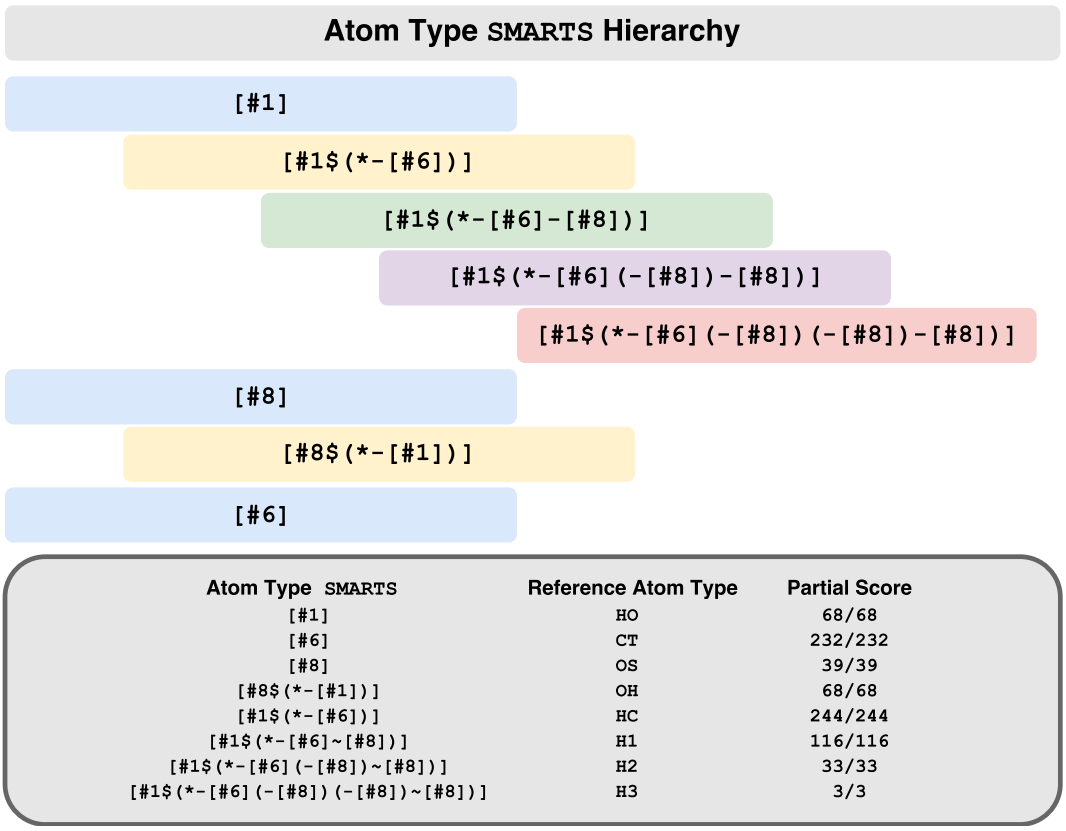


Figure 10. Frequency of success for SMARTY$_{orig}$ on the AlkEthOH and PhEthOH sets. Heat maps show the results of SMARTY$_{orig}$ for the AlkEthOH (A) and PhEthOH (B) molecule sets. We plot the fraction of simulation time that each reference atom type (x axis) is matched by a `SMARTS` pattern with a partial score of 100%. We tested eight different temperatures (y axis) and ran 10 simulations of 10,000 iterations each. Darker colors imply a higher rate of matching 100% of that reference atom type; see color bars at right. If we never found the reference atom type, the corresponding cell would be white, but this does not occur here.



Atom Type SMARTS	Reference Atom Type	Partial Score
[#1]	HO	68/68
[#6]	CT	232/232
[#8]	OS	39/39
[#8$(*-[#1])]	OH	68/68
[#1$(*-[#6])]	HC	244/244
[#1$(*-[#6]~[#8])]	H1	116/116
[#1$(*-[#6](-[#8])~[#8])]	H2	33/33
[#1$(*-[#6](-[#8])(-[#8])~[#8])]	H3	3/3

Figure 11. Example of a SMARTY hierarchy and inheritance in AklEthOH. This figure shows the final hierarchy achieved in one SMARTY simulation with the AlkEthOH set at $T = 0.0001$ after 10,000 iterations. The top shows the hierarchy of SMARTS strings discovered where each color represents a different level. For example, the initial types are represented in light blue, and the yellow squares are the working atom types derived from the initial types; the same logic applies for the other colors. For hydrogen, we see SMARTS patterns that match the parm99/parm@Frosst reference types HC (yellow), H1 (light green), H2 (lilac), and H3 (light red). The bottom (large gray box) shows all final working types and their respective reference atom types as well as partial score for that pairing.

all hydrogen atoms would be assigned to the more general "[#1]" type; the more complex pattern is only valuable when it is a child of and placed below the less complex pattern.

This point is further illustrated by considering the additional hydrogen types used in AlkEthOH. The hydrogen atom types used by parm99/parm@Frosst are relatively complex because previous work concluded the nonbonded parameters for these atoms *need* to be different,[102] and thus SMARTS patterns to match these must also be fairly complex. Specifically, these atom types require SMARTS strings extending out far enough to describe multiple atoms in the beta position. If SMARTY did not use SMARTS patterns already generated as parent types, finding these complex patterns would be impossible. Here, the relevant atom types cover hydrogen attached to carbon with zero (HC), one (H1), two (H2) or three (H3) electron withdrawing groups (Figure 3). For example, a SMARTS pattern which can recognize H3 (in AlkEthOH as shown in Figure 3 in red) is "[#1$(*-[#6](-[#8])(-[#8])-[#8])]". But arriving at H3 directly from HC ("[#1$(*-[#6])]") would be impossible without very complex move proposals (yellow to red in Figure 3). Instead, a move in SMARTY could use HC as a parent type and propose a move to create H1. A subsequent move could take H1 as a parent and generate H2 as a child then the same for creating H3 from H2. Of course, this will only work if uncovering each new intermediate type provides some incremental increase in total score, which it does here. Our assumption is that the chemistry of more diverse sets can be described in a similar way, starting with the most simple (or generic) SMARTS and then extending that pattern only where necessary.

The analysis so far has focused on the original SMARTY algorithm, which samples over all elements simultaneously. This approach is comprehensive, but leads to a combinatorial sampling problem. Imagine

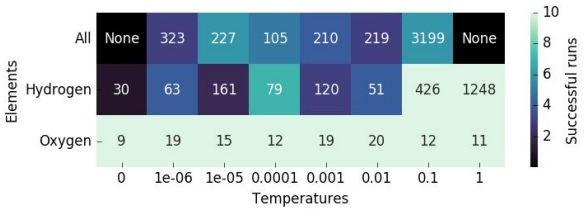


Figure 12. Comparison of SMARTY$_{elem}$ (hydrogen and oxygen) versus SMARTY$_{orig}$ (all atoms) on AlkEthOH. Here we determine the first iteration at which each sampler achieves a 100% total score (numerical values, out of 10,000 in total), then average this number for all runs which were successful in achieving a score of 100% at some point (out of 10 trials in total). The y axis shows which sampler was used (SMARTY$_{orig}$ for all atoms versus SMARTY$_{elem}$ for hydrogen and oxygen) and the x axis shows the temperature employed. In general lighter colors indicate higher success rates, and lower numbers indicate success was achieved more rapidly. Results for SMARTY$_{elem}$ for carbon is not shown since AlkEthOH has only one carbon atom type (`CT`).

---

we want to find the hydrogen atom types in AlkEthOH. The most complex of these is `H3`, which requires specifying one atom in the alpha position and three atoms in the beta position (Figure 11). When adding either an alpha or beta substituent, the probability of choosing a specific base type (i.e. correct element) is approximately 0.15. This means it could potentially take SMARTY more than $0.15^{-4} \approx 1,000$ moves to find `H3` when starting with a hydrogen parent type. Including the necessity of choosing the correct parent element decreases the probability of each move to 0.015, meaning it could take SMARTY on the order of 10,000,000 moves to generate the correct `SMARTS` pattern for `H3`. The elemental SMARTY method, SMARTY$_{elem}$, speeds the search by sampling only one element (base type) at a time. Thus, fewer iterations are required to first discover a `SMARTS` with a total score of 100%, as shown for AlkEthOH in Figure 12.

