ARTICLES

SYBYL Line Notation (SLN): A Single Notation To Represent Chemical Structures, Queries, Reactions, and Virtual Libraries

R. Webster Homer, Jon Swanson, Robert J. Jilek, Tad Hurst, and Robert D. Clark*

Tripos Informatics Research Center, 1699 S. Hanley Road, St. Louis, Missouri 63144

Received December 19, 2007

SYBYL line notation (SLN) is a powerful way to represent molecular structures, reactions, libraries of structures, molecular fragments, formulations, molecular queries, and reaction queries. Nearly any chemical structure imaginable, including macromolecules, pharmaceuticals, catalysts, and even combinatorial libraries can be represented as an SLN string. The language provides a rich syntax for database queries comparable to SMARTS. It provides full Markush, R-Group, reaction, and macro atom capabilities in a single unified notation. It includes the ability to specify 3D conformations and 2D depictions. All the information necessary to recreate the structure in a modeling or drawing package is present in a single, concise string of ASCII characters. This makes SLN ideal for structure communication over global computer networks between applications sitting at remote sites. Unlike SMILES and its derivatives, SLN accomplishes this within a single unified syntax. Structures, queries, compounds, reactions, and virtual libraries can all be represented in a single notation.

INTRODUCTION

SYBYL Line Notation (SLN)[1] was originally inspired by SMILES, but there are now many differences between the two. The most fundamental differences are in their handling of valence and aromaticity. Whereas SMILES supports implicit valences for atoms that are intrinsic to the language, SLN makes no assumptions as to an atom's valences; the choice to represent a particular set of bonds as aromatic or in an alternative Kekule form is left up to the user and the application. There are searching and database management routines in the UNITY[2] program suite that interconvert Kekule and aromatized structures, but the details of how those routines work is external to the language itself.

SLN's flexibility with respect to valences allows it to do the work of both SMILES and SMARTS. In SMILES aromaticity is treated as a property of atoms, while SLN treats aromaticity as an explicit property of bonds. Treating aromaticity as an atom property not only gives SMILES an economy of expression but also requires introduction of new syntax elements into SMARTS, including two different implied bond types, single and aromatic bonds. While not necessarily a problem for SMILES itself, this introduces ambiguity in the SMARTS query language. For example, the SMARTS string "cc" can represent either two aromatic carbons with an aromatic bond between them or two aromatic carbons joined by a single bond.

SLN has evolved substantially since the publication of its defining syntax in 1997.[1] New features have been added (see below), and some existing features have been modified in the interim. SLN continues to be a simple and concise way to store and communicate chemical structures, substructure queries, combinatorial libraries, and chemical reactions.

Here we provide a definitive description of current syntax for the SLN notation, its representation of atoms, bonds, rings, and chains. Many new features are extensions of the core syntax or exploit the intrinsic versatility of the notation. Combinatorial SLNs (cSLNs), for example, represent an elaboration of Markush atoms to file-based fragment lists, whereas the new capabilities for handling relative stereochemistry and stereomixtures take advantage of the inherent flexibility of atom and bond attributes.

SLN has been enhanced by the addition of new syntax elements for specifying reactions with atom-to-atom mappings and reaction centers. Markush syntax has been improved and simplified to better support combinatorial library specification, and Markush definitions can now be stored as database queries or as files of SLNs. New substructure query attributes such as "**is=**" and "**not=**" have been added as well as a new Group atom for queries, the **Rx** atom. Finally, an expanded syntax for specifying and searching incompletely specified stereochemistry has been added.

SLN OVERVIEW

An *SLN* is a character string that specifies how the atoms in a molecular structure or set of associated molecular structures are connected to one another as well as attributes of those atoms, of the bonds connecting them, or of the aggregate structures according to the specification of the SYBYL Line Notation language. A *connection table* (CT) is a fully connected structure within an SLN. Each SLN

* Corresponding author e-mail: bclark@bcmetrics.com. Current address: Biochemical Infometrics, 827 Renee Lane, St. Louis, MO 63141.

contains one primary CT, but it may also contain supplemental CTs, e.g., to define reactants or products. *Markush atoms* represent one or more substructures within a CT.

Atoms. An *atom identifier* is a substring that begins with an upper case letter; subsequent characters that are lowercase letters, digits, or the underscore character are included in the identifier.

Elemental atoms are atom identifiers that are valid atomic symbols. **C** specifies a carbon atom and **Cl** a chlorine atom, for example. Hydrogens must be specified in SLN, as no presumptions are made regarding valences. Shorthand notation modeled on standard organic notation is available for hydrogens in which the specification of the parent atom is followed by the letter **H** which may be followed by an optional count, as in **CH3** or **NH2**. This eliminates the need for explicit branches to hydrogen atoms. Hydrogen shorthand is not available for bridging hydrogens with bonds to multiple atoms or if the hydrogen has an attribute such as a charge or isotope specified. If any of these conditions are met, the connectivity of the hydrogen atoms must be specified in the same manner as for other elemental atoms.

Making hydrogen atoms explicit helps to dissociate SLN syntax from implicit chemical semantics, making it easier to parse. In contrast, some SMILES parsers fail to process **n1cccc1** properly, since a valid SMILES for pyrrole requires that the hydrogen be specified explicitly, e.g., as **[nH]1cccc1**.

Note that this shorthand notation can *only* be used for hydrogens; **CHCl3** is not a valid SLN.

Bonds. Most single bonds are implicit (**CH3CH3** represents ethane, for example), but other bonds are indicated by placing a special character between the bonded atom pair. The bond characters include the following:

•A hyphen ("**–**") indicates that a single bond exists between the flanking atoms - e.g., **CH3–CH3** is as an alternative representation for ethane. Such an explicit specification of a bond to a hydrogen atom using "**–**" prevents it from being parsed as hydrogen shorthand and may necessitate additional parentheses in the SLN. Explicit bonds are necessary to accommodate the three-center bonds involving hydrogen such as is found in diboranes. The formulation **BH3BH3** is equivalent to **BH3-BH3**; it indicates that each hydrogen is bonded to only one boron and that there is a single bond between the two boron atoms, neither of which is the case. In fact, the two boron atoms participate in two three-center bonds, each of which includes a bridging hydrogen. Such a three-center bond is represented as **B−H−B**[3] in SLN.

•An equals sign ("**=**") represents a double bond, as in **CH2=CH2** for ethylene.

•A pound sign ("**#**") represents a triple bond, as in **CH#CH** for acetylene.

•A period ("**.**") indicates the start of a new part of a structure. In essence, it represents a zero order bond, as in **CH3NH2.HCl** for methylamine hydrochloride.

•A colon ("**:**") indicates an aromatic bond; examples are given below.

Figure 1 shows examples of branched structures and their SLNs. Branches are indicated by parentheses. The parentheses set off a group which is connected to the atom immediately preceding the group. A final pair of parentheses is implicit and is ordinarily omitted; the atom or group which follows the closing parentheses is also connected to the atom which immediately precedes the open parenthetical group

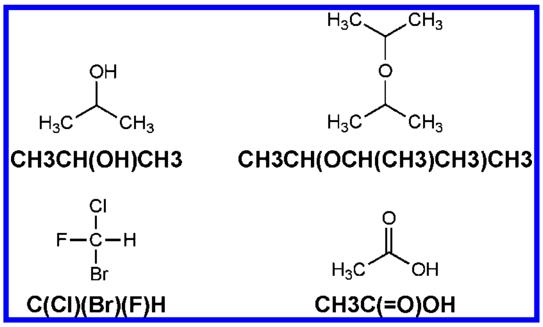


Figure 1. Specifying connectivity in branched molecules in SLN.

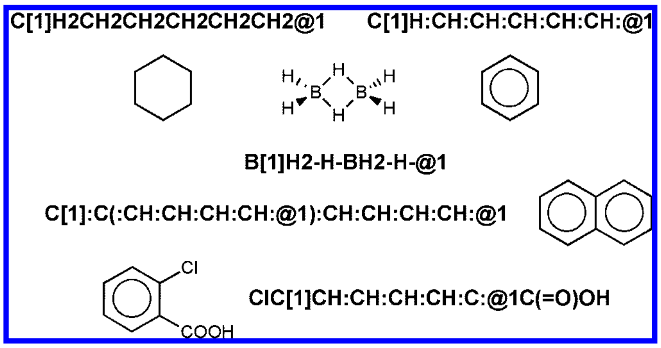


Figure 2. Describing cyclic structures in SLN.

(e.g., **CH3C(F)(F)F** is preferred over the otherwise equivalent **CH3C(F)(F)(F)**). Note that hydrogen atoms expressed in hydrogen shorthand are not considered atoms of attachment. Several sets of groups in parentheses may follow an atom, indicating multiple groups attached to the atom. The parentheses may be nested for branches that have branches.

Any atom that is the target of a ring closure must have an assigned ID number. The ID number is a positive integer placed in square brackets after the atom identifier. No two atoms in an SLN may have the same ID. Typically the atom ID is the index of the atom in the SLN string. The atom ID comes before any other atom attribute.

An "@" is used to indicate ring closure and is preceded by the bond type and followed by the ID of a previously defined target atom. Parentheses around the ring closure are allowed for clarity but are not strictly necessary, e.g. **O[1]CH2CH(@1)CH3** for propylene oxide. Figure 2 shows other cyclic structures with their SLNs. Note that aromaticity has been expressed explicitly in the 2D structures shown in Figure 2, just as it is in the SLNs. Aromatic structures elsewhere in the paper are shown in Kekule form for convenience but expressed as aromatic SLNs. In practice, structures are generally either standardized by aromatization to remove ambiguity before being entered into a corporate database or are aromatized on-the-fly during UNITY searches.

