4. SMARTS - A Language for Describing Molecular Patterns

Substructure searching, the process of finding a particular pattern (subgraph) in a molecule (graph), is one of the most important tasks for computers in chemistry. It is used in virtually every application that employs a digital representation of a molecule, including depiction (to highlight a particular functional group), drug design (searching a database for similar structures and activity), analytical chemistry (looking for previously-characterized structures and comparing their data to that of an unknown), and a host of other problems.

SMARTS is a language that allows you to specify substructures using rules that are straightforward extensions of SMILES. For example, to search a database for phenol-containing structures, one would use the SMARTS string **[OH]c1ccccc1**, which should be familiar to those acquainted with SMILES. In fact, almost all SMILES specifications are valid SMARTS targets. Using SMARTS, flexible and efficient substructure-search specifications can be made in terms that are meaningful to chemists.

In the SMILES language, there are two fundamental types of symbols: *atoms* and *bonds*. Using these SMILES symbols, once can specify a molecule's graph (its "nodes" and "edges") and assign "labels" to the components of the graph (that is, say what type of atom each node represents, and what type of bond each edge represents).

The same is true in SMARTS: One uses atomic and bond symbols to specify a graph. However, in SMARTS the labels for the graph's nodes and edges (its "atoms" and "bonds") are extended to include "logical operators" and special atomic and bond symbols; these allow SMARTS atoms and bonds to be more general. For example, the SMARTS atomic symbol **[C,N]** is an atom that can be aliphatic **C** or aliphatic **N**; the SMARTS bond symbol **~** (tilde) matches any bond.

4.1 Atomic Primitives

SMARTS provides a number of primitive symbols describing atomic properties beyond those used in SMILES (atomic symbol, charge, and isotopic specifications). The following tables list the atomic primitives used in SMARTS (all SMILES atomic symbols are also legal). In these tables **<n>** stands for a digit, **<c>** for chiral class.

Note that atomic primitive **H** can have two meanings, implying a property or the element itself. **[H]** means hydrogen atom. **[*H2]** means any atom with exactly two hydrogens attached

SMARTS Atomic Primitives

Symbol	Symbol name	Atomic property requirements	Default
*	wildcard	any atom	(no default)
a	aromatic	aromatic	(no default)
A	aliphatic	aliphatic	(no default)
D<n>	degree	<n> explicit connections	exactly one
H<n>	total-H-count	<n> attached hydrogens	exactly one[1]
h<n>	implicit-H-count	<n> implicit hydrogens	at least one
R<n>	ring membership	in <n> SSSR rings	any ring atom
r<n>	ring size	in smallest SSSR ring of size <n>	any ring atom[2]
v<n>	valence	total bond order <n>	exactly one[2]

X<n>	connectivity	<n> total connections	exactly one[2]
x<n>	ring connectivity	<n> total ring connections	at least one[2]
- <n>	negative charge	-<n> charge	-1 charge (-- is -2, etc)
+<n>	positive charge	+<n> formal charge	+1 charge (++ is +2, etc)
#n	atomic number	atomic number <n>	(no default)[2]
@	chirality	anticlockwise	anticlockwise, default class[2]
@@	chirality	clockwise	clockwise, default class[2]
@<c><n>	chirality	chiral class <c> chirality <n>	(nodefault)
@<c><n>?	chiral or unspec	chirality <c><n> or unspecified	(no default)
<n>	atomic mass	explicit atomic mass	unspecified mass

[1] Semantics of [H] changed in v4.5 [2] @ and @@ introduced in v4.1; r, v, X, and # in v4.3; x in v4.9

Examples:

C	aliphatic carbon atom
c	aromatic carbon atom
a	aromatic atom
[#6]	carbon atom
[Ca]	calcium atom
[++]	atom with a +2 charge
[R]	atom in any ring
[D3]	atom with 3 explicit bonds (implicit H's don't count)
[X3]	atom with 3 total bonds (includes implicit H's)
[v3]	atom with bond orders totaling 3 (includes implicit H's)
C[C@H](F)O	match chirality (H-F-O anticlockwise viewed from C)
C[C@?H](F)O	matches if chirality is as specified or is not specified

4.2 Bond Primitives

Various bond symbols are available to match connections between atoms. A missing bond symbol is interpreted as "single or aromatic".

