Semantic Dictionary for Substructure Matching of Chemical Structures with General Descriptors

Takashi Nakayama

Department of Information Science, Faculty of Science, Kanagawa University, 2946 Tsuchiya, Hiratsuka, Kanagawa, 259-12 Japan

Received November 5, 1993[⊗]

A semantic dictionary is introduced for substructure matching of chemical structures whose representation includes general descriptors like generic expressions, common names, topological relations, stereochemistry, etc. It is used primarily to describe structural contexts of reaction sites for transforms which constitute the key part of a knowledge base for computer-assisted synthesis planning, called SPEK in the author's laboratory. In general, the semantic dictionary may be used to describe, as well as in searching, chemical structures with general representation in data base systems, which will enble flexible access to chemical structures. The semantic dictionary is useful because it broadens the descriptive level range for chemical structures and provides users with a flexible interface to application systems.

1. INTRODUCTION

Building a large-scale knowledge base that has sufficient coverage for practical use is one of the major problems tackled in the field of knowledge base systems. In other words, efficient knowledge acquisition on a timely basis is a serious bottleneck for achieving expert systems that need a large-scale knowledge base.[1,2] The problem is the same for the synthesis-planning expert systems based on a knowledge base, that is, a collection of so-called "transforms".[3] Specific problems are to first obtain a sufficient quantity of transforms and to next maintain the quality for the transforms, as well as to maintain the integrity control with respect to the entire knowledge base. However, it does not mean that the data and knowledge representation and usage problem is completely overcome and that the desired results have been established. On the contrary, there are still difficult problems to be tackled. First, the available knowledge could always be incomplete, that is, all the information needed for synthesizing targets may not always be clarified. Second, the representation of abstract concepts that are used for describing transforms (specifically terms that stand for some kind of structural situations) is still left to be considered. Some of these concepts are generic chemical structures, states and/or relations, such as stereochemistry, three-dimensional structures, electronic structures, etc., in addition to topological relationships. Several approaches have been proposed to solve these problems. For instance, PC-SYNTREX,[4] which is classified into case-based reasoning systems, utilizes specific reaction data (actually, the number of reaction data contained is about 40 000 at present) rather than refined general rules. Another approach tries automatic knowledge extraction and self-organization for the knowledge base.[5–7] Several methods have been proposed for the representation of generic chemical structures.[8–11] In particular, a well-known series of works by Lynch and his group treats generic chemical structures in patents formally and supports a practical means for computer storage and retrieval.[12–16]

The semantic dictionary (called SD hereafter) is intended to solve the treatment method for generic structures and structural contexts, that is, structural relations or situations in chemical substructures. SD is a component of SPEK (computer-assisted synthesis planning system, based on empirical knowledge), which is under development at the author's laboratory, and its aim is to furnish flexibility to the description and application of transforms. The terms registered in SD are also used in the knowledge acquisition session in SPEK.[17] In SPEK, terms which stand for generic structures and abstract concepts are allowed to be used as descriptors of scope and limitations of transforms as well as of specific structures. SD is a collection of definitions of meanings of these terms, where the definitions mean that the terms are given those representations with which SPEK could interpret target structures. The interpretation of target structures means that SPEK segregates those substructures specified by the terms used in the applied transform. The meaning of a term is considered as a function which reveals its figures when it is used to represent some specific situation. Since the reason for defining term meanings in SPEK is substructure matching, the definition manner mentioned here makes sense. The meaning of a term corresponding to a specific substructure is clear. It only has to have a specific connection table in order to be matched with target structures at a specific level. The matching is implemented by a conventional subgraph matcher. On the contrary, the definitions of the meanings for generic structures and/or abstract concepts are complex, because they cannot be defined by simple connection tables to which a conventional subgraph matcher can be applied. They have to be given a matching mechanism of their own. Actually, SD gives those definitions of both specific and generic terms which appear in the description of structural contexts for transforms as predicates and their parameters. In other words, SD operates on a general basis in the description and interpretation for both reaction data bases and knowledge bases. Verbal expressions for topological relationships, stereochemistry, three-dimensional effect, and electronic effects are supported in conventional systems, such as LHASA, SECS, etc., as a feature of chemical data base language (CHMTRN, ALCHEM, etc.).[18,19] SD can be seen as an extended module for the feature, which is a generalized and independent module for flexible matching of chemical structures.

2. STRUCTURAL CONTEXTS

The transform knowledge base (TKB) is a major knowledge base used in SPEK, made of a collection of transforms. The major part of a transform is a transformation rule for chemical

[⊗] Abstract published in *Advance ACS Abstracts,* April 15, 1994.

structures, which is applied to a certain target structure to obtain its precursor structure. Usually, those rules are represented in the form of a general structural transformation with various restrictions.

The restrictions specify reaction conditions and structural situations expected to appear in targets, corresponding to the scope and limitations for reactions, which determine validities for a certain transform application to a target. Those structural situations are called *structural contexts*. Consequently, the rules are expressed formally as follows:

IF reaction site P with structural context SC is found in target T,

THEN apply transformation rule P $\rightarrow$ S to T,

WHERE validity for SC is evaluated properly.

Actually, several structural contexts are associated with a single transformation rule, giving variations of a transform. The WHERE part above is described in terms of pairs of structural contexts and corresponding rating values. Specifically, it is also expressed in the form of the IF–THEN rule, where the IF part represents a structural context and the THEN part indicates a rating. A structural context describes a relation between a reaction site and its surrounding substructure. The following examples are simple WHERE parts:

(1) IF alkyl group is adjacent to atom C1, THEN add −20 to rating.

(2) IF atoms C1 and C2 are part of a ring, THEN add +90 to rating.

(3) IF electron-withdrawing group W is NO_2, THEN add +60 to rating.

