Int. J. Human–Computer Studies (1994) **41**, 5–32

An object-based representation system for organic synthesis planning

AMEDEO NAPOLI
CRIN CNRS–INRIA Lorraine, B.P. 239, 54506 Vandœuvre-lès-Nancy Cedex, France

CLAUDE LAURENÇO
CICPE, rue de la Cardonille, 34094 Montpellier Cedex 5 and LIRMM, 161 rue Ada, 34392 Montpellier Cedex 5, France

ROLAND DUCOURNAU
Sema Group, 16 rue Barbès, 92126 Montrouge, France

In this paper, we present an application of object-based representation techniques to the design of a knowledge-based system for organic synthesis planning. The notion of an object-based representation system described in this paper relies on the integration of basic programming styles, such as message sending and access-oriented programming, with reasoning capabilities, such as classification. The basis for integration is the subsumption relation that is used jointly with inheritance to provide a two-dimensional universe in which representation and problem solving can be performed successfully.

1. Introduction

This article presents a theoretical and practical study in the field of object-based representation systems. The theoretical aspect is concerned with the definition of reasoning capabilities in an object-based environment. The practical aspect is concerned with the description of the principles ruling the design of a knowledge-based system for organic synthesis planning, i.e. handling synthetic pathways composed of ordered sequences of chemical operations leading from basic available molecules to a new target molecule. A prototype system has been built according to these principles. This paper is mainly a contribution to the study of the object-based representations of complex real-world concepts and the associated problem-solving schemes. Indeed, the design of a knowledge-based system for organic synthesis planning working on a real scale must take into account a large number of complex and evolving concepts—chemical objects and operations have an actual life—and requires many efforts and resources, e.g. description of thousands of chemical reactions and development of various sophisticated graphical interfaces. The first classical systems, e.g. LHASA (Corey, 1971) and SYNCHEM (Gelernter, Sridharan, Hart, Yen, Fowler & Shue, 1973), began to be designed more than twenty years ago. They are still under development and they still need to be improved. The use of an object-based environment is a valuable alternative to the programming schemes underlying classical systems, because it favors the representation of real-world domains including important, structured, complex and evolving knowledge. The paper gives details about a system relying on the association of an object-based representation with classification-based reasoning. This association enforces the

1071-5819/94/070005 + 28$08.00/0

representation and reasoning capabilities of object-based systems and facilitates the evolution and the maintenance of knowledge. The resulting system must not be viewed as a working system that is at the same level as LHASA or SYNCHEM, but rather as a system validating different and new representation ideas in the field of computer-aided organic synthesis planning, and as a basis for further developments.

In an object-oriented context, the *specialization* relation, ako in the following, is used to organize classes according to their generality in an inheritance hierarchy: a class is defined as a specialization of one or more superclasses, and it inherits the characteristics of its superclasses. In the context of terminological logics, the *subsumption* relation is used to organize concepts denoting sets of individuals in a subsumption hierarchy: a concept C subsumes a concept D if every individual described by D is also described by C. A classification process is used to compute the precise location of a new concept in the subsumption hierarchy. Subsumption is also used in predicate logics where it roughly corresponds to implication (Gottlob, 1987). Thus, subsumption involves a certain idea of specialization/generalization. From the preceding remarks, it follows that a parallel can be drawn between the specialization and subsumption relations, i.e. they are both a basis for a hierarchical organization of objects (taxonomies of classes or concepts), and between the associated inference mechanisms, i.e. inheritance and classification. The semantics of property sharing of inheritance can then be associated to subsumption, i.e. a concept D subsumed by a concept C inherits the properties attached with C. We have directly applied the preceding ideas to manage structural inclusion associated with the part–whole relation. Structural inclusion is used to compare and to organize structured objects, e.g. graphs. It can be considered as a specialization/subsumption relation, in which case: (1) it is possible to share information between structured objects, and (2) it is possible to take advantage of the reasoning capabilities associated with subsumption, i.e. classification-based reasoning. In this way, we work on a two-dimensional universe, specialization and inheritance for representation purposes (first dimension), subsumption and classification for problem-solving purposes (second dimension).

The paper is divided into two main parts: the basics of object-based representation systems, and a description of an application in the field of organic synthesis planning. The first part of the paper begins with a short presentation of the connections existing between class-based languages, prototype-based languages and terminological logics. In this context, we introduce our view of object-based representation systems that integrates both programming capabilities and representation and reasoning mechanisms. One can note that the expressions "object-based language" and "object-oriented language" have been defined in Wegner (1987) as, respectively: (1) a language that supports objects as a language feature, an object being defined as a set of "operations" and a "state" that remembers the effect of operations; and (2) an object-based language where objects belongs to classes and class hierarchies may be incrementally defined by an inheritance mechanism (object-oriented = objects + classes + inheritance). As we use objects defined by a state and a behavior, belonging to classes (actually frames), organized in an inheritance hierarchy, we could use the term "object-oriented representation system". However, the term "object-oriented" has a very strong programming flavor, and our concern is much more related to representation and reasoning.

Thus, we prefer to use the term "object-based representation system". In some studies, the term "object-centered representation" is also employed for the same purpose (Nebel, 1990).

The second part of the paper relates an application based on the ideas presented in the first part of the paper to organic synthesis planning. First, we summarize the principles of organic synthesis planning and give a brief overview of classical computer-aided synthesis systems. Then, we describe the object-based representation of basic chemical concepts, namely atoms, bonds, molecular structures and chemical transformations. Then, we explain *classification-based synthesis,* a process based on a problem-solving cycle involving structural inclusion. A synthetic pathway is built according to the structural characteristics or substructures of the target molecule. The recognition of these characteristics relies on the classification of the target molecule in a special hierarchy of generic substructures based on structural inclusion. In this hierarchy, properties are attached to substructures and shared among substructures just as in a usual inheritance hierarchy. A target molecule can then be processed according to the specific substructures that it includes. Thus, the chemical knowledge base can be seen as a two-dimensional space, one dimension being related to representation, i.e. the inheritance hierarchy organizing basic chemical objects, and the other dimension corresponding to structural inclusion of substructures.

In conclusion, we discuss the benefits of an object-based approach in organic synthesis planning and point out some important open problems in the field, such as the representation and the manipulation of synthetic strategies.

2. Object-based representation systems

2.1. OBJECTS, CLASSES, PROTOTYPES AND CONCEPTUAL DESCRIPTIONS

An object-based system allows one to represent real-world knowledge, i.e. concepts, individuals deriving from concepts, and relations between concepts, in a hierarchy of objects. Generic objects are used to describe concepts, and instances deriving from generic objects describe specific individuals. A generic object encapsulates a collection of structural and behavioral properties, i.e. data and procedures acting on these data. The set of generic objects is organized in an inheritance hierarchy, where the more specific objects are defined as specializations of the more general objects. Every generic object inherits the properties of one or more ascendants. Individuals usually depend on one single generic object, and they are the entities effectively handled in the applications. In the field of object-oriented programming, two main models are usually used to implement objects and hierarchies of objects, namely class-based and prototype-based models (Masini, Napoli, Colnet, Léonard & Tombre, 1991).

In the first model, generic objects are classes that define the structure, i.e. instance variables, and the behavior, i.e. methods, of a set of instances. Message sending allows one to handle instances and is at the heart of object-oriented programming.

In the second model, generic objects are usually three-level structures, frame-attribute-facet.† A frame-based language supplies a set of services to allow access to

† For an alternative implementation of prototypes, see for example Ungar and Smith (1987).

information, default reasoning, procedural attachment and access-oriented programming (Stefik, Bobrow & Kahn, 1986). These services are mainly built into primitives for accessing and updating frames, attributes and facets (reading, writing and removing values). Moreover, frames allow the handling of incomplete descriptions: relations and their inverse relations can be described by special attributes; simple mechanisms such as default values or if-needed reflexes (or demons) can be used to complete missing values in incomplete frame descriptions. Furthermore, frame-based languages have progressed and allow both access-oriented and object-oriented programming (Karp, 1993; Kaindl, 1994). Generic objects are frame-slot-facet structures where slots can be attributes or methods† (examples of frame-slot-facet structures are given in Section 3.4).

Relying more on a logical basis, *terminological logics* or *description logics* are used to reason about concepts, also called terms or descriptions, denoting sets of elements, called individuals or instances (Nebel, 1990; Woods & Schmolze, 1992). Concepts and individuals are related through binary relations called roles. Defined concepts are composite terms built from primitive concepts and restrictions on roles. Basic information such as the kinds or number of role fillers can be associated with roles. The subsumption relation is a partial ordering used to organize concepts and their instances, according to a set-theoretical interpretation (Levesque & Brachman, 1987); a concept C—the *subsumer*—subsumes a concept D—the *subsumee*—if and only if the set of individuals denoted by C, the extension of C, necessarily includes the set of individuals denoted by D. Subsumption organizes concepts in a hierarchy where subsumers are placed above their subsumees. Classification is the basis of terminological reasoning. It is used to manage the hierarchy of concepts and individuals, controlling the evolution and truth maintenance of the hierarchy. It must be added that the richness of the language describing concepts has a direct effect on the computational complexity of the classification process.