SMARTS Bond Primitives

Symbol	Atomic property requirements
-	single bond (aliphatic)
/	directional bond "up"[1]
\	directional bond "down"[1]
/?	directional bond "up or unspecified"
\?	directional bond "down or unspecified"
=	double bond
#	triple bond
:	aromatic bond
~	any bond (wildcard)
@	any ring bond[1]

[1] / and \ introduced in v4.1; @ in v4.6

Examples:

C	any aliphatic carbon
cc	any pair of attached aromatic carbons
c:c	aromatic carbons joined by an aromatic bond
c-c	aromatic carbons joined by a single bond (e.g. biphenyl)

4.3 Logical Operators

Atom and bond primitive specifications may be combined to form expressions by using logical operators. In the following table, **e** is an atom or bond SMARTS expression (which may be a primitive). The logical operators are listed in order of decreasing precedence (high precedence operators are evaluated first).

SMARTS Logical Operators

Symbol	Expression	Meaning
exclamation	!e1	not e1
ampersand	e1&e2	a1 and e2 (high precedence)
comma	e1,e2	e1 or e2
semicolon	e1;e2	a1 and e2 (low precedence)

All atomic expressions which are not simple primitives must be enclosed in brackets. The default operation is **&** (high precedence "and"), i.e., two adjacent primitives without an intervening logical operator must both be true for the expression (or subexpression) to be true.

The ability to form expressions gives the SMARTS user a great deal of power to specify exactly what is desired. The two forms of the AND operator are used in SMARTS instead of grouping operators.

Examples:

[CH2]	aliphatic carbon with two hydrogens (methylene carbon)
[!C;R]	(NOT aliphatic carbon) AND in ring
[!C;!R0]	same as above ("!R0" means not in zero rings)
[n;H1]	H-pyrrole nitrogen
[n&H1]	same as above
[nH1]	same as above
[c,n&H1]	any arom carbon OR H-pyrrole nitrogen
[X3&H0]	atom with 3 total bonds and no H's
[c,n;H1]	(arom carbon OR arom nitrogen) and exactly one H
[Cl]	any chlorine atom
[35*]	any atom of mass 35
[35Cl]	chlorine atom of mass 35
[F,Cl,Br,I]	the 1st four halogens.

4.4 Recursive SMARTS

Any SMARTS expression may be used to define an atomic environment by writing a SMARTS starting with the atom of interest in this form:

$(SMARTS)

Such definitions may be considered atomic properties. These expressions can be used in same manner as other atomic primitives (also, they can be nested). Recursive SMARTS expressions are used in the following manner:

*C	atom connected to methyl (or methylene) carbon
*CC	atom connected to ethyl carbon

[$(*C);$(*CC)]	atom in both above environments (matches CCC)

The additional power of such expressions is illustrated by the following example which derives an expression for methyl carbons which are ortho to oxygen and meta to a nitrogen on an aromatic ring.

CaaO	C ortho to O
CaaaN	C meta to N
Caa(O)aN	C ortho to O and meta to N (but 2O,3N only)
Ca(aO)aaN	C ortho to O and meta to N (but 2O,5N only)
C[$(aaO);$(aaaN)]	C ortho to O and meta to N (all cases)

4.5 Component-level grouping of SMARTS

SMARTS may contain "zero-level" parentheses which can be used to group dot-disconnected fragments. This grouping operator allows SMARTS to express more powerful component queries. In general, a single set of parentheses may surround any legal SMARTS expression. Two or more of these expressions may be combined into more complex SMARTS:

(SMARTS)
(SMARTS).(SMARTS)
(SMARTS).SMARTS

The semantics of the "zero-level" parentheses are that all of the atom and bond expressions within a set of zero-level parentheses must match within a single component of the target.

SMARTS	SMILES	Match behavior
C.C	CCCC	yes, no component level grouping specified
(C.C)	CCCC	yes, both carbons in the query match the same component
(C).(C)	CCCC	no, the query must match carbons in two different components
(C).(C)	CCCC.CCCC	yes, the query does match carbons in two different components
(C).C	CCCC	yes, both carbons in the query match the same component
(C).(C).C	CCCC.CCCC	yes, the first two carbons match different components, the third matches a carbon anywhere

These component-level grouping operators were added specifically for reaction processing. Without this construct, it is impossible to distinguish inter- versus intramolecular reaction queries. For example:

Reaction SMARTS expression	Match behavior
C(=O)O.OCC>>C(=O)OCC.O	Matches esterifications
(C(=O)O).(OCC)>>C(=O)OCC.O	Matches intermolecular esterifications
(C(=O)O.OCC)>>C(=O)OCC.O	Matches intramolecular esterifications (lactonizations)

4.6 Reaction Queries

Reaction queries are expressed using the SMARTS language. SMARTS has been extended to handle reaction query features in much the same fashion as SMILES has been extended to handle reactions.