Each structural context (IF part) of these examples is interpreted by a reasoning system in SPEK. Atoms or superatoms, corresponding to C1, C2, W, etc., are recognized in a target structure, as are generic terms corresponding to alkyl group, ring, electron-withdrawing group, etc. As the IF parts above show, structural contexts state topological relations (adjacency between atoms/substructures and other connectivities), *part-of* relation, *is-a* relation, etc., which can be expressed in the form of predicates as follows:

(1) adjacent(alkyl, C1)

(2) part-of((C1,C2),ring)

(3) is-a(W,NO_2)

where adjacent, part-of, and is-a are predicates, and where the generic (verbal) representations of chemical structures (alkyl, ring, W), atomic symbol (C1), and a set of atoms (C1, C2), functional group (NO_2), etc. are parameters applied to those predicates. The SD gives definitions of the meanings for these predicates and parameters so that they are interpreted appropriately as structural contexts. Conversely, both predicates and their parameters are terms for describing structural contexts. Further, a predicate itself could also be a parameter of another predicate. A predicate as a parameter is used to represent a kind of structural relation which cannot be expressed by a simple connectivity relation.

3. SEMANTIC DICTIONARY ORGANIZATION

SD can be utilized as an independent module by other application systems. The relationship between SD and its application system, i.e., MATCHING module in our system SPEK, is indicated in Figure 1. It shows that the module matches a substructure in a target with structural contexts of a transform in TKB by consulting SD. (If we substitute TKB and TARGET in Figure 1 with a chemical structure data base and a structural query, respectively, we get a data base system of generic chemical structures based on SD.) The MATCHING body module is a matching function, which

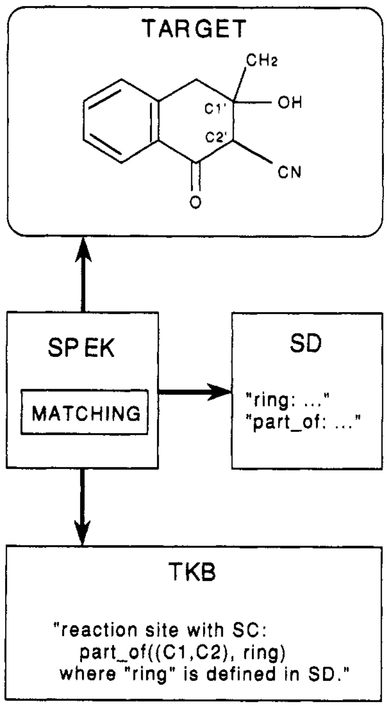

Figure 1. Relationship between SD (semantic dictionary) and MATCHING module in SPEK (computer-assisted synthesis planning system based on empirical knowledge). MATCHING module matches a reaction site in TARGET with SC (structural context) described in a transform in TKB, based on the meaning of "ring" defined in SD.

receives two parameters, a structural context and a target structure, expressed formally as

match(StructuralContext,TargetStructure)

where the parameter StructuralContext is a predicate implemented as a function and TargetStructure is expressed in the form of a connection table. Actually, the function match consists of a collection of matching functions prepared for various types of structural contexts. An appropriate matching function is invoked according to the given parameter StructuralContext, which is to be matched with a target structure specified by the parameter TargetStructure. In the matching process, SD is consulted to substantiate the StructuralContext, and appropriate procedures may be built dynamically when needed. The matching between StructuralContext and TargetStructure is executed atom by atom finally if the StructuralContext is given in the form of a specific structure. However, the matching process could be carried out at several specifications for StructuralContext with generic descriptions, which is the mechanism supported by SD.

The SD defines the meanings of descriptors for structural contexts (i.e., structural entities), which are categorized as: (1) specific/generic rings, (2) specific/generic substituents and functional groups, (3) specific/generic compounds, and (4) predicates (a subset of the predicates allowed for the parameter StructuralContext). A predicate which appears as a parameter of another predicate is interpreted as a structural entity which holds the relation stated by the parameter (predicate). Some geometries can be specified by this interpretation. Examples are shown in the explanation of predicates in the following section.

3.1. SD Term Contents. A structural entity defined in SD forms a unit in SD, called an SD term. Items of an SD term are shown in Figure 2. Not all the items are meaningful to each SD term, and some of the items may be omitted. Each individual item has the following meaning.

```
entry name
   name
   formula
   predicate
definition
   connection  table
   procedure
   extension
   attribute
superclass
subclass
synonym
formula
exception
contain
not-contained
contained-in
other associations
```

Figure 2. Transform items. Items other than an entry name are optional.

Entry Name. A headword for SD, which is also a notation itself of the term used in the description of structural contexts. Of course, an entry name is unique in SD, but its synonyms may be used freely, and various specificities may be used as modifiers for generic terms in structural contexts, provided that the synonyms are registered and recognized in SD and that the modifiers are parameters interpreted dynamically as restrictions in a matching process. The following three types of notations are allowed for entry names.

(1) Names: any kind of name could be registered in SD, but only if such names are used in the descriptions of structural contexts consistently. Systematic names, based on IUPAC nomenclature, are partly accepted at present. Primarily, however, trivial names, common names, and ad hoc names for ring systems employed in SPEK (described in Section 3.2) are expected, because SD is organized to accept customary and generic usage of terms.

(2) Constitutional Formula/Rational Formula: linear notations of a constitutional formula and/or a rational formula written in alphanumeric characters, including the special symbols (,), -, and =, are accepted, provided that they are interpreted uniquely. Names and constitutional formulas may become synonyms with regard to each other.

(3) Predicates: some of the predicates used for describing structural contexts are accepted as entry names. The difference between a predicate as an SD term and one as a structural context is that the former is interpreted as a structure itself, whereas the latter gives a Boolean value as a result of evaluating the structure. Further, the use of predicates as SD terms means that a structural context (i.e., a predicate) is allowed to have a predicate for its parameter, that is, a predicate may be expressed recursively. However, whether or not a recursive expression for a structural context makes sense is determined at the time the predicate is interpreted. For instance, the statement "present(cis(X,Y),anywhere)" makes sense, stating that a substructure contains X and Y in a *cis* relation. On the other hand, the statement "is-a(cis(X,Y),-trans(W,Z))" does not make sense, because it states that a substructure containing X and Y in a *cis* relation is a substructure containing W and Z in a *trans* relation. Meaningless statements like this could be checked and excluded at the time they are registered to TKB. In SPEK, however, they are free from the TKB registration, because whether or not a predicate (a structural context) makes sense can be detected automatically as a result of the matching between the structural context and a target structure.

