

SYBYL Line Notation (SLN): A Versatile Language for Chemical Structure Representation[†]

Sheila Ash,* Malcolm A. Cline, R. Webster Homer, Tad Hurst, and Gregory B. Smith

TRIPOS, Inc., 1699 S. Hanley, St. Louis, MO 63144

Received June 30, 1996[®]

SYBYL Line Notation (SLN) is an ASCII language used to represent chemical structures, including common organic molecules, macromolecules, polymers, and combinatorial libraries. SLN is also used to express substructural (2D) queries and includes a complete facility for Markush representation. This concise language is ideal for database storage of chemical entities as well as for network communication of structures and queries.

1. INTRODUCTION

Many chemical information and drug design programs require representation of chemical structures and queries. These programs often make extensive use of fragment recognition functionality. This functionality determines whether a substructure (2D) query is contained in a chemical structure or not. If the query is found (that is, if there is a "hit"), the mapping from the atoms of the query pattern to those of the hit structure is returned to the searcher.

To provide a simple and concise method for storing and communicating both chemical structures and substructural (2D) queries, SYBYL Line Notation (SLN) has been developed. This notation describes the structures and search patterns as a text string. The SLN strings are used to store structural information in databases and files and also provides a mechanism for communication of chemical entities between various programs. The concise nature of the SLN language minimizes the network bandwidth required for task-to-task communication and thus enables programs to communicate structural information interactively over global networks.

Many software programs use SLN for structure and query input or storage including UNITY,¹ SYBYL,¹ Chemeleon,² and CONCORD.³ It is generally not necessary for any user to know SLN in order to use Unity or other programs which use SLN, as graphical input is possible for most structures and substructure (2D) queries. Many users have become quite familiar with this language, however, and use it routinely for structure and query notation.

SLN is also used as an integral part of the 3D Searching query specification language SLN/3D. The SLN/3D language is not described here.

The following is a full description of Version 1.0 of the SLN Language. SLN was inspired by the SMILES notation described by Weininger⁴ but has several extensions and modifications which allow the specification of full substructure queries, including Markush⁵ structures and queries.

SLN is used to represent three entities:

1. Structures/fragments: Chemical entities described by atoms which are connected by bonds.

2. Patterns: The specifications for a substructure (2D) search, including special atom types, "Markush", and macro atom types and R- and X-group specification.
3. Combinatorial libraries: Representing in a compact format the possible products from a combinatorial synthesis in which some groups may represent a large list of possible fragments.

A set of atoms and their bonds is called a connection table (CT). An SLN contains a main CT and may contain additional Markush and macro atom definitions. These definitions, if present, contain other CTs.

There are many equivalent ways of specifying the same structure in SLN format. Database storage of structures is facilitated by producing a canonical form of the SLN. Software is available which will generate a unique ordering of the atoms in an SLN, so the users and programs which generate SLNs may do so without regard to the atom ordering. The SLN language itself does not require unique ordering, and the canonicalization software is not a formal part of the SLN language.

There are usually many ways to represent aromatic ring systems, either using aromatic bonds or with single and double bonds. Although software is available to normalize the aromaticity to a standard form, this normalization is not a formal part of the SLN language.

Most elements found in organic structures have a common valence. The SLN language itself makes no presumptions about what is chemically reasonable.

Many programs which use SLN allow the backslash character as a line continuation marker. This convention appears in the examples in this paper. However, the backslash character is not a formal part of the SLN language.

2. SPECIFICATION OF STRUCTURES

The specification of chemical structures and fragments using SLN is easy and intuitive. The atoms and bonds of the structures are described in a manner which allows a great deal of flexibility and customization. Six basic components can be used to specify chemical structures in SLN:

[†] Presented as a poster at the 4th International Conference on Chemical Structures, June 1996, Noordwijkerhout, The Netherlands.

[®] Abstract published in *Advance ACS Abstracts*, January 1, 1997.

Elemental atoms	specify atoms as nodes in a graph.
Bonds	specify the connectivity between atoms.
Branches	specify branch points for the structure.
Ring closures	specify bonds to previously defined atoms.
Attributes	include additional information about atoms, bonds, and CTs.
Macro atoms	allow specification of groups of atoms in a shorthand notation, such as Alanine (Ala). These may be expanded into the full atom form, if required.

2.1. Elemental Atoms. The atomic symbol is used as the name of an elemental atom. The first letter is uppercase. If there is a second letter, it is lowercase.