3.1.2 MiniDrugBank provides a more difficult molecule set for testing SMARTY

The MiniDrugBank molecule set was drawn from the DrugBank database, and contains a similar diversity of atom types in a smaller number of molecules (Section 2.4). The total number of reference atom types represented in MiniDrugBank, 37, is considerably larger than the number of reference atom types in AlkEthOH or PhEthOH, making this a more challenging test case. As a consequence, the average total scores run lower, often at 70-80%, than in the case of the simpler test sets (Table 2). We therefore focus further evaluation on the the elemental SMARTY sampler (SMARTY$_{elem}$), which reduces the combinatorial explosion of types and thus is particularly suitable for bigger and more complex molecule sets. Accordingly, the evaluation of these results is in terms of partial scores, rather than total scores (Section 2.4).

We find that SMARTY$_{elem}$ is capable of finding `SMARTS` patterns for the majority of reference atom types (Figure 13). Overall, SMARTY is more likely to match 100% of *all* reference atom types at relatively high temperatures. When we decrease the temperature, SMARTY's behavior changes and no longer finds many reference atom types (shown by white in the heat maps), but it matches 100% of some atom types more frequently (darker colors), until the temperature becomes too low.

SMARTY is able to generate multiple different `SMARTS` patterns matching typical reference atom types in MiniDrugBank. In 10 runs of 10,000 iterations each, we found 16,564 unique `SMARTS` strings which match at least some fraction of any reference atom type. When we look at the `SMARTS` patterns which match 100% of any reference atom type, the number is still rather large, at 801 unique `SMARTS` strings. This was true even for simple atom types, such as `CM`, which is represented by "`[#6X3]`" in parm99/parm@Frosst. For `CM`, SMARTY found 24 different `SMARTS` strings that matched 100% of this reference atom type. Some of these 24 `SMARTS` are very generic, such as "`[#6]`", others more complex ("`[#6X3$(*-[*])$(*=[#6])]`"). Our goal was not to replicate exactly the same `SMARTS` patterns written by human experts, but instead to determine what extent automated sampling can capture equivalent – or at least similar – chemical perception. SMARTY's ability to discover a wide diversity of `SMARTS` patterns corresponding to each reference atom type appears to demonstrate success in this regard.

When working types have only very generic `SMARTS` pattern they will match the most populated reference atom type due to our typing scheme and scoring function. The scoring function uses a bipartite graph where

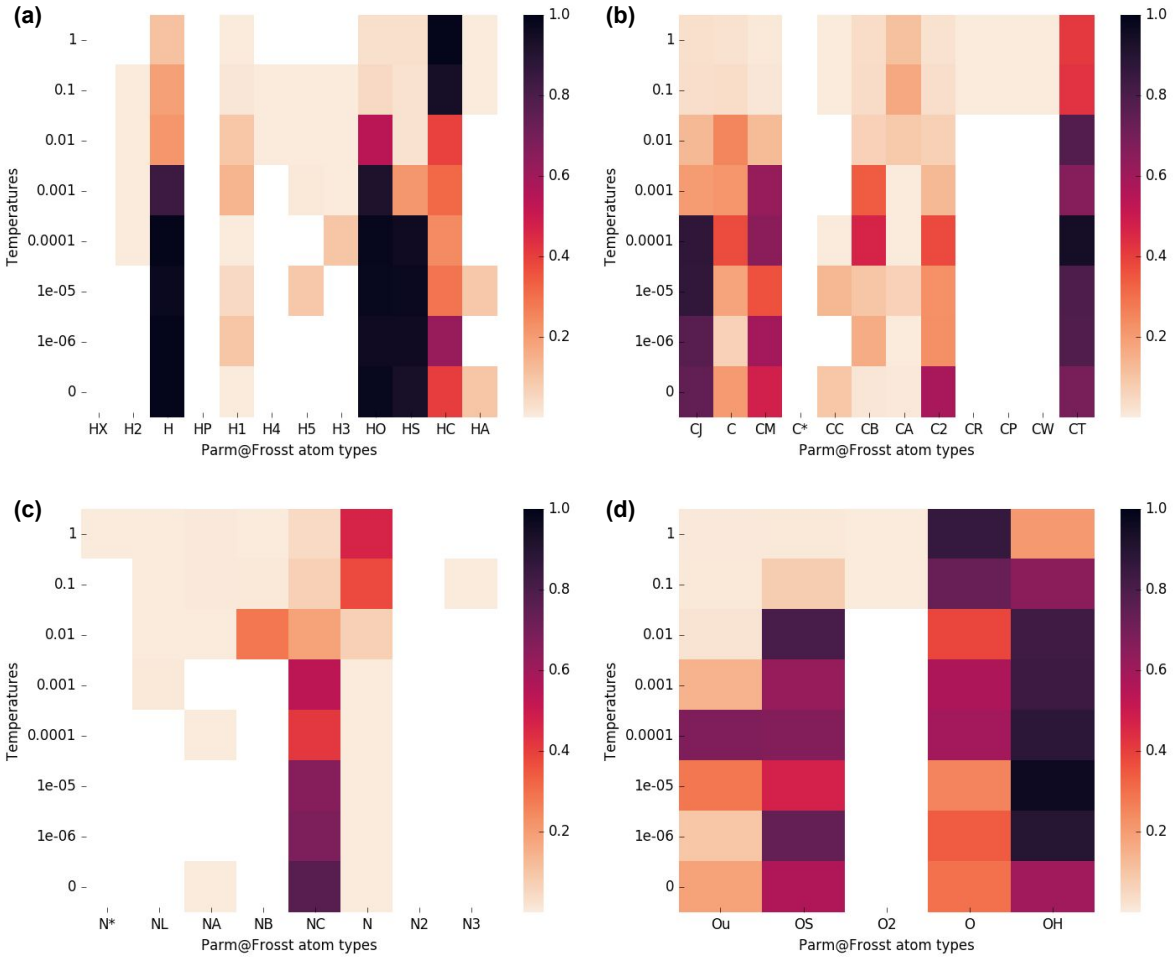


Figure 13. Frequency of success for SMARTY$_{elem}$ on the MiniDrugBank set. For SMARTY$_{elem}$ for MiniDrugBank, we plot the fraction of simulation time where the partial score is 100% for a given parm99/parm@Frosst atom type. The elements shown here are (a) hydrogen, (b) carbon, (c) nitrogen, and (d) oxygen. We used 10 simulations of 10,000 iterations each to construct the plots at different temperatures (y axis). At high temperatures, most of the reference atom types reach 100% partial score. However, at low temperatures, there are more well populated reference atom types but also more which never achieve 100% at any iteration.