ATOM ATTRIBUTES

Atom attributes are enclosed in square brackets ([...]) and, like all other attributes in SLN, are not case sensitive. There are three kinds of attributes which can be specified: the atom ID, static attributes (see below), and expression attributes. If an atom ID is specified, it must be the first attribute. An atom ID is separated from static attributes by a semicolon. Static attributes are separated from each other by a semicolon, as are expression attributes. Within the attribute list, expression attributes follow static attributes and atom IDs (if any)

Table 1. Structural Atom Attributes

attribute[a]	interpretation
+n	A shorthand form indicating a charge of +*n*. Equivalent to **charge=***n* (see below). "+" is equivalent to "**+1**", whereas "**+0**" denotes an atom bearing no formal charge.
–n	A shorthand form indicating a charge of -*n*. Equivalent to **charge=-***n* (see below). "-" is equivalent to "−1".
*	Shorthand indication of a doublet spin state; equivalent to **spin=d** (see below).
charge=*n*	Specifies formal charge. and '**-**'indicates a unit of negative charge. **+0** indicates a charge of 0. Example: **Ca[+2]** specifies a calcium dication.
I=	Specifies an atom isotope. **I=14** specifies isotope 14. Example: **C[I=14]** denotes ^{14}C. Only non-negative integer values are allowed.
fcharge=	Specifies a partial charge.
s=	Specifies stereochemistry at tetrahedral atoms. Allowed modes are **R** or **S** (CIP configurations); **N** or **I** (normal and inverted with respect to atom ID sequence); **D** or **L**; or **U** (unknown). Allowed modifiers are **E** (explicit); ***** or **R** (relative); or **M** (mixture).
spin= *or* *****	Specifies that an atom is a radical. A value of **s** indicates a singlet, a value of **d** signifies a doublet, and a value of **t** indicates a triplet.
other	Attributes bearing names that have not already been defined (nonreserved names) and that do not have associated value assignments are interpreted as user-defined Boolean atom attributes. Valid identifiers begin with an alphabetic character and consist of letters, digits, underscore ('_'), or periods. Case is ignored.
other=	Nonreserved attribute names with assigned values are interpreted as user-defined valued atom attributes. Assigned values are treated as simple text strings and can include any combination of characters and spaces provided only that the string is set off in quotation marks.

[a] Reserved attribute names are indicated in bold face type, whereas generic terms (such as "*n*" for a positive integer) are shown in italics. Attributes followed by an equals sign ("=") must be assigned a specific value; other attributes are Boolean.

and are separated from them by a colon. Table 1 lists atom attributes used in SLNs that represent individual structures, e.g. as searchable database entries. Examples of how they can be used include **CH3CH2C[*]HCH2Br** for bromobutyl radical; **CH3C(=O)O[–].Na[+]** for sodium acetate; **NH2C-[s=N]H(CH3)C(=O)OH** for alanine; and **ClC[spin=s]Cl** for dichlorocarbene.

Specifying Stereochemistry. The advent of sophisticated asymmetric synthesis methods and chiral separation technologies has meant that scientists can include stereochemistry as a design consideration in the development of biologically active compounds. In order to better leverage stereochemistry, stereochemical annotation in SLN has been expanded to include support for relative stereochemistry and full specification of enantiomeric composition. SLN and Conversational SMILES[4−6] both allow marking atom chirality based on Cahn-Ingold-Prelog[7,8] (CIP) rules. In the case of SLN, chiral classification is best expressed in N/I notation, where the atom priority is based on the order of atoms in the SLN string. There are several other examples of prioritization by atom position. Daylight Isomeric SMILES extends SMILES with @ and @@ atom flags, which define atom weighting based on atom order. MOL files[9] have an atom parity flag that defines atom chirality as either even or odd. Except for the provision of an unknown stereo atom attribute, there has been little provision for specifying structures where the absolute stereochemistry at each chiral center is not fully resolved or independently specifiable.

Registering stereoisomers into a database requires a broader context than that encompassed by "absolute" and "unknown" atom stereo attributes alone. One needs to be able to distinguish between several possibilities that a structure can represent: a single pure stereoisomer for which absolute stereochemistry is or is not known; a mixture of stereoisomers; or perhaps that nothing is known about the stereochemistry other than that the structure contains chiral centers. To provide this additional functionality SLN's chirality specification now includes support for incompletely resolved stereochemistry and mixtures. The new atom stereochemistry specification consists of the following three syntactic components:

1. Individual chirality designations such as **R**, **S**, **N**, **I**, **D**, **L**, or **U** that specify how groups are oriented in space around a chiral center.

2. Stereo mode modifiers that distinguish between atoms where chirality is known explicitly (absolute chirality is known) or when chirality is only known relative to other chiral centers. Modifiers include the following: **E** (explicit), when exact stereochemistry is known; ***** or **R** (relative), when exactly one isomer present but the absolute stereochemistry is unknown; and **M** for a mixture of stereoisomers.

3. Optional group modifiers allow specification of multiple groups within a molecule where relative stereochemistry is only known within the group. A group identifier cannot be used with either the unknown (**U**) chirality designator or the explicit (**E**) stereo mode modifier, nor can it be used in connection with **D** and **L** designations.

Allowed values for the stereochemistry attribute are as follows:

• **RE** or **SE**, where absolute stereochemistry is specified according to the CIP rules. For backward compatibility, the designation **S=SE** (**S=RE**) is treated as equivalent to **S=S** (**S=R**).

• **NE** or **IE**, where absolute stereochemistry is known and is specified based on ordering of atoms in the SLN. It is the same as in the *R/S* convention, except the priority of the attached atoms is determined by the order of their appearance in the SLN. **N** (normal) indicates clockwise presentation of the first three groups when considered in the order in which they appear in the SLN (i.e., in order of atom ID) when the "last" atom (the one with the highest atom ID) is positioned away from the viewer, whereas **I** (inverted) indicates a counterclockwise presentation. For backward compatibility, the designation **S=IE** (**S=NE**) is treated as equivalent to **S=I** (**S=N**). Attached atoms may be elemental, macro, query, or Markush atoms (see below).

• **DE** or **LE**, when absolute stereochemistry is known and specified based upon the similarity of the chiral center to D-glyceraldehyde. This form is not supported with mixture or relative group modifiers.

• **UE**, when the chirality of a center is unknown. The compound may be a single stereoisomer or it may be a mixture of stereoisomers.

• **R*** or **S***, where the center is unresolved. These represent single pure stereoisomers whose absolute configuration is unknown, but configuration of one chiral center is known relative to another chiral center, as specified using CIP rules. Normally, these attributes are applied only with multiple relative chiral centers. If a molecule contains two chiral centers, one **R*** and the other **S***, this represents either a single stereoisomer that is *R* at the first center and *S* at the other ***or*** a single isomer that is *S* at the first center and *R* at the other. If a group is not specified, the centers will be treated as being in group 0.

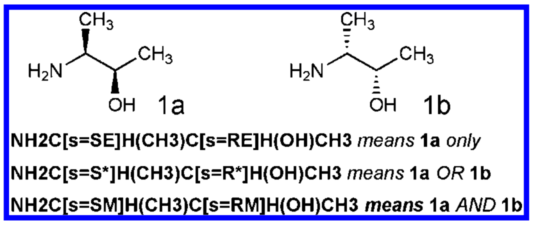
• **N*** ("normal) and **I***" ("inverted") indicate clockwise and clockwise presentation, respectively, when the center is unresolved, just as in the absolute case described above.

• **U***, where the center in question represents a single pure isomer but is of unknown stereochemistry.

• **RM** or **SM**, where a mixture of stereoisomers is present and is specified using CIP rules. These attributes are normally applied only with multiple relative chiral centers. If a molecule contains two chiral centers, one **RM** and the other **SM**, this represents a mixture of the stereoisomer that is *R* at the first center and *S* at the other ***and*** the isomer that is *S* at the first center and *R* at the other. If a group is not specified, the centers will be treated as being in group 0.

• **NM** or **IM**, for a mixture of stereoisomers specified based on ordering of atoms in the SLN.

• **UM**, for an unknown mixture in which both possible orientations at the atom are present.



NH2C[s=SE]H(CH3)C[s=RE]H(OH)CH3 *means* **1a** *only*

NH2C[s=S*]H(CH3)C[s=R*]H(OH)CH3 *means* **1a** *OR* **1b**

NH2C[s=SM]H(CH3)C[s=RM]H(OH)CH3 ***means*** **1a** *AND* **1b**

Figure 3. Specifying chirality in SLN.

Figure 3 illustrates the meaning of relative stereochemistry in SLN. SLN allows users to specify what *they know they do not know* about a chiral center. For example **S=UE** indicates that they know that they do not know the chirality of the center nor whether it is a mixture or pure compound. Users can specify that they know that an unknown center is pure (**S=U***) or a mixture (**S=UM**).

SLN can differentiate *cis* and *trans* relative stereochemistry in ring systems. For example, **C[1]H2CH2C[s=N]H-(CH2CH2C[s=N]H@1CH3)CH3** denotes a *cis*-cyclohexane and **C[1]H2CH2C[s=N]H(CH2CH2C[s=I]H@1CH3)-CH3** represents a *trans*-cyclohexane. *Meso* compounds such as tartaric acid can readily be represented in SLN, using absolute stereo attributes:

HOC[s=I]H(C[s=I]H(OH)C(=O)OH)C(=O)OH.

Wedged bonds are often used to indicate chiral centers in the literature, but until recently, if the molecule was not explicitly flagged as "chiral" in the file formats using this convention, all wedge bonds were taken to indicate mixtures.[8]

The V3000 MOL File format was extended in 2003[10] to support relative stereochemistry, using an approach analogous to that taken in SLN.[11] Chiral centers are marked with wedge bonds. The user uses a molecule sketcher to label the center as being: absolute; OR pure but with an unknown absolute orientation; or AND(&) a mixture. OR and AND centers must also be given a number indicating its group affiliation. Each grouping is placed into its own collection, with absolute centers placed into a collection of their own. For the most part SLNs with relative stereochemistry can be translated to V3000 MOL format and back to SLN with no loss in information. Information is lost in some cases because there is no way to distinguish between unknown mixtures, pure centers whose chirality is unknown, and centers whose purity and conformation are *both* unknown in the MOL file format.