The practical application of knowledge-based systems requires complex inferences, as well as the manipulation and the maintenance of large amounts of knowledge. As a knowledge base grows in size and complexity, it becomes more difficult to maintain and to extend. One approach to solve this kind of problem in frame-based systems is precisely to provide a classification tool that is able to acquire the description of new objects and to perform complex inferences, as will be explained in the next section.

2.2. A DEFINITION OF OBJECT-BASED REPRESENTATION SYSTEMS

Most of the time, inheritance is the primary and most powerful representation primitive in frame-based systems. Nevertheless, as pointed out in Patel-Schneider (1990), inheritance is a mechanism for knowledge sharing, but is deductively weak. By contrast, terminological logics are semantically well-founded knowledge representation tools and have powerful inference mechanisms relying on subsumption and classification-based reasoning. However, these representation systems lack the procedural component that one may need, for example, to model complex physical transformations of objects. Thus, we decided to integrate the basic deductive

† In early frame-based languages such as FRL (Roberts & Goldstein, 1977) or KRL (Bobrow & Winograd, 1977), frames did not have any global behavior defined by methods.

tools of terminological logics, namely subsumption and classification, in the context of frame-based systems. The representation capabilities of the resulting systems include a procedural component constituted by reflexes and methods activated by message sending.

More precisely, what we call an *object-based representation system* has the following features:

- The knowledge grain is defined by objects having a state (descriptive attributes) and a behavior (methods associated with objects and/or reflexes associated with attributes). Objects are divided into generic objects and instances.
- The primary organization of generic objects is an inheritance hierarchy.
- Instances are accessed through object-oriented programming (methods) or access-oriented programming (reflexes).
- Knowledge manipulation and reasoning rely on inheritance, classification and truth maintenance (the last point will not be discussed in the paper, see for example Russinoff, 1989). Matching is based on the comparison of object structures and is considered as a suboperation of classification.

In this next section, we make explicit the purpose of classification in the context of object-based representation systems.

2.3. CLASSIFICATION-BASED REASONING

We will assume that objects are organized in a hierarchy $\mathcal{H}$ according to a relation $\geq$ called *subsumption* as in terminological logics. The $\geq$ relation is supposed to be a partial ordering† and no other assumption is made about the semantics of $\geq$. Classification-based reasoning relies on a *classification cycle* making explicit the dependencies holding between a new object, say X, and the other objects lying in $\mathcal{H}$. The classification cycle proceeds with three main steps (inspired by Kaczmarek, Bates & Robins, 1986):

- *Instantiation*: the creation of a new object X depends on external information, e.g. X can be a query or a result produced by the system. X can be a generic or an individual object.
- *Classification*: the system first searches for the *most specific subsumers* of X (MSS), i.e. the most specific objects that are greater than X for $\geq$; then, it searches for the *most general subsumees* of X (MGS), i.e. the most general objects that are smaller than X for $\geq$. At last, X is inserted in the hierarchy under its MSS and above its MGS. When X is an individual object, the search for the MGS is useless.
- *Operations*: the modifications resulting from the insertion of X are spread in the whole hierarchy, possibly triggering either methods or reflexes. Usually, new objects are created, bringing the classification cycle back to its first step. The classification cycle continues until the goal is reached or no more information is available.

The search for the most specific subsumers and the most general subsumees can be performed using classical algorithms (Lipkis, 1982; Baader, Hollunder, Nebel,

† It could be more simply a preordering, i.e. a reflexive and transitive relation.

Profitlich & Franconi, 1992). Usually, the hierarchy $\mathcal{H}$ is explored depth-first and the exploration begins at the root of $\mathcal{H}$. For every object O*, a subsumption test is done to check whether O* subsumes X. If O* does not subsume X, none of the descendants of O* subsumes X, otherwise $\mathcal{H}$ would not be consistent ($\geq$ is transitive). Thus, the subhierarchy $\mathcal{H}_{O^*}$, whose root is O*, is pruned. If O* subsumes X, then the subsumption test is recursively performed in $\mathcal{H}_{O^*}$ in order to find the most specific subsumer. The exploration continues until all objects of the hierarchy have been compared to X. The search for the most general subsumees is constrained, first because the subsumption relation is transitive, i.e. the MGS of X are included in the descendants of the MSS of X, second, because it is sufficient to test descendants including the same set of properties as X. One can note that the MGS of an object are not necessarily the immediate descendants of its MSS (Finin, 1986). The insertion of a new object in $\mathcal{H}$ involves the creation of new links: the links between X and its MSS do not pose any particular problem, but the creation of links between X and its MGS can imply the removal of redundant transitivity links.

Our view of classification-based reasoning can be likened to taxonomic reasoning as presented in Attardi, Corradini, Diomedi and Simi (1986) and Attardi (1991): classification-based reasoning is an automated deduction procedure which exploits the properties of a partially ordered domain. A parallel can also be drawn with the representation formalism for n-ary relations, quantification and definition of concepts described in Gomez and Segami (1991). By contrast, classification as presented in Clancey (1985) and Chandrasekaran and Goel (1988) is a high-level problem-solving task especially used in diagnostic reasoning.

2.4. AN OBJECT-BASED VIEW OF SUBSUMPTION

Subsumption is one of the foundations of classification-based reasoning, and the extensional view of subsumption is quite well understood in the field of terminological logics. However, the general notion of subsumption involves another specific kind of subsumption, namely *structural subsumption* (Woods, 1991), related to the *intension* of generic objects, i.e. the set of properties associated with the object. In the following, we precisely define (what we mean by) subsumption in an object-based representation system. We assume that *definitional properties* of attributes are the domain, the range, the cardinality and the value of the attribute. Moreover, slots related to implementation purposes, e.g. attributes recording precedence lists, and slots associated with pieces of code such as methods or reflexes are not considered as definitional properties. An attribute with definitional properties is called a *definitional attribute.*

We are now ready to define subsumption for definitional attributes (inspired by slot specialization in CYC: Stephens, 1990).

Definition ($\geq_{sa}$)
An attribute $a1$ *subsumes* an attribute $a2$ if and only if $a2$ has at least the same or more specialized definitional properties as $a1$. The specialization of definitional properties can be stated as follows:

- The cardinality of $a2$ is a bounded interval included in the cardinality interval of $a1$. The domain and the range of an attribute depend on a type that can be

primitive such as *Number* (or one of its subtypes: *Integer, Rational, Float*), *Boolean, String, Character* and *Symbol,* or a user-defined generic object. The types associated with the domain and the range of $a2$ must be respectively specializations of the types associated with the domain and the range of $a1$.

- Attribute values must satisfy type and cardinality specifications. If the value of the attribute $a1$ is elementary or atomic, then the value of $a2$ must be the same. However, if the value of $a1$ is a list of elementary values, then the value of $a2$ can complete the value of $a1$ provided that the type and cardinality specifications are satisfied (in some cases, the value of $a1$ may be a subset of the value of $a2$).

One can note that (1) $a2$ may have an additional definitional property, and (2) $a1$ and $a2$ must have the same name and/or that $a1$ and $a2$ can designate the same attribute. The fact that the attribute $a1$ subsumes the attribute $a2$ will be denoted by $a1 \geq_{\mathcal{A}} a2$, where the ordering $\geq_{\mathcal{A}}$ relies on the rules given above.

We can now formally define the subsumption relation or *O-subsumption,* for Object-based subsumption, in the context of object-based representation systems.

Definition ($\geq_{\mathcal{O}}$)
Let $\mathcal{O}$ be the set of generic and individual objects, $\mathcal{A}$ the set of definitional attributes associated with the objects in $\mathcal{O}$ and $\mathcal{A}o$ the subset of definitional attributes associated with the object o. Let o and o' be two objects in $\mathcal{O}$. Then, o *O-subsumes* o', and we write $o \geq_{\mathcal{O}} o'$, if and only if:

- The set $\mathcal{A}o$ of definitional attributes of o is a subset of the set $\mathcal{A}o'$ of definitional attributes of o'.
- For every attribute a in $\mathcal{A}o$, there exists a corresponding attribute a' in $\mathcal{A}o'$ subsumed by a, i.e. for every $a \in \mathcal{A}o$, $\exists a' \in \mathcal{A}o'$ such as $a \geq_{\mathcal{A}} a'$.