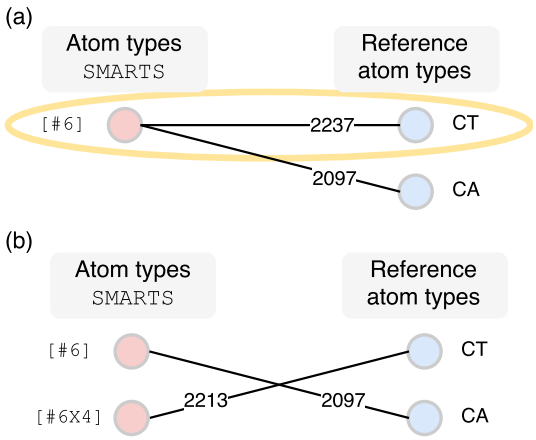


Figure 14. SMARTY scoring with the bipartite graph. We illustrate how SMARTY scores working atom types (red on the left) compared to reference atom types (blue on the right) using a bipartite graph. (a) When there is only one generic `SMARTS` string ("`[#6]`"), this pattern is assigned to both `CT` and `CA` atom types through the PATTY-adapted scheme, but it will be matched only to the more populated reference atom type (`CT` highlighted in yellow); (b) when SMARTY creates a new set of working types, containing both "`[#6]`" and "`[#6X4]`", the generic atom type will not match `CT` anymore, since it chemically matches the new working atom type ("`[#6X4]`") instead.

the matching is based on the maximum total weight and the weight on each edge is determined by the number of atoms assigned that working and reference type (Section 2.1.1). For this reason, a reference type with the highest population is more likely to be matched with a generic working type. To understand the implications of this, consider the frequency of two of the carbon types in MiniDrugBank, `CT` and `CA`. The parm99/parm@Frosst atom type `CT` is assigned to 2213 atoms, while `CA` is used for 2097. If we have only one carbon `SMARTS` string, such as the generic one "`[#6]`", all carbon atoms in the molecule set will be assigned as "`[#6]`", and after scoring, the `SMARTS` pattern will match the reference atom type `CT` because it leads to the highest total score. In that case, `CA` (and all other carbon atom types) are not matched to any `SMARTS`. But, if we discover a new working atom type that matches `CT` only, such as "`[#6X4]`", then the generic `SMARTS` will match `CA` (Figure 14).

For MiniDrugBank we were only partially successful; we were not able to find a 100% partial score for the `HX`, `N2`, `C*`, and `HP` atom types with SMARTY. It is worth briefly examining these types to understand why. `HX` was created to address a very specific problem occurring with hydroxyl hydrogens; to match it, a SMIRKS pattern would need to describe an atom in the gamma position (3 bonds away from the primary), but SMARTY sampling extends out only to the beta position. `N2` requires multiple `SMARTS` patterns to be able to match 100% (it combines several different chemical groups into a single atom type) and the bipartite graph matching only allows one working atom type per reference atom type, meaning that SMARTY will never discover it. `C*` is very similar to other five-membered ring carbon atom types (`CW`, `CC`, and `CR`) in our tests; it also occurs less frequently which would require SMARTY to generate and keep highly specialized `SMARTS` patterns that match the other atom types uniquely before it could be discovered. In order to find `HP`, the `SMARTS` pattern would need a decorator in the beta position, but SMARTY does not currently add decorators to the beta position making it impossible to match this particular atom type. More details about all of these atom types and why they never get a partial score of 100% can be found in the Supporting Information.

Nevertheless, SMARTY was fairly successful finding a 100% partial score for most reference atom types in parm99/parm@Frosst with only a few exceptions. In our tests, we were able to find patterns matching 33 out of 37 parm99/parm@Frosst atom types with a 100% partial score – overall an 89.2% success rate. While some reference atom types are not found, this seems to be less a commentary on our chosen sampling algorithm and move set; but a byproduct of the complex human decisions made in constructing parm99/parm@Frosst. The very complex atom typing employed in parm99/parm@Frosst may provide a further argument in favor of moving toward force field with direct chemical perception and away from those with human defined atom types.

3.2 SMIRKY

SMIRKY extends SMARTY, testing our ability to sample over chemical perception trees used for fragment types in SMIRNOFF format force fields. It automatically samples SMIRKS patterns for fragment types for bond, angle, torsion, and nonbonded types then compares the fragment type SMIRKS with that in a reference force field for the same set of molecules. As with SMARTY, the overall goal with SMIRKY is to ensure we can find SMIRKS patterns that sample comparable chemical complexity to that used in our reference force field, smirnoff99Frosst. Scoring works the same as with SMARTY—we evaluate both the total score and the partial score for each iteration. In the case of SMIRKY, a partial score of 100% corresponds to SMIRKY discovering a SMIRKS pattern that matches a single reference fragment type (Equation 1).

With SMIRKY, we were at least partially successful with all molecule sets as evidenced by our ability to find 100% partial scores for the majority of fragment types in smirnoff99Frosst. Initially, we performed simulations of comparable length with SMIRKY as with SMARTY; however, increased complexity made longer simulations necessary. Specifically, we began by performing 10 simulations with 10,000 iterations for each fragment type at all eight temperatures (0, 10^{-6}, 10^{-5}, 0.0001, 0.001, 0.01, 0.1, 1). However, because SMIRKY goes beyond atom types to fragment types, it necessarily must sample more complex chemical perception trees. For example, torsion fragment types require at least four atoms and potentially substituents as well; decorators may also be required for all these atoms. This results in SMIRKS patterns which involve four or more atoms (three or more bonds) all with decorators—a challenging problem. In SMARTY, the combinatorial problem of discovering sufficiently complex SMARTS patterns was tackled by creating the elemental sampler in order to limit the chemical space being searched. But with SMIRKY, we cannot currently sub-divide the chemical space for these more complex fragments (Section 2.3), so we instead increased the number of iterations. For the most complex molecule set, MiniDrugBank, we increased the number of iterations to 50,000 and then 100,000. In each case, we performed three simulations at each temperature for each fragment type.

Fragment Type	AlkEthOH	PhEthOH	MiniDrugBank
Bond	5/5	10/10	68/73
Angle	4/4	6/6	32/34
Torsion	11/11	16/16	117/136
Nonbonded	8/8	9/9	25/26

Table 3. Number of fragment types with 100% partial score in all SMIRKY simulations. This table summarizes results for all SMIRKY simulations. For each molecule set, we report the fraction of reference fragment types that get a 100% partial score during at least one simulation. It shows that we were successful in identifying all fragment types in the AlkEthOH and PhEthOH sets where all fractions are equal to one. As with SMARTY, we were still partially successful for MiniDrugBank given that we find the majority of fragment types.

3.2.1 SMIRKY is able to recover all smirnoff99Frosst types in AlkEthOH

As with SMARTY, AlkEthOH provides a useful toy example for testing our methods, and SMIRKY is successful based on total scores for all fragment types. In smirnoff99Frosst, there are nonbonded parameters corresponding to the same hydrogen parm99/parm@frosst atom types discussed above (Section 3.1.1). To distinguish all of these types, multiple beta position atoms must be identified. Thus, AlkEthOH is a good test set, although there are only a few fragment types in each category. SMIRKY discovers fragment type SMIRKS for all five bond, four angle, 11 proper torsion, and eight nonbonded fragment types required to type AlkEthOH with smirnoff99Frosst (Table 3). SMIRKY is able to generate sets of SMIRKS patterns which achieve a total score of 100% for all smirnoff99Frosst fragment types in AlkEthOH. An average total score of $> 90\%$ at multiple temperatures provides further evidence of SMIRKY's success (Table 4).

While SMIRKY is quite successful in general, its success is not universal at all temperatures. At moderate temperatures, SMIRKY is able to discover SMIRKS patterns complex enough to agree with the fragment typing used in smirnoff99Frosst, achieving a successful total score of 100%. But, as is expected with an MC

	AlkEthOH Average Score (%)*			
Temperature	**Bond**	**Angle**	**Torsion**	**Nonbonded**
1	86.7 ± 0.3	67.4 ± 0.3	57.4 ± 0.4	58.6 ± 0.2
0.1	91.8 ± 0.2	77.2 ± 0.4	68.8 ± 0.6	66.1 ± 0.2
0.01	99.66 ± 0.07	94.0 ± 0.4	90.9 ± 0.8	89 ± 1
0.001	99.95 ± 0.01	97.4 ± 0.7	97.7 ± 0.4	91 ± 2
0.0001	99.92 ± 0.03	97.9 ± 0.7	98.3 ± 0.4	93 ± 1
1e-05	99.93 ± 0.03	97.9 ± 0.7	97.3 ± 0.9	89 ± 2
1e-06	99.94 ± 0.02	97.9 ± 0.6	98.3 ± 0.4	87 ± 2
0	99.94 ± 0.02	94.6 ± 0.9	93 ± 1	76 ± 1

Table 4. Average of average total scores of 10 SMIRKY simulations with AlkEthOH. We took the average score at each temperature for each of our 10 simulations (10,000 iterations each), then averaged this across all 10 simulations.