Definition. The definition of the meaning for an SD term is specified here. In other words, the representation for a structural entity is given here. Four types of definition methods are prepared: (1) connection table, (2) procedure, (3) extension, and (4) attributes. As is suggested by the previous explanation, an appropriate procedure has to be invoked according to the type of the predicate parameter when StructuralContext is matched with TargetStructure. That is, a mechanism, which selects an appropriate matching method according to the type of an SD term, is supported if the parameter is an SD term. The type of SD term definition is checked when the method is selected. However, it is not necessarily specified exclusively, and multiple definitions are allowed.

(1) Connection table: specific structures are represented in the form of conventional connection tables. Simple generic terms and/or abbreviations for some pseudoatoms, such as "hal" for halogen atoms, "R" for alkyl group, "Ac" for acyl group, etc., may also be incorporated in the connection tables. Since pseudoatoms are expanded to specific substructures when they are matched with a target structure, the definition by connection tables is virtually combined with the one by procedures explained in the following. Further, in the case of a single value substitution for a pseudoatom, like Me for methyl, the value is assumed to be determined at the time it is bound to a corresponding matching object, although actually it has a specific definition.

(2) Procedure: structural entities which contain generic descriptions can be defined in the form of procedures. An SD term is regarded as a procedure name for this type of definition. If modifiers are added to the term, they are interpreted as parameters for the procedure. Various specificities for generic structures are represented with the facility of modifiers. For instance, "ring" may be registered as an SD term of the name without modifiers, and it is defined as a procedure which finds any rings in a target structure when interpreted in a structural context. If modifiers are added to it, like "6_ring", "aromatic_ring", etc., they (i.e., 6, aromatic, etc.) are set up as parameters and given to a ring-finding procedure to restrict the ring systems to be found. The syntax of modifiers are explained in the next section. In particular, structural entities, expressed in the form of predicates, always belong to this type of definition.

(3) Extension: when it is difficult to give a definition for a generic structure by a general procedure, or when it is desirable to add instances to a definition by a procedure, a set of subordinate concepts and/or instances may be specified as an extension. A set of specific structural entities formed in this way may be regarded as a kind of definition of a term. For instance, a term "electron_withdrawing_group" is given a set of instances, {NO_2, CN, =C—O, ...), which otherwise would have to be given a procedure by which to calculate electronic effect. The extension is useful because lexical matching with respect to those instances would be sufficient without calling an ad hoc procedure. For instance, a term "electron_withdrawing_group" in a structural context is matched lexically with NO_2 in a target by specializing the term "electron_withdrawing_group" and choosing the extension member NO_2 as a matching object.

(4) Attribute: it is possible to specify a category for structural entities as a combination of attributes. The attribute here is a property of a structural entity, which is extracted from the viewpoint of organic synthesis. For instance, a term "bulky_electron_repelling_group" is defined as an attribute

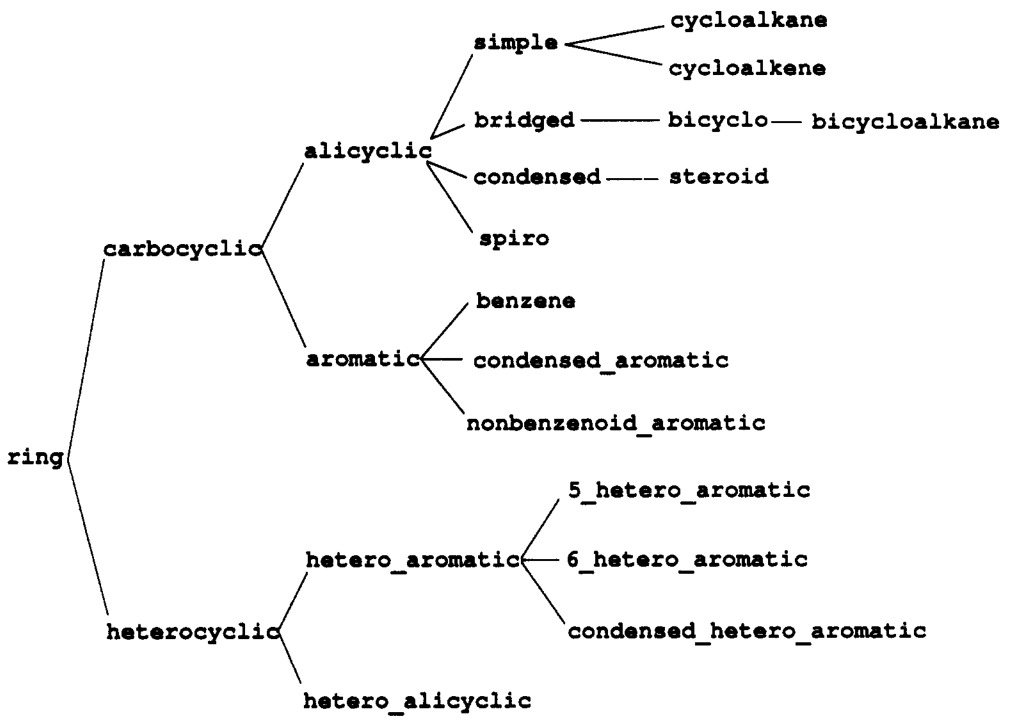

Figure 3. Is-a hierarchy example showing part of a ring hierarchy, each class node (ring system) having its superclass and/or subclass.

definition type, which means the intersection of two structural entity sets of constituent terms, "bulky" and "electron_repelling_group". It should be noted that a constituent term, which may be defined by a single attribute, is actually registered as an SD term of another definition type, as mentioned above. Terms like "electron_withdrawing", "bulky", etc., which give qualitative expressions, are attributes of some structural entities; such terms are also employed to define SD terms by extensions or procedures as mentioned above.

Superclass. A concept which has a more general or a broader meaning than the SD term in question is specified as a member of a superclass, provided that it is also registered in SD. Inversely, the SD term can be seen as an instantiation or a subclass of the superclass concept. The superclass and subclass described below together form a class hierarchy (generalization/specialization hierarchy; *is-a* hierarchy) in SD.[20] Any number of superclasses are allowed to be specified for a single term, so that any number of class hierarchies may be generated, resulting in a class lattice as a whole. Accordingly, multiple inheritance of attributes and methods is realized for SD.

