plementary material (Appendix A). This first program, INCOS1.FOR, extracts compound number, compound name, concentration, and flag. The second program, INCOS2.FOR, extracts date, station number, filename, level, matrix, dilution, and type. The third program, INCOS3.FOR, merges the data from the first two programs. And the fourth program, LOD.FOR, adds the limit of detection from an external table.

The last file created by this part of OARS is INCOS2.DB2. This file is fixed-format, delimited by double quotes. The file is ready for direct importing into Dbase III plus.

The second part of OARS is written in Dbase III plus[2] and is comprised of five separate Dbase programs given in the supplementary material (Appendix B).

The first Dbase program is a Dbase importation to get data into a file named INCOS. This program deletes compounds that have flag = "U", eliminating compounds looked for but not found. As can be seen in Figure 2, compounds that are detected in some samples but not in others are treated differently. This is essential to characterizing a waste site.

The second Dbase program adds results from the acid and base fractions. Many EPA methods for the analysis of semivolatile compounds in water samples call for sequential extraction of basic and acidic compounds. Some compounds may appear in both fractions. Hence the results are summed to give the concentration in the sample.

The third Dbase program merges samples that were analyzed at different dilutions; it can accommodate up to three different dilutions.

The fourth program averages the non-zero concentrations and calculates the percent relative difference or percent relative standard deviation. The analyst is then allowed to review the data on the screen and make corrections as necessary. Manual correction normally entails removing sample concentrations that were diluted below the instrument's limit of detection. Figure 4 shows an edit screen that is used at this stage.

The last program then produces a final report in the matrix format. An example of this final report is shown in Figure 5.

CONCLUSION

There is a difference between chemical data and chemical information. Data becomes information when the observer obtains knowledge. The NEIC OARS is a mechanism by which NEIC scientists can present chemical data to the nonscientist and the nonscientist acquires chemical information.

Supplementary Material Available: Four FORTRAN programs (Appendix A) and five Dbase programs (Appendix B) used in the OARS system (27 pages). Ordering information is given on any current masthead page.

REFERENCES AND NOTES

(1) Microsoft and Microsoft FORTRAN are registered trademarks of Microsoft Corp.
(2) Dbase and Dbase III plus are trademarks of Ashton-Tate.

Computer-Assisted Knowledge Acquisition System for Synthesis Planning

TAKASHI NAKAYAMA

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

Received January 3, 1991

A computer-assisted knowledge acquisition system for synthesis planning (KASP) is described and can be used by organic chemists to generate transforms interactively using a graphics user interface. KASP extracts transforms from a specific reaction database which contains about 30 000 records and some of its features are as follows: (1) It is a semiautomatic transform acquisition system. (2) It has a graphical and interactive language which will facilitate its use by chemists. (3) It has a generic-term dictionary that is helpful to describe chemical structures.

INTRODUCTION

Beginning with OCCS in 1967 by Corey,[1] a number of computer-based synthesis planning systems have been developed during the last two decades. They are divided into two categories, although several ideas have been proposed so far. One category is a logic-oriented approach, and the other is an empirical knowledge-oriented approach. For instance, EROS,[2] SYNGEN,[3], etc. belong to the former category, while LHASA,[4] SECS,[5] etc. belong to the latter one. Some of them are said to be in practical use at several companies. Knowledge-oriented systems in particular are said to be very prospective.[6] Specifically this type of system is based on a knowledge base consisting of empirical reaction rules called transforms. However, it seems that there are several problems impeding the development of a really practical system: First, the system must have enough processing speed; second, it must give results that are as precise as possible; and third, it must be able to process as many different targets as possible. The first problem is concerned with the combinatorial explosions that result from the opportunistic application of the rules (transforms). A second one is concerned with relevancy of the applied transforms. This suggests the importance of the quality of transforms and the selection mechanism. A third is concerned with the quantity of transforms. The second and third problems lead to the building of a knowledge base (transform base) with sufficient quality and quantity. In order to build a transform base, some ideas and practical systems have been proposed and implemented. Two of the practical systems are the well-known CHMTRN[4] and ALCHEM[5] which employ an English-like artificial language as a coding tool for transforms. Some favorable results have been reported concerning the utility of those languages.[7] In fact, however, it is not so easy to build a transform base on a practical scale, and this work is still continuing.[8-12]

In our laboratory, a computer-assisted synthesis planning system based on empirical knowledge (SPEK) is under development, using a reaction database which contains about 30 000 specific reaction data coded from *Organic Syntheses*

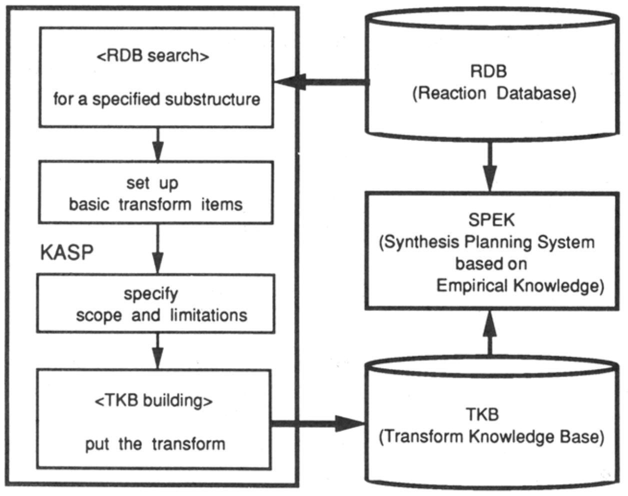


Figure 1. Relation between KASP and other components of computer-assisted synthesis planning system.

(Vols. 1–68) and *Shin Jikken Kagaku Kouza* (Vols. 14-I–14-V, Vols. 15-I–15-II; in Japanese).[13] The transform base in SPEK is acquired from this reaction database through the acquisition system KASP. Figure 1 shows the relation between KASP and other components of the system.

The importance of building a database and a knowledge base was recognized at an early stage[14] and is now one of the main themes in the field of knowledge engineering (called a knowledge acquisition problem). Knowledge acquisition usually requires enormous efforts from both computer scientists and experts in the field (organic chemists in this case). Interview systems with knowledge engineers, (semi-)automatic knowledge acquisition systems, and so on have been developed, but at present there are no satisfactory knowledge acquisition systems that are generally available for practical use. Therefore, a knowledge acquisition system that is specific to the domain of organic chemistry is needed.

The computer-assisted knowledge acquisition system for synthesis planning (KASP) reported in this paper was developed in our laboratory on a UNIX workstation. KASP generates transforms from reaction data mentioned above, cooperating with organic chemists. The main features of KASP are as follows:

(1) It is a semiautomatic transform acquisition system. A fixed part of the information from the reaction data is compiled automatically, and the result is edited interactively by chemists.

(2) It employs a graphical language which uses predicates and logical connectives, implemented as a GUI (graphical user interface).

(3) Description of the scope and limitations is a central task for KASP's users. Generic terms are defined in a term dictionary to help users specify scope and limitations. The use of generic terms and graphical operations are the key to the working environment.