Each atom which is the target of a ring closure (see below) must have an atom ID number. The ID number is a positive integer and is placed in square brackets after the atom name. No two atoms in a single CT may have the same ID. If an ID is specified for an atom which is not part of a ring closure, it is ignored.

Examples: **C[3], O[132]**

Hydrogens must be specified in SLN, as no presumptions are made regarding standard valences. A shorthand 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 **CH2** or **OH**. This eliminates the need for explicit branches to many hydrogens.

Examples: **CH3, OH, NH2, C[3]H2**

The hydrogen shorthand is not available if the hydrogen has more than one bond, its bond is not a single bond, or the hydrogen has some attribute specified. If any of these conditions are met, the hydrogen must be specified in the same manner as any other elemental atom.

In contrast to SMILES, aromaticity is not an atomic property, but a property of the bonds.

2.2. Specification of Bonds. Bonds are indicated by placing a special character between two atoms. The bond characters are as follows:

- Single bond. The “—” may be omitted. The specification of an explicit H atom may require the “—” to prevent it from being parsed as the “hydrogen shorthand.”
 - = Double bond.
 - # Triple bond.
 - : Aromatic bond.
 - .
- A period (.) may be used to indicate the start of a new part of the structure. In essence, the period represents a zero order bond.

Examples:

Ethane: **CH3CH3** or **CH3-CH3**
 Propane: **CH3CH2CH3**
 Ethylene: **CH2=CH2**
 Acetylene: **CH#CH**
 Sodium methoxide: **CH3O.Na**

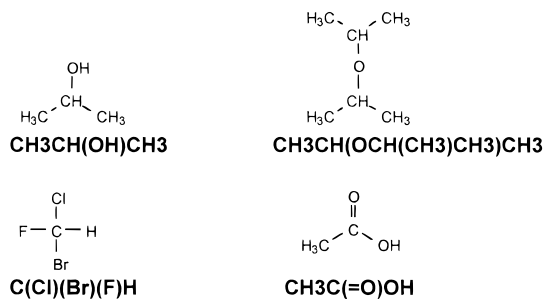


Figure 1. Examples of branching in SLN.

2.3. Branching. Branching is indicated by parentheses. The parentheses set off a group which is connected to the atom immediately preceding the group. The atom or group which follows the set of parentheses is also attached to the atom which precedes the open parenthetical group. Note that hydrogens expressed in the 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 which have branches (Figure 1).

2.4. Ring Closures. Ring closures are specified by a bond to a previously defined atom. The ID of the previous atom must be defined. An “@” is used to indicate the ring closure and is preceded by the bond type and followed by the ID of the previously defined atom.

Examples:

Benzene: **C[1]H:CH:CH:CH:CH:CH:@1**

Cyclohexane: **C[15]H2CH2CH2CH2CH2CH2@15**

o-Cyclobenzoic acid:

C1C[1]:CH:CH:CH:CH:C(:@1)C(=O)OH

Naphthalene:

C[1]:CH:CH:CH:CH:C(:@1):CH:CH:CH:CH:@1

2.5. Attributes for Structures. The atoms and bonds of a structure as well as the structure itself may have one or more attributes attached to provide information about the atom, bond, or structure. This allows arbitrary data to be associated with the atom, bond, or SLN, including biological or property data for the structure.

These attributes follow the entity being described. They are delimited by square brackets ([]) for atoms and bonds and by angle brackets (< >) for CTs. Lists of attributes are separated by semicolons (;).

Attributes may appear in any order, and the attribute names and values are not case-sensitive.

2.5.1. Atom Attributes. For elemental atoms, the attributes follow the atom ID, if present. The ID is followed by a colon (:), then by a list of attributes separated by semicolons (e.g., Atom[id: attr1 ; attr2; ...]).

The following are the predefined attributes for elemental atoms. In addition, users may define other attributes and assign them values.

charge= Indicates a formal charge. As a shorthand notation, the attribute name charge= may be replaced by a “+” or “-” sign. This sign indicator may be followed by an integer indicating the formal charge for the atom.

	Examples: Cl[11 : -] , O[-2] , Fe-[+3] , N[charge=+1]
fcharge=	Indicates a floating point charge. Example: N[fcharge=+1.231]
I=	Indicates an isotope. It must be followed by the isotope number. Example: Br[i=81] or C[I=13]
S=	Indicates a stereochemical center. It must be followed by one of the following chirality values: R or S where appropriate, D or L where appropriate, relative to glyceraldehyde N or I (parity) analogous to the R-S method, except that the priorities of the attached atoms are determined by their order of appearance in the SLN. A value of "N" (for normal) indicates clockwise presentation of the three dominant groups with the group appearing last in the SLN positioned away from the viewer, while "I" (inverted) indicates counterclockwise presentation. Internally, all chiral centers are represented with the N, I modes. The R, S and D, L specifications are converted to the standard internal form.
User defined	There are two kinds of user-defined attributes for atoms:
boolean	The attribute's name is specified, as in C[backbone] .
valued	The name is specified, followed by an equal sign and the value, as in C[chemshift=7.2] .