Subclass. The subordinate concepts of the term are specified here, provided that they are also registered in SD. They may coincide with members of a definition by extension. However, a subclass is defined along the class hierarchy (called also *is-a* hierarchy hereafter), whereas "extension" is formally independent from the hierarchy, since it is simply a list of subordinate terms and/or instances of a generic term. SD terms are implemented as objects generated from classes, defined on the basis of an object-oriented data model,[21] that are possibly placed somewhere in a class hierarchy and hold the same hierarchical relationship with the class hierarchy. For instance, *is-a* hierarchy among ring systems is shown in Figure 3, where each term (object) is generated from a corresponding class definition. There are cases when only one object is generated from one class definition, especially for generic terms.

Synonym. Synonyms are registered in this item in the form of a synonym list. This item is not used for access to SD terms. Instead, it is used for getting all the synonym members of an SD term. An SD term, which has this synonym item, is a key term for all the synonyms and is accessed from any member of the synonym list. This allows a flexible choice of terms in the description of structural contexts. Access via a synonym is realized by linking the synonym entry with this key term. Simple symbols, such as Me (methyl), Ac (acyl), etc., are also counted as synonyms.

Formula. A constitutional/rational formula, which is not registered as an entry name, may be set here just for visual convenience. It may include generic atoms.

Exception. A term which is not defined by a specific code but by other methods may include structural entities that should be excluded from the definition. Those exceptional entities are listed here. The exception list is made of SD terms. That is, only structural entities that are registered as SD terms are recognized as exceptions.

Contain. If a term contains characteristic substructures, they are set here as a list of entry names. This item, together with the following two items, *not-contained* and *contained-in*, forms a *part-of* hierarchy with respect to structural entities, which is different from an *is-a* hierarchy made of super/subclasses.

Not_contained. Contrary to the item *contain* above, this is for structural entities that should not be contained in the term. A list of entry names is set in this item.

Contained_in. If the term is contained in other structural entities as a component, those parent terms are listed here.

In addition to those items for the *is-a* hierarchies and/or the *part-of* hierarchies, other associations among structural entities, such as *derivative_of*, *residue_of*, *substitution_product_of*, *condensation_product_of*, etc., are prepared in order to support a wide range of search views. These associations form a set of structural entities of their own and could be utilized in analogical reasoning, which is under development as a component of SPEK, as a base of similarity perception, in addition to utilizing the *is-a* and/or *part-of* hierarchies. An SD term example for carboxylic acid is shown in Figure 4.

3.2. Generic Terms Representation. Generic terms are defined mainly by procedures. As described in the previous section, *modifiers* may be added to terms defined by procedures which express the specificity of genericness of structural representation. In particular, a naked generic term, which

```
carboxylic acid                  // entry name: name
1 1 1 6 8 8 200 2 1 1            // definition: connection table
formic acid,                     // definition: extension
acetic acid,
propionic acid,
butyric acid,
valeric acid,
oxalic acid,
malonic acid,
glutaric acid,
adipic acid,
acrylic acid,
methacrylic acid,
maleic acid,
fumaric acid,
benzoic acid,
phthalic acid,
terephthalic acid
acid                             // superclass
nucleocarboxylic acid,           // subclass
monocarboxylic acid,
dicarboxylic acid,
tricarboxylic acid,
chain carboxylic acid,
aromatic carboxylic acid,
saturated carboxylic acid,
unsaturated carboxylic acid,
fatty acid
RCOOH                            // synonym
RCOOH                            // formula
carboxyl group                   // contain
acid halide,                     // other association:
acid anhydride,                  //   derivative
ester,
amide,
nitrile
hydroxy acid,                    // other association:
oxo acid,                        //   substituted carboxylic acid
amino acid
```

Figure 4. SD term example for "carboxylic acid". A comment for an item is written on the right-hand side of //, suggesting the corresponding item. Elements of a set item are listed across lines delimited by a comma.

means a generic term without modifiers, gives the broadest coverage of the term. However, modifiers are not part of entry names for SD but are interpreted as parameters for the procedures which define those terms. The separation of modifiers from entry names, on the one hand, suppresses the number of terms to be registered in SD and, on the other hand, virtually results in a large collection of terms with flexible expressiveness. Major categories of generic terms and the syntax of modifier usages are described in the following.

Ring Structures. Three types of descriptors are prepared for ring structures as modifiers.

(1) Type: types of rings are specified from the viewpoints of ring characteristics shown in Figure 3 as a ring hierarchy, such as carbocyclic/heterocyclic, alicyclic/aromatic, simple/spiro/condensed/bridged.

(2) Size: a ring size has two meanings here. The first is the number of member atoms for simple rings. The second is the cardinality of SSSR (smallest set of smallest rings), that is, the number of ring closure bonds for condensed/bridged rings.

(3) Atomic composition: the number of heteroatoms for heterocyclic ring structures, that is, the number of oxygen/nitrogen/sulfur/phosphorus atoms.

Any combination of these descriptors may be used as a modifier for the root term "ring", and a set of modifiers for a ring structure is called a ring descriptor set, which is treated as a parameter set for the defining procedure to interpret, although very few descriptors are used in most cases. The syntax for the ring descriptor usage is as follows:

[type][size]ring[(([#O]O][[,#N]N][[,#S]S],[[#P]P])]
for simple rings

[size][type]rings[(([#O]O][[,#N]N][[,#S]S],[[#P]P])]
for others

Descriptors between brackets may be omitted. The viewpoints for *types* of rings are not always exclusive for a ring but are used in combination, like "spiro_heterocyclic_ring", where the order of attribute values (modifiers) can be arbitrary, because the content of the combination is obtained by intersection between the classes. The *size* of a ring is specified in terms of an integer for a rigid size or a pair of integers in case there is a range of sizes. A range of sizes is specified using hyphenation: 3-5, 5-*, *-6, meaning from 3 to 5, greater than or equal to 5, less than or equal to 6, respectively. The sequence for *type* and *size* and the stem part ("ring" and "rings") are different between the two syntaxes, which is intended to give a rather natural sense of words. In any case, *type* and *size* appear as prefixes for a term, whereas *atomic composition* appears as a suffix. Further, prefixes are concatenated by underscores and suffixes are enclosed by parentheses. The atomic composition is written in the form of a semilogical expression, with respect to heteroatomic occurrences. The number mark (#) means a multiplier (occurrence) of the atom, and the comma means conjunction (coexistence of specified heteroatoms), which may be replaced by a slash, meaning disjunction. The default interpretation for a multiplier, *i.e.*, a case where no multiplier is specified, is that at least one atom exists in the ring. Following are some examples for descriptions of ring systems with their interpretations:

6_ring	six-membered ring
aromatic_ring	aromatic ring
aromatic_5_ring	aromatic five-membered ring
5_ring (2N)	five-membered ring with two nitrogen atoms
2_rings (N, S)	ring system of cardinality 2, containing both N and S atoms

These expressions are used for describing structural contexts in transforms, which, accordingly, determine matching level (specificity) between structural contexts and target ring systems. The dynamic determination of a matching level is a principal feature of matching mechanism in SPEK, called an adaptive matching mechanism. In other words, the adaptive matching mechanism dynamically adjusts the matching level for a given target structure according to the descriptive level for the structural context in question. The method for implementing generic ring systems, together with general knowledge about ring structures, allows realization of the adaptive matching mechanism. Specific matching behaviors are described in the next section. Inversely speaking, the adaptive matching mechanism gives the definition for semantics regarding terms in SD, which are used as descriptors for structural contexts. For instance, a generic term "aromatic_ring" which appears in a structural context could match any of specific ring structures like benzene, pyridine, thiophene, naphthalene, etc. contained in target structures.

Trivial and/or common names which represent ring structures, such as benzene, pyridine, etc. as just mentioned, are also included in SD consistently by employing structural codes (connection table equivalents) as their definition.

Functional Groups. Terms for functional groups that contain generic expressions are also given their meanings in the form of procedures. Table 1 shows some generic functional groups in this category. The specificity of generic functional groups is indicated by specific parameters for each procedure, which is different from that for generic rings for which a ring descriptor set is commonly used. For instance, an acyl group could be defined as a connection table for "RCO-", where a symbol R would be employed as a generic atom and matched with target structures at this symbol level, if the acyl group in the targets is also expressed using the symbol R. However, if a target structure is represented specifically (not using R), then a substructure ">C=O" has to be found first and then examined further to determine whether or not it is connected to any hydrocarbon, which is actually the content of "R". A procedure FindHydrocarbon is activated for this purpose, which makes up a part of an entire procedure FindAcyl. That is, the definition for functional groups which contain generic symbols includes a procedure as its own part. Parameters as a constraint for the generic feature are given specifically to each procedure. For instance, the number of constituent carbon atoms (or a range of the number) is given to FindHydrocarbon.

Table 1. Examples of Generic Functional Groups Defined by Procedures

name	procedure
acyl/RCO	FindAcyl
aliphatic_hydrocarbon/aliphaticR	FindAliphaticHydrocarbon
alicyclic_hydrocarbon/alicyclicR	FindAlicyclicHydrocarbon
alkyl	FindAlkyl
alkylene	FindAlkylene
aryl	FindAryl
alkoxyl/RO	FindAlkoxyl
alkyoxycarbonyl/COOR	FindAlkoxylcarbonyl
hydrocarbon/R	FindHydrocarbon

Table 2. Examples of Generic Terms Defined by Extension

name	extension
hydrophobic/lipophilic/oleophilic	hydrocarbon
hydrophilic	hydroxyl, carboxyl, amino, carbonyl, sulfo
electron_withdrawing/ electron_attracting/ewg/eag	nitro, nitroso, carbonyl, carboxyl, trialkylammonium
electron_releasing/ electron_donating/ electron_repelling/erg/edg	amino, hydroxyl, methoxyl
acyl/RCO	formyl, acetyl, malonyl, benzoyl

A common part of descriptions for generic functional groups in structural contexts is the usage of symbol R, whose syntax is as follows:

$$R[(\#C)]\textit{specific-substructure}$$

where R means hydrocarbon, which contains carbon atoms of specified number (#C), and where #C gives the range for the number of carbon atoms in the form of *n*1–*n*2. For instance, both "acyl" and "RCO-" have no restriction on the number of carbon atoms, whereas "R(1-3)CO-" means an acyl group with R containing from one to three carbon atoms.

Extensions. As mentioned in Section 3.1, generic structures, for which implementing topological procedures are difficult or cumbersome, may be defined by extension, which is actually a set of instances and/or subordinate concepts. Moreover, any generic terms may have extensions in addition to their proper definitions. Table 2 shows some examples of functional groups which have definitions in the form of an extension. Every element in an extension should also be registered beforehand in SD as a term. Since members in an extension may coincide with members in an *is-a* hierarchy, the membership at some class would be inherited to classes of lower levels, though, formally, extensions are apart from *is-a* hierarchies. For instance, "hydrocarbon" is an element of an extension defined for hydrophobic_group in Table 2, and all its subclasses and their descendants, like alkyl, phenyl, methylene_groups, etc., belong to the extension.

Predicates. Table 3 shows predicate term examples. Topological relationships represent positions of substructures in question with respect to some reference atoms or substructures. Constitutional relationships represent presence/

Table 3. Examples of Generic Terms Defined by Predicate

category	predicate
topological relation	α–to, β–to, adjacent, distant, path, ortho, meta, para
constitutional relation	present, is–a, part–of
stereochemistry	cis, trans, Z, E, chiral, R, S, axial, equatorial, exo, endo
3-D relation	hindered, bulky

absence, part/whole, membership, etc. for some structural entities. The *cis*/*trans* relationships in stereochemistry are supported as well as the (*Z*)/(*E*) relationships for common use. Three-dimensional relationships represent the states of substructures in three-dimensional space qualitatively. The implementation of predictate terms is also in the form of procedures, which is similar to other procedural definitions. The point which characterizes predicate terms is that they assert some structural relationships which exist in given structures, whereas terms for other kinds state a substructure itself. In other words, predicates refer to structural situations (contexts), while the others refer to structural entities, that is, the components of the situations.

4. MATCHING MECHANISM

As already shown, meanings for SD terms are given in the form of structural entities/situations which are employed as structural context descriptors (SCDs). They are finally fixed through the evaluation of associated procedures, which determines whether or not the structural entities/situations are contained in target structures. That is, SD gives the basis for the substructural matching evaluation between SCD and target structures, where the structural representation and matching mechanism together form the meanings of SD terms (see Figure 1). The matching mechanism applies to both substructures and entire structures. Matching mechanism examples are shown in the following for each SD term category. The matching is carried out by evaluating the function,

match(StructuralContext,TargetStructure)

Since the first argument StructuralContext is given as a predicate (*i.e.*, procedure/function), match(...) is a function which applies the predicate to the second argument TargetStructure.