* uncertainties were estimated by the standard error over all 10 trials.

algorithm, we achieve relatively poor scores both at $T = 0$ and at the high temperature of 0.1. However, at the intermediate temperature of 0.001, SMIRKY eventually generates torsion `SMIRKS` with a 100% score and maintains that high score (Figure 15). In addition to the examples depicted here, multiple simulations at the temperatures of 10^{-6}, 10^{-5}, and 0.0001 also reached a 100% total score. The fact that multiple temperatures reach a total score of 100% indicates we have been successful in automatically sampling the chemical perception trees for torsion parameters in AlkEthOH. Details for all simulations are included in the Supporting Information.

3.2.2 PhEthOH demonstrates the complexity of SMIRNOFF parameters

PhEthOH adds a small amount of complexity relative to AlkEthOH, including aromatic rings in addition to alkanes, ethers, and alcohols. It requires a total of 10 bond, six angle, 16 torsion, and nine nonbonded smirnoff99Frosst parameters. As with AlkEthOH, we performed 10 SMIRKY simulations at each temperature for each fragment type. This small increase in complexity also increased the difficulty of matching all reference types simultaneously, as shown by the drop in average total score for all fragment types (Table 5). For bonds, angles, and nonbonded fragment types, SMIRKY is able to achieve a 100% total score; however, the same could not be said for torsions. However, SMIRKY is able to find a 100% partial score for all torsions at most temperatures (Table 3).

	PhEthOH Average Score (%)*			
Temperature	**Bond**	**Angle**	**Torsion**	**Nonbonded**
1	80.0 +/- 0.4	61.3 +/- 0.3	49.8 +/- 0.5	46.8 +/- 0.2
0.1	85.0 +/- 0.3	69.5 +/- 0.5	55.5 +/- 0.5	54.4 +/- 0.2
0.01	97.5 +/- 0.1	88.5 +/- 0.9	83.3 +/- 0.7	83 +/- 2
0.001	99.73 +/- 0.06	94.8 +/- 0.4	91.1 +/- 0.9	91 +/- 2
0.0001	99.78 +/- 0.04	95.5 +/- 0.5	93.0 +/- 0.5	90 +/- 3
1e-05	99.83 +/- 0.03	95.8 +/- 0.4	91.3 +/- 0.8	85 +/- 3
1e-06	99.83 +/- 0.04	96.8 +/- 0.6	92 +/- 1	90 +/- 3
0	99.7 +/- 0.1	92.8 +/- 0.8	87 +/- 2	71 +/- 3

Table 5. Average of average total scores of 10 SMIRKY simulations with PhEthOH. We took the average score at each temperature for each of our 10 simulations (10,000 iterations each), then averaged this across all 10 simulations.

* uncertainties were estimated by the standard error over all 10 trials.

A 100% partial score is evidence that SMIRKY can generate the `SMIRKS` of the same chemical complexity found in the smirnoff99Frosst. As with SMARTY, we examine the partial score for each reference torsion type and consider the fraction of iterations spent at 100% (Figure 16). For all but one reference torsion type (`t2`),

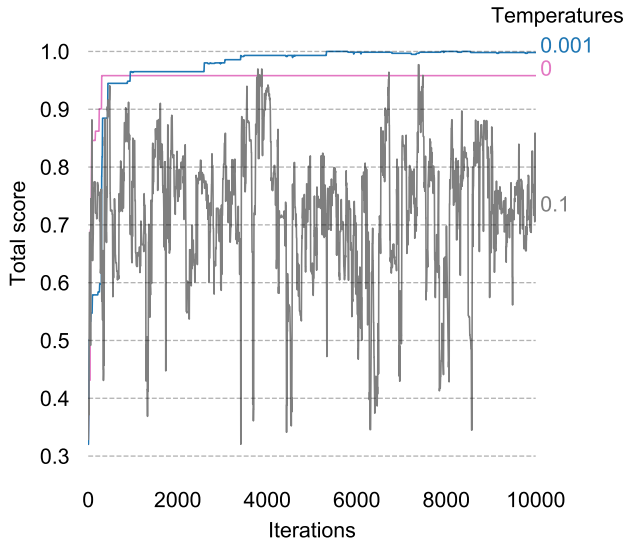


Figure 15. Total score versus iteration for SMIRKY sampling torsion fragment types in AlkEthOH. At a temperature of zero (pink line), SMIRKY works solely as a stochastic optimizer and gets stuck in a local optimum. When temperatures are too high, such as 0.1 (gray line), the probability of accepting moves that cause a decrease in score is very high, leading to dramatic changes in score throughout the simulation. At more moderate temperatures, such as 0.001 (blue line), a total score of 100% is eventually achieved.

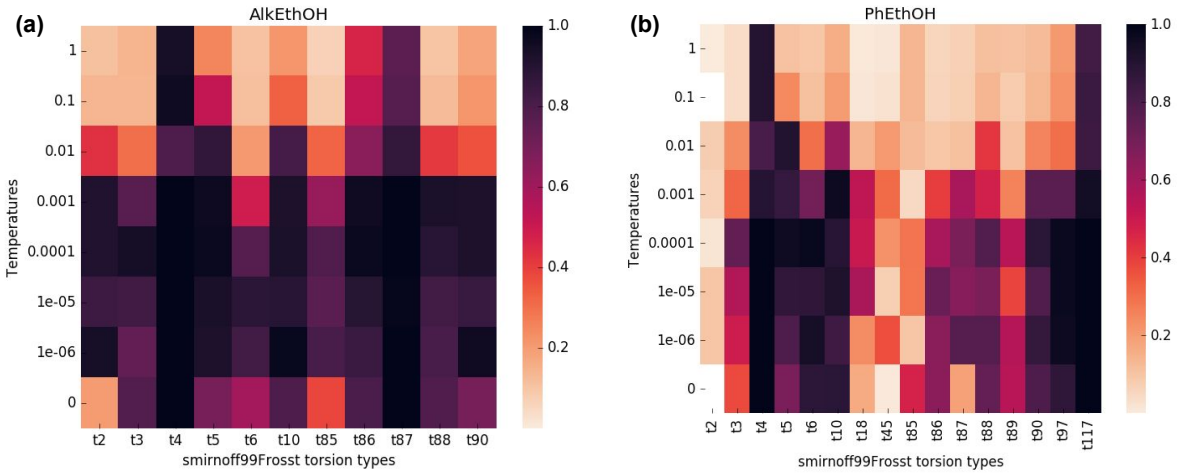


Figure 16. Frequency of success for SMIRKY simulations with AlkEthOH and PhEthOH. For torsion fragments in both AlkEthOH (a) and PhEthOH (b), we plot the fraction of simulation time where each reference fragment type has a torsion `SMIRKS` pattern that receives a partial score of 100%. For AlkEthOH, at temperatures below 0.001, all of the reference fragments spend at least 50% of the simulation time with a 100% partial score. However, for PhEthOH, many of the reference fragments spend a significant amount of simulation time without a 100% partial score.

SMIRKY is able to achieve a partial score of 100% for at least a fraction of time at all temperatures. It does achieve a 100% partial score for `t2` at most temperatures, only failing at temperatures of 0 and 0.1. However, PhEthOH is slightly more challenging, as evidenced by the fact that heat map colors are lighter in general.

Unlike in AlkEthOH, SMIRKY was not able to achieve a total score of 100% when sampling torsions in PhEthOH. PhEthOH only has six more torsion types, yet these make it substantially more challenging to find and keep sufficiently complex `SMIRKS`. In Section 3.1.1, we estimated the number of moves required to generate certain `SMARTS` patterns. When considering fragment types with more atoms, the combinatorial problem grows further since there are multiple atoms and bonds that require decorators. The lower frequency of 100% partial scores for torsions in PhEthOH is our first example of the implications of this combinatorial problem. For PhEthOH, we are able to generate `SMIRKS` for all reference torsions during at least some fraction of most simulations. However, if we are going to effectively sample chemical perception trees for the development of force fields, future move proposals will need to move through chemical space more efficiently. We will examine this problem more closely when analyzing SMIRKY's results on MiniDrugBank (Section 3.2.3).