REACTION DATABASE

The scheme of reaction data in the reaction database (abbreviated as RDB hereafter) which KASP manipulates is summarized in Figure 2a. Structures of the starting material and product are represented in the form of connection tables. Each atom of the structure is classified according to the degree of its relationship to the reaction. Most importantly, precise correspondence between atoms in the starting material and in the product is maintained by unique numbering. The full design of RDB was done by Matsuura, whose reaction database management system is called PC-SYNTREX.[15,16] Figure 2b is an example of reaction data which shows some atom

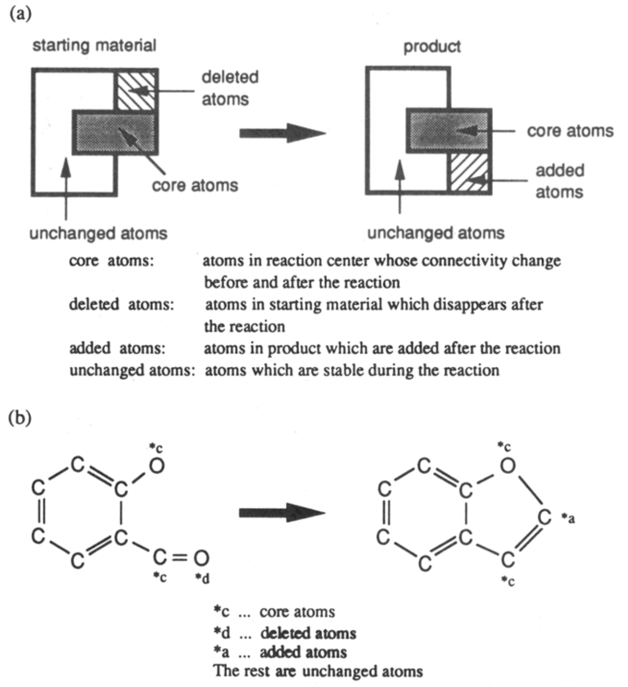

(a)
starting material
product
deleted atoms
core atoms
core atoms
added atoms
unchanged atoms
unchanged atoms
core atoms: atoms in reaction center whose connectivity change before and after the reaction
deleted atoms: atoms in starting material which disappears after the reaction
added atoms: atoms in product which are added after the reaction
unchanged atoms: atoms which are stable during the reaction
(b)
*c ... core atoms
*d ... deleted atoms
*a ... added atoms
The rest are unchanged atoms

Figure 2. (a) General representation of the scheme of reaction data in RDB. (b) Example of reaction data whose atoms are classified in four categories.

classes. (Atom classes are distinguished by color on a screen.) As a result, the core part of the structural transformation is extracted and set in the transform items by KASP automatically. (Building of the RDB in this way is still in progress.[13]) As a transform, however, this rigid form of structural transformation is not always sufficient, that is, the effects of peripheral substructures should be considered as scope and limitations. This is done with a graphical language in KASP.

REPRESENTATION OF TRANSFORMS

The contents that a transform represents are reaction information and its combinations extracted from the reaction database (RDB). Scope and limitations are typical examples of such combinations. A transform should accommodate as much such information as possible. The top level data structure of a transform includes the following items:

registry number
transform name
source reaction data
transform pattern
site structure
favorability
scope and limitations
conditions
editor's name

(1) **Registry number** is a unique sequential number given to a transform by KASP.

(2) **Transform name** is provided optionally by an editor.

(3) **Source reaction data** points to the list of RDB record identifiers that are the source records from which the transform is extracted. Editors can browse through relevant reaction examples from the RDB. The list itself is generated by KASP automatically at the first compilation of the RDB to make the fixed part of transform base. Browsing of RDB is also performed in the substructure search module of SPEK and is utilized in the editing session if necessary.

(4) **Transform pattern** represents a pattern of structural transformation corresponding to the reaction pattern. The following is the list of transform patterns for the present version.

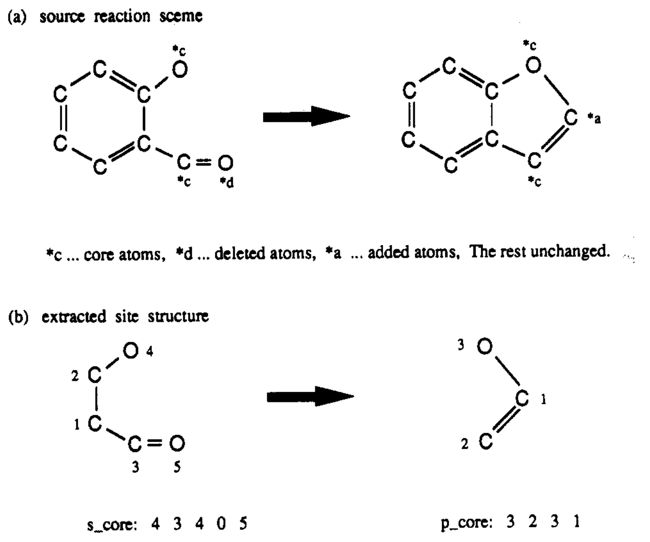

(a) source reaction sceme
*c ... core atoms, *d ... deleted atoms, *a ... added atoms, The rest unchanged.
(b) extracted site structure
s_core: 4 3 4 0 5
p_core: 3 2 3 1

Figure 3. Example of a site structure with its source reaction data.

connection of ring/chain
disconnection of ring/chain
rearrangement
interchange
transposition
removal
addition
inversion
other

These patterns are used to structure the transform base, which gives the basis of transform selection under some synthesis strategy.[17]

(5) **Site structure** represents the substructure of a reaction site, which is defined in KASP so that it contains both the core and its peripheral structure. Schematically

$$\text{site} = \text{core} + \text{peripheral}$$

The core structure consists of the reaction center and added (for products)/deleted (for starting materials) atoms. (See Figure 2.) The peripheral structure consists of atoms that have some effect on the reaction and are not necessarily adjacent to the core structure. Some default peripheral structures may be extracted automatically and set as part of an initial site structure: examples of this include a benzene/pyridine ring connected directly to the reaction center or atoms in the shortest path that connects two components when a core structure originally extracted at first is found disconnected. Though the peripheral structure is not part of the structure transformation, it should be considered as part of the area to be matched with the target. Four kinds of structural codes are set as transform items for a site structure: s_site, p_site, s_core, and p_core. Default structures are set for s_site and p_site for the first automatic extraction, but they could be modified later by editors. (Leading s_ and p_ means "starting material's" and "product's", respectively.) Further, s_core and p_core are the correspondence lists of atoms between core atoms in site structures, which enable structure transformation between products and starting materials (targets and precursors), that is, structure generation. This is necessary because code generation of s_site and p_site gives an independent numbering of atoms to each site structure. The method of structure generation in SPEK is virtually equal to the one that employs a so-called reaction matrix.[2] In SPEK, the equivalent of a reaction matrix is generated dynamically from the correspondence list of core atoms mentioned above when the structure transformation is performed. An example of a site structure is shown in Figure 3.

