PROCEEDINGS OF THE INTERNATIONAL CONFERENCE
ON FIFTH GENERATION COMPUTER SYSTEMS 1988,
edited by ICOT. © ICOT, 1988

CHEMILOG

- A Logic Programming Language/System for Chemical Information Processing -

Tatsuya Akutsu Setsuo Ohsuga

Research Center for Advanced Science and Technology.
The University of Tokyo,
4-6-1 Komaba Meguro-ku Tokyo, 153 Japan

ABSTRACT

In this paper, we show a logic programming language/system named "Chemilog" whose purpose is to support the development of chemical knowledge information processing systems. Chemilog is an extension of Prolog and is viewed as a CLP (constraint logic programming) language. In Chemilog, graphs representing chemical structures are treated as basic components and isomorphic graphs are identified. Such constraints as substructure recognition and replacement of substructures make it possible for users to easily write programs which handle chemical structures in a natural way and in a visual form. The Chemilog system consists of two major parts: an inference engine and a chemical structure database. Chemilog is a language in the domain of chemistry, but the method described in this paper can be applied to other engineering domains where graph representation is used.

1 INTRODUCTION

Knowledge information processing technology is applied to many engineering fields. In most knowledge information processing systems, languages with facilities of symbolic manipulation (for example, Lisp and Prolog) are used. To make chemical knowledge information processing systems, it is required to represent chemical structures. Graph representation is used in most chemical information processing systems. Further, the following facilities play a central role and algorithms for them have been investigated(Morgan 1965),(Sussenguth 1964),(Shelley 1977),(Randic and Wilkins 1979),(Funatsu et al. 1986) :

(1) testing isomorphism of two given structures

(2) pattern matching of a given structure and a given substructure

(3) exhaustive enumeration of unique structures consistent with structural information

(4) finding maximal substructures which are common with two given structures

So that, the systems for chemical knowledge information processing must represent and handle chemical structures as well as symbolic data.

Jaffar and Lassez (1986) showed a new approach to extend Prolog in the practical domains: *the constraint logic programming* (CLP) scheme. Unification is generalized to constraints solving in the CLP framework. They demonstrated its power in the domain of real arithmetics. They suggested that the CLP scheme can be applied to not only real arithmetics but also many practical domains.

Chemilog is a logic programming language/system which is made to include chemical structures as basic components in the CLP framework. The domain, which is the Herbrand universe in Prolog, is extended to the domain including graphs for chemical structure representation. In Chemilog, graphs of chemical structures are inputed with a graphic editor. Graphs are treated as basic components and users need not know how graphs are implemented in the system. Chemilog has been made to allow chemists to easily write and use their knowledges. As Chemilog is not difficult to learn, it will be a very powerful assistant for chemists.

2 AN OVERVIEW OF THE LANGUAGE

2.1 Chemical Graph

We briefly summarize *chemical graph* (called *chromatic graph* in (Dubois 1976)) which is used to represent chemical structures. We refer to chemical graph as graph in this paper. A (chemical) graph G is a quadruplet: $(V,E,\chi v,\chi e)$, where V is the finite set of the graph nodes; E is the finite set of edges linking the elements of V; χv is the function which, when applied to an element of V, expresses its atom name; χe is the function which, when applied to an element of E, expresses its kind of interatomic bond. Atom name is one of { C,H,O,N,F,Cl,Br,I,S,Na,K,Ca,Mn,Mg,A }, where 'A' is introduced as a wildcard to match any atom. But this is interpreted as so only in some builtin-predicates (constraints) which are explained in §2.3. The bond is one of {1,2,3,aromatic}. Aromatic bond is used for representing such as benzene ring.

Two chemical graphs G(V,E,χv,χe) and G'(V',E',χv',χe') are *isomorphic* if and only if the following conditions are satisfied:

(1) There is a bijection between V and V'. v' ($\in$V') denotes the corresponding node to v$\in$V.