Future move proposal engines could be made more efficient by taking advantage of basic chemical information. For example, if a double bond was already specified between a carbon and another atom, that carbon atom cannot have four bonds; however, as currently implemented, SMIRKY will occasionally propose moves which attempt to add a "`X4`" decorator to that carbon. We could take this concept one step further by learning which decorators can productively be applied to which atoms. For example, hydrogen can only have one bond and it is always single, so adding decorators to a hydrogen atom or its connecting bond will never usefully impact the separation of fragment types. Currently, SMIRKY makes many naïve moves, wasting many of its iterations generating `SMIRKS` patterns that cannot help separate fragments for these and other reasons.

3.2.3 MiniDrugBank is a significantly more challenging molecule set

MiniDrugBank covers significantly more chemical space than either of the other molecule sets with 73 bond, 34 angle, 136 proper torsion, and 26 nonbonded parameters from smirnoff99Frosst. It uses *all* parameters employed in molecules from the DrugBank database that can be atom typed by the parm99/parm@Frosst force field, as described in Section 2.4. This is a more complex set, so despite considerable success, we are often unable to obtain a total score of 100%. Specifically, we were unable to find a total score of 100% for bonds, angles, torsions or nonbonded fragment types with 10,000 iterations. We also saw a significant decrease in the average total score for all fragment types during our ten simulations (Table 6) relative to other molecule sets. For this reason we repeated the simulations with 50,000 and then 100,000 iterations. SMIRKY still did not find a total score of 100% during the longer simulations. However, overall, SMIRKY was partially successful for MiniDrugBank. On combining the results from every iteration in all simulations (10,000, 50,000, and 100,000 iterations at all temperatures) we find a 100% partial score for the majority of smirnoff99Frosst reference fragment types (Table 3). Heat maps for MiniDrugBank, like the examples shown for torsions in AlkEthOH and PhEthOH (Figure 16), are available in the Supporting Information. There are five bond, two angle, 19 torsion, and one nonbonded reference fragment types where SMIRKY never finds a 100% partial score. Potential reasons for missing these fragment types are summarized in this section and discussed in further detail in the Supporting Information. While it would be ideal to find all the fragment types, SMIRKY is able to find `SMIRKS` patterns that agree with 91% of the smirnoff99Frosst fragment types, indicating that we do recover a great deal of the target chemistry.

As with SMARTY, there are a few fragment types that SMIRKY never recovers. In the Supporting Information we provide information about these missing fragment types, exploring in more detail why some are so challenging and why others are undiscoverable with SMIRKY's current algorithm. In the following paragraphs we summarize the categories where SMIRKY's algorithm could be improved in order to discover these missing types.

The order of the fragment type hierarchy and relationship between parent and child types could help explain some missing types. The two missing angle types, `a6` ("`[*:1]~;!@[*;r3:2]~;!@[*:3]`") and `a12` ("`[*:1]~;!@[*;r5:2]~;@[*;r5:3]`"), highlight the importance of where newly generated `SMIRKS` are placed

	MiniDrugBank Average Score (%)*			
Temperature	**Bond**	**Angle**	**Torsion**	**Nonbonded**
1.0	67.0 ± 0.4	50.3 ± 0.3	32.7 ± 0.4	35.9 ± 0.3
0.1	70.3 ± 0.7	53.5 ± 0.5	34.9 ± 0.8	44.0 ± 0.4
0.01	82.9 ± 0.4	70 ± 1	47 ± 1	77 ± 1
0.001	90.1 ± 0.6	76.4 ± 0.9	55 ± 1	87 ± 2
0.0001	91.6 ± 0.4	81.4 ± 0.7	56 ± 1	86 ± 2
1e-05	91.9 ± 0.5	81 ± 1	54 ± 2	86 ± 1
1e-06	90.7 ± 0.6	77 ± 1	60 ± 1	86 ± 1
0	90 ± 1	81.7 ± 0.9	60 ± 2	79 ± 2

Table 6. Average of average total scores of 10 SMIRKY simulations with MiniDrugBank. We took the average score at each temperature for each of our 10 simulations (10,000 iterations each), then averaged this across all 10 simulations.

* uncertainties were estimated by the standard error over all 10 trials.

in the hierarchy. In smirnoff99Frosst these patterns always match carbon atoms in 3- and 5-membered rings, but atoms 1 and 3 can be any element, meaning it is easy to lose progress made toward these two patterns when patterns lower in the hierarchy also match the ring carbon. SMIRKY requires that when a child `SMIRKS` pattern is created the parent `SMIRKS` still matches at least some molecules. In most cases this prevents the creation of unnecessarily specific `SMIRKS`, but in some cases makes it impossible to OR decorators together. For example, we would be unable to find the exact pattern for `n7` ("`[#1:1]-[#6X4]~[*+1,*+2]`") because creation of a child with a second OR type on the beta atom will always empty the parent. This is also true for many of the other 19 missing torsions. SMIRKY is not trying to discover the exact `SMIRKS` patterns used in smirnoff99Frosst so it is possible our algorithm could still succeed finding `SMIRKS` matching these reference types using a different combination of moves. However, allowing moves to combine `SMIRKS` decorators could make distinguishing this type of chemistry easier. Another possible solution is allowing a child to completely replace a parent when the generated `SMIRKS` matches a larger section of chemical space, but not when it types the same group of fragments the parent typed.

The combinatorial problem that occurs in SMARTY is exacerbated in SMIRKY since multiple atoms may require decorators or alpha or beta substituents. It is difficult to identify a specific reason SMIRKY does not reach a 100% partial score for some of the torsion types and the five missing bond types. This suggests that the combinatorial problem described above (Section 3.2.2) is the most likely source of failure. Thus it seems likely that chemical perception sampling tools in future force field parameterization will need improved move sets with higher acceptance rates.

To estimate the size of the search problem posed by torsions in SMIRKY, we considered an example reference torsion `t78` ("`[*:1]-,:[#6X3:2]=[#7X2:3]-[*:4]`"), a deceptively simple example where SMIRKY never reaches a 100% partial score. SMIRKY simulations for torsions are initialized with the center two atoms specified and no bond information provided, so the initial torsion relevant to `t78` is "`[*:1]~[#6:2]~[#7:3]~[*:4]`" — that is, a carbon atom connected by any bond to a nitrogen atom. For the purpose of this exercise we will assume we want to generate this exact same `SMIRKS`; while that is not actually the goal of SMIRKY, it is a helpful example to illustrate the scope of the combinatorial problem. In this case, the probability of adding any one particular atom or bond decorator is approximately 0.01. In `t78`, there are an additional two atom decorators (`X3` on carbon and `X2` on nitrogen) along with four extra bond decorators. For each SMIRKY step, one torsion type from the working set is chosen (this is the parent type) and then the child fragment is created by making changes to this parent type (Figure 4). In order to generate this particular `SMIRKS` we also need to include the probability of choosing the correct parent type, which is about 0.02. Even if we assume the order of adding each decorator is unimportant, it will typically take SMIRKY over 1 billion steps to generate this exact `SMIRKS` pattern.

One way to simplify the combinatorial problem in SMIRKY or future packages for sampling chemical perception trees would be creating a more efficient move set. As discussed in Section 3.2.2, SMIRKY currently

considers all possible decorators for the atoms and bonds, whether or not they make chemical sense, and thus wastes considerable time proposing patterns that do not match any molecules. Considering the low probability of picking a given decorator, the fact that SMIRKY is able to find SMIRKS patterns which match 120 (88%) of the reference torsions is further evidence that we have been fairly successful in sampling the chemical perception tree effectively. However, if we could increase the probability of picking good decorators, we could decrease the number of moves required to sample the requisite chemistry. Future progress of these tools should leverage moves which only employ chemically reasonable combinations of decorators. For example, a double bond decorator should never be allowed next to a tetrahedral carbon which only has single bonds. This information would not necessarily need to be provided by a human expert; instead, molecule sets used for training could be used to extract this information.