SMILES has no syntax comparable to SLN's support for relative stereochemistry. It does provide a notation to support absolute stereochemistry and also supports the specification of the higher order (beyond tetrahedral) stereochemistry found in organometallics. In SMILES the atom attributes "@" and "@@" are used to designate chirality much as SLN uses "**S=I**" and "**S=N**".

Bond Attributes. Bond attributes are enclosed in square brackets ([...]) and are case insensitive. Table 2 summarizes bond attributes used for specifying structures. One critical attribute specifies bond stereochemistry, which is defined similarly to atom chirality. The stereo attribute **s** is followed by the "=" and a specification token:

• **C** (*cis*) or **T** (*trans*) presumes that the connected atom priority order is based on CIP priority, as do **E** (*entgegen*) or **Z** (*zusammen*).

• **N** (normal) or **I** (inverted) presume that connected atoms priority is based on the order in which they appear in the

Table 2. Structural Bond Attributes

attribute[a]	interpretation
s=	Indicates stereochemistry about a double bond. Values of **C** (*cis*), **T** (*trans*), **E** (*entgegen*), and **Z** (*zusammen*) define spatial distribution of substituent based on CIP priority; **N** (Normal) and **I** (Inverted) define a spatial distribution of substituent based on their position in the SLN, with **N** indicating that the groups appearing first on either end of the double bond (i.e., those on each side with the lowest atom IDs) are *trans* to each other and **I** indicating that they are *cis*; **U** indicates that the stereochemistry about the double bond is unknown, which may mean that a single configuration is present or that the SLN represents a mixture of configuration; and **U*** represents a single configuration where the actual stereochemistry about the double bond is unknown.
other	Nonreserved attribute names without assigned values are interpreted as user-defined Boolean bond attributes.
other=	Nonreserved attributes name with assigned values are interpreted as user-defined valued bond attributes. Assigned values are treated as simple text strings, as in **C-[bondstretch=20]C**.

[a] Literal attribute names are indicated in bold face type, whereas generic terms (such as "*n*" for a positive integer) are shown in italics. Attributes followed by an equals sign ("=") must be assigned a specific value; other attributes are Boolean.

SLN. **N** means the dominant groups on the ends of a double bond (lowest atom IDs) are *trans* to each other, and **I** means that they are *cis*.

• **U**: Stereochemistry about the double bond is unknown. The structure might represent a single configuration or may be a mixture of both configurations.

• **U***: Stereochemistry about the double bond is unknown; however, the structure represents a single configuration.

SLN does not currently support *syn* and *anti* nomenclature for oximes and other imino compounds. Users can assign their own bond types to bonds, however. This is accomplished with the bond attribute "**type=**" which is followed by a string identifier. For example

C[1]H:CH:CH:CH:CH(:@1)=[type=dative]
Pt(=[type=dative]C=O

is one way to represent the adduct between cyclopentadienyl cation, palladium, and carbon monoxide.

Connection Table Attributes. Just as standard atoms and bonds can have attributes, so too can structures. SLN provides a rich vocabulary of CT attributes as well as the ability to define others. This allows arbitrary data to be associated with an SLN. CT attributes follow the SLN entity being described and are set off by angle brackets (<···>). When multiple attributes are present, they are separated from each other by semicolons. Attributes in the brackets can appear in any order and are not case sensitive. There are three classes of CT attributes: valued, reorderable, and private. A valid CT attribute name begins with a letter and can contain letters, digits, underscore characters, and periods.

The syntax for separating the name from the value is the means by which the different attribute classes are differentiated. When the name is separated from the value only by an equal sign ("="), it is a simple valued attribute. When the name is separated from the value by a colon and an equals sign (":="), the attribute is a "re-orderable" attribute. Reorderable attributes contain lists of atom IDs. Software that can reorder the CT (e.g., for canonicalization) must keep these attributes in sync with the original order of the atoms in the CT. Additionally, a string value can be associated with the list of atoms. For example the following associates "**red**" with specific atoms in an SLN:

CH3CH2OCH2CH3<good_ones:=red:1,5,8>

"Private" attributes represent a third class of CT attribute. These retain information from other molecule formats to be used if a molecule was translated back into the format from which it came. Private attributes are simple name/value pairs. They are distinguished from user-supplied attributes by placing " ^=" between the name and the value. Table 3 lists names and definitions of reserved CT attribute names. Examples include the following:

N[1]H:CH:CH:CH:CH:@1<EXPLICIT_H:=2>;
BrC[s=ie]H(F)Cl<wedge_up:=2,4;name="freonX">;
CCC<COORD2D=(2.821,−4.900),(3.237,−4.187),
(3.427,−3.729)>;
and **FS(F)(Cl)(Cl)(Cl)Cl<sym^=axial>**

The private attribute in the last example might be used in a file conversion program to incorporate the octahedral stereochemistry of a sulfur hexahalide into the SLN.

Specifying Compounds. SLN supports syntax suitable for specifying compounds, mixtures, and formulations. In this form, global CT attributes such as REGID or NAME are at the beginning of the SLN string. These are followed by the atom bond connectivity specification for each compound component. Each component has its own set of CT attributes. If a component has no CT attributes it must use empty angle brackets ("<>") to separate components. The following example illustrates the syntax:

<regId="Sodium Chloride";name="Table Salt">
Na[+]<coord2D=(0.000,0.000)>
Cl[-]<coord2d=(0.500,0.500)>

Applications can define CT attributes for each component to support mixtures and formulations. Applications can even assign meaning to the order of the components in the SLN to support formulation work.

Structure databases often include information about compounds for which the structure is completely unknown. SLN supports this with CT attributes with no atom connectivity. An example of such a null structure is **<regId="SCS000123-A";COCKLEBUR_ACTIVITY=45>**

Table 3. Glossary of Connection Table (CT) Attributes

attribute[a]	interpretation	value
REGID=[b]	Registration ID for the SLN	string
RXNID=	Reaction ID for an SLN reaction	string
name=	Compound name	string
coord2d=	Collection of 2D coordinates for the atoms in the CT. There will be two floating point numbers for each atom in the CT. Number pairs are generally set off by parentheses for clarity.	list of floating point numbers
coord3d=	The 3D coordinates for the atoms in the CT. There will be three floating point numbers per atom in the CT. Number triplets are generally set off by parentheses for clarity.	list of floating point numbers
other	A user-defined Boolean CT attribute. The name can be any valid identifier (a letter followed by alphanumeric characters or an underscore) that is not among the reserved CT attributes listed in this table.	none
other=	A user-defined valued CT attribute. See "*other*" for constraints on which names can be used.	string
other:=	A user defined reorderable Boolean attribute of a group of atoms. See "*other*" for constraints on which names can be used. An example is "**<Explicit_H:=2,4,5,6,7>**"	list of atom IDs
other:=**text**:	A user defined reorderable valued attribute for a group of atoms, where *text* specifies the value applied to each atom in the appended list. See "*other*" for constraints on which names can be used. "**<Exchange:=fast:2,4>**"	value to assign plus a list of atom IDs
wedge_up:=	A reorderable attribute. The first atom in the list is the FROM atom, i.e., the chiral center. This denotes a solid ("up") wedge bond.	list of two atom IDs
wedge_down:=	Same as "**wedge_up:=**", except that the wedge is to be down (hashed).	list of two atom IDs
wedge_either:=	Same as "**wedge_up:=**", except that the wedge is rendered as a "squiggle" or "both" bond.	list of two atom IDs
v=	Valence attribute for Markush CTs. This attribute lists the atom ids in the Markush choice to indicate the atoms which attach to atoms external to the Markush.	Markush

[a] Literal attribute names are indicated in bold face type, whereas generic terms (such as "*n*" for a positive integer) are shown in italics. Attributes followed by an equals sign ("=") must be assigned a value; other attributes are Boolean. [b] Capitalized attributes can only appear once in each SLN.

COMPLEX ATOMS

Macro atoms are shorthand notations for groups of atoms such as amino acids. A macro atom starts with an uppercase letter, followed by lowercase letters, digits, or underscore characters. Structures expressed in macro atom format contain all the information needed to be expand into full atom format. Examples include **Monomer_1, Ala, Gly**, and **His**.

The syntax for a macro atom definition is **{***macro_name***:** *sln***<***list***>}**,[12] where *macro_name* is the macro atom name, *sln* is the atom bond connectivity string, and *list* is a list of CT attributes, the most important to macro atom definitions being the list of valence atom IDs. Examples are **{Ala: NHC[s=l]H(CH3)C=O<*v*=1,9>}** and **{Gly:NHCH2C=O <*v*=1,6>}** for alanine and glycine peptide residues, respectively.

References to macro atoms are made by using the macro name in place of the group it represents. The syntax is *macro_name*[*macro_attributes*], where *macro_name* is the name of the Macro and *macro_attributes* is a list of Macro atom attributes. Just as for any other atom, users can specify an atom ID, static attributes, and expression attributes. Macro atoms also support the static valence attribute. Examples include **HHisCys[2]IleCys[v=1,3,2]@2Gly[3]OH**, **HPva-PvaPvaH{Pva:CH2CHOH<*v*=1,4>}**, and **HAlaHisGly-OH**.

The order in which the bonds attached to the macro atom appear in the CT determines which valence of the macro atom they satisfy. In the first example above, **Ala** is connected to two atoms: a hydrogen atom and a histidine residue (**His**). The bond to the hydrogen appears first in the CT, so it is connected to the first valence position of the **Ala** atom, which is its *N*-terminus. The bond to the **His** macro atom appears second, so the histidine residue is attached to the second valence of the alanine residue, the *C*-terminus. This type of connectivity is referred to as the "natural order" and is specified by the macro atom definition.