(6) **Favorability** is a standard value for rating the applicability of a transform, which corresponds to a certainty factor in the field of expert systems. This is initially set to the smallest yield value in the set of corresponding source reaction data from which the transform is extracted, and is increased or decreased according to structural contexts and reaction conditions. The initial value could also be modified later by editors.

(7) **Scope and limitations** express the scope and limitations of the transform just like that of reaction data. The detail of scope and limitations in KASP is explained later. It is implemented as a mechanism that changes favorability according to the variation of site structures (called structural contexts; that is, scope and limitations in KASP are defined from the viewpoint of the target structure itself). In other words, scope and limitations give a basis for control of the transform application to the target.

(8) **Conditions** are interpreted as environmental contexts of the transform-like structural contexts. If a transform is derived from several reaction data, reaction conditions in those source data give a variety of environmental contexts, and the favorability is increased or decreased correspondingly.

(9) **Editor's name** indicates who specified the scope and limitations of the transform, by which we can trace back to the knowledge source with respect to the expert who edited the transform.

The operation for structure transformation is not included in this representation of a transform except for correspondence lists of atoms in core structures between starting material and product described in the site structure. In some systems, this function is explicitly described with a name like "manipulation", but in KASP the operation list is generated dynamically when the transform is actually applied. Transform base is used both in the forward and backward directions by this implementation.

GRAPHICAL LANGUAGE

The most characteristic aspect of KASP as a graphical language appears when scope and limitations are specified in the interactive session, so we show the specification of scope and limitations in the following manner. Informally, scope and limitations are expressed in the form of **if-then** type rule: **if** structural context is ... **then** increase/decrease favorability by ...

In KASP, we call the **if** part a **structural context**. This if-then form of expression was employed straightforwardly in CHMTRN[4] and ALCHEM.[5] Those artificial languages are said to be complete enough to express any structural contexts. But from the viewpoint of the user interface, those "English-like" languages are not so easy for end users. It seems indispensable to give a more facile and prompting user interface to encourage popular use. Here, "prompting" means that the system might prompt the editor to get ideas of scope and limitations during the editing session. It is natural to consider that structural contexts are understood and processed by human chemists in the form of a graphical image, and therefore, it is desirable to implement the editing system so that it can manipulate structural contexts directly in terms of graphical operations instead of substituting them by language expressions. KASP is intended to provide such a tool.

(1) **Predicate menu**: Structural contexts are expressed in terms of predicates in KASP. A predicate menu is prepared, and a menu icon is assigned to each predicate. The list of predicates is shown in Table I with syntax and arguments. Syntax and operations of the predicate menu are as follows:

Arguments *X*, *Y*: Atoms, functional groups, other substructures, and their combinations are assigned to these arguments. If a prompt indicates a brace "{" or parenthesis "(",

Table I. Predicate Menu and Its Syntax

menu	syntax	X	Y
α_to	X is α to Y	{...}[a]	in site atom
β_to	X is β to Y	{...}	in site atom
γ_to	X is γ to Y	{...}	in site atom
adjacent	X is adjacent to Y	{...}	in site atom
attach	X is attached to Y	{...}	in site atom
distant	X is D distant from Y	{...}	in site atom
is_a	X is Y	in site atom	{...}
part	X is part of Y	(...)[b]	{...}
present	X is present ⟨...⟩[c]	{...}	
and	af1[a] &&		af2[d] ...
or	af1[b] ||		af2[d] ...
(not)	double click of mouse button[e]		

[a] {...} means that a set of values is required for X or Y whose members are interpreted disjunctively to be assigned to the arguments. (A single value is also allowed.) [b] (...) means in-site atoms whose members are interpreted conjunctively to be assigned to the arguments. (A single value is also allowed.) [c] ⟨...⟩ is specified in a submenu of menu item "present" as "in site", "elsewhere", or "anywhere". [d] af1, af2, ... means atomic formulas that represent a predicate expression (e.g., X is adjacent to Y). [e] If a predicate menu doubly clicked, it is interpreted that the predicate was negated (e.g., not part, not present, etc.).

like "x = {" or "x = (", a set value is requested, otherwise, i.e., for prompt X = or Y =, a single value is requested to be entered. (A single member is, of course, allowed for a set value.) The members in braces are assigned to the argument disjunctively, whereas the members in parenthesis are assigned conjunctively.

Input for X, Y: Five ways are prepared for input atoms and substructures according to their type.

Atom Number. An atom in a site structure can be specified by a unique identification number (called "atom number" hereafter). A substructure in a site structure is thus identified with a set of atom numbers. A set of constituent atoms is enough to identify the substructure because the matching mechanism between the site structure and corresponding substructure in the target has already been completed at this stage. The connected subgraph consisted of the set of specified atoms is regarded as the substructure.

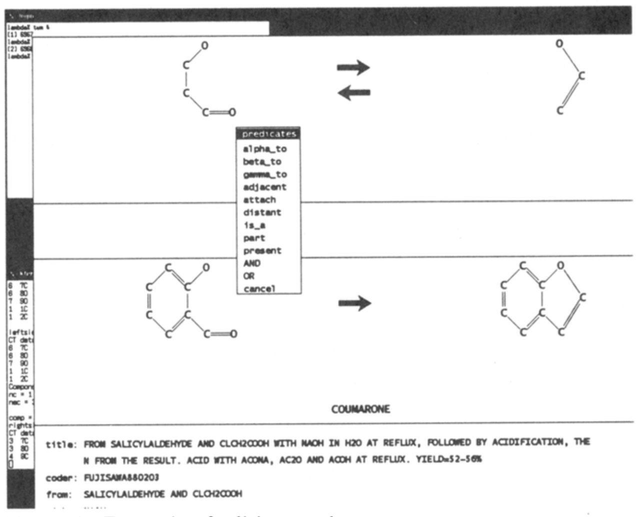
Visual Object. An atom in a site structure is also identified by pointing at the object, that is, the atomic symbol on the screen, using a pointing device (a mouse). A substructure is identified in a site structure in the same way. This is the default operation to specify atoms and substructures on the screen.

Name. A substructure which has a trivial name is specified by the name (a character string), which starts with a lower case letter. This name is supposed to be registered in the term dictionary described below.

Chemical Formula. A chemical formula that is composed only of atomic symbols and their multipliers (which means it does not contain bond symbols) can be specified as a name of the corresponding substructures (e.g., COOR, CN, etc.).

Structure Diagram. A substructure can be input in the form of structure diagram by using a structure input module. Substructures that have no trivial names are specified by structural diagram (e.g., CH═CHW, C═C—, etc. are specified in this manner).