A reaction query may be composed of optional reactant, agent, and product parts, which are separated by the ">" character. In this case, the parts of the reaction query match against the corresponding roles within the reaction target, as expected. Note that it is also quite reasonable to search a set of reactions by giving a molecule query. In this case, the answer is a hit if the molecule SMARTS matches anywhere within the reaction target. In effect, matching a molecule SMARTS against a reaction target is a query where the role of the SMARTS is unspecified.

Example Reaction SMARTS:		
Query:	Target:	Matches:

C>>	CC>>CN	2
>C>	CC>>CN	0
>>C	CC>>CN	1
C	CC>>CN	3

The atom mapping for a reaction query is optional. When included in the definition of the pattern, it is used for searching.

If atom maps are used for a SMARTS match, their only effect is to potentially eliminate answers from the result. Atom maps can never, under any circumstance cause the addition of hits to an answer set. Conceptually, one can consider the atom map matching as a post-processing step after a "normal" match. Each of the hits is examined to make sure the atom map classes match on the reactant and product sides of the reaction.

In SMARTS, the atom map has unusual semantics. An atom map is a property which must be evaluated on a global scope during the match. One can not know if the map is correct without considering every atom in the match, in effect requiring the enumeration of every possible path before testing. This is much more computationally expensive than the current SMARTS implementation, which tests the paths as they are built and stops as soon as a path fails to match.

In order to avoid this computational trap, the expressiveness of SMARTS for atom maps has been limited to a low-precedence and operation. That is, only expressions of form: "[expr:n]" or "[expr:?n]" are allowed, where "expr" is any legal atomic expression excluding atom maps and "n" is a map class value. This expression is a low-precedence logical AND between "expr" and the map expression ":n". The following examples illustrate other nuances of the semantics:

Example Reaction SMARTS:			
Query:	Target:	Matches:	Comment:
C>>C	CC>>CC	4	No maps, normal match.
C>>C	[CH3:7][CH3:8]>> [CH3:7][CH3:8]	4	No maps in query, maps in target are ignored.
[C:1]>>C	[CH3:7][CH3:8]>> [CH3:7][CH3:8]	4	Unpaired map in query ignored.
[C:1]>>[C:1]	CC>>CC	0	No maps in target, hence no matches.
[C:?1]>>[C:?1]	CC>>CC	4	Query says mapped as shown or not present.
[C:1]>>[C:1]	[CH3:7][CH3:8]>>[CH3:7][CH3:8]	2	Matches for target 7,7 and 8,8 atom pairs.
[C:1]>>[C:2]	[CH3:7][CH3:8]>> [CH3:7][CH3:8]	4	When a query class is not found on both sides of the query, it is ignored; this query does NOT say that the atoms are in different classes.
[C:1][C:1]>>[C:1]	[CH3:7][CH3:7]>> [CH3:7][CH3:7]	4	Atom maps match with "or" logic. All atoms get bound to class 7.
[C:1][C:1]>>[C:1]	[CH3:7][CH3:8]>> [CH3:7][CH3:8]	4	The reactant atoms are bound to classes 7 and 8. Note that having the first query atom bound to class 7 does not preclude binding the second atom. Next, the product atom can bind to classes 7 or 8.
[C:1][C:1]>>[C:1]	[CH3:7][CH3:7]>> [CH3:7][CH3:8]	2	The reactants are bound to class 7. The product atom can bind to class 7 only.

The last example is the most confusing. Since there is no "or" logic for atom maps, the behavior when checking the maps is as follows: the query reactants can be bound to any classes in the target. These bindings form the set of allowed product bindings. The product query atoms are then tested against this list. If all of the product atoms pass, then the path is a match. The effect of this procedure is to provide the "logical-OR" semantics for atom maps within the simple implementation. The downside of this implementation is that it can be confusing to the user. Fortunately, the simple pairwise atom maps will suffice for most users.

Finally, atom map labels in molecule SMARTS and unpaired atom map labels in reaction SMARTS are ignored. Stated another way, since the atom maps express the idea of a global association of atoms across a reaction, atom maps on a molecule query have no meaning. Similarly, a lone atom map on a reaction atom which doesn't correspond to any other atoms in the query has no meaning. In both of these cases, the query is identical to the query written without the meaningless atom maps.

In recursive SMARTS, reaction expressions are not allowed. The reasons for this are twofold: first, it isn't clear that the meaning of a recursive SMARTS for a reaction would have any useful expressiveness and second, there is a practical problem with the lexical definitions of reactions: given the strict left-to-right definition of reactant-agent-product, how would one express a product atom in a vector binding?? Of course we can change the syntax for recursive SMARTS or reactions to accommodate this if it becomes clear that it is useful.