If the connectivity does not take the natural order for the macro atom, it must be specified using the static *atom* attribute, "**v=**". This designation is followed by a comma separated list of the valence numbers for the connected atoms in order of the connected atoms. For example, if an amino acid sequence were to be specified in reverse order, the modifier **[v=2,1]** would be used. **HOGly[v=2,1]Ala [v=2,1]His[v=2,1]H** is equivalent to **HHisAlaGlyOH**, for example.

Note the valence specification in the above example. The "**v=1**" indicates that the macro is connected to other atoms with bonds to the first atom in the macro definition valence list.

The valence of a macro atom is fundamentally different from that of elemental atoms in organic compounds. Whereas all valences of a nonchiral tetrahedral carbon are equivalent, the valences for a macro atom are distinct and may or may not be on the same atom. Macro atom valences are specified with the static atom attribute "**v=**" which is followed by comma-separated list of the valence numbers for the connected atoms in the order of the connected atoms. In the example above, **Gly[v=2,1]** indicates that first bond to **Gly** is to the second valence atom in the definition of **Gly**, the carbonyl; and the second bond to **Gly** is to the first valence atom in **Gly**, the nitrogen.

If the macro definition has no valence specified, the connections to the macro are assumed to be in the same order as the atoms in the definition. For example, in the macro definition **{Peroxide:O–O}**, no valence designation is necessary, since the first valence is connected to atom 1 and the second is connected to atom 2. In the absence of a valence CT attribute, all atoms with open valences are considered potential attachment points.

MARKUSH ATOMS

Markush atoms are shorthand notations for sets of alternative substructures including cases such as **Hal** (halogens) or **Het** (heteroatom) where each substructure consists of a single atom. Like any other atom type, a Markush name is case sensitive and begins with an uppercase letter which is followed by zero or more lower case letters, digits, or underscores. Any group atom name is allowed so long as it does not conflict with an elemental atom type, a reserved macro atom name (**Any**, **Hev**, or **Lp** (lone pair); **R**, **X**, or **Rx** or some a derivative query attachment atom name (see below); or a name that is already in use for another Markush. UNITY and SYBYL applications preload several standard global Markush definitions from an editable external file (*markush.defs*), including **Hal** and **Het**, but these are not an innate part of the language.

Markush atoms are defined by specifying a list of possible fragments which may be matched. The simplest syntax for a Markush definition is

{*markush_name*:*sln1*<*ct_attrs*>|*sln2*<*ct_attrs*>|. **..**}

where *markush_name* is the name of the Markush being defined; *sln1*, *sln2*... are the allowed choices for the Markush definition, each of which is parsed into a separate CT; and *ct_attrs* specifies the valence(s) and range of occurrence for the CT. Other CT attributes may be included as well. **{Bead:}** is an example of a Markush suitable for use in solid phase chemistry.

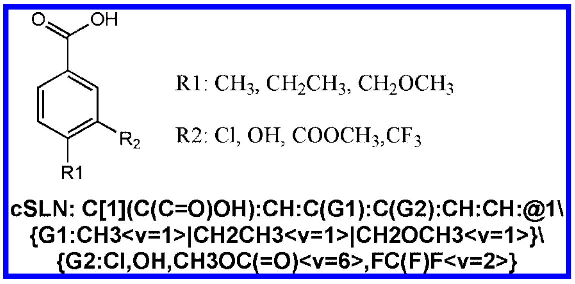
A Markush or macro atom can be defined globally prior to use in an SLN, or it can be included in the SLN specification itself. If the Markush atom is defined as part of the SLN, the scope of the definition is the SLN, and that particular Markush is undefined if referenced in a different SLN.

References to Markush atoms are made by using the Markush name as an atom. The syntax is identical to Macro atom references.

SLN allows Markush atoms to be defined in different "namespaces". When this is done, otherwise ambiguous Markush names will not collide with elemental symbols.

H{@AA}GCA{@}OH, for example, represents a tripeptide, where **G** is defined as **{@AA:G:C(=O)CH2NH <*v*=6,1>}**; **C** is defined as **{@AA:C:NHC[s=l]H(C=O) CH2SH<*v*=1,5>}**; and **A** is defined as **{@AA:A: C(=O)C[s=l]H(CH3)NH<*v*=9,1>}**

In the example above the **{@AA}** switches to a new **AA** namespace within which the macro atoms **A**, **C**, and **G** are



```
cSLN: C[1](C(C=O)OH):CH:C(G1):C(G2):CH:CH:@1\
{G1:CH3<v=1>|CH2CH3<v=1>|CH2OCH3<v=1>}\
{G2:Cl,OH,CH3OC(=O)<v=6>,FC(F)F<v=2>}
```

Figure 4. An example of a combinatorial SLN.

defined as indicated. The **{@}** switches back to the default namespace. Namespaces are optional, with the default namespace being unnamed.

SPECIFYING VIRTUAL COMBINATORIAL LIBRARIES

Markushes can be used to specify virtual libraries, which allows one SLN to specify a complete virtual library. The SLN representing the structure in Figure 4 is referred to as a combinatorial SLN (cSLN). Note that the substituents in the cSLN are named **G1** and **G2**, not **R1** and **R2**. This is because **R1** and **R2** are reserved as special group atom names for use in queries (see below), just as the symbols of the elements (**H**,**O**, **Cl, Xe,** etc.) are reserved in SLNs for structures and queries; hence they cannot be used as Markush names. **R**, **X**, and **Rx** are also reserved group names, as are their indexed variants.

A drawback to this approach is that the cSLNs can become very large and difficult to modify. Software used to create cSLNs often worked from a single set of files containing lists of fragments, with each file containing all of the choices for a particular Markush. This meant that the information was duplicated in the fragment files and the cSLN. To streamline this workflow and to limit the duplication of data a new type of Markush was introduced, the *FILE Markush*.

In a file-based Markush, the Markush is defined by a file specification. The file holds SLNs that represent fragments (fSLNs). In this form the definition contains a filename of a text file wherein each line in the file is one Markush choice. Hence **{Mfile:@FILE="***filename***"}** defines a FILE Markush.

The ability to cross-reference fragment lists in this way comes with the administrative overhead of maintaining an external file system, a cost which can become substantial for large research organizations or where there is a need to keep the files systematically updated (e.g., in response to reagent availability). This has been addressed in part by the introduction of *TABLE Markushes*.

A TABLE Markush is used in applications that make use of relational databases. It is similar to a FILE Markush definition except that each Markush choice is a row in a table rather than an entry in a file. The TABLE definition is a SQL SELECT statement, which retrieves one Markush choice per database fetch. This provides support for database hosted virtual libraries. Each row returned by the SELECT statement is one choice:

{Mtable:@TABLE="SELECT *Markush_choice*, mark_sequence **FROM** *tbl_markush_sln* **ORDER BY** *mark_sequence***"}** (all on one line).

The order of the choices in a TABLE Markush will depend upon the presence of an ORDER BY clause in the select statement, but if order is not important the ORDER BY clause can be left out. For the returned choices to be ordered in a meaningful way the SELECT statement can select a second column to be used in the ORDER BY clause.

FILE and TABLE Markush definitions are both useful when building applications to support combinatorial chemistry. They support a variant on Markush choices known as fragment SLNs where the valence attribute is replaced by attachment atoms. Each fSLN file contains fragments for a point of variability in the core. If a core has multiple related substitutions, then the fSLN file will contain fSLNs with the appropriate number of attachments. For example, an ethyl group written with a single attachment atom is **A1CH2CH3**.

Attachment atoms serve a role similar to the valence CT attribute (**v**=), though they carry more information than the valence attribute can. Atoms bound to an attachment atom are the valence atoms. This allows for more information in the Markush part as it keeps the information about how the fragment is connected to a core. Another important difference is that this approach can support chirality attributes on the valence atom. Although valence CT attributes continue to be supported, the use of attachment atoms is preferred.

The attachment atom is specified by an "**A**" optionally followed by a unique positive integer. In an fSLN this number indicates the fragment's valence order, just as the **v**= CT attribute does. Each attachment atom can only have one bond. Attachment atoms serve as proxies for the atoms in a core to which the fragment attaches. In a substructure search query an attachment atom with no number will match any attachment atom, but an attachment atom with a number will only match another attachment atom with the same number. Examples of fSLNs include the following: **A1N[2]CH2CH2N(A2)CH2CH2@2**(*N,N*-disubstituted piperazine); **A1OCH2CH(CH3)NHA2** (*O,N*-disubstituted 2-methylethanolamine); and **A1C[s=i]H(NHCH2CH2C(=O)OA3)A2** (a potentially chiral disubstituted *N*-alkyl *β*-alanine ester). Substructures in disjoint R-groups that are mechanistically linked are separated by a period within the fSLN; **A1CH3.A2C(=O)OCH3**, for example, would appear in a Diels−Alder cycloaddition library in which methyl isocrotonate (**CH3CH=[s=Z]CHC(=O)OCH3**) was used as a reagent.