(2) There is an edge (v1,v2)$\in$E iff. there is an edge (v1',v2')$\in$E'. e' ($\in$E') denotes the corresponding edge to e$\in$E.

(3) For all v$\in$V, χv(v) = χv'(v') .

(4) For all e$\in$E, χe(e) = χe'(e') .

Graph G(V,E,χv,χe) is a *subgraph* of G'(V',E',χv',χe') if and only if the following conditions are satisfied:

(1) There is an injection from V to V'. v' ($\in$V') denotes the corresponding node to v$\in$V.

(2) For all v1,v2 $\in$V, there is an edge (v1,v2) iff. there is an edge (v1',v2') . e' ($\in$E') denotes the corresponding edge to e$\in$E.

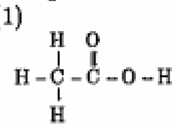
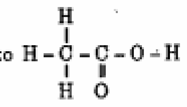
(3) For all v$\in$V, χv(v) = χv'(v') .

(4) For all e$\in$E, χe(e) = χe'(e') .

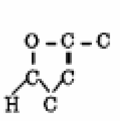
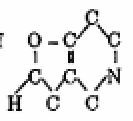
Note that the definition of subgraph is different from that of usual graph theory.

examples

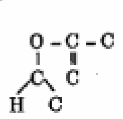
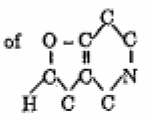
(1)

 is isomorphic to 

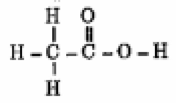
(2)

 is a subgraph of 

(3)

 is not a subgraph of 

For the convenience of description, some obvious edges are omitted. The structure:



for example, can be written as CH_3-COOH or CH_3COOH. And, in this paper, we use X,Y,Z,U,V,W,... to denote variables and C,H,N,O,A... to denote nodes of graphs and G,G' to denote graphs.

2.2 Syntax and Domain in Chemilog

Because Chemilog is an extension of Prolog and one of the CLP languages, its syntax is the same as CLP except that a term may contain chemical graphs and nodes in terms (the definition of a term is explained later). A Chemilog clause is of the form:

p(...):-q1(...),q2(...), ... ,qn(...) .

where p is a predicate symbol and the qi's are either predicates or constraints. But, we may use the term predicate or builtin-predicate instead of constraint because in Chemilog, there is no difference between constraint and builtin-predicate when lazy evaluation is not included.

In Chemilog, the Herbrand universe is extended to include chemical graphs and their nodes. Our domain is constructed in a way similar to Prolog. It is a set of all ground terms. A term is defined inductively as follows:

(1) A variable is a term.

(2) A constant is a term.

(3) A chemical graph is a term.

(4) A node is a term.

(5) If f is an n-ary function and t_1,...,t_n are terms, then f($t1_1$,...,t_n) is a term.

Of course, integer and real number (floating point number) are considered in our implementation. But we omit them to simplify our discussion. Chemical graphs and nodes must satisfy the following conditions:

(1) Isomorphic graphs are identified. So that, there are no two isomorphic graphs.

(2) If $G(V,E,\chi v,\chi e)$ and $G(V',E',\chi v',\chi e')$ are not isomorphic, V and V' have no common nodes. So that, a node does not belong to more than one graph.

The condition (1) is very important to give a simple and natural declarative semantics to Chemilog. To preserve condition (1), a special mechanism are introduced into Chemilog. The details are explained in §2.5 .

Although edges are not defined as basic objects, information about them can be got by using builtin-predicate:

getedge(Node1,Node2,KindOfEdge)

The meaning of this predicate is obvious.