Logical connectives: A predicate like β_to, is_a, etc. selected from this predicate menu gives only an atomic formula that corresponds to a single sentence. Compound sentences, that is, compound structural contexts, are made of these atomic formulas using logical connectives **and**, **or**, and **not**. **not** is used just to negate an atomic formula. (In KASP, **not** is applied to the predicate if the double click of a mouse button were detected at the time of menu selection.) A compound sentence is generated in the order that the constituent predicates are selected: After the arguments of a predicate are specified, if a logical connective **and** or **or** is selected rather than the context input completion indicated, the predicate just specified is supposed to be connected with the following predicate by the logical connective (e.g., "X is part of $Y1$." **and** "X is not part of $Y2$.", etc.).



title: FROM SALICYLALDEHYDE AND CLCH2COOH WITH NAOH IN H2O AT REFLUX, FOLLOWED BY ACIDIFICATION, THE N FROM THE RESULT. ACID WITH ACONA, AC2O AND ACOH AT REFLUX. YIELD=52-56%
coder: FUJISAWA880203
from: SALICYLALDEHYDE AND CLCH2COOH

Figure 4. Example of editing session.

For instance, if the menu predicate **α_to** is selected, then a sentence "X is α to Y." is displayed as a syntactical meaning of the menu. Further, the prompt "X = {" is indicated to input atoms/fragments, and finally, the prompt "Y =" is indicated to input an atom in site structure which is α to one of the members in X. They are specified either by pointing at the visual object (using a mouse: this is the default for this predicate) or with an identification number (using the keyboard). **β_to** and **γ_to** are treated likewise. The term **adjacent** means "X is adjacent (connected directly) to Y", and arguments X and Y are forced to be filled in just the same way as described above. The term **attach** is equivalent to adjacent in meaning, but is prepared to accommodate conventional terms used in restricted manner, like "attached hydrogen", etc. The term **distant** has a rather general meaning, where distance (D) between X and Y should be specified as range rather than a fixed length. (The lower and upper limits are specified.) The phrase **is_a** expresses assignment/substitution of X by Y, which means "X is Y," and **part** is used to express the idea that X is part of Y, where X is a set of atoms in a site structure. Further, **present** specifies that some atoms/groups/substructures exist somewhere in the target structure. The area where X is present is restricted to one within the site, outside the site, or anywhere in the structure. This discrimination of area is specified through the submenu of predicate **present** and is set in scope and limitations.

An example of an editing session is shown in Figure 4, where a core transform is displayed with its source reaction data and summary data (reaction conditions) described in the RDB. It is a stage where the predicate menu is displayed by pressing a mouse button to specify a structural context. The window just under the one for the core transform is used for key input for detailed specification of the structural context.

(2) Data structure: A data structure for scope and limitations is shown schematically in Figure 5a. As mentioned earlier, scope and limitations are expressed in the form of the **if-then** rule, and the data structure is implemented so that it represents the rule format straightforwardly. The form of the **if-then** rule is represented hierarchically in three levels (in four levels beginning with a transform level). The first level gives an entry for each rule, while an increment/decrement value is also set in the entry. That is, the **then** part of the rule is implemented in the first level. The second level, which represents the **if** part of the rule, is accessed through "contextp" from the first level. The contents of the second level are as

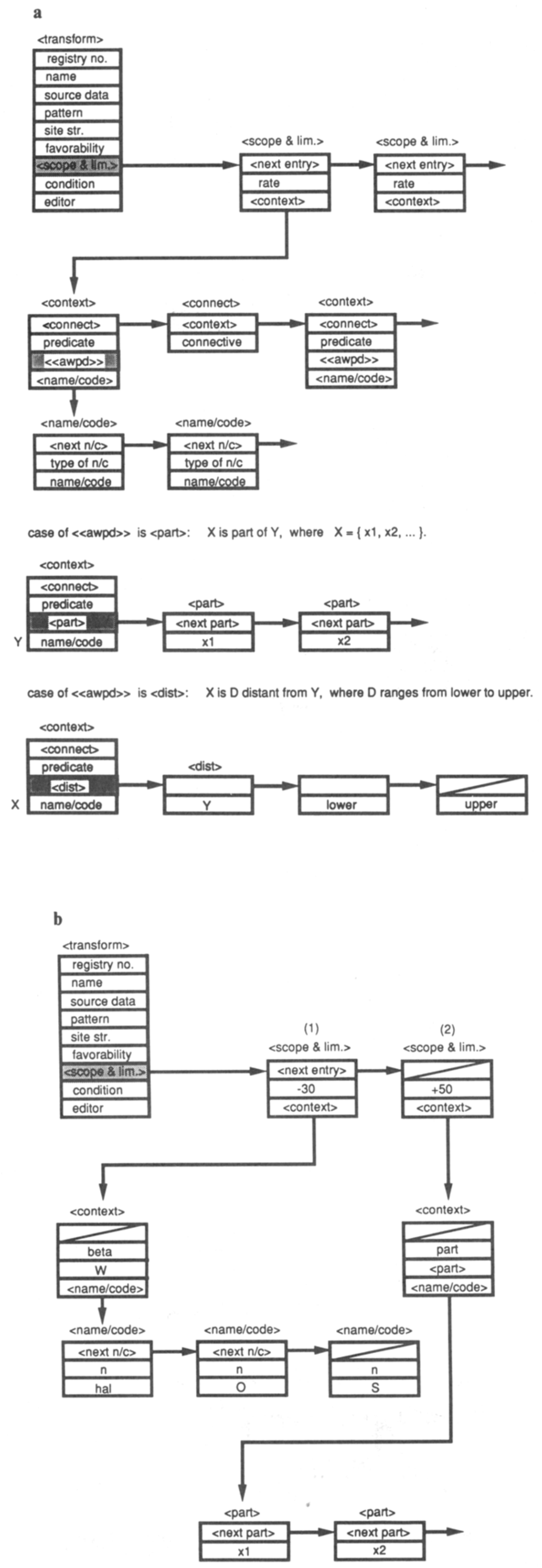
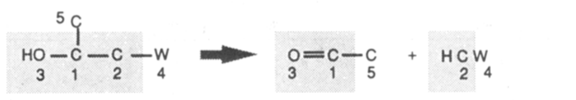


Figure 5. (a) Schematic representation of data structure for scope and limitations. (b) Sample data structure that represents the following two statements: (1) If hal, O, or S is β to *W*, then decrease favorability by 30. (2) If C1 and C2 are in 5- and 6-membered ring, then increase favorability by 50.



```
RATING DECREASED BY:
  (1) Hal, O or S β to W;
  (2) W = CONHR, CN, COOR;
  (3) C=C- on C1;
  (4) Alk at C2;
  (5) W = CH=CHW;
  (6) presence acid- or base-sensitive groups elsewhere;
  (7) C1-C2 in ring whose ring size is other than
      5 or 6.

RATING INCREASED BY:
  (8) additional W at C2;
  (9) W = NO2;
 (10) no hydrogen on C5;
 (11) C1-C2 in ring whose ring size is 5 or 6.
```

Figure 6. Example of scope and limitations from Aldol condensation.[18]

follows: The first item "connectp" is used as a pointer to logical connectives (and/or) when the **if** part is a compound sentence. The "predicate" represents the type of the predicate used in the rule. The "atom_no"/"where"/"partp"/"distp" is a shared part which is used in four ways according to the type of the predicate. Each member of the shared part has the following meaning: (i) **atom_no** is an identifier of an atom which corresponds to argument *X* or *Y* in Table I. (ii) **where** is an integer corresponding to the argument *X* of the predicate **present**, which represents where the fragment specified in argument *Y* exists. (0 = within the site; 1 = elsewhere outside the site; 2 = anywhere.) (iii) **partp** is a pointer to a list of constituent atoms (argument *X*) that should be part of the parent structure specified in argument *Y*. (iv) **distp** is a pointer to a list which is made of three cells: the first one is for argument *Y* (in-site atom); the second and third one are for the lower and upper limit, respectively, of the distance between argument *X* and *Y*.