Examples 1–4, shown below, assume the notation, match(predicate(...),TS), which is an abbreviation for the same function above, where predicate(...) is a function which represents a structural context (SC), and TS means a connection table for TargetStructure. Similarly, abbreviations are used appropriately to make the notations simple.

4.1. When SCD Indicates Specific Trivial Names.

SC example: adjacent(nitro,C1).

Interpretation: the predicate asserts that a nitro group is connected to carbon atom C1 directly, where C1 is an atom discriminated uniquely in a transform's structure.

Matching procedure: function match(adjacent(nitro,C1), TS) is evaluated as follows. (Actually, this indicates a usual specific substructure search case.)

s1. The first parameter for match(...), *i.e.*, adjacent(nitro, C1), is evaluated.

s11. The first parameter for adjacent(...), *i.e.*, nitro, is evaluated. Since it is known that a first parameter of adjacent is always an SD term from the predicate syntax, its meaning is looked up in SD. As a result, "nitro" is found to be defined by a connection table as a specific term.

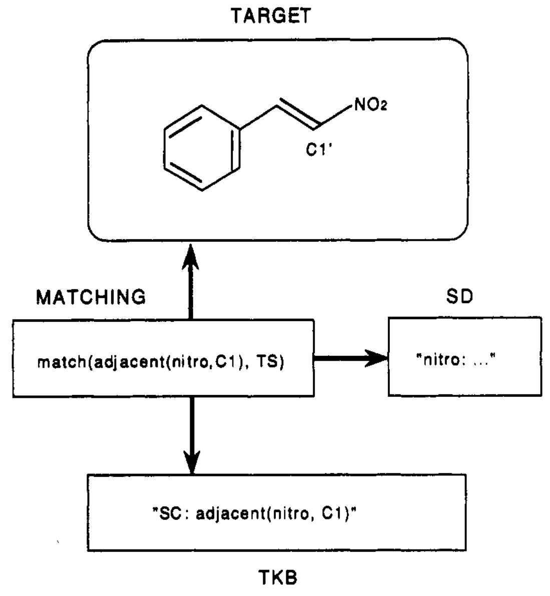

Figure 5. Substructural matching between the target structure and structural context: adjacent(nitro,C1).

s12. A carbon atom (C1′) corresponding to the second parameter (C1) is searched for on the target, as shown in Figure 5. (This step is independent from SD, being accomplished in relation to the core reaction site of the transform in question.)

s2. The structural context (SC), interpreted as above, is searched for on the target structure. Since the C1′ is already fixed, nitro group, which is adjacent to C1′ on the target, is searched. Formally, the following function is invoked,

FindAdjacent(CT(nitro),C1′,TS)

where CT(nitro) represents a connection table for the nitro group and FindAdjacent(...) tries to find a nitro group which is adjacent to C1′ on TS.

4.2. When SCD Indicates Predicates.

SC example: present(cis(COOH,OH),anywhere).

Interpretation: the predicate asserts that COOH and OH groups that are in *cis* relation exist anywhere.

Matching procedure: function match(present(cis(COOH, OH),anywhere),TS) is evaluated as follows.

s1. The first parameter for match(...), *i.e.*, present, is evaluated.

s11. The first parameter, cis(...), is evaluated. Since it is known that the first parameter for present is always an SD term, its meaning is looked up in SD. As a result, cis(COOH,OH) is found, where it is defined by procedure, *i.e.*, function FindCis(...) as a predicate. It should be noted that the term cis(COOH,OH), to which specific parameters COOH and OH are bound, is not registered in SD, but the parameters are dynamically bound to the relevant function.

s111. COOH and OH are bound to variables X and Y for the predicate, cis(X, Y).

s112. Parameters COOH and OH are looked up in SD, because it is known that they are SD terms from the syntax of FindCis(...). These parameters are defined by connection tables as specific terms and are accessed in the form of CT(COOH) and CT(OH).

s12. The second parameter for present(...), *i.e.*, anywhere, is evaluated. This is interpreted as indicating that the search range for the first parameter cis(COOH,OH) is assumed to be the entire structure TS (with no constraint).

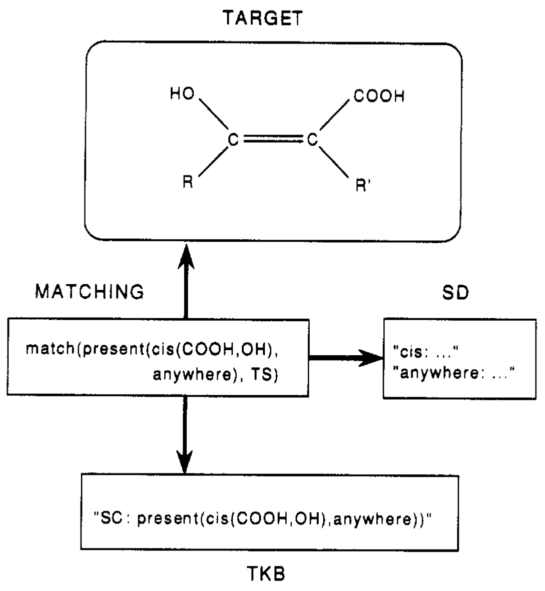
TARGET
HO
COOH
C
C
R
R'
MATCHING
SD
match(present(cis(COOH,OH), anywhere), TS)
"cis: ..."
"anywhere: ..."
"SC: present(cis(COOH,OH),anywhere))"
TKB

Figure 6. Substructural matching between the target structure and structural context: present(cis(COOH,OH),anywhere).