2.3 Graph Manipulation in Chemilog

It is very important to represent and handle transformation rules of chemical structures such as chemical reaction formula. In earlier implementation, we introduced the concept *graph-term* which was an extension of chemical graph. In a graph-term, nodes may be variables. For example, the assertion: 'X is a carboxylic-acid'(i.e. 'X has COOH as its subgraph') can be described as follows:

X = Y-COOH

The assertion: 'Z is got by replacing -COOH of X by -OH' can be described as follows:

X = Y-COOH, Z = Y-OH

It seemed elegant. But, we found that graph-term was not sufficient to represent every transformation rules and efficient implementation of it was difficult. We gave up using graph-term. In place of using graph-term, we explicitly write transformation rules by using builtin-predicate:

convert(In,InPat,Map,OutPat,Out)
where
In: A list of input graphs
InPat: A list of input graph subpatterns
Map: A list of pairs of corresponding nodes in input graph patterns and output graph patterns
OutPat: A list of output graph subpatterns
Out: A list of output graphs

Although we use the terms input and output, the system can work reversely. This predicate implicitly uses the subgraph matching algorithm.

As formal description takes much space, we will explain with examples.

(1)

Primary alcohol is made by deoxidizing arudehyde.

$$R\text{-}CH{=}O \xrightarrow{[H]} R\text{-}CH_2\text{-}OH$$

This transformation rule can be written as follows:

convert([X],[A^1-CH=O],[(A^1,A^2)],
[A^2-CH_2-OH],[Y]). ,

where the superscript on a node is an index to distinguish different nodes of the same atom name. For example, if X is bound to CH_3-CH_2-CH=O, then Y will be bound to CH_3-CH_2-CH_2-OH. But, if the rule is as follows:

convert([X],[A^1-CH=O],[ ],[A^2-CH_2-OH],[Y]). ,

then Y will be bound to A-CH_2-OH.

(2)

Ester is made by reaction between acid and alcohol.

$$R\text{-}OH + R'\text{-}COOH \longrightarrow ROCOR' + H_2O$$

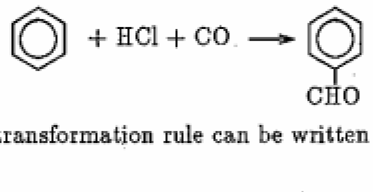
This transformation rule can be written as follows:

convert([X,Y],[A^1-OH, A^2-COOH],
[(A^1,A^3),(A^2,A^4)],[A^3-OCO-A^4],[Z]).

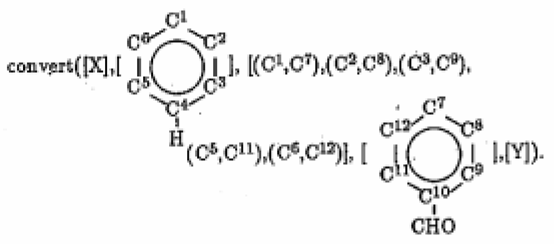
For example, if X is bound to CH_3-OH and Y is bound to CH_3-CH_2-COOH, then Z will be bound to CH_3-OCO-CH_2-CH_3. Conversely, if Z is bound to CH_3-OCO-CH_2-CH_3 at first, then X will be bound to CH_3-OH and Y will be bound to CH_3-CH_2-COOH.

(3)

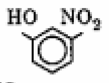
Aromatic arudehyde is made from aromatic compound by the Gettermann-Koch method.

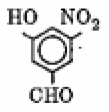


This transformation rule can be written as follows:



convert([X],[
C^1 C^2 C^3 C^4 C^5 C^6
H
], [(C^1,C^7),(C^2,C^8),(C^3,C^9),
(C^5,C^{11}),(C^6,C^{12})], [
C^7 C^8 C^9 C^{10} C^{11} C^{12}
CHO
],[Y]).

For example, if X is bound to 

then Y will be bound to 

It is also important to test graph-subgraph relation. Although testing graph-subgraph relation can be done by the 'convert' predicate, we introduce builtin-predicate: contain(Graph,SubGraph) for the purpose of efficient execution.

2.4 The Delay Mechanism

The key point of CLP scheme is as follows:

(1) The concept of syntactic unification in the Herbrand universe is replaced by constraint satisfaction in the domain of application.