The third level of the data structure makes part of the second level, which gives the contents of arguments *X* and *Y* in the structural contexts. They are a list of constituent atoms in part structure (**struct part *partp**); a list for distance (**struct dist *distp**); and a list of name/code of fragments specified in argument *X* or *Y*.

A sample diagram to explain the total data structure for structural contexts is shown in Figure 5b.

(3) Example: One of the main features of KASP is that it handles generic terms registered in the term dictionary, which is explained in the following section. To show the graphical, interactive, and flexible facilities of KASP, we borrow an example from Corey[18] in Figure 6, where some generic terms are employed to describe structural contexts. The sentences from 1 to 11 describe structural contexts that lead to increase/decrease favorability of the transform. The problem is how to specify these structural contexts to the system. The procedure of specifying structural contexts using KASP is as follows:

s1. Set up site structures: A set of reaction data of Aldol condensation in RDB can be accessed through the list of source reaction data in the core transform. Here we can start with this core transform to be edited. Atoms in the hatched area in Figure 6 are indicated as the core structure, where C_1, C_2, and O_3 are the constituent atoms, whereas atoms C_5 and W are not. However, C_5 and W could be included in the site structure if the editor judges this to be better. So, here, we include C_5 and W in the site structure for demonstration.

s2. Select predicate menu and describe structural context: Predicates to describe original expressions are listed in Figure 7 with corresponding regulated sentences and editing session.

original expression	regulated sentence	predicate	editing session

Hal, O or S β to W.	hal, O or S is β to w.	β_to	X is β to Y.
			X = {hal, O, S}
			Y = w

W = CONHR, CN or COOR.	w is CONHR, CN or COOR.	is_a	X is Y.
			X = w
			Y = {CONHR, CN, COOR}

C=C- on C1.	C=C- is adjacent to C1.	adjacent	X is adjacent to Y.
			X = {<str. input>} *
			Y = C1

Alk at C2.	alk is adjacent to C2.	adjacent	X is adjacent to Y.
			X = {alk}
			Y = C2

W = CH=CHW.	w is CH=CHw.	is_a	X is Y.
			X = w.
			Y = {<str.input>} *

presence acid- or base-sensitive group elsewhere.	acid- or base-sensitive group is present elsewhere.	present	X is present <elsewhere>. X = {asg, bsg}

C1-C2 in ring whose ring size is other than 5 or 6.	C1 and C2 are part of a ring
	
	and
	not part of 5_ring nor 6_ring.
		part
		
		and
		^part
			X is part of Y.
			X = {C1, C2}
			Y = {ring}
			
			X is not part of Y.
			X = {C1, C2}
			Y = {5_ring, 6_ring}

additional W at C2.	w is adjacent to C2.	adjacent	X is adjacent to Y.
			X = {w}
			Y = C2

W = NO2	w is NO2.	is_a	X is Y.
			X = w
			Y = {NO2}

no hydrogen on C5.	H is not attached to C5.	^attach	X is not attached to Y.
			X = {H}
			Y = C5.

C1-C2 in ring whose ring size is 5 or 6.	C1 and C2 are part of 5_ring or 6_ring.	part	X is part of Y.
			X = {C1, C2}
			Y = {5_ring, 6_ring}

(*) <str.input> means that the substructure is input using structure input module rather than character string.

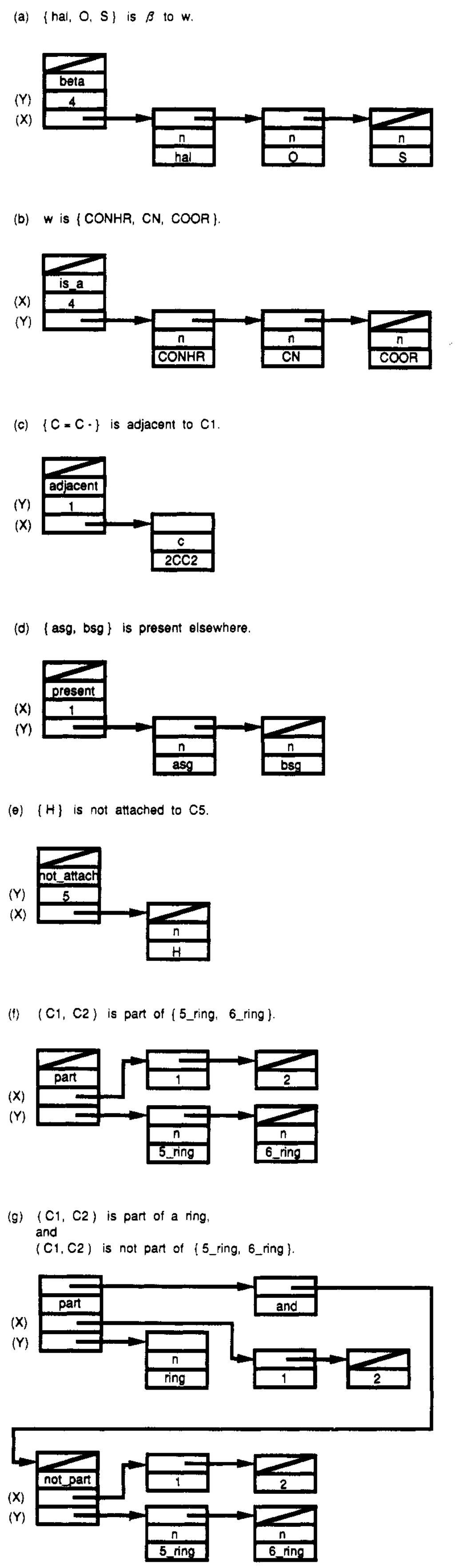
Figure 7. Predicate menu and editing session for structural contexts in Figure 6.

For instance, the original expression "Hal, O, or S β to W." is interpreted as "hal, O, or S is β to w.", and predicate **β_to** is selected. This selection is done by the editor's judgement, that is, the editor points out the predicate **β_to** in the predicate menu by using a mouse. Then, KASP replies with syntax "*X* is β to *Y*", and asks for input to the arguments *X* and *Y* as explained in the previous section. In this example, *X* = {hal, O, S) and *Y* = w is the answer to the prompts (underlined strings), where actually w is pointed out by using a mouse on the screen. This sentence for the structural context is fully described using only one predicate (**β_to**). Other sentences are interpreted and treated likewise.