2.5.2. Bond Attributes. Bond attributes may follow the bond symbol and are enclosed in square-brackets ([]). The following bond attributes are predefined:

Type=	This attribute overrides the type specified by the bond character. Predefined values for the "Type=" attribute are as follows: 1, 2, 3, aromatic. The bond character may be used as well (e.g., -[type=3] is equivalent to -[type=#]). In addition to these values, you may supply any other string as the value to the Type= attribute. For example: -[type=2] and #[type=2] both refer to a double bond, and [type=ligand] is a user-defined bond type for a ligand bond.
S=	This attribute indicates stereochemistry about a double bond. This attribute is meaningless except for double bonds. The allowed values are: C or T Cis or Trans E or Z Entgegen or Zusammen

N or I	Normal or Inverted—similar to E or Z, except that the priorities of the connected atoms are determined by their position in the SLN, rather than by the Cahn–Ingold–Prelog rules. For "N", the two atoms (one from each end of the double bond) which appear first in the SLN are on opposite sides of the double bond, while for "S" they are on the same side. Thus, <i>trans</i> -butene could be expressed by any of the following SLNs: CH3CH=[s=t]CHCH3 CH3CH=[s=i]CHCH3 CH3CH=[s=n]C(CH3)H
User defined	There are two kinds of user-defined attributes for bonds:
boolean	The attribute's name is specified, as in -[rotatable] .
valued	The name is specified and is followed by an equal sign and the value, as in -[bondstretch=20] .

2.5.3. CT Attributes. Attributes can be attached to the complete structure to indicate the name, bioactivity, molecular properties, etc. CT (Connection Table) attributes follow the CT and are delimited by angle brackets (< >). There are five predefined CT attributes for structures:

regid	the registration ID. Any string can be used.
name	the name of the CT. Any string can be used.
type	the type of CT. The type is used to classify the structure into broad groups, such as proteins, small molecules, polysaccharides, etc. Any string may be supplied as the type.
coord2d	the 2D coordinates for the atoms in the CT. There will be two floating point numbers for each atom in the CT. Pairs of numbers can be offset by parentheses.
coord3d	the 3D coordinates for the atoms in the CT. There will be three floating point numbers for each atom in the CT. Triplets of numbers can be offset by parentheses.

Examples:

```
CH3CH2CH3<regid=C10123;name=propane;\
type=small_molecule;coord2d=1.23,3.4,5.6...>
CCC<coord2d=(1.2000,0.500),(1.200,1.100),\
(1.95,1.25)>
CCC<coord3d=(1.2000,0.500,0.00),(1.200,1.100,0.00),\
(1.95,1.25,0.450)>
```

As with atoms and bonds, user-defined attributes are allowed and may be either boolean or valued.

2.6. Macro Atom Definition. Macro atoms are shorthand notations for groups of atoms, such as amino acids. A

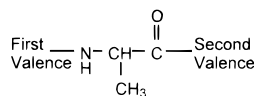


Figure 2. The distinct valences of an alanine fragment.

macro atom starts with an uppercase letter, followed by one or more lowercase letters, digits, or underscores. Structures expressed in macro atom format can be expanded into full atom format. Full atom structures can be contracted to macro atom encoded forms without loss of atomic level information.

Examples: **Ala, Gly, His, Monomer1**

2.6.1. Macro Atom Definition. The syntax for a macro atom definition is as follows:

```
{macro_atom_name:ct<v=valence_specs>}
  where:
  macro_atom_name = name of the macro atom being
    defined
  ct = CT for the full atom form of the macro atom
  valence_specs = valence of the macro atom.
```

Examples:

```
{Ala:NHCH(CH3)C(=O)<v=1,9>}
{Gly:NHCH2C(=O)<v=1,6>}
```

The valence of a macro atom is fundamentally different from that of most elemental atoms in organic compounds. Whereas all valences of a tetrahedral carbon atom are equivalent, the valences of a macro atom are distinct and different. For example, a macro atom representing alanine (Ala) has a valence of 2. The first valence is to the nitrogen, while the second is to the carbonyl (Figure 2).