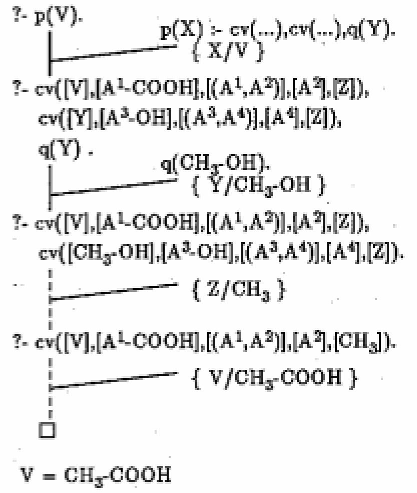
(2) Evaluation of the constraints are delayed until the constraint solver can determine whether the constraints are solvable or unsolvable.

As to (1), the Herbrand universe is extended to include chemical graph in Chemilog. Although there are no corresponding mechanism to the linear programming solver in CLP(*R*), many builtin-predicates such as those mentioned in §2.3 are added for manipulating graphs in Chemilog. As to (2), the delay mechanism is also introduced into Chemilog to preserve declarative semantics. For example, 'convert' predicates can not be executed until either all variables in the first argument are bound to graphs or all variables in the last argument are bound to graphs.

Consider a simple example:

p(X):-
  convert([X],[A^1-COOH],[(A^1,A^2)],[A^2],Z),
  convert([Y],[A^3-OH],[(A^3,A^4)],[A^4],Z),
  q(Y).
q(CH_3-OH).
?- p(V).

where p(X) in the first formula means that X is a graph which is got by replacing -OH of a graph Y that have property q(...) by -COOH. Figure 1 illustrates one of execution traces for this example.

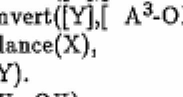


```
?- p(V).
                          p(X) :- cv(...),cv(...),q(Y).
                           { X/V }
?- cv([V],[A1-COOH],[(A1,A2)],[A2],[Z]),
   cv([Y],[A3-OH],[(A3,A4)],[A4],[Z]),
   q(Y) .
                          q(CH3-OH).
                           { Y/CH3-OH }
?- cv([V],[A1-COOH],[(A1,A2)],[A2],[Z]),
   cv([CH3-OH],[A3-OH],[(A3,A4)],[A4],[Z]).
                           { Z/CH3 }
?- cv([V],[A1-COOH],[(A1,A2)],[A2],[CH3]).
                           { V/CH3-COOH }
  □

 V = CH3-COOH
```

Figure 1: A trace example of Chemilog program (The predicate 'convert' is abbreviated as 'cv')

Note that evaluation of constraints is delayed until the previously described condition is satisfied. Although the system returns not only the expected answer but also return the unexpected answers. The following is an unexpected answer:



The reason is that a node A^2 matches not only C in CH_3- but also H in CH_3-. To get only the expected answer, we rewrite this program as follows:

p(X):-
  convert([X],[A^1-COOH],[(A^1,A^2)],[A^2],Z),
  convert([Y],[A^3-OH],[(A^3,A^4)],[A^4],Z),
  balance(X),
  q(Y).
q(CH_3-OH).

where balance(X) is a builtin-predicate in Chemilog. It is used to test whether all atoms (nodes) of X use exactly the same number of bonds associated with them, for example, 4 for 'C' and 3 for 'N'. Of course, it does not solve all problems. There are more complex cases for which we must write somehow complicated programs. But, in most cases, it can be described by using 'convert' predicate only.

2.5 Other Features of Chemilog

Chemilog has many builtin-predicates for manipulating graphs. We list some important ones that have not been mentioned yet.