Whether encoded as fully inline cSLNs or as file based cSLNs, information about the individual choices for each substructure may be recorded as CT attributes. This can be used to track the catalogue numbers of the starting material or any other fragment based information. For example:

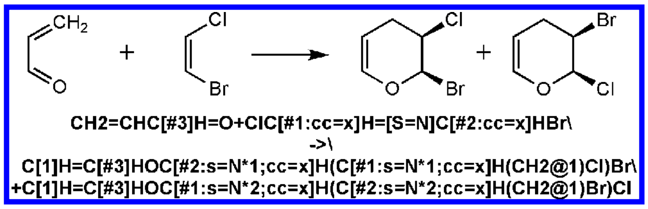
C[1]:C:C:C:N:C:@1Grx{Grx:CH3
<Ald="160−2314">|CH2CH3<Ald="895−34−2">

Markushes are also useful for specifying bioisosteres. A bioisostere is a set of atoms that can be interchanged at a site in a molecule without losing biological activity. For example, the following Markush implements a tetrazole group as a bioisostere of the carboxylate moiety:

{Carboxy_isostere:C(=O)OH<*v*=1>
|C(=O)O[-]<v=1>|C[1]:N:N:N:NH:@1<*v*=1>}

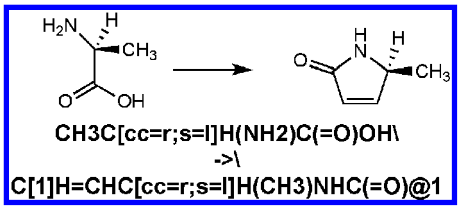
SPECIFYING REACTIONS

SLN now includes a special syntax for specifying reactions. Reactants are separated from products by the presence of "−>" which represents the reaction arrow. Individual reactants and products are separated from each other by a "+". Each reactant and product can have CT attributes. The reaction SLN as a whole can have CT attributes; when present, these global attributes precede the rest of the SLN.



```
CH2=CHC[#3]H=O+ClC[#1:cc=x]H=[S=N]C[#2:cc=x]HBr\
->\
C[1]H=C[#3]HOC[#2:s=N*1;cc=x]H(C[#1:s=N*1;cc=x]H(CH2@1)Cl)Br\
+C[1]H=C[#3]HOC[#1:s=N*2;cc=x]H(C[#2:s=N*2;cc=x]H(CH2@1)Br)Cl
```

Figure 5. Atom mapping example for a Diels−Alder reaction.



```
CH3C[cc=r;s=I]H(NH2)C(=O)OH\
->\
C[1]H=CHC[cc=r;s=I]H(CH3)NHC(=O)@1
```

Figure 6. Effect of reaction on stereochemistry in a specific reaction.

The SLN for the Diels−Alder reaction depicted in Figure 5 is

C[#4]H2=[rc=c]C[#3]H–[rc=c]C[#2]H=[rc=c]O[#1]
+ClC[#6]H=[rc=c]C[#5]HBr−>
C[1;#3]H=[rc=c]C[#2]H–[rc=c]O[#1]–[rc=x]C[#5]
H(–[rc=c]C[#6]H(–[rc=x]
C[#4]H2–[rc=c]@1)Cl)Br+C[1;#3]H=[rc=c]C[#2]
H–[rc=c]O[#1]–[rc=x]
C[#6]H(–[rc=c]C[#5]H(–[rc=x]C[#4]
H2–[rc=c]@1)Br)Cl

Atoms in the reactant can be mapped explicitly to atoms in the product. Atom mappings are specified with a static attribute, the *atom map*. The atom map syntax is specified by a "**#**" character followed by a positive integer. Atom mappings in the reactants should be unique. A single atom in the reactants can map to multiple atoms in the products. In the Diels−Alder shown in Figure 5, the oxygen mapped to atom **#1** in the first reactant appears in both of product CTs

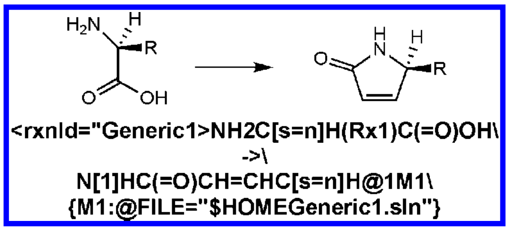
The SLN language has been further extended by the addition of a bond expression attribute (**rc**) that indicates which bonds are being broken, created, or changed in a reaction. A value of **n** indicates that the bond is not a reaction center, a value of **x** indicates that the bond is being broken or created, and a value of **c** indicates that the bond type changes in the course of the reaction.

Figure 6 depicts a reaction where a chiral center is carried through a reaction. SLN provides a syntax that specifies what happens to a chiral center during a reaction. The SLN for the alanine cyclization reaction shown in Figure 6 is:

CH3C[cc=r;s=I]H(NH2)C(=O)OH–>
C[1]H=CHC[cc=r;s=I]H(CH3)NHC(=O)@1

The chiral conversion atom attribute (**cc**) specifies how chiral centers behave. It can take the following values: **i** if the chiral center inverts; **r** if chirality is retained; **x** if a chiral center is created or removed; and **m** if the chirality is scrambled to some degree, in which case the product SLN represents a mixture.

Generic Reactions and Virtual Libraries. Generic reactions (Figure 7) are a useful and intuitive way to specify a virtual library. The reactants can be treated as templates and



```
<rxnId="Generic1>NH2C[s=n]H(Rx1)C(=O)OH\
->\
N[1]HC(=O)CH=CHC[s=n]H@1M1\
{M1:@FILE="$HOMEGeneric1.sln"}
```

Figure 7. Example of a generic reaction.

serve as queries to select lists of reagents. For each matching reagent, the reactant core is clipped off leaving only the portion that matched the variable part of the generic reactant. The core atoms in the clipped reactant are replaced by attachment atoms to create fSLNs. A generic reaction can be used to create a virtual library using a molecule repository as input where the repository can be any searchable molecule data source. The combinatorial reaction might look like:

<rxnId="Generic1>NH2C[s=n]H(Rx1)C(=O)OH–>
N[1]HC(=O)CH=CHC[s=n]H@1M1
{M1=1:@FILE="$HOME/Generic1.sln"}

MDL's RG[8] file syntax has been used to represent virtual libraries or their subsets. RG files are most commonly used in RGROUP queries.

Applications such as SMILIB[13] have also extended SMILES for use in creating virtual libraries, often adding their own proprietary language extensions.

SPECIFYING QUERIES

SLN provides a rich syntax for posing questions about molecular structures. It has several classes of atoms specifically for use in queries, including the wildcard atoms **Any** and **Hev**. These match individual atoms, while the **R**, **X**, and **Rx** group atoms match groups of connected atoms. SLN supports the ability to group atom and bond attributes together to form expressions which are evaluated during a search.

Query Attributes and Expressions. Lists of bond and atom attributes can be formed into complex expressions using a set of Boolean operators, of which the semicolon is a special case. The order of precedence starting with the highest level of binding is as follows:

"**!**" (logical "not") Negates the subsequent attribute, as in "**[!R]**" and "**[!type=2]**".

"**&**" (logical "and") True if the preceding and following attribute expressions are both true.

"**|**" (logical "or") True if either the preceding or the following attribute expression is true.

"**;**" (logical "and") True if the preceding and following attribute expressions are both true. This is a low-binding version of the "and" operator and is used as a separator for lists of attributes.

Parentheses may be used to modify the default precedence of the operators.

Atom Attributes in Queries. Any atom or bond attribute that can be used in a structure can be used in a query, but there are several reserved attributes that only have meaning within queries. Atom and bond query attributes that can be interpreted more broadly in queries than in structural SLNs are laid out in Table 4, whereas attributes that appear only in queries are laid out in Table 5.

Example expressions include the following:

• **O[-|+0&r]** matches an oxygen that is either an uncharged ring atom or has a charge of minus one.

• **N[is=N[hc≤2]C,CN*CC;not=NC={{O,S,N}},NC:Hev]** matches a nitrogen which has at most two hydrogens attached and one or two carbons singly bonded to it. The nitrogen cannot be attached to a carbon with a double bond to an oxygen, sulfur, or nitrogen nor can it be singly bonded to an aromatic carbon. The "*" in the **not=** and **is=** attributes identifies which atom in the attribute is the anchor atom for the search. The anchor atom is the atom being qualified.

• The bond expression in **CC~[(type=2&s=t)|type=3]CC** will match either a *trans* double bond or a triple bond.

• **C[r]-[!r]C[r]** will match two carbons which are each contained in a ring but are connected by a bond that is not contained in any ring (e.g., in biphenyl).

• **Mg[charge>+1&charge>4]** specifies a magnesium atom with a positive charge that greater than one and less than four.

Bond expressions of the form **~[type=2|type=1|type=3]** are quite common. In these expressions the type is restricted to a list of possibilities. If a bond expression contains only "or" operators and only **type=** attributes, it may be expressed in a shorthand notation, the bond characters for the allowed types are listed without separators. Hence **C=:C** is equivalent to **C~[type=2|type=aromatic]C**, and **C-=#C** is equivalent to **C~[type=1|type=2|type=3]C.**

The bond characters for any bond ("~") and for no bond (.) may not be included in the bond pattern shorthand.

Static Atom Attributes. Static attributes (Table 5) are used to determine how the atom is searched in partitioned searches, to direct how Markush atoms are connected, and to specify how atoms are mapped in reactions. If present, a static attribute is separated from an atom ID by a semicolon ("**;**") and from any expression attributes by a colon ("**:**"). Static attributes may not be grouped into expressions.

Partitioned Searching. Some problems require partitioning the query space into separate patterns. The same target will then be searched by applying multiple patterns sequentially. Partitioning keeps track of which atoms and bonds have already been matched. Patterns used in partitioned searches need to control how targets matched by a previous pattern can match the current one. SLN provides support for this in the static coverage and noncovering flags. The coverage flag allows the user to specify that an atom must already be covered (**c=y**), must not be covered (**c=n**), or that coverage is open and does not matter (**c=o**). The noncovering Boolean flag (**n**) specifies whether a matched atom will be marked as having been covered by the search.

One use of partitioned searches is in atom typing, where a single structure will be searched by an ordered list of queries. Atoms matched by previous queries may be important to a query's search, but should not be available for matching in the current query.