Figure 7 shows the summary of such editing sessions corresponding to those structural contexts in which the original expressions in Figure 6 are regulated to sentences which use predicates provided in the predicate menu; predicates selected by an editor and used in those regulated sentences are listed in the third column; editing sessions to specify structural contexts are shown in the fourth column in Figure 7. For instance, the original expression: "C_1-C_2 in ring whose ring size is other than 5 or 6" is converted to a regulated compound sentence: "C_1 and C_2 are part of a ring" and "not part of 5-ring nor 6-ring", which is expressed in terms of conjunction of two single sentences as shown in the editing session in Figure 7.

s3. Generate data structures for the structural context: Figure 8 shows data structures that are generated for the specification in Figure 7. **beta, is_a, adjacent**, etc. are function names for predicate **β_to, is_a, adjacent**, etc. and evaluated when called by matching mechanism with arguments *X* and *Y*.

The matching mechanism here is implemented as a function



(a) {hal, O, S} is β to w.
beta
4
(Y)
(X)
n
hal
n
O
n
S
(b) w is {CONHR, CN, COOR}.
is_a
4
(X)
(Y)
n
CONHR
n
CN
n
COOR
(c) {C=C-} is adjacent to C1.
adjacent
1
(Y)
(X)
c
2CC2
(d) {asg, bsg} is present elsewhere.
present
1
(X)
(Y)
n
asg
n
bsg
(e) {H} is not attached to C5.
not_attach
5
(Y)
(X)
n
H
(f) {C1, C2} is part of {5_ring, 6_ring}.
part
(X)
(Y)
1
2
n
5_ring
n
6_ring
(g) {C1, C2} is part of a ring, and {C1, C2} is not part of {5_ring, 6_ring}.
part
(X)
(Y)
and
n
ring
1
2
not_part
(X)
(Y)
1
2
n
5_ring
n
6_ring

Figure 8. Examples of data structures for the structural contexts in Figure 7.

for each predicate. For instance, for a sentence (predicate) which states that "*X* is β to *Y*", a function *beta* is invoked with arguments *X* and *Y*, and the topological relation "*X* is β to *Y*" is checked atom-by-atom for the target structure. Likewise, a specific matching function is dispatched according to each structural context. Other matching mechanisms are summarized in the next section.

TERM DICTIONARY

The term dictionary supports nominal expressions in editing sessions to identify specific and generic chemical (sub) structures. As is shown in structural context description in the previous section, generic terms (like hal, w, R, ring, 5-ring, 6-ring, acid-sensitive-group, etc.) can be used in the specification of structural contexts as well as specific terms. This facility is supported by the term dictionary. That is, generic terms or expressions that are registered in the term dictionary can be used as descriptors of structural contexts. The use of generic terms gives great flexibility to the description of structural contexts, though, at the moment, they are restricted to symbols or names that are represented in the form of character strings. For instance, the editor (a chemist) can concentrate on his work of structural contexts using familiar terms commonly used by organic chemists. Further structural contexts cannot always be described precisely in terms of specific chemical structures. Some of them are forced to be described by qualitative terms, such as "rings", "bulky", "acid sensitive", etc. These qualitative descriptions can be done with the term dictionary, which allows various definitions of terms. Correspondingly, the matching mechanism for target structures is provided for each definition, which realizes substructure matching at various specificity levels of structural description. Any terms that represent a group of (sub)structures could be defined in the dictionary by giving appropriate names. The matching mechanism here consists in part of the matching mechanism mentioned in the previous section.

(1) Registered Objects and Notations. Items that are registered in the term dictionary are groups of atoms (e.g., hal = {F, Cl, Br, I}), functional groups, and other substructures. The notations of these items are names, chemical formulas (alphanumeric string), and linear structural diagrams with bond symbols (e.g., C=C—), respectively. Names are represented by a string of lower case letters (consequently, "hal" is employed instead of "Hal", and spaces must not be used). A chemical formula here is a string of upper case letters (except for the second letter of atomic symbols). A linear structural diagram is a string that includes names, atomic symbols, generic symbols, and other bond symbols. This string is interpreted as an entry name for the term dictionary.

(2) Definitions. There are four types of definitions for entry names:

Definition by code: If a name is registered with its structural data (a connection table, specifically), that name is defined by a code (Morgan code derived from the connection table). For instance, the name "cyano" may be defined by structural data —C≡N (which is actually entered by using a graphical input module). Further a string "CN" may be defined by —C≡N, and so "cyano" and "CN" become synonyms.

Definition by procedure. Whereas names representing specific structures are defined by codes, generic terms such as "ring" or "bulky_substituent" are difficult to define as specific code strings, though the terms represent some structural characteristics.

This type of term representing some general connectivity relationship is defined by binding a procedure (a function name, specifically) with the entry name. For instance, "ring" is defined by a function which finds the smallest ring containing specified atoms, and "bulky_substituent" is defined by a function which finds a bulky substituent at a specified position.

Definition by property: Generic terms that have a common structural feature may be defined by a procedure which tries to find that feature in the target. However, with respect to generic terms that have no common structural feature to find, we cannot apply procedures. Even in that case, if the generic term represents some common property (reactivity), we may give a property value list for it. We may use the property value list as a definition of the term, although a unique property value list is not always assigned to the generic term. It is supposed that the names of generic substructures of this kind are defined by the editor, that is, they are user-defined names, because usually they do not have names of their own. For instance, "bulky_erg" could be defined as "erg" (electron releasing group) that has a large volume like $C(CH_3)_3$, and its property list is represented as a tuple of two values: "erg" and "bulky".

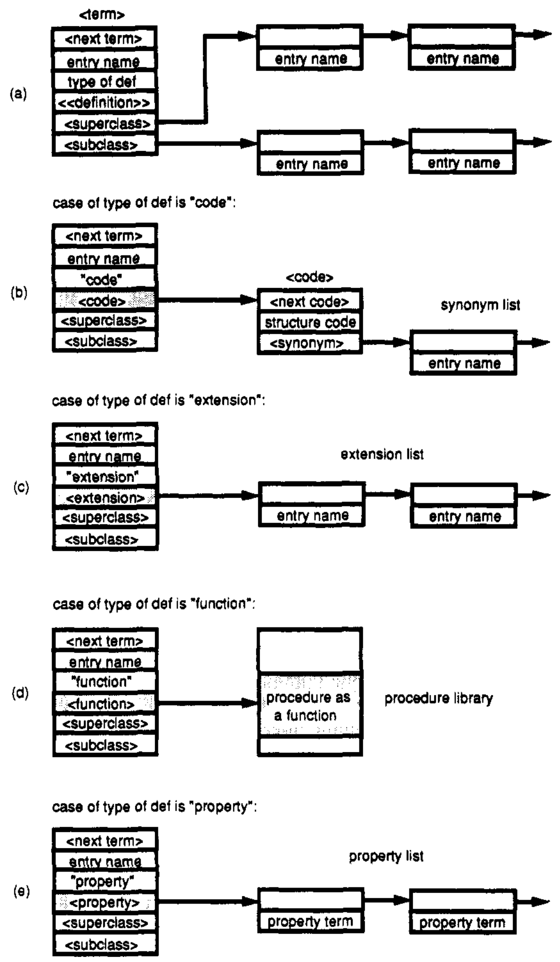


Figure 9. Schematic representation of data structure for terms dictionary.