Overall, SMIRKY's ability to find over 90% of the smirnoff99Frosst fragment types in MiniDrugBank shows it can automatically sample suitable chemical perception trees. Despite the shortcomings discussed above, SMIRKY is clearly capable of generating complex chemical perception like that historically generated by hand.

4 Conclusions

SMARTY and SMIRKY have proven capable of automatically sampling chemical perception trees relevant to general small molecule force fields. Both of our tools were tested first with relatively simple sets (AlkEthOH and PhEthOH) and then using MiniDrugBank, which provided a test set of molecules encompassing the diversity of the DrugBank database. SMARTY generated SMARTS patterns covering 89.2% of the atom types of parm99/parm@Frosst force field for MiniDrugBank, and 100% for AlkEthOH and PhEthOH.

In some cases, the significant chemical complexity encoded in certain existing atom types limits our ability to find these types via individual SMARTS patterns. This may in part be because of overly complex atom typing, which, in traditional force fields, needs to encode the chemistry required for all force field parameter types simultaneously. Switching to direct chemical perception may allow for much more straightforward parameter assignment.[73]

Our SMIRKY tool for working with direct chemical perception also had considerable success. SMIRKY generated SMIRKS patterns which capture more than 90% of the fragment types in smirnoff99Frosst for MiniDrugBank and 100% for AlkEthOH and PhEthOH. Coverage was less than perfect in part because there is a significant combinatorial problem in sampling chemical perception trees. This combinatorial problem is most pronounced in SMIRKY's results on torsions, where many atoms and bonds may require multiple decorators. The ability to automatically recover the chemical perception in these existing force fields is a promising first step toward sampling over chemical perception as part of force field parameterization.

SMARTY and SMIRKY are prototypes created as a part of the Open Force Field Initiative,[42] which aims to create open source parameterization tools that can provide accurate force fields without human experts being required in the parameterization process. The overall initiative is expected to have a significant impact on the quality of biophysical modeling and molecular design for a variety of fields, and facilitate force field science.

Historically, many force fields used atom types, a form of indirect chemical perception, where all atom types were generated by hand by an expert. As discussed in the Introduction, the complexity of these atom types has been a major contributor to the difficulty of force field development and a barrier to cross-comparing force fields. The new SMIRNOFF format, with direct chemical perception, is an important step forward, but still relies on hand-encoded SMIRKS patterns to assign parameters. In order to automate force field development, the generation of these SMIRKS patterns must be done automatically. SMARTY and SMIRKY are able to generate SMARTS and SMIRKS with comparable chemical complexity to parm99/parm@Frosst and smirnoff99Frosst. These tools are a promising first step to determining chemical perception for general force fields without relying on human expertise.

While SMARTY and SMIRKY both show considerable promise, they both face challenges with the size of the combinatorial search problem, suggesting room for further improvement. The combinatorial problem becomes especially important for fragment types involving the most potential atoms and decorators — angles and torsions. Here, the sampling problem is exacerbated by the fact that move proposals are not

necessarily chemically sensible, so future work will need to make more reasonable chemical moves to improve efficiency. Instead of using any `SMARTS` decorator, these tools should take advantage of basic chemistry. For example, no molecule will have a tetrahedral carbon with a double bond, so such a move should never be proposed.

Our ultimate goal is to move past comparisons to existing force fields and to develop new force fields which hopefully provide improved accuracy. While SMARTY and SMIRKY are both promising examples of tools which sample over chemical perception trees, our work here used them in the context of existing force fields. Longer-term, they will instead be used to help sample over chemical perception as well as force field parameters while fitting force fields to allow development of next generation force fields in a completely automated way.

5 Acknowledgements

The authors are grateful to Patrick B. Grinaway (ORCID: 0000-0002-9762-4201), Kyle A. Beauchamp (ORCID: 0000-0001-6095-8788), Robert McGibbon (ORCID: 0000-0003-3337-954X), Kim Branson (ORCID: 0000-0002-5326-8911), and Vijay S. Pande (ORCID: 0000-0003-2774-1178) for insightful discussions on Bayesian approaches to sampling atom types. We would like to also acknowledge Victoria T. Lim (ORCID: 0000-0003-4030-9312) for advice on code organization and help reviewing code. We are also thankful for helpful discussions with all the members of the Open Force Field Initiative.

DLM, CCB, and CZ appreciate the financial support from the National Science Foundation (CHE 1352608) and the National Institutes of Health (1R01GM108889-01) and computing support from the UCI GreenPlanet cluster, supported in part by NSF Grant CHE-0840513. CCB is also supported by a fellowship from The Molecular Sciences Software Institute under NSF grant ACI-1547580. CZ also appreciates the Brazilian Science without Borders scholarship (Capes - BEX 1865612-9). MKG appreciates financial support from the NIH (GM061300). JDC appreciates support from the Sloan Kettering Institute and NIH grant P30 758 CA008748, Cycle for Survival, and the Sloan Kettering Institute. JF acknowledges support from NSF grant CHE-1738979. MRS acknowledges support from NSF grant CHE-173897. CIB acknowledges support by OpenEye Scientific Software for his sabbatical term contributing to this work. MKG has an equity interest in, and is a co-founder and scientific advisor of, VeraChem LLC. The contents of this publication are solely the responsibility of the authors and do not necessarily represent the official views of the NIH. DLM serves on the scientific advisory board of OpenEye Scientific Software. JDC serves on the scientific advisory board of Schrödinger.

References

[1] Klepeis, J. L.; Lindorff-Larsen, K.; Dror, R. O.; Shaw, D. E. *Curr. Opin. Struct. Biol.* **2009**, *19*, 120–127.

[2] Sellers, B. D.; James, N. C.; Gobbi, A. *J. Chem. Inf. Model.* **2017**, *57*, 1265–1275.

[3] Fischer, N. M.; van Maaren, P. J.; Ditz, J. C.; Yildirim, A.; van der Spoel, D. *J. Chem. Theory Comput.* **2015**, *11*, 2938–2944.

[4] Christ, C. D.; Mark, A. E.; van Gunsteren, W. F. *J. Comput. Chem.* **2010**, *31*, 1569–1582.

[5] Pohorille, A.; Jarzynski, C.; Chipot, C. *J. Phys. Chem. B* **2010**, *114*, 10235–10253.

[6] Mohamed, N. A.; Bradshaw, R. T.; Essex, J. W. *J. Comput. Chem.* **2016**, *37*, 2749–2758.

[7] Jiao, D.; Golubkov, P. A.; Darden, T. A.; Ren, P. *PNAS* **2008**, *105*, 6290–6295.

[8] Hansen, N.; van Gunsteren, W. F. *J. Chem. Theory Comput.* **2014**, *10*, 2632–2647.

[9] Mishra, S. K.; Calabró, G.; Loeffler, H. H.; Michel, J.; Koča, J. *J. Chem. Theory Comput.* **2015**, *11*, 3333–3345.

[10] Mobley, D. L.; Gilson, M. K. *Annu. Rev. Biophys.* **2017**, *46*, 531–558.

[11] Piana, S.; Lindorff-Larsen, K.; Shaw, D. E. *Biophys. J.* **2011**, *100*, L47–L49.

[12] Senftle, T. P.; Hong, S.; Islam, M. M.; Kylasa, S. B.; Zheng, Y.; Shin, Y. K.; Junkermeier, C.; Engel-Herbert, R.; Janik, M. J.; Aktulga, H. M.; Verstraelen, T.; Grama, A.; van Duin, A. C. T. *npj Comput. Mater.* **2016**, *2*, 15011.

[13] van Gunsteren, W. F.; Berendsen, H. J. C. *Angew. Chem. Int. Ed. Engl.* **1990**, *29*, 992–1023.