Queries used for atom typing nitrogen in SYBYL include the following:

N[c=n;!n](Any[c=o;n])(Any[c=o;n])(Any[c=o;n])
Any[c=o;n]<Atomtype="N.4">
N[c=n;!n:f](=O[c=o;n])(=O[c=o;n])Any[c=o;n]
<Atomtype="N.pl3">
N[c=n;!n:f](=N[c=o;n])=N[c=o;n]
<Atomtype="N.1">

Table 4. Glossary of Atom Attributes Whose Usage May Differ between Structural and Query SLNs

attribute[a]	interpretation
charge[b]	Specifies a formal charge or range of charges: **charge>0** specifies a positive charge and **charge<−1** specifies a negative charge less than ∃1, for example.
fcharge[b]	Specifies a partial charge or range of partial charges, e.g., **fcharge≥-.125** and **fcharge<−0.5**.
i=*n*	Specifies an atomic isotope. A value of **0** indicates the normal isotope for an atom. **C[I=0]** will match both **C** and **C[I=12]**. The isotope query **Any[!I=0]** matches any atom with a rare isotope.

[a] Literal attribute names are indicated in bold face type, whereas generic terms (such as "*n*" for a positive integer) are shown in italics. Attributes followed by an equals sign ("=") must be assigned a value; other attributes are Boolean unless otherwise indicated. [b] This attribute must take a relational operator (+, >, <, ≥, ≤, != or =.) and one or more arguments.

N[c=n;!n]=Any[c=o;n]<Atomtype="N.2">

N[c=n;!n]:Any[c=o;n]<Atomtype="N.ar">

N[c=n;!n]C[c=o;n]={{O,S,N,P}}[c=o;n]\ <Atomtype="N.am">

Another example of partitioned searching involves the substructure searches used by UNITY to identify hydrophobic centers and other pharmacophoric elements.[14]

Wildcards. The two wildcard atoms **Any** and **Hev** are used to match single elemental atoms of any type and of any type other than hydrogen, respectively. Any attribute that can be applied to an elemental atom can be applied to **Any** and **Hev**. Both **Hev** and **Any** support hydrogen shorthand.

The wildcard bond ("∼") matches any bond between two atoms. Example: **C∼C** matches **C−C**, **C=C**, **C:C**, and **C#C**

Group Atoms. SLN supports three classes of group atoms, **R**, **X**, and **Rx**, all of which specify points of broad variability in a pattern. These groups match substructures comprised of atoms that do not otherwise map to an atom in the query pattern, thereby providing place holders for side chains. They match all atoms in groups starting at the indicated position that have not already been matched by a pattern atom, including single hydrogen atoms. Several query attributes take on slightly different meanings when applied to group atoms.

The names of the group atoms must be prefixed by **R**, **Rx**, or **X**. Appended numeric characters serve to differentiate groups within a query; the class of the group atom is determined by its name prefix.

An **R** atom matches all atoms and bonds in a substructure connected to the rest of the query CT *only* by way of the explicit bond to the **R** atom. An **X** atom matches all atoms and bonds in a substructure connected to the rest of the query CT by way of one or more bonds, including the explicit bond to the **X** atom, excluding any atoms explicitly included in the CT. An **Rx** atom matches all atoms and bonds in a substructure connected to the rest of the query CT *only* by way of the explicit bonds to that **Rx** *and* to the other **Rx** atoms bearing the same numeric extension.

Figure 8 illustrates how **R**, **X**, and **Rx** atoms match groups of atoms in a substructure search. *o*-Xylene hits the first three queries shown along the side of the figure but lacks the methane bridge in the fourth query. That query matches dihydrobenzofuran, as do the middle two. The first query misses, however, because **R** group atoms can only match substituents that do not cycle back to the phenylene group. It misses the third target structure, a tetracyclic lactam, for the same reason.

The **Rx** group behaves like the **X** group except that when the group constitutes a ring the ring closure must be matched by a second, matching **Rx** atom in the pattern. For **Rx** atoms to match a ring there must be at least two of them in a query that have matching numeric extensions (or no extension). The third target structure in Figure 8, for example, is missed by the third query because two of the reattachment points are not specified as such in the query. This is in contrast with **X** groups that will match any connected structure atoms not matched by other pattern atoms (Figure 8).

C[1]H2CH2CH2CH2X@1, for example, matches any ring with at least four adjacent methylene groups; **C[1]:C:C:C:C:C:@1R[mw=18−127]** will match a phenyl with a side chain that has a molecular weight between 18 and 127; and **C[1]:C:C:C:C:C:@1R[mw=15-*;tac ≤ 50]** will match a phenyl group bearing a side chain with a molecular weight greater than 15 but having no more than 50 atoms.

Markush Queries. The most common use of Markush atoms is in queries, where they are generally defined at the end of the SLN or in an external file. In particular, **{Hal:F|Cl|Br|I}** defines a Markush which will match any halogen atom, and **{Het:O|N|P|S}** defines a Markush that matches hetero atoms. A Markush that will match methyl, ethyl, propyl, or butyl, respectively, is defined by;

{Alkyl24:CH3<*v*=1>|CH2CH3<*v*=1>|CH2CH2\ CH3<*v*=1>|CH2CH2CH2CH3<*v*=1>}

An inline Markush, in contrast, has no name, since it is defined in place. Users specify an inline Markush by placing the definition inside paired curly braces ({{...}}). A nitrogen connected to a carbon which has a double bond to either an oxygen, sulfur, or nitrogen, for example, can be written as **NC={{O,S,N}}.**

SLN originally supported complex negation expressions in Markush syntax, such as **{Nsimple:N&!NC=O&!N:Any}** and **"{Nitro:N&!(!N(=O)=O)}**, but this syntax proved too complex and difficult to support, especially the double negative (**!!**) in the second example. The intent of the syntax was to support atom typing queries to limit the extended connectivity around specific atoms. This is now accomplished using the atom attributes "**is=**" and "**not=**". The above Markushes can be simplified to qualified atoms, obviating the need for any Markush expression - i.e., **N[not=NC=O,N:Any]** and **N[is=N(=O)=O]**, respectively. The old syntax is no longer supported; the SLN parser in UNITY returns an error to alert the user to cases where it needs to be

Table 5. Glossary of Atom and Bond Attributes Used Only in Query SLNs

attribute[a]	interpretation
#1, **#2**...	The atom map number is a new static attribute that associates a map number to an atom. It is used in reactions to provide atom to atom mapping. Each atom map number must be unique with respect to reactants, but a reactant atom may be mapped to multiple product atoms.
c=	A static atom attribute that controls how atoms in the target are matched ("covered") by pattern atoms. A value of **n** indicates that the atom must not have been matched previously; a value of **o** indicates that the atom's coverage flags are ignored; and a value of **y** indicates that the atom must be covered by previous search
f	Boolean attribute indicating that the atom is filled; matching atoms must have only the bonds listed in the query SLN.
hac[b]	The total number of attached heavy atoms (heavy atom count). A value of **f** indicates that the pattern atom's heavy atom attachments will be used in the match. Inequality expressions are supported, so **hac** ≥ **3** indicates that the atom must have three or more heavy atoms attached to it. When used with a group atom, **hac** limits the number of heavy atoms matched by the group atom.
hc[b]	The total number of attached hydrogens (hydrogen count) needed to match the qualified atom. A value of **f** indicates that the number of hydrogens in the pattern SLN will be used.
htc[b]	The heteroatom count constrains the number of attached hetero atoms (oxygen, nitrogen, sulfur, or phosphorus). When attached to a group atom it indicates the total number of hetero atoms that the group must match for the group atom to match.
is=	Specifies patterns that the qualified atom in a query must match if it is to match the query atom; an atom matches if it matches *any* of the queries in the appended cooma-separated list of queries. Example: in **Any[is=Any*=O,Any*=S,C*#N,N*(~O[f]) ~O[f]]** the **Any** atom must have a double bond to an oxygen or sulfur, *or* it can map to the carbon of a nitrile group, *or* it can map to the nitrogen in a nitro group. When necessary to resolve ambiguity an asterisk following an atom indicates which atom in the individual query expression corresponds to the atom being evaluated. The **is** attribute can apply to elemental, **Any**, or **Hev** or it can apply to group atoms, in which case the **is** attribute indicates that the patterns specified must occur within the atoms matched by the group.
mw[b]	The molecular weight attribute is used with group atoms (**R**, **X**, and **Rx**) to specify the cumulative atomic weights that atoms that match the group atom to match the query. "**R1[mw=18−36]**" will match any R group with a molecular weight between 18 and 36, inclusive.
n	The noncovering static atom attribute prevents an atom that matches a query atom from being covered. **!n** means that the atom is not noncovering, i.e., that it is covered by the search, which is the default interpretation.
not=	Specifies patterns that the qualified atom in a query must *not* match; it prevents an atom from matching if *any* of the queries in the appended list match the extended connectivity around the atom being evaluated. For example, **N[not=N*C=O,N*C: Hev]H2C** prevents a primary amine from being matched by an amide or anilinic nitrogen.
ntc[b]	The nonterminal count specifies the number of nonterminal atoms. OH, NH2, and CH3 are considered terminal, as is as any atom having only one valence. It is primarily used with group atoms to limit the size of the groups being matched. **R[ntc** > **1&ntc** ≤ **5]**, for example, specifies an R atom that includes more than one but no more than five nonterminal atoms.
r	Boolean attribute used to specify that a bond or atom is in a ring. **!r** specifies that the bond or atom must not be in a ring, i.e., is in an open chain.
rbc[b]	The number of ring bonds (ring bond count) that must be present for the atom to match. A value of **f** indicates that the number of ring bonds is taken from the pattern, whereas **rbc** > **2** picks out atoms in fused ring systems.
src[b]	The size of the smallest ring an atom can be in to match, i.e., the smallest ring count.
tac[b]	The total number of atoms attached to the qualified atom (total atom count). A value of **f** indicates that no more attached atoms than are explicitly shown in the pattern are allowed; **tac** > **2** indicates that there must be at least three attached atoms.
tbo[b]	Constrains the overall (total) bond order of an atom.
type=	Overrides the bond type specified by the bond character it qualifies. Predefined values are **–**, =, **#**, **:**, **1**, **2**, **3**, or **aromatic**, but other values can be supplied as well. For example: **-[type=2]** and **#[type=2]** both refer to a double bond, **[type=ligand]** is a user-defined bond type for a ligand bond. This attribute cannot be assigned null (.) or unspecified (~) bond values.
v=	Conveys Markush and macro atom valence information. Indicates which bond to a Markush reference is to the corresponding valence atom in the Markush definitions. For example, in **OHAla[v=2,1]H** the oxygen is attached to the second valence of alanine.