(1) getallnodes(Graph,ListOfNodes)
    getting a list of nodes in 'Graph'

(2) adjacentnodes(Node,ListOfNodes)
    getting a list of nodes adjacent to 'Node'

(3) connectnodes(Graph,Node1,Node2,Bond,NewGraph)
    getting a new graph by connecting 'Node1' and 'Node2' of 'Graph'

(4) getatomname(Node1,AtomName)
    getting an atom name of 'Node1'

As mentioned in §2.2, isomorphic graphs are identified in Chemilog. Because we think that it is natural to identify isomorphic graphs. Let us consider the following example:

```
p(X):-
  convert([$CH_3$-COOH],[$A^1$-COOH],[($A^1$,$A^2$)],[$A^2$-OH],[X]).
q(X):-
  convert([$CH_3$-COOH],[$A^3$-COOH],[($A^3$,$A^4$)],[$A^4$-OH],[X]).
?- p(X), q(Y), X=Y.
```

If we identify isomorphic graphs, the answer is 'Yes'. Otherwise, the answer is 'No'. It seems natural to adopt the former interpretation. Next, consider the following goal:

```
?- X=Y, p(X), q(Y).
```

In this case, 'q(Y)' will work as test. Even if we do not identify isomorphic graphs, the answer will become 'Yes'. A logic programming system is not a good one if answers depend on evaluation order. So we identify isomorphic graphs.

To identify isomorphic graphs, two methods can be considered:

(1) Extending unification algorithm to include the facility of checking graph isomorphism.

(2) When a new graph is made by evaluating such predicates as 'convert' and 'connectnodes', the database which contains chemical graphs is searched for the graph. If an isomorphic graph is found, the new graph is abandoned. Otherwise, the new graph is registered.

Using a morgan index file (it is explained in §3.1) and binary search, searching an isomorphic graph is carried out in O(log(n)) time, where n is the number of graphs. But, if we employ a method (2), we must consider corresponding nodes between isomorphic graphs. For that purpose, we must maintain map of corresponding nodes or we must make canonicalization of nodes. It would make implementation very difficult. So we employ method (1). In fact, we have made some experimentations and found that execution time increased by only 20-30% when isomorphic structure search is considered.

2.6 Examples

In this section, we show some short examples of Chemilog programs. One of the features of Chemilog is that chemical structures can be written in a visual style and it does not require such data structures as list, atom and function symbols to represent chemical structures.

(1)

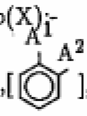
```
primary_alcohol(X):-
  convert([X],[ $A^1$-$CH_2$-OH ],[($A^1$,$A^2$)],[ $A^2$ ],[Y]),
  alkyl_group(Y).

secondary_alcohol(X):-
  convert([X],[ $\begin{smallmatrix}A^1\\A^2\end{smallmatrix}$>CH −OH ],
       [($A^1$,$A^3$),($A^2$,$A^4$)],[ $A^3$, $A^4$ ],[Y,Z]).
  alkyl_group(Y),
  alkyl_group(Z).
```

In these programs, 'primary_alcohol(X)' ('secondary_alcohol(X)',respectively) means that the molecule 'X' is primary alcohol (secondary alcohol, respectively). 'alkyl_group(Y)' means that 'Y' belongs to alkyl group, whose molecular formula is C_nH_{2n+1}. You will see easily that this predicate can be written in Chemilog.

(2)

is_ortho(X):-

convert([X],[],[(A^1,A^3),(A^2,A^4)],[A^3,A^4],[Y,Z]).

In this program, 'is_ortho(X)' means that molecule 'X' is ortho-aromatic compound.

(3)

```
    how_to_make( Molecule ,[ ] ):-
        mol_in_db( Molecule ).
    how_to_make( New ,[RuleId | Rules]):-
```

covert_rule(RuleId, Original, New),
how_to_make(Original, Rules).

This is a much simplified chemical synthesis program. The meaning of mol_in_db(X) is that the graph 'X' is registed in the database and how to make it is already known. The meaning of convert_rule(RuleId,Orig,New) is that the molecule 'New' is made from the molecule 'Orig' by using the rule whose rule ID is 'RuleId'.