Valence is specified with the CT attribute "V=", which is followed by a comma-separated list of the attachment points for the macro atom (the number corresponds to the number of the atom as written, left to right, in the SLN.) In the example above, the attribute <v=1,9> indicates that the first valence of Ala is connected at atom 1 (the nitrogen) and the second valence is connected at atom 9 (the carbonyl carbon).

If no valence attribute is specified, the connections to the macro atom are assumed to be in the order of appearance of the atoms in the CT. For example, in the macro definition {Peroxide:O—O}, no valence designation is needed, since the first valence is connected at atom 1, while the second is connected at atom 2. In the absence of a valence attribute, all atoms are considered possible attachment points.

2.6.2. Macro Atom References. References to macro atoms are made by using the macro atom name as an atom. The syntax is:

```
macro_atom_name[macro_atom_attributes]
  where
  macro_atom_name = name of the macro atom
  macro_atom_attributes = used to specify the atom ID
    and the connectivity.
```

An integer ID is allowed as for other atom types. If it is specified, it must be unique in the CT.

Examples:

```
HAlaHisGlyOH
HHis[1]Ile[2]Gly[3]OH
HPvaPvaPvaH
where Pva is:
```

{Pva:CH2-CH(OH)<v=1,4>}

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 and a His. The bond to the hydrogen appears first in the CT, so it is connected to the first valence position of the Ala macro atom, which is the N-terminus. The bond to the His atom appears second, so the His atom is attached to the second valence of the Ala, the C-terminus. This type of connectivity is referred to as the "natural order" and is determined by the macro atom definition.

If the connectivity does not follow the natural order for the macro atom, it must be specified following the reference to the macro atom using the attribute V=. This designation is followed by a comma-separated list of the valence numbers for the connected atoms in the order of the connected atoms. For instance, if an amino acid sequence were to be specified in reverse order, the modifier [V=2,1] would be used.

Example:

```
HOGly[v=2,1]Ala[v=2,1]His[v=2,1]H
is equivalent to:
HHisAlaGlyOH
```

The definition of macro atoms can be made globally prior to use in an SLN, or it can be part of the SLN specification itself. If the macro atom is defined as part of the SLN, the definition is not valid in other SLNs (the scope is only for the current SLN).

Example: **HPvaPvaPvaH{Pva:CH2-CH(OH)<v=1,4>}**

It is possible to expand CTs containing macro atoms to full atom forms or to contract full representations to macro atom encoded forms. Thus, structures in databases can be stored in macro atom reduced formats and still retain full atomistic information.

Random copolymers are expressed in SLN using the macro atom facility. The monomeric units are specified in a disconnected state and are separated by a null bond (.). The type for the CT is then set as type=polymer.

Information about the probabilities of occurrence for the monomeric units can be supplied as CT attributes.

Examples:

```
A co-polymer consisting of 40% Ethylene and 60%
  Propylene:
Ethylene.Propylene<type=polymer;Ratio_1=.4;\
  Ratio_2=.6>
A co-polymer with specified markovian probabilities:
Ethylene.Propylene<type=polymer;\
  Markov=a,a,0.6,b,a,0.4,a,b,0.7,b,b,0.3>
```

3. SPECIFYING 2D SUBSTRUCTURE QUERIES

Pattern SLNs are the query specifications for a substructure search. If all the atoms and bonds of the pattern can be matched with atoms and bonds in the structure, a "hit" is found.

Any valid structure SLN is also a valid pattern SLN. In addition to the constructs used for full structure and fragment specification, there are several constructs specific to search patterns. They are as follows:

Any-atom pattern Matches any atom.

Attribute expressions	Expressions of attributes can control matching of atoms and bonds. For example, an attribute expression can be made which matches a carbon atom that has a charge of 1 or 2 and is in a ring.
Covering and covered flags	Allows specification of the important atoms in a pattern and the atoms of a structure which are available to match.
R and X atoms	Matches all atoms in the side chain or group not matched by other parts of the pattern. These are useful in reporting the matched atoms of side chains.
Markush atoms	Allows points of controlled variability in searching patterns to be defined.

3.1. "Any" Atoms The "Any" atom matches any single atom in a structure. All elemental attributes may be applied to an "Any" atom.

Example:

Match any structure with an ethyl group attached to an atom with a charge of -1:

CH3CH2Any[charge=-1]

3.2. Pattern Attributes. Special atom and bond attributes are available for patterns beyond those which are available for structures. Attribute expressions always follow the object of the expression and are delimited by square brackets ([]). All attributes are named and the names and values are not case-sensitive.