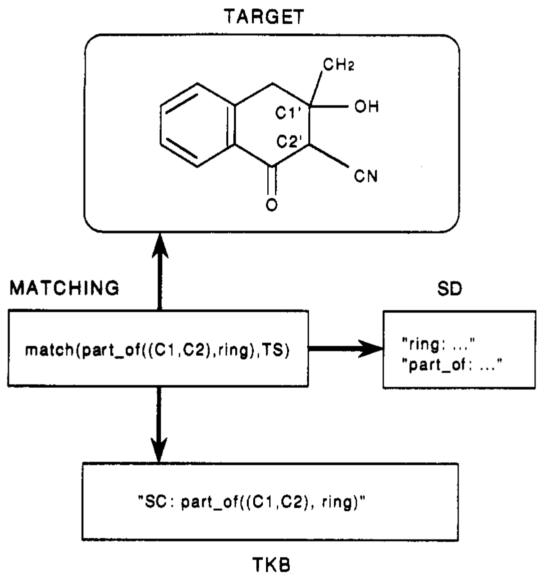

Figure 7. Substructural matching between the target structure and structural context: part_of((C1,C2),ring).

s2. The SC, interpreted as above, is searched for on a targetstructure as shown in Figure 6. Formally, the function below is invoked,

FindCis(CT(COOH),CT(OH),TS)

where CT(COOH) and CT(OH) are connection tables for COOH and OH and FindCis(...) searches for COOH, OH, and a path between them on TS to see if they are in a *cis* relation.

4.3. When SCD Indicates Generic Ring Systems.
SC example: part_of((C1,C2),ring).

Interpretation: the predicate asserts that two specified carbon atoms (C1,C2) are part of any ring system. (Any rings will do, because individual ring modifiers are not specified.) The C1 and C2 are fixed atoms in the transform's structure.

Matching procedure: Function match(part_of((C1,C2), ring,TS) is evaluated as follows.

s1. The first parameter for match(...), *i.e.*, part_of((C1, C2),ring), is evaluated.

s11. A set of carbon atoms (C1′,C2′) which corresponds to the first parameter (C1,C2) is searched for on the target, as shown in Figure 7. The result is SET=(C1′, C2′), where SET is a data type designating a set of atoms. This evaluation behavior is different from those for the two examples above because of the difference in the predicate syntax.

s12. The second parameter, ring, is evaluated. It is known that the second parameter is always an SD term from the syntax for predicate part_of. "Ring" is defined by a procedure FindRing(...) as a generic term. (At this stage, if modifiers are added to "ring", it is interpreted and set to a ring descriptor set.)

s2. The SC, interpreted as above, is searched for on a TS. Formally, the function below is invoked,

FindRing(SET,RDS,TS)

where RDS represents a ring descriptor set. Since there are no modifiers to the "ring" in this case, RDS is not set up. The first parameter, SET, represents a set of atoms (C1′,C2′) which has to be contained as members of the "ring" to be found. It is not a general classifier of ring systems but is given from the outside and is distinguished from RDS, although it can be seen as a restriction to ring systems. FindRing(...) searches for a minimum ring on a TS which contains C1′ and C2′.

4.4. When SCD Indicates Generic Functional Groups.
SC example: distant(acyl,5,C1).

Interpretation: the predicate asserts that the acyl group is connected to carbon atom C1 with distance 5, meaning that a path exists which contains five bonds between C1 and the acyl group. The carbon atom C1 is fixed in the transform's structure.

Matching procedure: function match(distant(acyl,5,C1), TS) is evaluated as follows.

s1. The first parameter for match(...), *i.e.*, distant(acyl,5, C1), is evaluated.

s11. Since the first parameter, acyl, is an SD term (known as such from the syntax of distant), the meaning is consulted in SD. As a result, "acyl" is found to be a generic functional group, defined by a procedure FindAcyl(...). At the same time, the term "acyl" has an extended connection table, which includes a generic symbol R, that is, a connection table corresponding to R-(C=O)-. The latter definition could be applied to such TS, since the symbol R is used directly to represent its structure and needs not be expanded to specific substructures. Otherwise, the former definition for a procedure FindAcyl(...) would be applied, and as is done in this example in Figure 8. This is an example of multiple definition of a term.

s12. Find a carbon atom C1′ on TS corresponding to the second parameter C1.

s2. The SC, interpreted as above, is searched for on TS.

s21. Formally, the following function is invoked,

FindPath(FindAcyl,5,C1′,TS)

where the first parameter, FindAcyl, is a function to which the successive parameters 5, C1′, and TS are passed:

FindAcyl(5,C1′,#C,TS)

where 5 designates the distance from the C1′, and where #C specifies the number of carbon atoms contained in the acyl group. (The #C is not specified in this case, because no modifiers are specified in the first SC expression). The behavior of FindPath(CT(RCO),...)

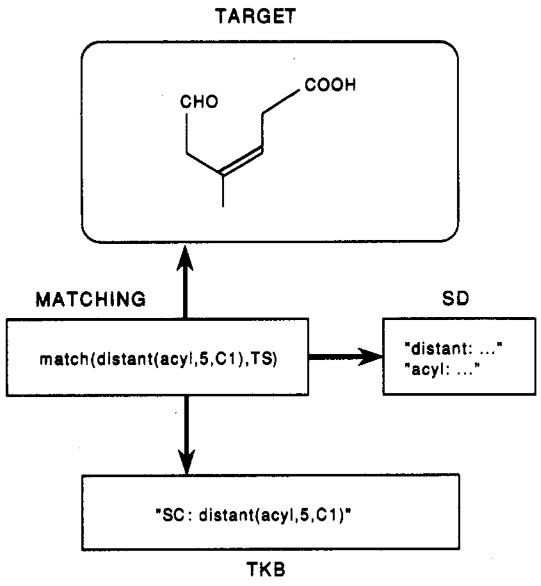

Figure 8. Substructural matching between the target structure and structural context: distant(acyl,5,C1).

would be different from the one shown above, which is similar to that of FindAdjacent(CT(nitro),...) in example 1. This is another example of multiple definition of a function, which is discriminated by the value of the first parameter to be a CT(...) or a function name.

s22. The function FindAcyl(...) is dynamically constructed as a specialization of a procedure FindR(...), which finds a hydrocarbon (R):

FindR(CT(C=O),#C,TS)

That is, FindAcyl(...) is regarded as the derivative of FindR(...), which takes connectivity constraint -(C=O)- and the number of carbon atoms contained in R as its parameters. FindAcyl(...) tries to find a substructure,

C1′-C-C-C-C-(C=O)-R

by invoking the FindR(...), as specified above. In this manner, the search for a generic term is carried out as consecutive evaluations on generic terms, which should appear in parameter lists recursively. The evaluations may be in the form of dynamic function calls or dynamic bindings to specific substructures. The evaluation process terminates finally when an equivalent generic structure is found in the target structure.