For example, let $o \equiv (a(o) \wedge b(o))$, where $a(o)$ denotes the definitional attribute a in o, and let $o' \equiv (a(o') \wedge b(o') \wedge c(o'))$, then $o \geq_{\mathcal{O}} o'$ if and only if $a(o) \geq_{\mathcal{A}} a(o')$ and $b(o) \geq_{\mathcal{A}} b(o')$. Note that $\mathcal{A}o = \{a, b\} \subseteq \mathcal{A}o' = \{a, b, c\}$, and that only definitional attributes are taken into account.

The O-subsumption relation is based on inclusion of sets of definitional attributes and on restriction of definitional properties. Thus, O-subsumption is a reflexive and a transitive relation, i.e. $\geq_{\mathcal{O}}$ is a preordering, and it organizes objects in a hierarchy named $\mathcal{H}_{\mathcal{O}}$. By convention, the hierarchy $\mathcal{H}_{\mathcal{O}}$ is upper bounded by the so-called root of the hierarchy, that subsumes every object belonging to $\mathcal{H}_{\mathcal{O}}$.

Furthermore, *O-subsumption = ordering + property sharing*, i.e. the O-subsumption relation must also allow property sharing in $\mathcal{H}_{\mathcal{O}}$. If $o \geq_{\mathcal{O}} o'$, then the value of any definitional attribute $a(o')$ can be deduced from $a(o)$. We will consider three main kind of property sharing rules:

- $\mathcal{S}s$ (standard property sharing): unless completed in o', the value of $a(o')$ is equal to the value of $a(o)$. This sharing rule is monotonic.
- $\mathcal{S}c$ (cumulative property sharing): the value of $a(o')$ is determined by collecting, e.g. set union, the values of a in the subsumers of o' in $\mathcal{H}_{\mathcal{O}}$. This sharing rule is also monotonic.
- $\mathcal{S}cnm$ (cumulative nonmonotonic property sharing): the value of the attribute

a depends on a collection of positive and negative values, where negative values are used to override or cancel positive values. Then, the value of a is computed according to a specific user-defined formula (an example is given in Section 3.5.2). This sharing rule is nonmonotonic.

2.5. O-SUBSUMPTION AND INHERITANCE

In object-based representations, the inheritance mechanism is primarily used for knowledge sharing and code factorization. In this section, we exhibit the links between specialization, i.e. the relation supporting inheritance, and O-subsumption. For now, we assume that the set of objects $\mathcal{O}$ is organized in an inheritance hierarchy $\mathcal{H}$ and that specialization is a monotonic operation in $\mathcal{H}$, i.e. only definitional attributes are considered and values cannot be overridden. Then, an object o O-subsumes an object o' in $\mathcal{H}$ if one of the following holds:

- $(C1)$: o is an ancestor of o' in $\mathcal{H}$, i.e. an inheritance path exists between o' and o, defined by an ordered set of objects $x_1, x_2, \ldots, x_n$, lying in $\mathcal{H}$, where $o' = x_1$, x_i ako x_{i+1} for every $i \in [1, n-1]$, and $x_n = o$.
- $(C2)$: for every definitional attribute $a(o)$ in o, there exists a corresponding definitional attribute $a(o')$ in o' such that $a(o) \geq_{\mathcal{A}} a(o')$.

If $(C1)$ holds, then $(C2)$ must hold too, otherwise $\mathcal{H}$ would not be consistent, i.e. one of the basic subsumption relations holding on definitional attributes would not be true. Conversely, if $(C2)$ holds, then $\mathcal{H}$ can be modified by addition of a new specialization relation between o' and o. However, this last operation is not equivalent to a classification operation: there is no guarantee that o is one of the most specific subsumers of o' in $\mathcal{H}$.

Thus, monotonic specialization is an O-subsumption relation. Conversely, O-subsumption gives rise to a hierarchy $\mathcal{H}$ allowing monotonic property sharing that relies on the following principles: property inheritance is uniform, and when o O-subsumes o' in $\mathcal{H}$, o' inherits every property associated with o. Thus, an object inherits a property p if it is O-subsumed by an object owning p. These principles are in agreement with those presented in Ducournau and Habib (1991).

Furthermore, the conditions $(C1)$ and $(C2)$ are equivalent if a strong hypothesis is satisfied, namely monotonic specialization for definitional attributes. If nonmonotonic specialization is allowed for definitional and nondefinitional attributes, then the two conditions are no longer equivalent. Moreover, if nondefinitional slots are considered, then a new definition of O-subsumption should take into account rules for checking the specialization of methods and reflexes.

2.6. STRUCTURAL OBJECTS AND CO-SUBSUMPTION

A composite object is an object whose parts are represented by other objects. The part-whole or composition relation, *part-of,* and its inverse relation, *composed-of*, are important basic relations with many interpretations. In the study presented in Markowitz, Nutter and Evens (1992), relations are divided into three main categories: queueing, similarity and part–whole relations. The first express orders or

sequences; the second expresses correspondence between referents; and the third has four subrelations, namely functional composition, member-set, subset-set and slice. In the study presented in Winston, Chaffin and Herrmann (1987), the part–whole relation has six main subrelations: functional composition, member-collection, portion-mass, stuff-object, feature-activity and place-area.

In the following, we will restrict our study to functional composition in the domain of structured objects: a part is in a specific spatial position with respect to the whole and it can be disconnected from the whole. A structured object is supposed to be represented by a graph whose vertices and edges are labeled.†

The *co-subsumption* relation describes structural composition as follows: a structured object O1 *co-subsumes* a structured object O2 if O1 describes a part of the structure of O2. The co-subsumption relation can be likened to substructure/structure or structural inclusion. The co-subsumption is related to functional composition, to the subset/set relation if structured objects are considered as sets of vertices and edges, and, lastly, to the slice or portion-mass relations, because a part and the whole have the same type, namely structured objects.

We must make precise that the co-subsumption relation does not capture the general part–whole relation, but is restricted to inclusion of structured objects. Moreover, the subsumption and specialization formalisms do not apply to the general part–whole relation for at least two reasons. First, subsumption and specialization are reflexive relations used to compare objects of the same type. By contrast, the part–whole relation is not reflexive and it can be used to compare objects that do not have the same type, e.g. individual–collection. Second, property sharing in the part–whole relation is usually not oriented.

The co-subsumption relation is well suited to organize graphs from a subgraph/graph point of view. Let us suppose that a graph G = (V, E) is represented by a frame with two definitional attributes, vertices and edges. The value of these two attributes is respectively the set V of vertices and the set E of edges of G. Moreover, every vertex and every edge are represented by an object. We can give a formal definition of co-subsumption for graphs.

Definition ($\geq_{\mathcal{G}}$)
A graph G1 = (V1, E1) *co-subsumes* a graph G2 = (V2, E2) if:

- A subgraph G = (V, E) of G2 is isomorphic to G1.
- $\forall v \in V1$, $v(G1) \geq_{\mathcal{V}} v(G)$, where $\geq_{\mathcal{V}}$ is a partial ordering depending on the type of the vertex v, v(G1) being the vertex in G1 corresponding to the vertex v(G) in G through the isomorphism.
- $\forall e \in E1$, $e(G1) \geq_{\mathcal{E}} e(G)$, where $\geq_{\mathcal{E}}$ is a partial ordering depending on the type of the edge e, e(G1) being the edge in G1 corresponding to the edge e(G) in G through the isomorphism.

The co-subsumption relation is a special form of O-subsumption: $\mathcal{O}$ is a set of graphs and $\mathcal{A}$ is the set of definitional attributes {vertices edges}. The co-subsumption is a preordering that organizes graphs in a hierarchy, where a

† A formal description of structured objects can be found in Shapiro and Haralick (1981) or in Vosselman (1990).

subgraph g co-subsumes a graph g′ including g. Note that in this particular case, the value of a(g) (a ∈ {vertices edges}) is a set included in the value of a(g′). We will see in Section 3.4.4 how the co-subsumption relation can be adapted to molecular graphs. Lastly, let us mention that co-subsumption for graphs can be viewed as an instance–instance relation and can be contrasted with specialization or subsumption, viewed as generic object–generic object relations.

3. Knowledge-based systems for organic synthesis planning

3.1. AN INTRODUCTION TO ORGANIC SYNTHESIS PLANNING

The object of chemical synthesis is to build up molecules. Once a target molecule has been chosen, the chemist searches for a *synthetic pathway* that is constituted by a sequence of reactions leading from simple or commercially available starting materials to the target molecule. The main approach used to find such a pathway is *analytical* or *retrosynthetic*: it takes as input a structured description of the target molecule and breaks it down into simpler structures called *precursors*. The target molecule is disconnected by a *transform*, i.e. the formal inverse of a chemical reaction (Corey, Long & Rubenstein, 1985). A precursor that is not identified as a starting material is in turn broken down into simpler precursors, still using a transform. This process generates a *retrosynthetic tree* of molecules, whose root is the target molecule and leaves are starting materials. From a computational point of view, the retrosynthetic approach is a goal-directed problem-solving process, related to problem reduction and AND/OR graphs (Nilsson, 1980).