2.7 Soundness and Completeness

One of the most important features of Logic Programming is that its declarative and operational semantics are both simple and elegant. Further, they coincide in a natural way (i.e. strong completeness of SLD-resolution(Lloyd 1984)). In Chemilog, the soundness and completeness must be considered so that user can write their knowledges declaratively. Because we have not fixed the specification of Chemilog yet and we are making modifications, we cannot discuss the semantics of Chemilog formally. Of course, we are taking care to preserve soundness. We do not expect Chemilog to be complete in the sense that the system answers 'yes' if and only if a given goal is unsatisfiable. Because it is very difficult to determine whether the given constraints are satisfiable or not. Even constraints are limited to 'convert' predicate only, we have not yet found an algorithm to determine whether the given constraints are solvable or not. The following example shows this difficulty.

?-convert([U],[A^{1}-A^{2}],[(A^{1},A^{3}),(A^{2},A^{4})],[A^{3},A^{4}],[X,Y]),
  convert([U],[A^{5}-A^{6}],[(A^{5},A^{7}),(A^{6},A^{8})],[A^{7},A^{8}],[Z,W]),
  convert([V],[CH3-A^{9}-A^{11}],[(A^{9},A^{11}),(A^{10},A^{12})],
                    [A^{11}, A^{12}],[X,Z]),
  convert([V],[NH2-A^{13}-A^{14}],[(A^{13},A^{15}),(A^{14},A^{16})],
                    [A^{15}, A^{16}],[Y,W]).

This goal is not solvable, but it is difficult to determine. So, Chemilog is currently using the delay mechanism and the completeness does not hold in a simple way. Of course, it is possible to return remaining constraints which can not be simplified as answer constraints.

Further, if we claim that Chemilog is a CLP language in the strict sense, we must prove some conditions(Jaffar and Lassez 1986) . Although we do not prove them, we think that Chemilog can be made to satisfy these conditions because the domain of Chemilog has no limit elements.

3 AN OVERVIEW OF THE SYSTEM

The Chemilog system consists of two major parts.

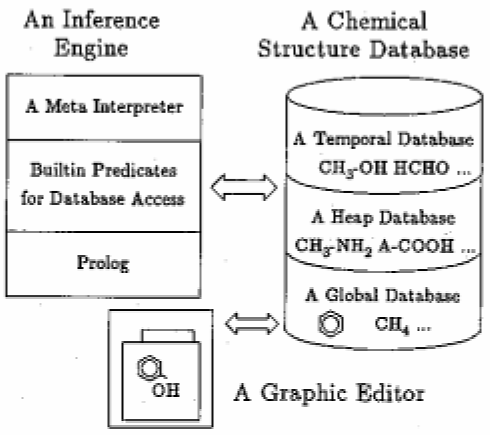


Figure 2: System Overview

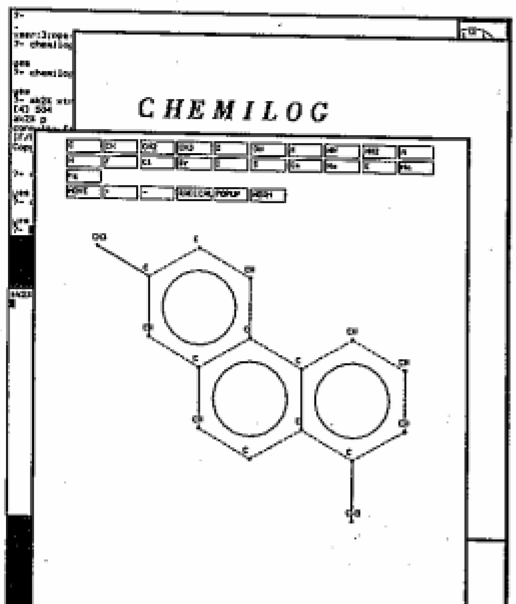


Figure 3: The Graphic Editor

One is a database which contains chemical structures represented by graphs. It is called the chemical structure database and it is implemented in C language (see Figure 2). The other is an inference engine which is implemented in Prolog and many builtin-predicates are added for interaction with the database. Chemical structure is passed in the forms of pointer between the inference engine and the chemical structure database. The Chemilog system is built on the SONY NEWS workstation with 68020 CPU and UNIX-4.2BSD. A graphic editor (Figure 3) for chemical structure input and output is implemented with X-Window.