5. CONCLUSION

The semantic dictionary (SD) implementation method and its application to substructure matching for generic structures are described in this paper. The SD content is a collection of terms with definitions of their structural meanings, which are actually structural entities and/or situations. SD itself is an independent and open-ended module and could be utilized in various application systems. At present, however, those terms registered in SD are assumed to appear in structural contexts (SCs) of transforms, which are matched with target structures.

The essential part of SD is the definition method for generic terms, implemented as context-sensitive procedures using the C++ programming language. The SD could be viewed as an object-oriented data base system, whose objects are chemical (sub)structures and their properties.[21]

Thus SD makes it possible to implement a flexible matching mechanism and the user interface familiar to chemists by defining structural meanings for generic terms. It resembles the communication among human beings using natural language. In addition, the similarity-based reasoning (analogical reasoning[22]) could be realized by extending SD. Actually, as is shown in the previous section, the matching mechanism for generic structural contexts suggests one aspect of the analogical reasoning, which adjusts the matching level for chemical structures between targets and structural contexts.

REFERENCES AND NOTES

(1) Motoda, H.; Mizoguchi, R.; Boose, J.; Gaines, B. Knowledge Acquisition for Knowledge-Based Systems. *IEEE Expert.* **1991**, *6*, 53–64.
(2) Lenat, D. B.; Guha, R. V. *Building Large Knowledge-Based Systems*; Addison-Wesley Publishing Company, Inc.: Reading, MA, 1990.
(3) Corey, E. J.; Long, A. K.; Rubenstein, S. D. Computer-Assisted Analysis in Organic Synthesis. *Science* **1985**, *228*, 408–418.
(4) Matsuura, I. Development of PC-SYNTREX, A CAD System for Organic Synthetic Route, Based on Reaction Database. *Proc. Symp. Inf. Chem. 13th* **1990**, 17–20 (in Japanese).
(5) Wilcox, C. S.; Levinson, R. A. A Self-Organized Knowledge Base for Recall, Design, and Discovery in Organic Chemistry. *ACS Symp. Ser.* **1986**, *306*, 209–230.
(6) Blurock, E. Computer-Aided Synthesis Design at RISC-Linz: Automatic Extraction and Use of Reaction Classes. *J. Chem. Inf. Comput. Sci.* **1990**, *30*, 505–510.
(7) Ai, C.-S.; Blower, P. E., Jr.; Ledwith, R. H. Extracting Reaction Information from Chemical Databases. In *Knowledge Discovery in Databases*; Piatetsky-Shapiro, G., Frawley, W. J., Eds.; AAAI Press: California, 1991; pp 367–381.
(8) Gordon, J. E.; Brockwell, J. C. Chemical Inference. 1. Formalization of the Language of Organic Chemistry: Generic Structural Formulas. *J. Chem. Inf. Comput. Sci.* **1983**, *23*, 117–134.
(9) Gordon, J. E. Chemical Inference. 2. Formalization of the Language of Organic Chemistry: Generic Systematic Nomenclature. *J. Chem. Inf. Comput. Sci.* **1984**, *24*, 81–92.
(10) Lynch, M. F.; Barnard, J. M.; Welford, S. M. Generic Structure Storage and Retrieval *J. Chem. Inf. Comput. Sci.* **1985**, *25*, 264–270.
(11) Tokizane, S.; Monjoh, T.; Chihara, H. Computer Storage and Retrieval of Generic Chemical Structures Using Structure Attributes. *J. Chem. Inf. Comput. Sci.* **1987**, *27*, 177–187.
(12) Lynch, M. F.; Barnard, J. M.; Welford, S. M. Computer Storage and Retrieval of Generic Chemical Structures in Patents. 1. Introduction and General Strategy. *J. Chem. Inf. Comput. Sci.* **1981**, *21*, 148–150.
(13) Barnard, J. M.; Lynch, M. F.; Welford, S. M. Computer Storage and Retrieval of Generic Chemical Structures in Patents. 2. GENSAL: A Formal Language for the Description of Generical Chemical Structures. *J. Chem. Inf. Comput. Sci.* **1981**, *21*, 151–161.
(14) Barnard, J. M.; Lynch, M. F.; Welford, S. M. Computer Storage and Retrieval of Generic Chemical Structures in Patents. 4. An Extended Connection Table Representation for Generic Chemical Structures. *J. Chem. Inf. Comput. Sci.* **1982**, *22*, 160–164.
(15) Lynch, M. F.; Gillet, V. J.; Downs, G. M.; Holliday, J. D.; Barnard, J. M. Computer Storage and Retrieval of Generic Chemical Structures in Patents. 11.Theoretical Aspects of the Use of Structural Language in a Retrieval System. *J. Chem. Inf. Comput. Sci.* **1991**, *31*, 233–253.
(16) Holliday, J. D.; Downs, G. M.; Gillet, V. J.; Lynch, M. F. Computer Storage and Retrieval of Generic Chemical Structures in Patents. 15. Generation of Topological Fragment Descriptors from Nontopological Representations of Generic Structure Components. *J. Chem. Inf. Comput. Sci.* **1993**, *33*, 369–377.
(17) Nakayama, T. Computer-Assisted Knowledge Acquisition System for Synthesis Planning. *J. Chem. Inf. Comput. Sci.* **1991**, *31*, 495–503.
(18) Pensak, D. A.; Corey, E. J. LHASA—Logic and Heuristics Applied to Synthetic Analysis. *ACS Symp. Ser.* **1977**, *61*, 1–32.
(19) Wipke, W. T.; Ouchi, G. I.; Krishnan, S. Simulation and Evaluation of Chemical Synthesis—SECS: An Application of Artificial Intelligence Techniques. *Artif. Intell.* **1978**, *11*, 173–193.
(20) Booch, G. *Object-Oriented Design with Applications*; The Benjamin/Cummings Publishing Company, Inc.: Redwood City, CA, 1991.
(21) Kim, W. *Introduction to Object-Oriented Databases*; The MIT Press: Cambridge, MA, 1990.
(22) Winston, P. H. Learning New Principles From Precedents and Exercises. *Artif. Intell.* **1982**, *19*, 321–350.