There are two classes of attributes:

Static attributes:	Each object has a value for each static attribute, whether specified or not. Each static attribute has a default.
Expression attributes:	Each object type may have one or more occurrences of these kinds of attributes, or may have no occurrence. Complex expressions of these attributes using boolean operators are possible in 2D substructure queries.

Lists of attributes are separated by semicolons (;). Static attributes must come first, followed by the expression-type attributes. If both types of attributes are present, the two groups are separated by a colon (:).

3.2.1. Pattern Atom Attributes. The following atom attributes are used for the specification of search patterns and control of searching functions:

F	(expression-class) This character indicates that the atom has filled valences. Matching structure atoms must have only the bonds explicitly listed in the pattern. For example, the pattern CH3SCH3 will match the structures CH3SCH3 , CH3S(=O)-CH3 , and CH3S(=O)(=O)CH3 , whereas CH3S[F]CH3 will match only CH3SCH3 . This expression class is also used to define a radical.
R	(expression-class) This character indicates that the matching structure atom must be in a ring.
C	(static-class) The "covered" attribute is used to specify which structure atoms are not available for matching. Possible values are C=Y , C=N , and C=O . See the full discussion of covered and noncovering in Section 3.3.
N	(static-class) This is called the "non-covering" attribute and is used to specify which pattern atoms do not cause the matching structure atoms to be marked as covered. Thus structure atoms that match noncovered pattern atoms are still available for matching in subsequent searches.

3.2.2. Pattern Bond Attributes. One attribute is pre-defined for use in bond patterns:

R	(expression_class) This character indicates, for patterns, that the matching bond in the structure must be in a ring. !R indicates that the bond must not be in any ring.
----------	--

In addition, the "Any" bond type is defined. This type bond is signified by the tilde (~) character and will match any bond type.

Example:

Match a carbon bonded in any manner to an oxygen atom:

C~O

3.2.3. Attribute Expressions. While the list of static attributes must always be separated by semicolons, the expression attributes can be formed into complex expressions for queries using a set of special boolean operators, of which the semicolon is a special case.

The order of precedence of the operators, starting with the highest binding, is

!	(logical "not") Negates the subsequent attribute, as in [!backbone] and [!type=3]
&	(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 alter the default binding of the operators.

Examples of atom attribute expressions:

O[charge=-1|charge=0&r]—Matches an oxygen atom which is either an uncharged ring atom or has a charge of -1

C[((charge=-1|charge=-2)&r)|!backbone]—Matches a carbon atom which is either not a backbone atom, or is a ring atom with a charge of -1 or -2

Examples of bond attribute expressions:

C-[(type=2&s=c)|type=3]C—Matches two carbons connected by a cis double bond or a triple bond.

C-[!r]C—Matches any two carbons connected by a single bond which is not in any ring.

3.2.4. Bond Pattern Shorthand Notation. Bond expressions of the form

-[type=2 | type=3 | type=aromatic]

are quite common. In these expressions, the bond 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 the bond patterns shorthand form. In the shorthand notation, the bond characters for the allowed types are listed without separators.

Examples:

C=:C is equivalent to **C-[type=2 | type=aromatic]C**

C=#C is equivalent to **C-[type=1 | type=2 | type=3]C**

The bond characters for any bond (~) and for no bond (.) must not appear in the bond pattern shorthand.

3.3. Covered and Noncovering Flags. In some applications, it is necessary to “cover” or partition a structure with an ordered list of patterns. To be fully covered, each atom of the structure must be mapped to one and only one atom of one of the patterns.

In most cases, the structure atom must not be already covered in order for a match to the pattern atom to occur. Thus, the allowed values of the C= attribute for structures are C=Y (already covered) and C=N (not yet covered, the default).

Some patterns require atoms which must be present for a match but are not really part of the fragment to be covered. These atoms may have already been covered by a previous search. For example, a pattern for an amide nitrogen must include, in addition to the nitrogen atom, the supporting carbonyl group. To facilitate this, pattern atoms can have three values of the c= attribute:

c=n The default; the structure atom must not be covered for a match to occur.



Figure 3. Use of R-groups. The pattern on the left will match the structure on the right and R1 will map to the atoms of the ethyl group.

c=y The structure atom must be covered for a match to occur.

c=o Open; the structure atom may be either covered or not covered.