3.1 The Chemical Structure Database

The Chemical Structure Database is a database which contains graph representations of chemical structures. The purpose of this database system is to make fast search of chemical structures and substructures.

Chemical structure search problem includes the graph isomorphism testing problem which has been famous as an open problem that whether there is a polynomial time algorithm. But efficient algorithms have been developed by chemists. Although complexity of these algorithms is unlikely to be in polynomial time, these algorithms run efficiently in most cases. Most of these algorithms are modified versions of the Morgan algorithm(Morgan 1965). The most important feature of this algorithm is that it makes the index (i.e. unique naming) of the given chemical structure. So we use a modified Morgan algorithm (Shelley 1977) for searching the given chemical structure.

As well as chemical structure search, chemical substructure search is important for chemical database systems. Although unlikely to be in polynomial time, efficient algorithms for chemical substructure matching have been developed by chemists. The set reduction algorithm(Sussenguth 1964) and the atom by atom algorithm(Randic and Wilkins 1979) are famous ones. These algorithms not only examine subgraph-graph relation but also find correspondences of nodes. Our algorithm combines some features of the set reduction algorithm to the atom by atom algorithm. This algorithm plays a central role in 'convert' and 'contain' predicates.

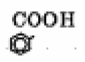
The database is divided into three parts:

A Global Database
A Heap Database
A Temporal Database

All of them contain chemical structure data and miscellaneous information which is concerned with structure data. The global database contains a large number of permanent data. The heap database contains data existing only when the system is running. Data in program clauses and data generated by such builtin-predicates as 'assert' are registed in this database. The temporal database is used as a stack for the inference engine. Chemical structures which are generated during inference process are registed in it until backtracking occurs. Chemical structures are represented by adjacency lists in the databases, because this representation of graph is reasonably economical and most graph algorithms are efficiently executed with adjacency lists(Aho et al. 1974) .

To make structure and substructure search faster, the global database and the heap database have many index files. One of the index files contains Morgan index. This file is sorted and used to find isomorphic graphs by making binary search.

By the way, most of the other index files are lists of pointers whose structures contain the special substructures such as benzene ring. For example, consider the case that we want to find a structure which contains:



At first, the system get a list of structures which contain benzene ring and a list of structures which contain COOH by using two index files. Next, the system applies a join operation to get a list of structures which contain both benzene ring and COOH. At last, the system searches the list by using the substructure matching algorithm.

3.2 The Inference Engine

The inference engine makes much use of Prolog inference mechanism. The (meta) interpreter is implemented on Prolog system. If we do not use chemical structures, programs are treated as usual Prolog programs. The interpreter interprets Chemilog programs and controls the delay mechanism. The constraints (builtin-predicates) are tested whether they can be evaluated by their own routines. When they can not to be evaluated, evaluation of them is delayed. After each derivation step, all the delayed constraints are tested and if any of the constraints can be evaluated, they are evaluated. If no predicates can be resolved and no constraints can be evaluated, the interpreter returns them to the user.

The current interpreter is an experimental version and the system is being developed. The central part is described in Prolog and not much attention has been paid to efficiency. So, we sometimes write programs procedually instead of using delay mechanism and writing programs declaratively. Efficient implementation is an important ongoing subject of our research.