Definition by extensions: If a generic term implies a class of instances or subclasses (members), it can be interpreted as a representative of the class. Inversely, each member of the class can be seen part of the contents of the term. Extensions could play a role of definition of generic terms. Extensions are often given duplicately with other definitions. For instance, "electron-withdrawing group" is a class of members: "cyano", "nitro", "carbonyl", etc., those members together define the term as its extensions.

(3) Organization of Term Dictionary. The data structure for each dictionary term is shown schematically in Figure 9. The body of structure ⟨**term**⟩ consists of five members except for a pointer to the next entry:

Name represents an entry name/formula that is used in structural contexts.

Superclass is a name of the super class of the term and is also registered in the term dictionary. For instance, "electron_withdrawing_group" is a superclass of "cyano".

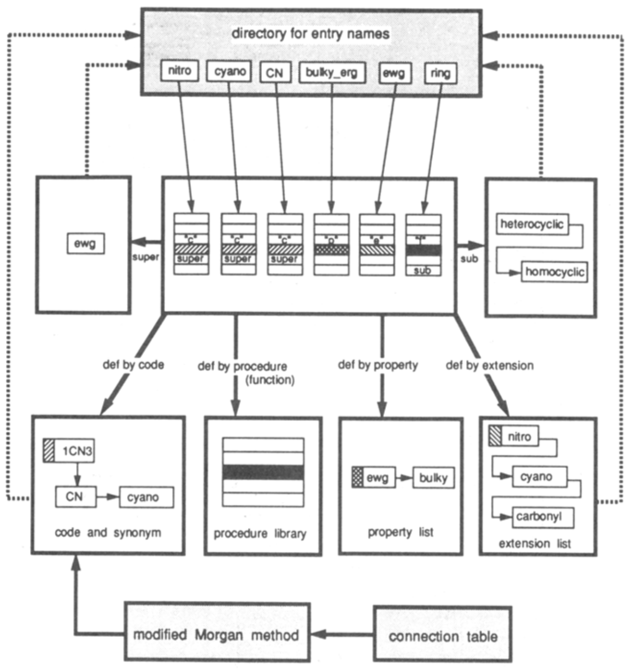


Figure 10. Conceptual diagram of term dictionary organization. A dashed line to the directory block indicates the reference to terms from the definition blocks.

Each term may have multiple superclasses, and **superclass** is actually a pointer to a list of superclass names.

Type of definition indicates the type of definition of the term: c = definition by code; e = definition by extension; *f* = definition by procedure(function); p = definition by property. This is necessary because the definition entry described below is shared by pointers for four types of definitions.

Definition is a pointer to a code entry, a function name, a property list, or an extension (instance) list according to the definition type. A modified Morgan code is generated for specific substructures that is an entity of some term. That is, the term is defined by its structural code. When a term of this type is registered to the dictionary, synonyms could also be set at the same time or later as an attribute of the code. Synonyms, in turn, are entry names of the dictionary. A procedure (function) name is set for the term defined by procedure, which is located in the procedure library. An extension list means a set of specific terms, which is interpreted disjunctively, whereas a property list means a set of generic terms, which is interpreted conjunctively.

Subclass is a name of the subclass of the term and is also registered in the term dictionary. For instance, "primary amine", "secondary amine", and "tertiary amine" are subclasses of "amine". Subclass is actually the pointer to a list of subclass names. A conceptual diagram of the term dictionary is given in Figure 10. The term dictionary has a directory for entry names. The body of term representation is below the directory indicated by an arrow. There is another type of directory for code and synonym entries to access from the code representation of substructures, which is shown in the lower part of Figure 10.

An entry name "nitro" in Figure 10 has the superclass "ewg", which means "electron-withdrawing group", and the entry name "ewg", in turn, is defined by an extension which has three instances: nitro, cyano, and carbonyl. The entry name "cyano", and "CN" are defined by the same code (1CN3), whose attributes are the synonymous terms: "CN" and "cyano", so "CN" and "cyano" can be used indiscriminately. An entry name "ring" is defined by procedure and has a pointer to a ring-finding function that is stored in the procedure library. Finally, "amine" has a subclass list of "primary amine", "secondary amine", and "tertiary amine", which should also be registered in the dictionary.

(4) Matching Mechanism. Terms used in the context structures have to be matched with substructures in a target structure when the transform's applicability is checked. Several matching mechanisms are provided for KASP to support generic expressions of terms, which correspond to the definition of terms.

Code matching: A term of a specific substructure is matched with its code, in addition to matching with the term itself. The use of synonyms is enabled by this method.

Generic matching: Generic matching here means the matching between terms registered in the dictionary: specific terms, generic terms, or whatever. For instance, a substructure —C≡N extracted from a target structure is matched with "ewg" in a structural context through the following steps (see Figure 10): (1) conversion to a modified Morgan code from a connection table; (2) access to its code and synonym entry from code directory; (3) find an entry name (a term) of the code in the synonym list; (4) find "ewg" in the superclass list of the term. If "ewg" were not found in its immediate superclass, the search would be continued to the upper limit if possible.

Procedural matching: A term defined by a procedure is matched by applying the procedure (calling an appropriate function) to the target structure. In most cases, the procedure tries to find a substructure that satisfies the constraints given by the corresponding structural context. For instance, a ring-finding function is called for the structural context: "C_1 and C_2 are part of a ring" and tries to find the smallest ring which contains two atoms corresponding to C_1 and C_2.

Property matching: A property list is a conjunction of generic terms, so property matching is a conjunction of the results of those member-term matching. For instance, the matching of property list: "erg" and "bulky" is performed with respect to "erg" and "bulky", which are performed through generic matching and procedural matching, respectively.

CONCLUSION

As long as computer-assisted synthesis planning systems are developed on the basis of a knowledge (transform) database, it is necessary to generate transforms of good quality and in sufficient quantity. This was recognized at the early stage of the development of this kind of system. Although the work of building a transforms database has been tried at many places, it cannot be claimed that the work has been fulfilled successfully. The reasons why we tried this difficult problem are as follows: first, we are able to utilize a reactions database which contains sufficient information for extracting reaction centers and related substructures from each datum; second, the rapid growth of computer power and environments like a large capacity of disks, high-resolution bit map displays, and pointing devices that are available for personal use. The approach to develop KASP is as follows:

(1) It must be easy for chemists to use. So it employs a graphical and interactive language that allows the use of generic terms to describe structural contexts.

(2) A quantity of transforms should be secured at the first step. The refinement and reorganization of the transform set is the next important step, which must be related to the strategic knowledge base that could control the application of transforms.