Thus, the pattern for matching an amide nitrogen is **NC-[c=o]=O[c=o]**, in which the carbonyl is required for matching, but is not considered part of the group matched. The carbonyl group may have already been covered by a previous search. In contrast, the pattern **NC=O** matches an amide group, not an amide nitrogen.

The N attribute is used to control whether or not a structure atom matching a pattern atom is marked as covered as a result of the search. In the example of the amide nitrogen pattern, the carbonyl carbon and oxygen should not be marked as covered. Thus the pattern should include the N (noncovering) attribute for these atoms and should appear as **NC[n;c=o]=O[n;c=o]**. The “C=” attribute controls what will match, and the “N” attribute controls whether the structure atom will be marked as covered when a match does occur.

As a simple example, consider the following fragment list used to “cover” a structure:

C(=O)(N)N	(ureas)
C(=O) N[n;c=o]	(amides)
C(=O)	(other carbonyls)
Any	(all other atoms)

The [n;c=o] attributes on the amide pattern allows the amide carbonyl to match even if the nitrogen atom was covered by the urea pattern, as would happen when covering the structure: **NH2C(=O)NHC(=O)CH3**. The second carbonyl is an amide carbonyl but shares its nitrogen with the urea group.

3.4. R and X Groups. R and X groups specify points of variability in a pattern. These groups can match multiple structure atoms and provide place holders for the side chains and groups which vary in a list of hits from a search (Figure 3).

R and X groups match all atoms, starting at the indicated position, which are not already mapped to a pattern atom. They may match a single hydrogen atom.

In SLN the names of these groups must begin with R or X and may have an associated number (i.e., R1, R2, X3). The digits in R and X groups are used for labeling purpose only and have no effect on a search.

An R group is allowed to connect to the structure at only one point (it is a terminal side chain). X groups can attach several times, thus forming rings or connecting various parts of the structure (Figure 4). No two R or X groups can be connected to each other.

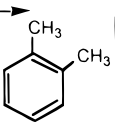
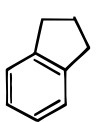
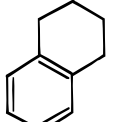
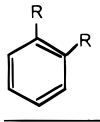
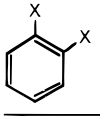
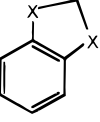
Structure			
Search Pattern			
	Matches	Misses	Misses
	Matches	Matches	Matches
	Misses	Matches	Matches

Figure 4. The difference between R and X groups. When the patterns along the left side are used to search the structures along the top, the results shown in the table are found.

Examples:

Match any ring which contains at least four methylene groups in a row.

C[1]H2CH2CH2CH2X1@1

Match anything with a methyl group and an ethyl group.

CH3XCH2CH3

3.5. Markush Atoms. Markush atoms provide points of constrained variability in a pattern. Like macro atoms, Markush atoms are named with a capital letter followed by one or more lowercase letters, digits, or underscores. Markush atom names cannot be the name of an element (a real atom), nor can they be an "R" or "X" followed by an integer, or the reversed word "Any".

Markush atoms may be defined by the user. As with macro atoms, Markush atoms may be defined globally prior to use in the SLN or may be defined locally within the SLN. If defined within the SLN, the scope is limited to that SLN.

3.5.1. Markush Atom Definition. Simple Markush atoms specify a list of possible fragments which may be matched. The syntax for the simple form of the Markush atom definition is

```
{markush_atom_name:ct1<ct_attrs>|ct2<ct_attrs>|ct3-
  <ct_attrs>...}
```

where:

markush_atom_name is the name of the Markush atom being defined.

ct1, ct2, ct3... are the various choices for the Markush atom.

ct_attrs specify the valences of the CT and the range of occurrence for the CT. Other CT attributes may be included as well.

Examples:

Define a Markush atom which will match any halogen atom.

{Hal:F|Cl|Br|I}

Define a Markush atom which will match any heteroatom.

{Het:O|N|P|S}

Define an atom which will match methyl, ethyl, propyl, or butyl.

{Alkyl1to4:CH3|CH2CH3|CH2CH2CH3|CH2CH2CH2CH3}

Match phenyl or 2-pyridyl.

{Group1:C[1]:CH:CH:CH:CH:@1|\C[1]:N:CH:CH:CH:CH:@1}

This simple form of Markush atom definition is a special case of the general form for Markush atom definition in which the Markush atom name is followed by an *expression* of CTs. All the expression operators are available (! & |; and ()). The definition of a Markush atom may contain other Markush and macro atom references.