The elaboration of a synthetic pathway requires a large amount of chemical knowledge because a single generic transform can have hundreds of specific cases. It also depends on chemical practice that cannot be defined by simple rules. Because of this great richness of information, there is a danger of generating arbitrarily large retrosynthetic trees. Prevention of this combinatorial explosion requires the use of synthetic strategies. There are five main types of strategies (Corey *et al.*, 1985; Corey & Cheng, 1989). The transform-based strategy identifies a powerful simplifying transform that is well suited to modifying the current target molecule. The functional group-oriented strategy supposes that transforms are applied only if the target molecule includes some specific substructures named *functional groups*. A functional group determines a chemical function—the *functionality* of the molecule—and usually characterizes the behavior of a family of molecules, e.g. it conditions the application of transforms. The structure-goal strategy identifies a potential starting material that can lead to the structure of the target molecule. The topological strategy identifies one or more bonds whose disconnection can lead to major molecular simplifications. The stereochemical strategy can reduce the stereochemical complexity of the target molecule. Two or more of the above strategies can be used concurrently to develop simple synthetic pathways. Lastly, let us mention that the formalization of synthetic strategies is far from being complete and must still be improved. This is one of the most serious limitations to the capabilities of knowledge-based systems for organic synthesis planning.

In our case, we will mainly use a retrosynthetic approach relying on a functional group-oriented strategy, i.e. the selection of valid transforms depends on the

perception of functional groups lying in the target molecule. Thus, functional groups are of primary importance for the categorization of molecular structures.

3.2. COMPUTER-AIDED SYNTHESIS SYSTEMS

The retrosynthetic approach has been used as a guideline for the construction of a series of computer systems, usually called *computer-aided organic synthesis* systems or CAOS systems (Corey *et al.*, 1985; Barone & Chanon, 1986). The main goal of these systems is to assist the chemist building a synthetic pathway. Most of the existing CAOS systems are based on procedural representations, and their work can be likened to the selection-choice-activation cycle of an inference engine in a rule-based expert system. Given the structure of a target molecule and a retrosynthetic approach, the system searches for appropriate transforms to be applied to the structure, selects and then applies the valid transforms to generate a level of precursors. The chemist communicates with the system through the use of interactive computer graphics. First, the chemist draws a target molecule. The system stores information about the position, type and connection of each atom and bond in the molecule, in arrays of numbers called *connection tables.* The system uses this basic information about atoms and bonds to assemble a body of chemical knowledge about the target. This process is called the *perception* of the target molecule. The system checks the environment of each atom, perceives functional groups, synthetic significant cycles and stereochemistry. The information resulting from perception is stored by the system to be used during the process of transform selection. The representation of a transform consists of a series of items reflecting the scope and limitations of the transform, e.g. a utility rating of the transform in standard synthetic contexts, a list of key functional substructures conditioning the execution of the transform. A transform is selected on the basis of the comparison between its characteristics and those of the target molecule resulting from perception. When a transform has been chosen and applied, precursors are generated and displayed in a retrosynthetic tree, from which the chemist can choose any structure for further decomposition. In this usual interactive retrosynthetic approach, the chemist is responsible for choosing a strategy when several strategies are available by the system, for controlling and completing the evaluation of the results and for choosing one or more precursors to be processed in the next step of the synthesis planning. The system is responsible for computing and applying the set of valid transforms, for displaying and for evaluating the resulting precursors. This evaluation consists in eliminating redundancies, i.e. similar precursors, chemically unfeasible and unstable precursors, and in sorting precursors for further processing according to chemical preference criteria.

Procedural computer-aided synthesis systems are usually efficient, but they present some limitations. The manipulation of molecular structures is based on the use of a programming language. First, this language is not easily and directly accessible to the chemist. Second, situations and actions are described by formulas including keywords that can be ambiguous, e.g. the different neighbors of an atom can be very hard to distinguish. Third, as usual with procedures, it is difficult to maintain the coherence and to control the redundancy and the evolution of knowledge embodied in procedures. The last point is important because chemical knowledge and practice evolve everyday. Therefore it could be worth automatically

decoding the available knowledge in order to update and to reuse it in different contexts or systems.

Besides procedural systems, a few rule-based systems have been designed (Napoli, Laurenço & Heitz, 1986; Wipke & Dolata, 1986). However, the representation formalisms on which these systems rely—propositional logic, predicate logic and pro-„formalisms on which these systems rely—propositional logic, predicate logic and pro-duction rules—are not well-suited to the design of organic synthesis planning systems.

3.3. ORGANIC SYNTHESIS PLANNING REVISITED

The primary task of a CAOS system is to give assistance to a chemist planning the synthesis of a new organic molecule. The system must be able to handle chemical objects as well as synthetic pathways. Synthesis planning can be seen as a process evolving from an initial state, the target molecule, towards a final state, the starting materials, passing through a list of intermediate states constituted of sets of atoms and bonds. From the users' viewpoint, there is no difference between our approach and the classical one, i.e. in both cases, the model of synthesis planning is based on the simulation of chemical practice. However, the most noticeable difference can be seen from the software engineers' and expert chemists' viewpoints, i.e. it lies in the design of the knowledge base and the choice of the problem-solving schemes.

In our view of synthesis planning, every chemical object is described by an abstract object and implemented by a frame (Napoli, 1990). Transforms are at present implemented as methods attached to frames representing specific functional substructures. All generic and individual frames describing chemical entities are manipulated by classification-based reasoning. Solving a synthesis planning problem relies on the retrieval of functional substructures lying in a target molecule, say TM. To retrieve this information, the system uses a specific classification process involving molecular structures and co-subsumption. The classification of TM produces a set of valid transforms that can be applied to TM. A transform is selected and applied to TM, generating a set of precursors. The synthesis planning problem is solved when a synthetic pathway leading from the target molecule TM to precursors known as starting materials is found. In this approach, the emphasis is placed on the description of molecular structures and functional substructures: the representation of transforms is no longer central within the whole process of representation, as is the case in procedural systems.

In the following, we detail the implementation of chemical concepts using the formalism of an abstract object-based representation formalism.

3.4. AN OBJECT-BASED REPRESENTATION OF CHEMICAL OBJECTS

3.4.1. Atoms

A molecular structure can be seen as a graph whose vertices and edges correspond to the atoms and to the bonds of the molecular structure. The generic atom is represented by the `Atom` frame (see Figure 1). It includes the attributes `valence`, `atomic-number`, `charge`, `incident-bonds` (i.e. the list of bonds that are incident to the current atom) and `structure`, i.e. the name of the molecular structure including the current atom. The value of `ako`, the specialization relation, is filled with the direct ascendant(s) of the frame in the inheritance hierarchy: `Atom` is a subframe of `Chemical-object`, the root of the inheritance hierarchy including chemical objects, also called the *chemical taxonomy* or $\mathcal{CT}$ (see Figure 2). The `Atom`

```
(defmodel Atom
  (ako ($value Chemical-object))
  (valence ($a Integer)
           ($interval [1 6]))
  (atomic-number ($a Integer)
                 ($interval [1 103]))
  (charge ($a Integer)
          ($domain [-2 -1 0 1 2])
          ($value 0))
  (incident-bonds ($list-of Bond)
                  ($inverse-link ends)
                  ($value nil))
  (structure ($a Structure)
             ($inverse-link atoms)
             ($value nil))
  (degree ($method +degree))
  (neighbors ($method +neighbors))
  (creation ($method make-atom))
  (subsumep ($method a-subsumep))
  (matchp ($method a-matchp)))
```

FIGURE 1. The `Atom` frame describes the generic atom. A frame is defined with the `defmodel` primitive. It is composed of slots that can be attributes or methods. The type of an attribute is introduced by the facets `$a` or `$list-of`; type restrictions are indicated by `$domain` and `$interval`, and values are introduced by the `$value` facet.