[a] Literal attribute names are indicated in bold face type, whereas generic terms (such as "*n*" for a positive integer) are shown in italics. Attributes followed by an equals sign ("=") must be assigned a value. [b] This attribute must take a relational indicator (+, >, <, ≥, ≤, != or =.) and an argument.

modified. Unfortunately the inherent ambiguity in the old logic precludes reliable automatic translation.

Sometimes it is necessary for the pattern to specify several points at which a Markush atom can be attached yet still limit the number of times a part of the Markush definition can be used to match the Markush atom references.

Suppose, for example, that one wishes to create a query that will "hit" all dichloro substituted toluenes, regardless

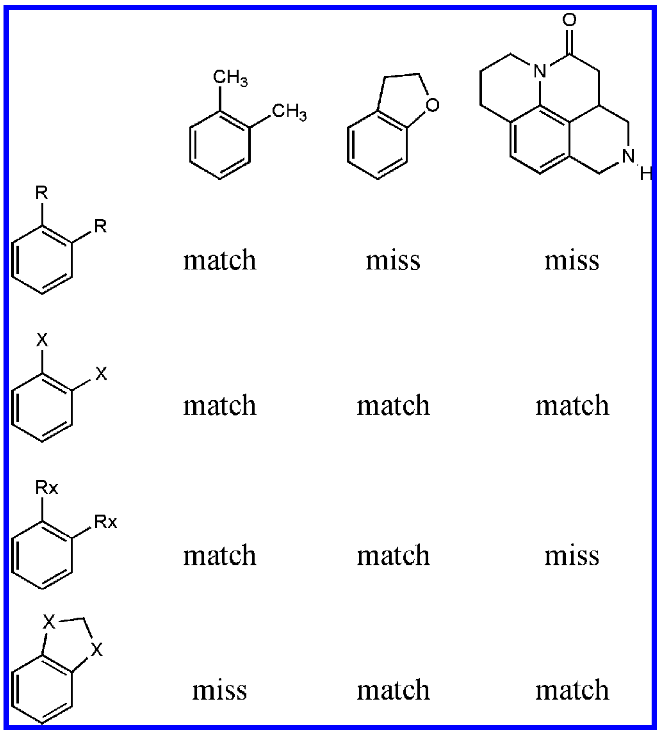


Figure 8. Matching behavior of different kinds of group atoms in SLN queries. An entry of “match” indicates that the query shown to the left of the entry matches the structure shown above it. Structures that do not match the corresponding query are indicated by an entry of “miss”.

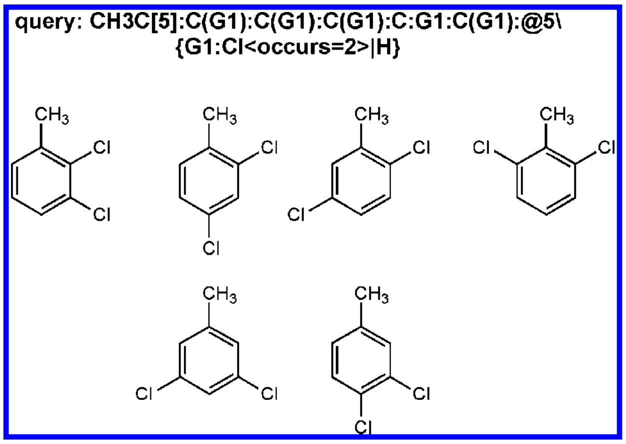


Figure 9. Example of a range-of-occurrence query and the structures it matches.

of the position of substitution. The range of occurrence specification can be used to indicate the restrictions on the number of times a particular Markush choice may be used. Range of occurrence is specified as a CT attribute, **occurs**. The following query SLN, for example, is a very efficient way to find all six isomers of dichloro toluene (Figure 9):

CH3C[5]:C(G1):C(G1):C(G1):C:G1:C(G1):@5/
{G1:Cl<occurs=2>|H}

The **occurs** CT attribute can take a discrete value, a list of discrete values, or a range of values. In the above example, **G1** must match two chlorine atoms, and the rest of the matches must be hydrogen. If we needed the query to match toluene as well as dichloro toluene, then the **occurs** would be <**occurs=”0,2”**>. The **occurs** attribute can also specify ranges, even unbounded ranges, e.g., <**occurs=”2,4−6”**> and <**occurs=”1,3-*”**>. The first example allows for two occurrences *or* between four and six, whereas the second the specifies either a single occurrence or at least three.

REACTION QUERIES

All atom and bond query attributes are supported in reaction queries. In addition users can specify reaction specific queries. The reaction query **–>N[1]H:C:C:C:C:@1**, for example, will match any reactions where the products contain pyrrole rings, and **N[+0;not=NC=Hev,NC:Hev]H2CX1–>** will match reactions that have primary amines as a reactant.

A common problem with molecular databases is the identification of duplicate structures where the duplicates have different resonance forms or are different tautomers. It is often desirable to identify a normalized form and have compounds be standardized before being added to the corporate database. One way to accomplish this is to specify the required transformations as reactions. The reactant is used as a query, and the reaction product is used as a template for the transformation.

For example, nitro groups are often encountered in the rather ungainly “double double bond” form, which includes a pentavalent nitrogen. These can be converted to the generally more acceptable, charge-separated form by applying the transform

N[#1](=O[#2])=[rc=c]O[#3]–>
N[#1:+](=O[#2])–[rc=c]O[#3:–]

Daylight SMARTS supports a full query syntax for reactions, but SMILES syntax does not support bond attributes for reaction centers. SMIRKS was created as an extension to SMARTS specifically to provide support reaction transformations. RXN files include support for reaction centers, but RGROUP query syntax is not fully supported in RXN files.

DISCUSSION

Comparison of SLN to SMILES and SMARTS. In its simplest form a SMARTS string is identical to a SMILES string, the difference in meaning depending upon the context in which the string is used. For example the SMILES for benzene is **c1ccccc1** and the SMARTS for phenyl is also **c1ccccc1**. In SLN the string to represent benzene is **C[1]H:CH:CH:CH:CH:CH:@1** and the string for the carbon framework of a phenyl group is **C[1]:C:C:C:C:C:@1**. Of course there is far more to SMARTS than simply being SMILES in a query context. SMARTS has its own rich syntax for specifying substructure search queries, just as SLN does.

The SMARTS approach to aromaticity not only gives it an economy of expression but also requires the addition of extra syntactical elements when the user wants to match substructures regardless of whether the target is specified as being aromatic. For example, the “*****” wildcard is equivalent to “**A,a**”, and the token **#6** is equivalent to “**C,c**”.

SLN supports all of the basic query features of SMARTS other than octahedral and other high-order chiralities and certain kinds of tetrahedral chirality matches. Tables 6 and 7 show how the atomic primitives in SMARTS map to equivalent SLN constructs.

Many SLN atom attributes support inequality relations. For example, SLN allows for queries specifying an atom with more than two ring bonds (**Hev[rbc > 2]**), whereas SMARTS and SLN both only allow atom and bond queries to group attributes into complex expressions with logical operators (“**&**”, “**;**”, “**|**” and “**,**”)

Table 6. How SMARTS Atomic Features Map into SLN

SMARTS	interpretation	SLN
+<*n*>, -<*n*>	formal charge	**charge**=*n*, **charge**=-*n* ,+*n*, -*n*, +, -
@, @@	indicates the sense of tetrahedral chirality around an atom, excluding relative group stereochemistry	**s=I**, **s=N**
@<*c*><*n*>	nontetrahedral chirality	*none built in*
<*n*>	specifies atomic mass	**I**=*n*
#n, #X	**#6**, **#7**... specifies an atom by its atomic number, with **#X** indicating a non-hydrogen, noncarbon atom.	Elemental atomic symbols. **#X** is specified by **Hev[not=C]**
*	wildcard atom that matches any heavy atom	**Hev**, **Any**
A,a	wildcard atom that matches any heavy atom	**Hev**, **Any**
D<*n* >	The degree indicates the number of bonds to heavy atoms; defaults to 1 if *n* is not specified.	**hac**=*n*
H<*n*>,**h**<*n*>	Number of hydrogens, where explicit (**H**) and implicit (**h**) hydrogens are distinguished from each other; defaults to 1 if *n* is not specified.	**hc**=*n*
R<*n*>	Indicates the number of rings an atom is included in.	ring bond count (**rbc**=*n*) query attribute
r<*n*>	Smallest ring, simply indicating membership in some ring if *n* is not specified.	smallest ring count (**src**=*n*) and Boolean **r** attributes
v<*n*>	total bond order	**tbo**=*n*
X<*n* >	total number of connections to an atom	**tac**=*n*
x<*n*>	total number of ring connections to this atom	**rbc**=*n*

Table 7. How SMARTS Bond Features Map into SLN

SMARTS feature	interpretation	SLN	comment
-	single bond	**-**	bond syntax is the same in all cases
=	double	=	
#	triple	#	
:	aromatic	**:**	
~	any bond	~	
/	up and down bonds	**s=i,**	specifies double bond stereochemistry
\	Used to specify double bond stereochemistry, C\C=C/C is a cis double bond. C\C=C\C is trans	**s=n**	
\\?	Match bond stereochemistry or match unspecified double bond.	—	SLN lacks this specific sense of matching. Instead, a bond matches if the stereochemistry matches or if the target's stereochemistry is unspecified.
/?			
@	ring bond	**r**	bond must be in ring

Recursive SMARTS allows for the specification of an atom's extended connectivity as an atomic property. A recursion is indicated with a leading **$** and is enclosed in parentheses. **C[$(aaO)]**, for example, specifies a carbon *ortho* to an oxygen atom. The SLN equivalent of this functionality is a combination of **is=** and **not=** atom attributes. The SLN equivalent of **C[$(aaO)]** is **C[is=C*Hev:HevO]**. The SMARTS string **C[$(aaO);$(aaaN)]** specifies a carbon atom *ortho* to an oxygen and *meta* to a nitrogen. The equivalent SLN is **C[is=C*Hev:HevO;is=C*Hev:Hev:HevN]**.