4.7 SMARTS Versus SMILES

All SMILES expressions are also valid SMARTS expressions, but the semantics changes because SMILES describes molecules whereas SMARTS describes patterns. The molecule represented by a SMILES string is usually, but not always, matched by the same string when used as a SMARTS.

SMILES is interpreted as a molecule, and it is the resultant molecule (not the SMILES string) which is subject to searching. Similarly, SMARTS is interpreted as a pattern; it is this pattern (not the SMARTS string) which is matched against molecules. For instance, the SMILES "C1=CC=CC=C1" (cyclohexatriene) is interpreted as the benzene molecule. This molecule will be matched by the SMARTS c1ccccc1, which is interpreted as the pattern "6 aromatic carbons in a ring". The SMARTS "C1=CC=CC=C1" makes a pattern ("six aliphatic carbons in a ring with alternating single and double bonds") which will *not* match benzene. It will, however, match the nonaromatic phenylate cation with SMILES C1=CC=CC=[CH+]1.

When atoms are specified without brackets in SMILES, default values are used; in SMARTS, unspecified properties are not defined to be part of the pattern. For instance, the SMILES O means an aliphatic oxygen with zero charge and two hydrogens, i.e. water. In SMARTS, the same expression means any aliphatic oxygen regardless of charge, hydrogen count, etc, e.g. it will match the oxygen in water, but also those in ethanol, acetone, molecular oxygen, hydroxy and hydronium ions, etc. Specifying [OH2] limits the pattern to match only water (this is also the fully specified SMILES for water).

There are a few anachronisms in most SMILES interpreters which can also lead to confusion. Some SMILES interpreters allow implicit hydrogens to be added as explicit atoms on input as a shortcut. E.g., the SMILES for 1H-pyrrole is "[nH]1cccc1" which is matched by itself as SMARTS and by "n1cccc1". The current Daylight SMILES interpreter will also accept "Hn1cccc1" for (not very good) reasons of historical compatibility; this generates the same (hydrogen-suppressed) molecule as does "[nH]1cccc1" and is matched by the same SMARTS. However, the SMARTS "Hn1cccc1" does not match this molecule.

Most SMARTS expressions are not valid SMILES expressions. For instance, the string "cOc" is a valid SMARTS, matching an aliphatic oxygen connected to two aromatic carbons as part of a larger molecule (e.g. diphenyl ether). However, "cOc" does not describe a molecule per se, and is therefore not a valid SMILES.

4.8 Efficiency Considerations

The Daylight 4.x SMARTS Toolkit provides a function, `dt_smarts_opt()`, which automatically optimizes a SMARTS by reordering, expanding, and/or consolidating atom and bond expressions. Programs which use this feature (e.g. the Merlin program) can be expected to be near optimal in terms of the time used to search typical organic structures.

When this optimization method is not used, there are some things which can be done to facilitate efficient (fast) searching operations using SMARTS. It is important to recognize that SMARTS target strings are

processed in strictly left-to-right order. For this reason, substantial gains in speed can be achieved by following these guidelines:

- Uncommon atoms or bond arrangements should be placed early in SMARTS targets.
- In an "and-expression", the less common atom or bond specifications should be placed early.
- In an "or-expression", the less common atom or bond specifications should be placed last.

4.9 Examples

cc	any pair of attached aromatic carbons
c:c	aromatic carbons joined by an aromatic bond
c-c	aromatic carbons joined by a single bond (e.g. biphenyl).
O	any aliphatic oxygen
[O;H1]	simple hydroxy oxygen
[O;D1]	1-connected (hydroxy or hydroxide) oxygen
[O;D2]	2-connected (etheric) oxygen
[C,c]	any carbon
F,Cl,Br,I]	the 1st four halogens.
[N;R]	must be aliphatic nitrogen AND in a ring
[!C;R]	(NOTaliphatic carbon) AND in a ring
[n;H1]	H-pyrrole nitrogen
[n&H1]	same as above
[c,n&H1]	any arom carbon OR H-pyrrole nitrogen
[c,n;H1]	(arom carbon OR arom nitrogen) and exactly one H
!@	two atoms connected by a non-ringbond
@;!:	two atoms connected by a non-aromatic ringbond
[C,c]=,#[C,c]	two carbons connected by a double or triple bond

Go To Next Chapter... 5. SMIRKS - A Reaction Transform Language

Daylight Headquarters Other Locations
Daylight Chemical Information Systems, Inc.
PO Box 7737, Laguna Niguel, CA 92607
tel +1 949-831-9990 - fax +1 949-831-9902 - info@daylight.com