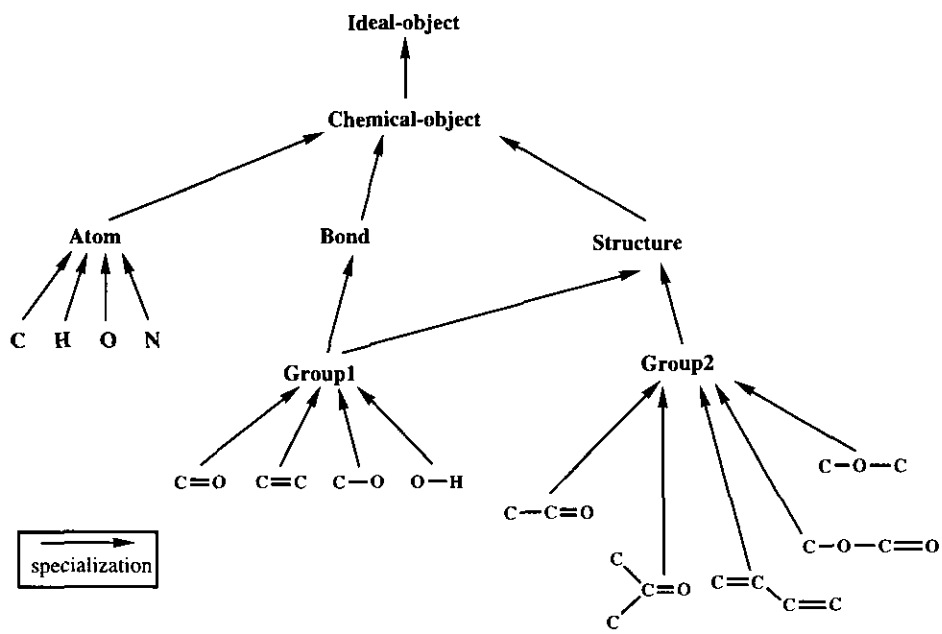

FIGURE 2. Basic chemical objects are organized in the chemical taxonomy $\mathscr{CT}$.

frame has several specializations that will not be detailed here. Actually, all atoms of the Mendeleev table involved in current applications are represented as frames.

The attributes `incident-bonds` and `structure` hold an `$inverse-link` facet. They define a relation between two frames and the value of the `$inverse-link` facet is filled with the name of the inverse relation.† The `incident-bonds` attribute defines a relation between `Atom`, the *domain* of the relation, and `Bond`, the *range* of the relation, the name of the inverse relation being `ends`. The `structure` attribute defines a relation between `Atom` and `Structure`, with `atoms` as inverse relation: for every atom, the value of `structure` is automatically filled with the name of the structure to which the atom belongs.

The `Atom` frame has five methods whose selectors are `creation`, `degree`, `neighbors`, `subsumep` and `matchp`. Methods are introduced by the special `$method` facet and are activated by message sending. The name of the slot corresponds to the selector of the method. The `creation` method is used to create instances. The `degree` method is used to check the structural consistency of the environment of every atom in a structure. It computes the number of virtual bonds that can be accepted by an atom, i.e. the more the degree is high (up to the value of `valence`), the less the atom is constrained. The `neighbors` method is used to find the atoms bonded with the current atom (using the value of the `incident-bonds`).

The `subsumep` method checks that an atom is more *general* than another atom, and the `matchp` method checks the equivalence of two atoms:

- An atom `a1` *subsumes* an atom `a2` if `a1` and `a2` have the same atomic number (`atomic-number`), and if the degree of `a1` is greater than the degree of `a2`.
- An atom `a1` *matches* an atom `a2` if `a1` and `a2` have the same atomic type and the same degree.

The co-subsumption relation, as defined in Section 2.6, depends on two partial orderings. Considering that a molecular structure is represented by a graph, the `subsumep` method in `Atom` can be seen as an implementation of the partial ordering $\geq_{\mathcal{V}}$ depending on the type of vertices (see also the definition of $\geq_{\mathcal{MG}}$, Section 3.4.4). This distributed implementation of the co-subsumption relation contrasts with the classical single built-in subsumption function usually used in terminological logics (Nebel, 1990).

3.4.2. Bonds

The generic bond is described by the `Bond` frame (see Figure 3). The `order` attribute denotes the bond type, i.e. simple (`order=1`, denoted by the symbol "–"), double (`order=2`, denoted by the symbol "="), triple (`order=3`, denoted by the symbol "≡") and aromatic (`order=1.5`, denoted by the symbol "%"). The `ends` attribute specifies the atoms located at the ends of the bond.

As said in the preceding section, the frames `Atom` and `Bond` are related through the attributes `incident-bonds` and `ends`, e.g. whenever a new bond `b` between atoms `a1` and `a2` is built, the value of `ends` in `b` is filled with the pair `(a1 a2)`, and

† Whenever the value of one attribute is modified, the value of the related attribute is consequently modified. The parallel updating of the values of related attributes is realized by reflexes (Masini *et al.*, 1991).

```
(defmodel Bond
  (ako ($value Chemical-object))
  (order ($a Number)
         ($domain [0 1 1.5 2 3])
         ($value 0))
  (ends ($list-of Atom)
        ($inverse-link incident-bonds)
        ($value nil))
  (structure ($a Structure)
             ($inverse-link bonds)
             ($value nil))
  (creation ($method make-bond))
  (subsumep ($method b-subsumep))
  (matchp ($method b-matchp)))
```

FIGURE 3. The Bond frame.

in parallel, the value of incident-bonds in a1 and in a2 is filled with b. As this is the case for Atom, the structure attribute in Bond records the name of the molecular structure including the bond.

The Bond frame has three methods whose selectors are creation, subsumep and matchp. The first is used to create an instance of Bond, given two atoms and an order, provided that the conditions of bond formation are met. The subsumep and the matchp methods implement the following definitions;

- A bond b1 *subsumes* a bond b2 if b1 and b2 have the same order and if the atoms ending b1 subsume the corresponding atoms ending b2 (according to a standard ordering of atoms). Again, the subsumep method in Bond can be seen as an implementation of the partial ordering $\succeq_{\mathscr{E}}$ depending on the type of edges in the co-subsumption relation (see Section 2.6).
- A bond b1 *matches* a bond b2 if b1 and b2 have the same order and if the atoms ending b1 match the corresponding atoms ending b2.

The Bond frame has only one important specialization, namely Group1, that describes generic bonds having special reactive properties, called *simple functional groups,* e.g. carbon–carbon bonds such as C=C and C≡C, heteroatom–carbon bonds† such as C–C, C=O or C–X (the symbol X denotes the family of halogens), heteroatom–heteroatom bonds such as O–O and heteroatom–hydrogen bonds such as O–H. The Group1 specialization does not have an actual chemical semantics, but is rather used as an implementation trick for matching simple and complex functional groups in a molecular structure. The subsumep method is overridden in Group1: a simple group g1 subsumes a bond b1 if g1 and b1 have the same order and if the atoms ending g1 are of the same atomic type as the atoms ending b1.

There is no other specialization of Bond. In particular, generic frames describing simple, double, triple and aromatic bonds are not defined. The main reason is that it

† Every atom, except carbon and hydrogen, is an heteroatom.

```
(defmodel Structure
  (ako ($value Chemical-object))
  (atoms ($list-of Atom)
         ($inverse-link structure)
         ($value nil))
  (bonds ($list-of Bond)
         ($inverse-link structure)
         ($value nil))
  (transfos+ ($list-of Symbol)
             ($value nil))
  (transfos- ($list-of Symbol)
             ($value nil))
  (transfos ($method comp-transfos))
  (creation ($method make-structure))
  (+bond ($method add-bond))
  (-bond ($method rem-bond))
  (subsumep ($method s-subsumep))
  (matchp ($method s-matchp)))
```

FIGURE 4. The Structure frame.

is costless to update the order value of a bond b evolving in a synthesis, rather than moving b within the network of instances.

3.4.3. Molecular structures

A molecular structure is a composition of atoms and bonds, described by the Structure frame (see Figure 4). The atoms and bonds attributes hold respectively the list of the atoms and the list of the bonds of the structure. Every atom is an instance of Atom, while every bond is an instance of Bond. The Structure frame has two other attributes, transfos+ and transfos−, and six methods whose selectors are transfos, creation, +bond, −bond, subsumep and matchp. The attributes transfos+, transfos− and the transfos method are used in the classification-based process producing synthetic pathways. Their roles will be made explicit in Section 3.5.2.

To create an instance of Structure, a creation message is sent to the Structure frame with a *connection list* as argument. The connection list is built during the drawing of the structure. The atoms of the structure are created according to their type, and the bonds are created according to their order and to their extremities. The value of the structure attribute is filled in every bond and atom with the name of the newly created structure.

There are two *surgery* operations to alter a molecular structure: adding and removing a bond. All surgery operations are based on the following principle: when a structure ms is changed, a new structure is built, depending on the atoms and bonds of the old structure ms. Thus, a trace of every operation altering a chemical object is kept, and the full history of the state of objects involved in a synthesis is known. The sending of a −bond message with argument b to ms breaks b in ms.

One or two new molecular structures, ms1 and ms2, are built, depending on the new connectivity† of the atoms of ms after the breaking of b. The sending of a +bond message, with an order n and two atoms a1 and a2 as arguments, creates a new bond, say b, of order n, between a1 and a2. The formation of b includes three main steps: verifications of the bond formation conditions, construction of b and construction of a new molecular structure.