Neither SMILES nor SMARTS support the specification of relative stereochemistry. SLN lacks the ability to query chiral centers that match if the target has the chirality specified and simultaneously match if *no* chirality is specified. This feature is useful when working with generic reactions, where a common scenario with reaction component searches has chiral generic reactants matching achiral reagents. SLN only directly supports tetrahedral chirality,[15] whereas SMILES can express higher order chirality.

SMARTS has nothing comparable to the SLN **R**, **X**, and **Rx** query atom types.

Both notations have rich query syntaxes that cover huge query space. Most SMARTS queries can be expressed in SLN quite easily, and most SLN queries can be expressed in SMARTS.

SMARTS has two implied bond types, single and aromatic bonds. For example, the SMARTS string "**cc**" will match two carbons with an aromatic bond or two carbons in aromatic systems with a single bond between them. This can lead to queries that are ambiguous. For this reason it is common for SMARTS strings to have explicit bonds between aromatic atoms.

It is important for a query to be able to address the types of questions that chemists actually ask. A synthesis chemist, for example, might need to find primary amines with a molecular weight between 150 and 500. In SLN this can be expressed as **N[is=N*C]H2R[mw=133−485]**. That simple query will bring back amides and anilines as well, however, so a more useful version might look like

N[is=N*C;not=N*C={{O,N,S}},N*C=:#Hev]\
H2R[mw=133−485; not=NC={{O,N,S}},NC=:#Hev]

which will exclude amides and anilines from matching. The "**not=**" attribute on the **R** group prevents them from matching the **R** group as well.

InChI and SLN. Though the output of the program is a text string, the IUPAC International Chemical Identifier (InChI[16]) is primarily intended to produce a nonproprietary, canonical identifier for chemical structures, not to be a line notation *per se*. Hence SLN and InChI were designed to address different kinds of problems, which makes a direct comparison difficult. In particular, InChI is designed to generate an identifier for linking together references to specific structures in disparate databases, not to support substructure searching within a single database or set of similar databases. UNITY does include a canonicalization that can be used to aid in exact-match searching, but SLN was primarily designed to represent a connection table in a way that supports efficient *substructure* searching. Canonicalization is of little help in that context, in part because there can be no guarantee that an arbitrary shared substructure will produce identical substrings in the corresponding SLN and in part because queries are often necessarily more or less ambiguous.

In InChI, the chemical formula and heavy atom connectivity information is placed at the beginning of the string and set off from the subsequent stereochemical, mobile hydrogen information and other attributes. As a result, the leading blocks of characters will be the same for different tautomers, and exact-match substring searches will find all structures that could be tautomers or alternative protonation states. In SLN, the equivalent query is generated by stripping away all attributes and hydrogens and converting all bonds to type "any" (~). In either system, the search will retrieve all isomers that differ in bond order and hydrogen atom placement, but only some of these are actually tautomers in the usual sense, i.e., they do not readily interconvert. Faster searching is supported by InChIKeys (condensed, 25 character representations of the full string), the leading portions of which are the same for structures which differ only in the placement of "mobile hydrogens".[17] Unfortunately, the converter program[18] does not always recognize labile hydrogens as mobile: the InChIKey generated for the activated carbonyl 3-oxopentanenitrile, for example, is KOYWUYIBEXFRFH-UHFFFAOYAT, whereas the InChI key for its enol tautomer is DXIYCAQUQLUBJF-HWKANZROBQ.

The two critical differences between InChI and SLN are: that it is readily extensible—e.g., by creating new user-defined atom or bond attributes—whereas InChI is necessarily standardized; and that the intended audience for InChI is composed of machines rather than human beings. The InChI for propanoic acid, for example is[18]

C3H6O2/c1−2−3(4)5/h2H2,1H3,(H,4,5)/f/h4H

which is considerably less transparent to a human being than is the corresponding SLN, **CH3CH2C(=O)OH**.

This direct interpretability and simplicity is, of course, less evident in complex structures and may vanish completely in complicated expressions, but it lies at the heart of the SLN syntax nonetheless.

Software Applications that Utilize SLN. SLN is an excellent method for encoding chemical structures over local and even worldwide computer networks. UNITY, CONCORD,[19] SYBYL,[20] Galahad,[21] Tuplets,[22] and Benchware Data Miner[23] all make use of SLN, and more applications are continually being developed. SYBYL uses SLN to process and analyze molecular structures as well as to build queries for searching UNITY databases. SLN is also useful for storage of molecular structures in relational databases. Since SLNs are strings, they reduce the overhead of using Large Object (LOB) data types in such databases. The SYBYL Force Field Engine uses SLN for both atom typing and assigning force field parameters. The ChemNavigator[24] program suite consists of a number of applications that utilize SLN, including Cnctranslate, 3DPL, CncDiverse, CncMcs, and CncConnect.

ACKNOWLEDGMENT

The authors greatly appreciate the heroically detailed suggestions for improvement provided by our anonymous reviewers and wish to thank them for their careful consideration of the manuscript as a whole.

REFERENCES AND NOTES

(1) Ash, S.; Cline, M. A.; Homer, R. W.; Hurst, T.; Smith, G. B. SYBYL Line Notation (SLN): A Versatile Language for Chemical Structure Representation. *J. Chem. Inf. Comput. Sci.* **1997**, *37*, 71–79.
(2) *UNITY*; Tripos International: St. Louis, MO, 2008.
(3) The cyclic diborane structure is best specified in SLN as **B[1]H2-H-BH2-H-@1**, with the bridging hydrogens set off by explicit single bonds (Figure 2).
(4) The original SMILES syntax did not include attributes for atom or bond chirality. Two separate extensions are now in widespread use: "Conversational SMILES" by Robert Pearlman and coworkers at the University of Texas, Austin and "Isomeric SMILES" by Daylight Information Systems.
(5) Weininger, D. SMILES: A Chemical Language and Information System. *J. Chem. Inf. Comput. Sci.* **1988**, *28*, 31–36.
(6) Weininger, D.; Weininger, A.; Weininger, J. L. SMILES II. Algorithm for Generation of Unique SMILES Notation. *J. Chem. Inf. Comput. Sci.* **1989**, *29*, 97–101.
(7) Cahn, V. R. S.; Ingold, C.; Prelog, V. Specification of Molecular Chirality. *Angew. Chem., Int. Ed.* **1966**, *5*, 385–415.
(8) Prelog, V.; Helmchen, G. Basic Principles of the CIP-System and Proposals for a Revision. *Angew. Chem., Int. Ed.* **1982**, *21*, 567–583.
(9) Dalby, A.; Nourse, J. G.; Hounshell, W. D.; Gushurst, A. K. I.; Grier, D. L.; Leland, B. A.; Laufer, J. Description of Several Chemical Structure File Formats Used by Computer Programs Developed at Molecular Design Limited. *J. Chem. Inf. Comput. Sci.* **1992**, *32*, 244–255.
(10) MDL Information Systems, Inc. MDL's Enhanced Stereochemical Representation. http://www.mdli.com/products/pdfs/Enhanced_Stereochemical_Representation.pdf (accessed Dec 12, 2007); copyright 2003.
(11) MDL Information Systems, Inc. MDL CT File Formats. http://www.mdl.com/solutions/white_papers/ctfile_formats.jsp (accessed Dec 1, 2007).
(12) *Italics* are used herein to indicate descriptive terms for syntactical elements in an SLN, whereas **boldface** is used for literal characters.
(13) Shuller, A.; Schneider, G.; Byvatov, E. SMILIB: Rapid Assembly of Combinatoral Libraries In SMILES Notation. *QSAR Comb. Sci.* **2003**, *22*, 719–721.
(14) Abrahamian, E.; Fox, P. C.; Nærum, L.; Christensen, I. T.; Thögersen, H.; Clark, R. D. Efficient generation, storage and manipulation of fully flexible pharmacophore multiplets and their use in 3-D similarity searching. *J. Chem. Inf. Comput. Sci.* **2003**, *43*, 458–468.
(15) Other kinds of chirality can be specified using user-defined attributes.
(16) InChI is a trademark of the International Union of Pure and Applied Chemistry (IUPAC).
(17) Williams, A. Does InChI Account for Tautomers? http://www.chemspider.com/blog/does-inchi-account-for-tautomers.html (accessed Sept 2, 2008), ChemSpider Blog.
(18) InChI converter. http://www.inchi.info/converter_en.html (accessed Sept 2, 2008).
(19) Pearlman, R. S.; Rusinko, A.; Skell, J. M.; Balducci, R. *Concord*; Tripos International: St. Louis, MO, 2008.
(20) *SYBYL*;Tripos International: St. Louis, MO, 2008.
(21) *Galahad*; Tripos International: St. Louis, MO, 2008.
(22) *Tuplets*; Tripos International: St. Louis, MO, 2008.
(23) *Benchware DataMiner*; Tripos International, St. Louis, MO, 2008.
(24) *ChemNavigator*; ChemNavigator: San Diego, CA, 2008.

CI7004687