Examples:

Match a nitrogen which is not part of an amide and is not in an aromatic ring (note the use of "Any" as the arbitrary end of the aromatic bond):

{Nsimple:N&!NC=O&!N:Any}

Match a non-amide nitrogen or any oxygen atom:

{Group1:(N&!NC=O)|O}

As with macro atoms, the valences of a Markush atom are distinctly different from those of most common organic atoms. Each of the bonds to a Markush atom is a distinct connection. It is therefore necessary to indicate which valence of the Markush atom is occupied by each connection to the Markush atom. As with macro atoms, this is done with the *v=* CT attribute. The attribute designation is followed by a comma-separated list of the atoms to which each of the valences connects.

Examples:

Define a Markush atom, "Arom5", which will match any five-membered mono hetero aromatic ring with five free valences starting at the hetero atom:

{Arom5:C[1]:C:C:C:Het:1(v=5,4,3,2,1)}

Match chains of 1, 2 or 3 methylene groups connected at the ends.

{Alkylchain1to3:CH2(v=1,1);CH2CH2(v=1,4);CH2CH2CH2(v=1,7)}

If the connection information is not specified, it is assumed to be in order of appearance of the atoms in the CT. Thus, the SLN **C-C** is equivalent to **C-C(v=1,2)**.

Some standard Markush atoms are predefined, including

Het	defined as {Het: O S N P} , it matches any hetero atom.
Hal	defined as {Hal: F Cl Br I} , it matches any halogen atom.
Hev	defined as {Hev: Any & !H} , it matches any non-hydrogen atom. Note: Any specification forces this Markush atom to match a single atom.

3.5.2. Markush Atoms References. References to Markush atoms are made using the name defined.

Examples:

Match any halo-benzene:

C[1]H:CH:CH:CH:CH:C:@1Hal

Match any heteroatom with two methyl groups connected to it, as in dimethyl ether and dimethyl amine:

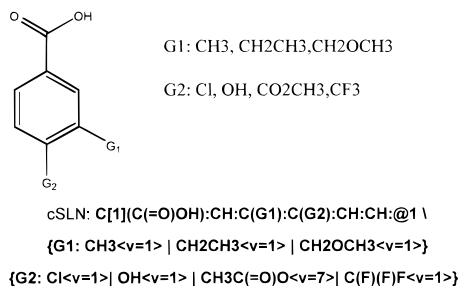


Figure 5. An example of combinatorial SLNs (cSLNs).

CH₃HetCH₃

Markush atom attributes are placed in square brackets after the Markush atom reference. An integer ID may precede the attribute list as with other atom types and, if indicated, must be unique within the CT.

Examples: **Hetero[13]** **Inductive[3]** **Heteroaromatic[31]**

If the bonds connected to the Markush atom are not connected in the natural order, the order must be specified with the Markush atom attribute *v*=. The attribute specification is followed by the comma-separated list of the valences occupied by the connected atom, in order of their appearance in the CT referencing the Markush atom, and maps to the points of connection in the Markush definition.

Examples:

HOCarbonchain[v=1,2]SH

where Carbonchain is defined as

{**Carbonchain:CC|CCC(v=1,3)|CCCC(v=1,4)**}

CH₃Heteroaromatic[v=2,5]CH₃

where the Markush atom is defined as

{**Heteroaromatic:C[1]:C:C:C:Het:@1**
 {v=5,1,2,3,4}}

4.0. SPECIFICATION OF COMBINATORIAL LIBRARIES

The specification of combinatorial libraries is accomplished using markush atoms. A limited set of markush atoms are required: each definition contains a simple list of the groups which are used. No new constructs are required for expression of combinatorial libraries in SLN. An SLN which represents a combinatorial library is often referred to as a combinatorial SLN (cSLN) (Figure 5).

Example: A hexapeptide:

CH₃C(=O)AaAaAaAaAaNH₂(regid="T14382")

Aa is redefined as:

{**Aa:C(=O)CH₂NH(v=6,1)|**
C(=O)C[s=I]H(CH₃)NH(v=9,1)|
C(=O)C[s=I]H(CH(CH₃)CH₃)NH(v=15,1)|
C(=O)C[s=I]H(CH₂CH(CH₃)CH₃)NH
(v=18,1)|
C(=O)C[s=I]H(CH(CH₂CH₃)CH₃)
NH(v=18,1)|
C[1:s=s]H(CH₂CH₂CH₂N@1)C=O(v=12,13)|
C[1]H:CH:CH:C(:CH:CH:@1)CH₂CH(NH)C=O
(v=17,19)|
C[1]:C(:CH:NH:@1)CH₂C[s=I]H(NH)C=O):
CH:CH:CH:CH:@1(v=13,15)|
NHC[s=I]H(C=O)CH₂CH₂SCH₃(v=1,5)|
NHC[s=I]H(C=O)CH₂OH(v=1,5)|
NHC[s=I]H(C=O)CH(OH)CH₃(v=1,5)|}