From a retrosynthetic point of view, the initial state of a synthesis is described by a target molecule and the final state by available starting materials. The intermediate steps of the synthesis result from the application of chemical transforms to a *reactor*, i.e. a set of atoms and bonds at a given time t. A synthesis can then be seen as a temporal sequence of reactors: the reactor existing at time $t+1$ depends on the reactor existing at time t and on the transform applied. A tool describing and handling the temporal evolution of a set of objects has been developed, but this tool must still be improved and will not be detailed here (see Napoli, 1992).

3.4.4. The molecular co-subsumption and the functional partonomy

In the Structure frame, the methods subsumep and matchp are used to compare molecular structures, i.e. the first method implements the co-subsumption of molecular structures, while the second implements the matching of molecular structures. Following the lines of Section 2.6, let $\mathscr{A}$ denote the set of the atomic types (Atom and its specializations) and $\geq_{\mathscr{A}}$ the subsumption relation defined on $\mathscr{A}$; let $\mathscr{B}$ denote the set of bond types (Bond and its specializations) and $\geq_{\mathscr{B}}$ the subsumption relation defined on $\mathscr{B}$.

Definition ($\geq_{\mathscr{MS}}$ and $=_{\mathscr{MS}}$)

A molecular structure ms1 = (A1, B1) *co-subsumes* a molecular structure ms2 = (A2, B2) if:

- A substructure ms = (A, B) of ms2 is isomorphic to ms1, i.e. the subgraph associated with ms is isomorphic to the graph associated with ms1.
- For every a ∈ A1, a(M1) $\geq_{\mathscr{A}}$ a'(M), where a(M1) denotes the atom (vertex) a in ms1 corresponding to the atom a'(M) in ms (⊆ms2), with respect to the isomorphism.
- For every b ∈ B1, b(M1) $\geq_{\mathscr{B}}$ b'(M), where b(M1) denotes the bond (edge) b in ms1 corresponding to the bond b'(M) in ms (⊆ms2), with respect to the isomorphism.
- ($=_{\mathscr{MS}}$): a molecular structure ms1 *matches* a molecular structure ms2 if there exists a structural isomorphism between the graph associated with ms1 and the graph associated with ms2, and if every atom and bond in ms1 match the corresponding atom and bond in ms2.

The implementation of the subsumption relation for chemical objects is distributed among atoms, bonds and molecular structures. Moreover, a general method subsumep always returning nil is attached to Chemical-object, i.e. two incomparable objects cannot subsume each other. The general subsumep method is then overriden in Atom, Bond, Group1 and Structure.

† Actually, the transitive closures of the incident-bonds attributes associated with the extremities of b, say a1 and a2, are used to determine the new molecular structures ms1 and ms2. A new structure is built only if it contains at least one bond.

In Section 3.4.2, we introduced simple functional groups. More generally, a functional group is a composition of atoms and bonds, and is represented as a molecular structure. Functional groups are divided into two main categories: simple groups or single bonded groups, specializations of `Group1`; and complex groups including at least two bonds, specializations of `Group2`. A complex group can be a linear structure, such as the alcohol group `C–O–H`, or a more complex structure such as carboxyls or esters. The frames `Group1` and `Group2` are both specializations of `Structure`. Thus, it is possible for a single group to co-subsume a molecular structure in the following way: *a simple group* `g1` *co-subsumes a molecular structure* `ms` if `ms` includes a bond `b` subsumed by `g1`.

The practical use of molecular co-subsumption is to perceive functional substructures in a molecular structure `ms` that determine the transforms modifying `ms`. This perception relies on a classification process in a specific hierarchy named the *functional partonomy* or $\mathcal{FP}$. Contrasting the chemical taxonomy, the functional partonomy† reflects the co-subsumption relations holding between functional substructures, i.e. functional groups depending on `Group1` and `Group2` (see Figure 5).

On the one hand, the specialization relation is used for general frame organization, code factorization and knowledge sharing in the chemical taxonomy. On the other hand, the co-subsumption relation organizes functional substructures in the functional partonomy according to structural inclusion. The functional partonomy is then the basis of a classification-based process producing synthetic pathways. The construction of the partonomy $\mathcal{FP}$ is dynamic and results from the application of the classification-based cycle using molecular co-subsumption on functional substructures.

3.4.5. Transforms

A transform can be seen as a process by which a target molecule is split into precursors. A transform can require light or pressure conditions, or other more complicated reactional conditions. For the sake of simplicity, reactional conditions will be ignored in the following. Transforms can be divided into *simplifications* and *exchanges*. The first are used to decompose a structure, e.g. the Wittig transform is used to break down the functional group `C═C` into two precursors: `C═C` ⇒ `C–X+C═O`.

Exchange transforms are used to modify the functionality of a molecular structure, replacing a functional group with another, e.g. the replacement of the alcohol function `C–O–H` with the ketone function `C═O`. The modification of the functionality of a molecular structure `ms` can be seen as a preparation of `ms` allowing the application of a simplification transform to the structure of `ms`.

From an implementation point of view, a transform `T` is described by a method associated with the object describing the particular substructure that is decomposed or modified by `T`, e.g. the method implementing the Wittig transform is attached to the functional group `C═C` (see Figure 6). Note that the representation of a transform is nothing more than an ordered sequence of elementary surgery operations `+bond` and `−bond`.

† We borrow the word *partonomy* from Tversky and Hemenway (1984).

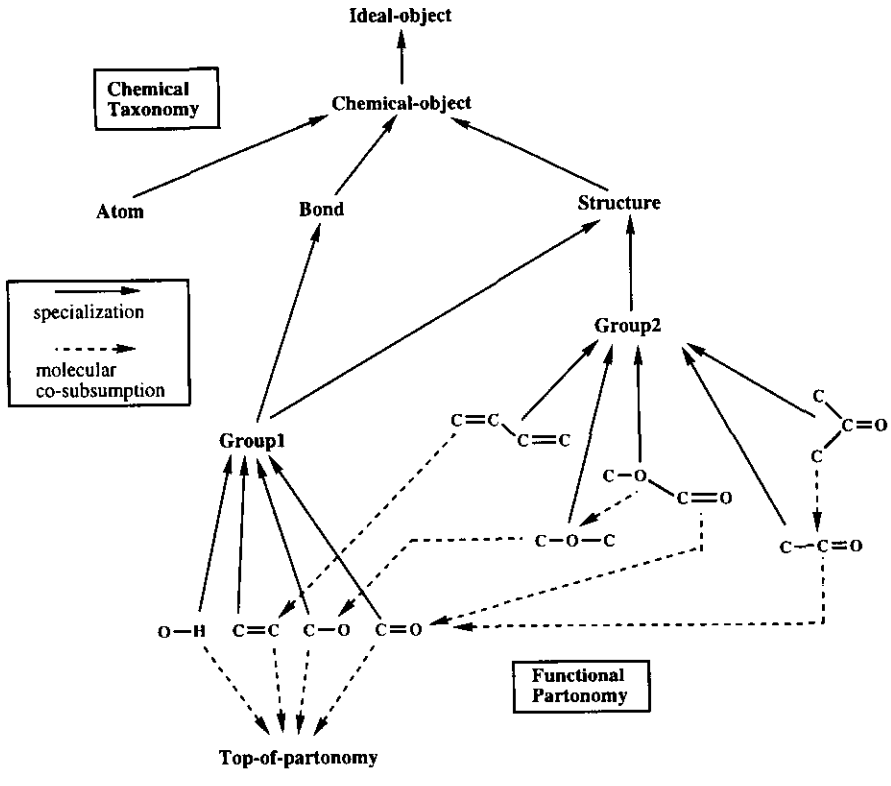

FIGURE 5. Functional groups are divided into simple and complex groups. A simple group co-subsumes the complex groups in which it is included. The functional partonomy $\mathcal{FP}$, whose root is `Top-of-partonomy`, reflects the co-subsumption relations existing between functional substructures. The partonomy $\mathcal{FP}$ is *orthogonal* to the chemical taxonomy $\mathcal{CT}$.

```
method wittig (target C1=C2)
  C1=C2 is a copy of C=C lying in the target molecule
  A reactor memorizes the set of atoms and bonds of target
  reactor ← (concatenate (atoms target) (bonds target))
  Deletion of C1=C2 in target
  reactor ← (-bond target C1=C2)
  Construction of a new bond C1=O
  reactor ← (+bond target C1 (creation O) 2)
  Construction of a new bond C2-X
  reactor ← (+bond target C2 (creation X) 1)
  Construction of precursors
  precursors ← (creation (connected-parts reactor))
```

FIGURE 6. A Lisp-like description of the Wittig transform: C=C ⇒ C-X+C=O. The method implementing the transform is associated with the generic functional group C=C. The form `(attribute frame)` returns the value of `attribute` associated with `frame`.