[14] Dey, A.; Nath Pati, N.; R. Desiraju, G. *CrystEngComm* **2006**, *8*, 751–755.

[15] Lindorff-Larsen, K.; Piana, S.; Palmo, K.; Maragakis, P.; Klepeis, J. L.; Dror, R. O.; Shaw, D. E. *Proteins* **2010**, *78*, 1950–1958.

[16] Vanommeslaeghe, K.; Yang, M.; MacKerell, A. D. *J. Comput. Chem.* **2015**, *36*, 1083–1101.

[17] Wang, L. et al. *J. Am. Chem. Soc.* **2015**, *137*, 2695–2703.

[18] Aldeghi, M.; Heifetz, A.; Bodkin, M. J.; Knapp, S.; Biggin, P. C. *Chem Sci* **2016**, *7*, 207–218.

[19] Bannan, C. C.; Burley, K. H.; Chiu, M.; Shirts, M. R.; Gilson, M. K.; Mobley, D. L. *J.Comput. Aided Mol. Des.* **2016**, *30*, 1–18.

[20] Wu, S.; Angelikopoulos, P.; Papadimitriou, C.; Moser, R.; Koumoutsakos, P. *Phil. Trans. R. Soc. A* **2016**, *374*, 20150032.

[21] Anisimov, V. M.; Vorobyov, I. V.; Roux, B.; MacKerell, A. D. *J. Chem. Theory Comput.* **2007**, *3*, 1927–1946.

[22] Monticelli, L.; Tieleman, D. In *Biomolecular Simulations*; Monticelli, L., Salonen, E., Eds.; Methods in Molecular Biology 924; Humana Press, 2013; pp 197–213.

[23] Ponder, J. W.; Wu, C.; Ren, P.; Pande, V. S.; Chodera, J. D.; Schnieders, M. J.; Haque, I.; Mobley, D. L.; Lambrecht, D. S.; DiStasio, R. A.; Head-Gordon, M.; Clark, G. N. I.; Johnson, M. E.; Head-Gordon, T. *J. Phys. Chem. B* **2010**, *114*, 2549–2564.

[24] Wang, L.-P.; Martinez, T. J.; Pande, V. S. *J. Phys. Chem. Lett.* **2014**, *5*, 1885–1891.

[25] Wu, J. C.; Chattree, G.; Ren, P. *Theor. Chem. Acc.* **2012**, *131*, 1138.

[26] Halgren, T. A. *J. Comput. Chem.* **1996**, *17*, 490–519.

[27] Halgren, T. A. *J. Comput. Chem.* **1996**, *17*, 520–552.

[28] Damm, W.; Frontera, A.; Tirado–Rives, J.; Jorgensen, W. L. *J. Comput. Chem.* **1997**, *18*, 1955–1970.

[29] Cailliez, F.; Pernot, P. *J. Chem. Phys.* **2011**, *134*, 054124.

[30] Geballe, M. T.; Guthrie, J. P. *J. Comput. Aided Mol. Des.* **2012**, *26*, 489–496.

[31] Rizzi, F.; Najm, H.; Debusschere, B.; Sargsyan, K.; Salloum, M.; Adalsteinsson, H.; Knio, O. *Multiscale Model. Simul.* **2012**, *10*, 1460–1492.

[32] Deublein, S.; Metzler, P.; Vrabec, J.; Hasse, H. *Mol. Simul.* **2013**, *39*, 109–118.

[33] Köster, A.; Spura, T.; Rutkai, G.; Kessler, J.; Wiebeler, H.; Vrabec, J.; Kühne, T. D. *J. Comput. Chem.* **2016**, n/a–n/a.

[34] Hopkins, C. W.; Roitberg, A. E. *J. Chem. Inf. Model.* **2014**, *54*, 1978–1986.

[35] Hédin, F.; El Hage, K.; Meuwly, M. *J. Chem. Inf. Model.* **2016**, *56*, 1479–1489.

[36] Baskin, I. I.; Winkler, D.; Tetko, I. V. *Expert Opin. Drug Discovery* **2016**, *11*, 785–795.

[37] Huang, J.; Rauscher, S.; Nawrocki, G.; Ran, T.; Feig, M.; de Groot, B. L.; Grubmüller, H.; MacKerell Jr, A. D. *Nat Meth* **2017**, *14*, 71–73.

[38] Folgoc, L. L.; Delingette, H.; Criminisi, A.; Ayache, N. *IEEE Trans. Med. Imaging* **2017**, *36*, 607–617.

[39] Lesch, V.; Diddens, D.; Bernardes, C. E. S.; Golub, B.; Dequidt, A.; Zeindlhofer, V.; Sega, M.; Schröder, C. *J. Comput. Chem.* **2017**, *38*, 629–638.

[40] Smith, J. S.; Isayev, O.; Roitberg, A. E. *Chem. Sci.* **2017**, *8*, 3192–3203.

[41] Xu, P.; Guidez, E. B.; Bertoni, C.; Gordon, M. S. *J. Chem. Phys.* **2018**, *148*, 090901.

[42] Chodera, J. D.; Gilson, M. K.; Mobley, D. L.; Shirts, M. R.; Wang, L.-P.; Kroenlein, K.; Bayly, C. I. Open Force Field Group. openforcefield.org, 2018.

[43] Ren, P.; Ponder, J. W. *J. Comput. Chem.* **2002**, *23*, 1497–1506.

[44] Grossfield, A.; Ren, P.; Ponder, J. W. *J. Am. Chem. Soc.* **2003**, *125*, 15671–15682.

[45] Noskov, S. Y.; Lamoureux, G.; Roux, B. *J. Phys. Chem. B* **2005**, *109*, 6705–6713.

[46] Yu, H.; Geerke, D. P.; Liu, H.; van Gunsteren, W. F. *J. Comput. Chem.* **2006**, *27*, 1494–1504.

[47] Laury, M. L.; Wang, L.-P.; Pande, V. S.; Head-Gordon, T.; Ponder, J. W. *J. Phys. Chem. B* **2015**, *119*, 9423–9437.

[48] Lemkul, J. A.; Huang, J.; Roux, B.; MacKerell, A. D. *Chem. Rev.* **2016**, *116*, 4983–5013.

[49] Liu, C.; Li, Y.; Han, B.-Y.; Gong, L.-D.; Lu, L.-N.; Yang, Z.-Z.; Zhao, D.-X. *J. Chem. Theory Comput.* **2017**,

[50] Liu, Y.; Tao, L.; Lu, J.; Xu, S.; Ma, Q.; Duan, Q. *FEBS Letters* **2011**, *585*, 888–892.

[51] Angelikopoulos, P.; Papadimitriou, C.; Koumoutsakos, P. *J. Chem. Phys.* **2012**, *137*, 144103.

[52] Olubiyi, O. O.; Strodel, B. *Data Brief* **2016**, *9*, 642–647.

[53] Wang, L.-P.; McKiernan, K. A.; Gomes, J.; Beauchamp, K. A.; Head-Gordon, T.; Rice, J. E.; Swope, W. C.; Martínez, T. J.; Pande, V. S. *J. Phys. Chem. B* **2017**, *121*, 4023–4039.

[54] Wildman, J.; Repiščák, P.; Paterson, M. J.; Galbraith, I. *J. Chem. Theory Comput.* **2016**, *12*, 3813–3824.

[55] Qi, R.; Wang, Q.; Ren, P. *Bioorg. Med. Chem.* **2016**,

[56] Botu, V.; Batra, R.; Chapman, J.; Ramprasad, R. *J. Phys. Chem. C* **2017**, *121*, 511–522.

[57] Zahariev, F.; De Silva, N.; Gordon, M. S.; Windus, T. L.; Dick-Perez, M. *J. Chem. Inf. Model* **2017**,

[58] Pande, V. *Biophys. J.* **2014**, *106*, 44a.