NHC[s=I]H(C=O)CH₂SH(v=1,5)|
C[1](C:CH:CH:C(:CH:CH:@1)CH₂C[s=I]H(NH)
C=O)OH(v=16,18)|
CH₂(C[s=I]H(NH)C=O)C(=O)NH₂(v=6,8)|
NHC[s=I]H(CH₂CH₂C(NH₂)=O)C=O
(v=1,16)|
NHC[s=I]H(C=O)CH₂C(=O)OH(v=1,5)|
NHC[s=I]H(C=O)CH₂CH₂C(=O)OH(v=1,5)|
NHC[s=I]H(C=O)CH₂CH₂CH₂CH₂NH₂
(v=1,5)|
NHC[s=I]H(C=O)CH₂CH₂CH₂NHC(=NH)NH₂
(v=1,5)|
NHC[s=I]H(C=O)CH₂C[1]:NH:CH:N:CH:@1
(v=1,5)}

Information about the individual choices for each group may be recorded as CT attributes. This can be used to track the catalogue number of the starting material or any other fragment-based information.

Example:

C[1]:C:C:C:C:C:@1Group1|
{Group1:CH₃(aldrich=160-23-14) |CH₂CH₃
(aldrich=895-34-2)}

5.0. SLN SOFTWARE APPLICATIONS

SLN is an ideal method for chemical structure representation over local and even worldwide computer networks. A number of commercial applications (CONCORD, Chemeleon, UNITY, Alchemy 2000) already convert structural formats to and from SLN, with more in development. The advantages of SLN for computer communication include

- low traffic and network loads
- access to remote applications
- no loss of structural or textural information

A number of applications on the World Wide Web use SLN as the medium for transferring structural information quickly and accurately to browsers, Java applets, and other applications around the world. There are routines available from Tripos, Inc. (URL <http://www.tripos.com>) for converting SLNs to GIF graphics files and award winning applications like "Sketch and Fetch" for searching chemical databases from the internet (or intranets). SYBYL uses SLN for processing and analyzing molecular structures as well as for building database queries for searching UNITY databases. The SYBYL Force Field Engine uses SLN searching as a pattern matching routine for both atom typing and assignment of force field parameters. LeapFrog *de novo* design in SYBYL relies heavily on SLN for special force field computations as well as for fragment additions to the growing ligand. The Legion combinatorial builder creates computational libraries of compounds for analysis and searching so that combinatorial synthesis laboratories can maximize the return on their synthetic efforts.

6. CONCLUSION

SYBYL Line Notation is a powerful language which represents molecular structures, libraries of structures, molecular fragments, and structural queries for database searching. Nearly any chemical structure imaginable, including macromolecules, polymers, catalysts, pharmaceuticals, and

even the multiple structures of combinatorial libraries can be written as an SLN string.

SLN provides two- and three-dimensional database queries with full Markush, R-group, X-group, and macroatom capability. This ability is present in the basic language itself, obviating the need for learning multiple languages or applications.

SLN stores full sets of 2D and 3D coordinates. All the information necessary to recreate the structure in a modeling or drawing package is present in a single, terse string of characters. This makes SLN ideal for structure communication over global computer networks between applications sitting at remote sites.

The constructs above have proven useful for specification of structures and queries in many software programs. SLN is currently being revised to include more searching query

and structure representation features. The new version (SLN 3.0) is expected to be released in 1997.

REFERENCES AND NOTES

- (1) Available from Tripos, Inc., St. Louis, MO.
- (2) Available from Exographics, Newark, NJ.
- (3) Developed by Dr. R. Pearlman *et al.*, University of Texas at Austin. Available from Tripos, Inc., St. Louis, MO.
- (4) Weininger, D. J. SMILES, a Chemical Language and Information System. 1. Introduction of Methodology and Encoding Rules. *J. Chem. Inf. Comput. Sci.* **1988**, 28, 31–36.
- (5) Lynch, M. F.; Downs, G. M. Chemical patent database systems. In *Chemical Structure Systems Computational Techniques for Representation, Searching, and Processing of Structural Information*. Ash, J. E., Warr, W. A., Willett, P., Eds.; Ellis Horwood Limited: 1991; pp 126–153.

CI960109J