3.5. CLASSIFICATION-BASED SYNTHESIS

3.5.1. The classification cycle for organic synthesis planning

The retrosynthetic approach relies on the identification of functional substructures in a target molecule TM. This identification is a consequence of the classification of TM in the functional partonomy $\mathcal{FP}$: the functional substructures of TM are the most specific co-subsumers of TM in $\mathcal{FP}$. The computation of the set $\mathcal{T}$ of potential transforms modifying TM depends on a specific property sharing rule $\mathcal{S}_{\mathcal{FP}}$ associated with molecular co-subsumption. This problem-solving process, called *classification-based synthesis,* can be seen as an instantiation of the classification-based cycle (see Section 2.3):

- *Instantiation*: the target molecule is described by a new instance TM of the `Structure` frame.
- *Classification*: TM is classified in the functional partonomy $\mathcal{FP}$. The co-subsumers of TM determine the functional substructures of TM to which transforms can be applied. These transforms are either attached to the most specific co-subsumers of TM or shared with the ancestors of these co-subsumers in $\mathcal{FP}$.
- *Operations*: the set $\mathcal{T}$ of valid transforms that can be applied to TM is computed according to the property sharing rule $\mathcal{S}_{\mathcal{FP}}$. A transform is chosen and applied to break the target molecule into precursors. These precursors become new targets if they are not recognized as readily available starting materials.

The temporal evolution of classification-based synthesis gives birth to a retrosynthetic tree: the root of the tree is the target molecule, while the nodes of the tree are intermediate precursors and the leaves are known starting materials.

The classification-based synthesis process relies on a double use of the classification-based reasoning process. On the one hand, the retrosynthetic tree associated with the target molecule TM depends on the classification of TM in the functional partonomy $\mathcal{FP}$. The co-subsumers of TM own the transforms that can be applied to modify TM. This first view is related to the problem-solving capabilities of classification-based reasoning. On the other hand, the classification process organizes functional substructures in $\mathcal{FP}$, according to molecular co-subsumption. This second view is related to the conceptual clustering capabilities of classification-based reasoning.

3.5.2. Property sharing in the functional partonomy

We are going to detail the rule $\mathcal{S}_{\mathcal{FP}}$ controlling the sharing of transforms in $\mathcal{FP}$. Recall that two attributes named `transfos+` and `transfos−`, as well as a method named `transfos`, are associated with the `Structure` frame (see Figure 4). In the following, `transfos+(FS)` and `transfos−(FS)` respectively denote the value of the attributes `transfos+` and `transfos−` associated with the frame describing the functional substructure FS, while `transfos(FS)` denotes the value delivered by the `transfos` method when applied to FS.

Given a functional substructure FS, `transfos+(FS)` is filled with the set (of the names) of the transforms modifying FS. In a dual way, `transfos−(FS)` is filled with the set of transforms that modify the co-subsumers of FS in $\mathcal{FP}$, but that

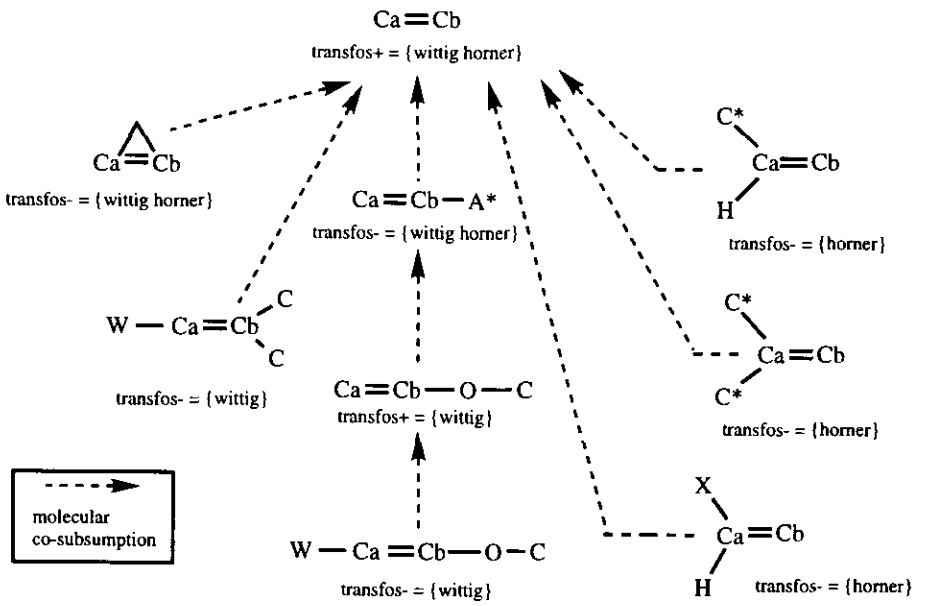

FIGURE 7. Functional substructures associated with the Wittig and the Horner transforms. The two transforms have the same profile: $C_a{=}C_b \Rightarrow X{-}C_a{+}C_b{=}O$, but they are not applicable in the same contexts. This example shows the importance of chemical contexts.

cannot be applied to FS itself. In other words, transfos− is used to override or cancel values associated with transfos+.

For example, in Figure 7, the value transfos+(C=C) is equal to {wittig horner}, i.e. the Wittig and the Horner transforms can currently be used to break down C=C in a target molecule. By contrast, the value transfos−(W–C=C(–C)–C) is also equal to wittig, i.e. C=C can no longer be broken down by the Wittig transform when C=C lies in the chemical context† W–C=C(–C)–C.

A transform T is *positive* for a functional substructure FS if T ∈ transfos+(FS) or if T ∈ transfos+(FS′), where FS′ is a co-subsumer of FS in $\mathscr{FP}$. In the same way, a transform T is *negative* for FS if T ∈ transfos−(FS) or if T ∈ transfos−(FS′), where FS′ is a co-subsumer of FS in $\mathscr{FP}$. At last, a transform T is *inheritable* for FS if T ∈ transfos(FS): the value transfos(FS) can be interpreted as the set of the valid transforms that can modify FS. This value is calculated by the formula:

$$\texttt{transfos(FS)} = \texttt{transfos+(FS)} \cup [\texttt{transfos(SUPER)} - \texttt{transfos-(FS)}] \quad (\mathscr{S}_{\mathscr{FP}})$$

The symbol ∪ denotes set union and SUPER denotes the most specific co-subsumer of FS in $\mathscr{FP}$. This most specific co-subsumer is obtained by a linearization‡ of the set of the co-subsumers of FS in $\mathscr{FP}$. Thus, the most specific positive or negative information is retrieved by the rule $\mathscr{S}_{\mathscr{FP}}$. For example, the fact that the Wittig transform is positive for C=C–O–C overrides the fact that the same

† The symbol W is used to denote a family of specific substructures named *W-groups*. The notation $C_a{=}C_b(-C_c)-C_d$ is used to represent bondings greater than 2: C_b has three neighbors, C_a, C_c and C_d.

‡ The linearization of the set of co-subsumers relies on the linearization used to build the precedence list of a frame in an inheritance hierarchy, as presented in Ducournau and Habib (1991).

transform is negative for C=C-A* (A* denotes the generic heteroatom), because C=C-O-C is co-subsumed by C=C-A* in $\mathcal{FP}$ (see Figure 7). The rule $\mathcal{S}_{\mathcal{FP}}$ is then a kind of cumulative nonmonotonic property sharing (see Section 2.4).

A property sharing *conflict* may appear when a transform T ∈ transfos+(FS) ∩ transfos−(FS). In this case, T is no longer a valid transform that can be applied to FS.† For example, if FS1 and FS2 are two co-subsumers of FS in $\mathcal{FP}$ such that T ∈ transfos+(FS1), T ∈ transfos−(FS2) and FS1 ∩ FS2 ≠ ∅, then T cannot be applied to FS. On the other hand, if FS1 ∩ FS2 = ∅, then T can be applied to the structural part of FS matching FS1, because T ∈ transfos+(FS1). The fact that T ∈ transfos−(FS2) only forbids the application of T to the structural part of FS matching FS2.

From a practical point of view, the retrieval of the functional substructures lying in a target molecule TM is ordered. It begins with the retrieval of simple groups and then continues with the retrieval of complex groups including these simple groups. When TM includes more than one functional substructure FS of the same type, then every copy of FS determines a specific context. In this case, the value transfos(FS) must be calculated for every context and the selection of the transforms applicable to TM is made accordingly.