[59] Wang, L.-P.; Head-Gordon, T.; Ponder, J. W.; Ren, P.; Chodera, J. D.; Eastman, P. K.; Martinez, T. J.; Pande, V. S. *J. Phys. Chem. B* **2013**, *117*, 9956–9972.

[60] Torabifard, H.; Starovoytov, O. N.; Ren, P.; Cisneros, G. A. *Theor. Chem. Acc.* **2015**, *134*, 101.

[61] Mu, X.; Wang, Q.; Wang, L.-P.; Fried, S. D.; Piquemal, J.-P.; Dalby, K. N.; Ren, P. *J. Phys. Chem. B* **2014**, *118*, 6456–6465.

[62] Bradshaw, R. T.; Essex, J. W. *J. Chem. Theory Comput.* **2016**, *12*, 3871–3883.

[63] Albaugh, A. et al. *J. Phys. Chem. B* **2016**, *120*, 9811–9832.

[64] Artemova, S.; Jaillet, L.; Redon, S. *J. Comput. Chem.* **2016**, *37*, 1191–1205.

[65] Avendaño-Franco, G.; Romero, A. H. *J. Chem. Theory Comput.* **2016**, *12*, 3416–3428.

[66] Cole, D. J.; Vilseck, J. Z.; Tirado-Rives, J.; Payne, M. C.; Jorgensen, W. L. *J. Chem. Theory Comput.* **2016**, *12*, 2312–2323.

[67] Vanommeslaeghe, K.; MacKerell, A. D. *J. Chem. Inf. Model.* **2012**, *52*, 3144–3154.

[68] Zhang, Q.; Zhang, W.; Li, Y.; Wang, J.; Zhang, L.; Hou, T. *J. Cheminform.* **2012**, *4*, 26.

[69] Wang, J.; Wang, W.; Kollman, P. A.; Case, D. A. *J. Mol. Graph. Model.* **2006**, *25*, 247–260.

[70] Yesselman, J. D.; Price, D. J.; Knight, J. L.; Brooks, C. L. *J. Comput. Chem.* **2012**, *33*, 189–202.

[71] Daylight Chemical Information Systems Inc., *Daylight Chemical Information Systems Inc.* **2011**, *Daylight Version 4.9*.

[72] Ehrlich, H.-C.; Rarey, M. *J. Cheminform.* **2012**, *4*, 13.

[73] Mobley, D.; Bannan, C. C.; Rizzi, A.; Bayly, C. I.; Chodera, J. D.; Lim, V. T.; Lim, N. M.; Beauchamp, K. A.; Shirts, M. R.; Gilson, M. K.; Eastman, P. K. *bioRxiv* **2018**, 286542.

[74] Wang, J.; Wolf, R. M.; Caldwell, J. W.; Kollman, P. A.; Case, D. A. *J. Comput. Chem.* **2004**, *25*, 1157–1174.

[75] Böcker, S.; Bui, Q. B. A.; Truss, A. *Theoretical Computer Science* **2011**, *412*, 1184–1195.

[76] Wildman, S. A.; Crippen, G. M. *J. Chem. Inf. Comput. Sci.* **1999**, *39*, 868–873.

[77] Pedretti, A.; Villa, L.; Vistoli, G. *Theor. Chem. Acc.* **2003**, *109*, 229–232.

[78] Lipkowitz, K.; Boyd, D. *Reviews in Computational Chemistry*; Reviews in Computational Chemistry v. 14; Wiley, 2009.

[79] Brown, I. D. *IUCrJ* **2017**, *4*, 514–515.

[80] Jin, Z.; Yang, C.; Cao, F.; Li, F.; Jing, Z.; Chen, L.; Shen, Z.; Xin, L.; Tong, S.; Sun, H. *J. Comput. Chem.* **2016**, *37*, 653–664.

[81] Bush, B. L.; Sheridan, R. P. *J. Chem. Inf. Comput. Sci.* **1993**, *33*, 756–762.

[82] OpenEye Scientific Software, I. OEChem. http://www.eyesopen.com, 2010; www.eyesopen.com.

[83] Marcou, G.; Rognan, D. *J. Chem. Inf. Model.* **2006**, *47*, 195–207.

[84] Stahl, M.; Mauser, H. *J. Chem. Inf. Model.* **2005**, *45*, 542–548.

[85] Green, P. J. *Biometrika* **1995**, *82*, 711–732.

[86] Binder, K.; Heermann, D. *Monte Carlo Simulation in Statistical Physics: An Introduction*, 5th ed.; Graduate Texts in Physics; Springer-Verlag: Berlin Heidelberg, 2010.

[87] Farrell, K.; Oden, J. T.; Faghihi, D. *J. Comput. Phys.* **2015**, *295*, 189–208.

[88] Ferguson, A. L. *J. Comput. Chem.* **2017**, *38*, 1583–1605.

[89] Metropolis, N.; Rosenbluth, A. W.; Rosenbluth, M. N.; Teller, A. H.; Teller, E. *J. Chem. Phys.* **1953**, *21*, 1087–1092.

[90] Kroese, D. P.; Brereton, T.; Taimre, T.; Botev, Z. I. *Wiley Interdiscip. Rev. Comput. Stat.* **2014**, *6*, 386–392.

[91] Diestel, R. *Graph Theory*, 5th ed.; Springer, 2016.

[92] Jensen, F. *Introduction to Computational Chemistry*; John Wiley & Sons, 2016.

[93] Galil, Z. *ACM Comput. Surv.* **1986**, *18*, 23–38.

[94] Glover, F. *Naval Research Logistics* **1967**, *14*, 313–316.

[95] Bayly, C. I.; Bannan, C. C.; Mobley, D. L. smirnoff99Frosst. doi.org/10.5281/zenodo.1186466, 2016.

[96] Wang, J.; Cieplak, P.; Kollman, P. A. *J. Comput. Chem.* **2000**, *21*, 1049–1074, parm99.

[97] Bayly, C.; McKay, D.; Truchon, J. An Informal AMBER Small Molecule Force Field: Parm@ Frosst. http://www.ccl.net/cca/data/parm_at_Frosst/, 2010.

[98] Wishart, D. S.; Knox, C.; Guo, A. C.; Shrivastava, S.; Hassanali, M.; Stothard, P.; Chang, Z.; Woolsey, J. *Nucleic Acids Res.* **2006**, *34*, D668–672.

[99] Wishart, D. S.; Knox, C.; Guo, A. C.; Cheng, D.; Shrivastava, S.; Tzur, D.; Gautam, B.; Hassanali, M. *Nucleic Acids Res.* **2008**, *36*, D901–906.

[100] Knox, C.; Law, V.; Jewison, T.; Liu, P.; Ly, S.; Frolkis, A.; Pon, A.; Banco, K.; Mak, C.; Neveu, V.; Djoumbou, Y.; Eisner, R.; Guo, A. C.; Wishart, D. S. *Nucleic Acids Res.* **2011**, *39*, D1035–1041.

[101] Law, V. et al. *Nucleic Acids Res.* **2014**, *42*, D1091–1097.

[102] Veenstra, D. L.; Ferguson, D. M.; Kollman, P. A. *J. Comput. Chem.* **1992**, *13*, 971–978.

A Supporting Information Available

The supporting information consists of a PDF file with extended details and images considered too long for the main text. There are images of all molecules in each of the molecule sets tested for these results: AlkEthOH (42), PhEthOH (200), and MiniDrugBank (371). Additionally, we provided further details and analysis of the reference atom types (SMARTY) and fragment types (SMIRKY) which never receive a 100% partial score. We also provide heat map images for all SMIRKY fragment types and molecule sets not shown here in the same form as Figures 10, 13, and 16.

Additional supporting information is available free of charge online through the University of California Dash at https://doi.org/10.7280/D1CD4C. It includes input files, output files, and score vs. iteration plots for all SMARTY and SMIRKY simulations with every molecule set discussed. This also includes molecule files that are human and machine readable. It contains heat map results both as images and as computer readable csv files, as well as all of the Python scripts used to analyze these results and to create the figures in this paper.