(3) It must be designed to evolve to an advanced knowledge processing system. It means that the synthesis planning system has to be improved so that it can manipulate a wide variety of knowledge.

(4) KASP's language is a set of predicates. New predicates should be added at any time so that any structural contexts

can be described by the language.

So many interesting but difficult problems have to be tackled and solved to achieve the goal of a synthesis planning system.[19] Although KASP has been developed for a specific reaction database (RDB), the system's implemented facilities such as generic term manipulation and its matching mechanism should be useful for other similar systems.

ACKNOWLEDGMENT

This research is based on a reaction database that is a result of the working group for a reaction database system in Molecular Design Society in Japan, and I am grateful to Prof. Fujiwara of the University of Tsukuba, Dr. Matsuura of the Chuugai Corporation Company, and other members of the society.

REFERENCES AND NOTES

(1) Corey, E. J.; Wipke, W. T. Computer-assisted design of complex organic syntheses. *Science* **1969**, *166*, 178-192.
(2) Gasteiger, J. Computer-assisted synthesis design. *Chim. Ind. (Milan)* **1982**, *64*, 714-721.
(3) Hendrickson, J. B.; Braun-Keller, E.; Toczko, G. A. A Logic for Synthesis Design. *Tetrahedron* **1981**, *37* (Supplement 1), 359-370.
(4) Pensak, D. A.; Corey, E. J. LHASA-Logic and Heuristics Applied to Synthetic Analysis. *Computer-Assisted Organic Synthesis*; ACS Symposium Series 61; American Chemical Society: Washington, DC, 1978; pp 1-32.
(5) Wipke, W. T.; Ouchi, G. I.; Krishnan, S. Simulation and Evaluation of Chemical Synthesis-SECS. *Artif. Intell.* **1978**, *11*, 173-193.
(6) Lee, T. V. Expert Systems in Synthesis Planning: A User's View of the LHASA Program. *Chemom. Intell. Lab. Syst.* **1987**, *2*, 259-272.
(7) Gund, P.; Grabowski, E. J. J.; Hoff, D. R.; Smith, G. M. Computer-Assisted Synthetic Analysis at Merck. *J. Chem. Inf. Comput. Sci.* **1980**, *20*, 88-93.
(8) Braun, H. W. Private communication.
(9) Yanaka, M.; Tomonaga, A. Organic Synthesis Design Program (SECS) and Its Knowledge Base Building. *The 8th Symposium on Chemical Information*, Kanazawa, Japan; Chemical Society of Japan: Tokyo, 1985; pp 138-141.
(10) Yanaka, M.; Kurumisawa, A.; Nakamura, K.; Wipke, W. T. Automatic Knowledge Base Building for the Organic Synthesis Design Program (SECS). *The 11th Symposium on Chemical Information*, Kyoto, Japan; Chemical Society of Japan: Tokyo, 1988; pp 72-75.
(11) Yanaka, M. An Application of SECS: Automatic Building of Knowledge Base. *Mol. Des.* **1989**, *11*, 2-57.
(12) Yanaka, M.; Nakamura, K.; Kurumisawa, A.; Wipke, W. T. Automatic Knowledge Base Building for the Organic Synthesis Design Program (SECS). *QSAR: Quantitative Structure-Activity Relationships in Drug Design*; Fauchere, J. L., Ed.; Alan R. Liss: New York, 1989; pp 147-150.
(13) The work of building RDB has been performed by the working group for a reaction database system that is an informal organization consisting of 16 company and university members of Molecular Design Society in Japan.
(14) Bersohn, M.; Mackay, K. Step toward the Automatic Compilation of Synthetic Organic Reactions. *J. Chem. Inf. Comput. Sci.* **1979**, *19*, 137-141.
(15) Matsuura, I. Computer-Assisted Reaction Design System Based on Reaction Database. *Mol. Des.* **1986**, *7*, 2-31.
(16) Matsuura, I. Development of PC-SYNTREX, A CAD System for Organic Synthetic Route Based on Reaction Database. *The 13th Symposium on Chemical Information*, Toyohashi, Japan; Chemical Society of Japan: Tokyo, 1990; pp 17-20.
(17) Structuring the transform base is not discussed in this paper.
(18) Corey, E. J. Computer-Assisted Analysis of Complex Synthetic Problems. *Chem. Soc. Rev.* **1971**, *25*, 455-482.
(19) Corey, E. J.; Long, A. K.; Rubinstein, S. D. Computer-Assisted Analysis in Organic Synthesis. *Science* **1985**, *228*, 408-418.

General Formulas for the Wiener Index

ISTVÁN LUKOVITS

Central Research Institute for Chemistry, Hungarian Academy of Sciences, P.O. Box 17, H-1525 Budapest, Hungary

Received February 8, 1991

The Wiener Index *W* of a graph is most often calculated by numerical methods. In this paper a method is proposed to deduce analytical formulas for *W* and for the connectedness index. It is shown that expressions for *W* can not include fourth or higher order terms in the number of vertices of the constituent strings. Formulas have been derived for graphs containing one, two, and three strings, and the general rules based on these formulas are discussed. Several numerical examples are given.

INTRODUCTION

The Wiener Index *W* is the sum of topological distances in the graph representing the structural formula of a molecule:[1]

$$W = \sum_{i<j} d_{ij} \qquad (1)$$

where d_{ij} denotes the topological distance between atoms *i* and *j* ($i, j = 1, 2, ..., N$; *N* denotes the number of atoms, *H* atoms are neglected). The distances between all $N(N-1)/2$ pairs of atoms must be added. It has been found that in a series of alkane isomers *W* accounts for the compactness of the molecules,[1,2] the smaller the value of *W* the more compact the molecule.

W has been used to explain the variation in the physical and chemical properties of alkanes.[3-8] It has also been used to correlate the pharmacological properties of compounds with their structure.[9-11] Molecular branching[12] and molecular cyclicity[13,14] are topological characteristics that are properly accounted for by the Wiener Index. Congruence relationships for the Wiener Index of tree graphs (i.e., acyclic molecules) have also been investigated.[15] Algorithms have been proposed[16,17] supporting the reconstruction of the underlying graph from the actual value of the Wiener Index. Most applications of the Wiener Index are related to saturated hydrocarbons, but attempts have been undertaken to extend its definition for unsaturated hydrocarbons[18,19] and for heteroatoms.[20,21] For a series of molecules possessing a common parent structure, *W* could be decomposed into three portions. These are related to the parent structure, the substituents, and the interaction terms between the latter.[11]

Most often, numerical methods are used to calculate *W*. The first method suggested by Wiener,[1] can be used only for trees. The easiest way to obtain *W* is by eq 1, and the majority of the methods may be classified by the approach used to construct the matrix of distances. d_{ij} may be obtained by using the adjacency matrix approach,[22] the method of linked lists,[23] the algorithm by Müller et al.[24] or the Warshall algorithm.[25] Analytical expressions may be used to obtain *W* directly. The formulas were used to study the effects of structural variations and to find different structures with an identical *W*. Explicit