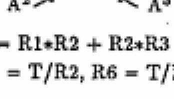
4 CONCLUSIONS

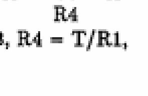
Chemilog is a logic programming system which handles graphs of chemical structures in an elegant way. Chemilog is an extension of Prolog and is viewed as a constraint logic programming language. The domain of CLP(R)(Jaffar and Lassez 1986) is real arithmetic. In Chemilog, the domain contains graphs of chemical structures. Chemilog is equipped with such constraints as substructure recognition and replacement of substructures which makes it possible for users to easily write chemical knowledges in a natural way and in a visual form. Although there are several problems remained, they can be solved.

The important subject we must do is to make a practical chemical application system on Chemilog. For that purpose, we must extend Chemilog to handle more information such as the stereochemical information. Chemilog should also have more facilities for handling chemical structures. Facilities (3) and (4) mentioned in §1 have not been yet implemented in Chemilog. Now we are making two applications to examine Chemilog's ability. One is bidirectional transformation between IUPAC names and chemical structures. For example, IUPAC name of molecule $C(CH_3)_4$ is '2,2-dimethylpropane'. The other is a system for chemical synthesis. Of course, domains are limited in both applications because it is too difficult to cover all the domains of chemistry. This is part of the project "knowledge based system for chemical compound design" sponsored by ministry of science and technology in Japan.

By the way, the method described in this paper can be applied to other domains. Graph is a general data structure in many engineering fields such as electric circuits, traffic lines, control system, computer networks, and so on. For example, if we make a modified version of Chemilog, the star-delta transformation rule in electric circuit theory can be written as follows:

star_delta(S,D):-

convert([S],[

],[(A¹,A⁴),...],[

],[D]),

T = R1*R2 + R2*R3 + R1*R3, R4 = T/R1,
R5 = T/R2, R6 = T/R3.

Combining with user friendly graphic editors, users will be able to write down their knowledge in a natural way.

ACKNOWLEDGEMENTS

We wish to thank Prof. Sasaki and Dr. Funatsu in Toyohashi University of Technology for their valuable discussions and suggestions on chemical information processing systems. We are grateful to Prof. Hori for his valuable discussions on knowledge information systems. We also want to thank Mr. C. Y. Li for his program of displaying chemical structures.

REFERENCES

[1] J. Jaffar and J-L. Lassez, "Constraint Logic Programming", in *Proc. Conf. on Principle of Programming Languages*, pp. 111-118, 1986.

[2] H. L. Morgan, "The Generation of a Unique Machine Description for Chemical Structures - A Technique Developed at Chemical Abstracts Service", *J. Chemical Documentation*, vol. 5, pp. 107-113, 1965.

[3] E. H. Sussenguth Jr., "A Graph-Theoretic Algorithm for Matching Chemical Structures", *J. Chemical Documentation*, vol. 5, no. 4, pp. 36-43, 1964.

[4] C. A. Shelley and M. E. Munk, "Computer Perception of Topological Symmetry", *J. Chemical Information and Computer Science*, vol. 17, no. 2, pp. 110-113, 1977.

[5] M. Randic and C. L. Wilkins, "Graph-Based Fragment Searches in Polycyclic Structures", *J. Chemical Information and Computer Science*, vol. 19, no. 1, pp. 23-31, 1979.

[6] K. Funatsu, C. A. Del Carpio, and S. Sasaki, "Computer-assisted Structure Elucidation Based on the Interdependent Analysis of H- and C-NMR Spectra", *Computer Enhanced Spectroscopy*, vol. 3, pp. 133-140, 1986.

[7] J. E. Dubois, "Ordered Chromatic Graph and Limited Environment Concept", in *Chemical Applications of Graph Theory*, ed. A. T. Balaban, pp. 333-370, 1976.

[8] J. W. Lloyd, *Foundations of Logic Programming*, Springer-Verlag, 1984.

[9] A. V. Aho, J. E. Hopcroft, and J. D. Ullman, *The Design and Analysis of Computer Algorithms*, Addison-Wesley, 1974.