3.5.3. A simple example

The TM target molecule shown in Figure 8 has four simple co-subsumers in $\mathcal{FP}$, namely C-O, C=O, C=C and C-X. From the point of view of C-O and C=O, the ester

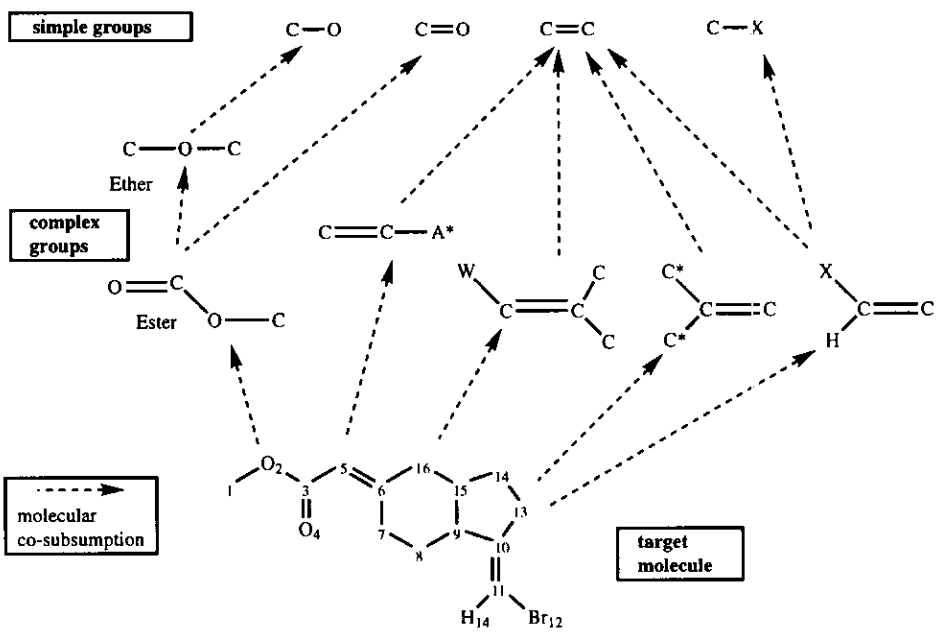

FIGURE 8. The target molecule TM and its co-subsumers in the functional partonomy. For a better reading, the symbol C has been omitted in the target molecule.

† Priority is given to negative information. This choice can also be likened to the choices made to handle inheritance exceptions in nonmonotonic inheritance theories (Ducournau & Habib, 1991).

group O=C-O-C co-subsumes the substructure O4=C3-O2-C1 and becomes the most specific co-subsumer in this branch of $\mathcal{FP}$. The simple group C-X co-subsumes the bond C11-Br12 because X denotes the generic halogen, and bromine (Br) is a member of the halogen group. The point of view of C-X can be combined with the point of view of C=C, actually the most interesting in this example. Four cases are to be considered because the C=C group is symmetric, i.e. Ca=Cb co-subsumes C5=C6, C6=C5, C10=C11 and C11=C10 (carbons of the functional substructure C=C are indexed by letters, and carbons of the target molecule are indexed by integers):

- Ca=Cb co-subsumes C5=C6: the functional substructure W-C=C(-C)-C co-subsumes the substructure C1-O2-C3(=O4)-C5=C6(-C7)-C16 (the ester group C1-O2-C3(=O4) bonded to C5 is a special kind of W-group and thus is co-subsumed by W). The Wittig transform is negative for W-C=C(-C)-C, and then negative for the target molecule. On the other hand, the Horner transform is positive and can be applied to produce two precursors (case (1), Figure 9).
- Ca=Cb co-subsumes C5=C6: the substructure C*-C(-C*)=C co-subsumes C7-C6(-C16)=C5. This time, the Horner transform is negative but the Wittig transform is positive (case (2), Figure 9).
- Ca=Cb co-subsumes C10=C11: the substructure C=C-A* co-subsumes C10=C11-Br12. The Wittig and Horner transforms are negative for C=C-A* and cannot be applied to TM (case (3), Figure 9).

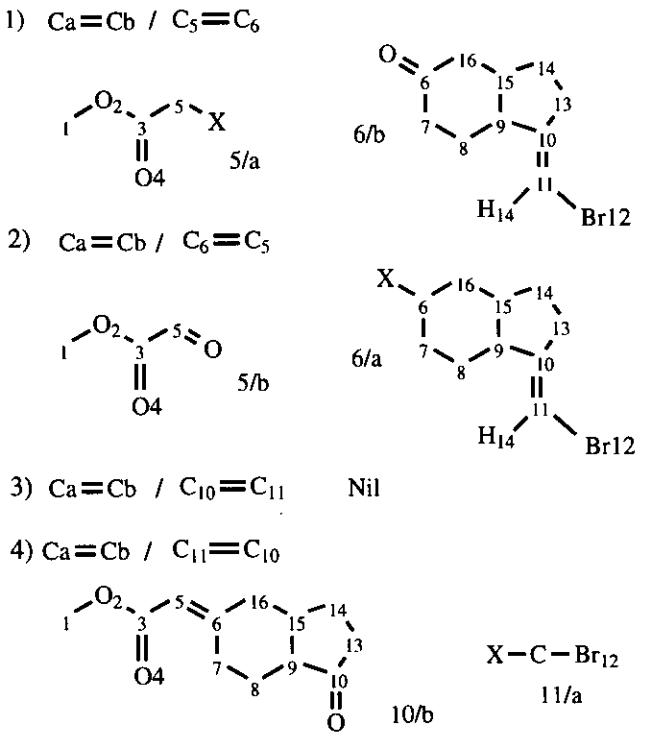

FIGURE 9. The four application cases of the Wittig and Horner transforms.

- Ca=Cb co-subsumes C11=C10: the substructure X-C(-H)=C co-subsumes Br12-C11(-H14)=C10. The Horner transform is negative for the substructure X-C(-H)=C, and then negative for TM. However, the Wittig transform is positive and can be applied to produce two precursors (case (4), Figure 9).

In case (3), neither the Wittig nor the Horner transforms are applicable. However, in the three other cases, the Horner transform (case (1)) and the Wittig transform (cases (2) and (4)) are applicable. The precursors that are produced are all chemically acceptable. Moreover, precursors 5/a (case (1)) and 11/a (case (4)) are commercially available. Thus, in order to develop an economical retrosynthetic tree, it could be worth trying to simplify first precursor 6/b (case (1)) and/or precursor 10/b (case (4)).

3.5.4. A first analysis of object-based synthesis

As we said in the introduction, the system described above was especially built for validating new ideas in the field of computer-aided organic synthesis planning. The system being still a prototype, it is difficult to give an analysis of its performance, particularly in comparison to classical systems under development for a number of years. However, it is quite possible to give a first evaluation of the benefits resulting from the use of the object-based approach.

In classical CAOS systems, the emphasis is on the implementation of transforms. First, the chemist has to research and to prepare the available chemical knowledge (this is a long and difficult knowledge-acquisition task). Then he or she has to implement the collected knowledge with a programming language that is far from being the usual graphical language employed everyday. By contrast, we rely strictly on object-based principles for designing the chemical knowledge base: the main task is to represent molecular structures (or objects) first. The chemist still has to collect chemical knowledge but he or she is no longer constrained to enclose his or her knowledge within sequences of if-then-else rules. He or she has only to describe structural environments, an everyday task: the chemist draws functional substructures and associates with these structures positive and negative information related to transforms. In this way, chemical knowledge is more easy to encode, to read and to modify. It can be more easily controlled and updated by the chemist, because the system and the chemist both use the same "graphical" language (by extension, one can imagine methods for automatically controlling, evaluating and updating this chemical knowledge). Thus, the object-based approach has an important advantage from the point of view of knowledge expressiveness. This advantage can also be explained by arguments resulting from the declarative/procedural controversy (Winograd, 1975): the object-based approach is more declarative than the classical procedural approach and has, then, to be preferred.

Lastly, the search for the available chemical information, e.g. transforms, is more constrained in the object-based approach. When the co-subsumers of a target molecule are found in the functional partonomy, all potential transforms are known at the same time, using a property sharing rule. Thus, structural matching tests are done only one time for every transform. On the other hand, in the classical approach, structural matching tests must be done for every potential transform, possibly to discover that the transform is not applicable.

To summarize, the object-based approach associated with classification-based reasoning are promising tools, and one can expect that they will bring more efficiency and more economy in the chemical knowledge-acquisition, retrieval and evolution processes.

3.5.5. Related works

Only a few systems for organic synthesis planning rely on an object-based approach: among them, let us mention RESYN (Vismara, Regin, Quinqueton, Py, Laurenço & Lapied, 1992), and PPA, an early work, presented in Wipke and Rogers (1984) „& Lapied, 1992), and PPA, an early work, presented in Wipke and Rogers (1984) where the object-oriented language Simula was used to implement a parallel subgraph search, but where molecular graphs were still represented as connection tables.

The organization of molecular structures according to a substructure/structure point of view has been used in some systems, e.g. the DARC system for chemical information storage and retrieval (Dubois, 1976). The design of a general knowledge base for labeled graphs and molecular structures is presented in Levinson (1985) and Wilcox and Levinson (1986), while a framework for building and refining knowledge bases including molecular structures and chemical reactions, using induction and conceptual clustering methods, is described in Rose and Gelernter (1989) and in Gelernter, Rose and Chen (1990).

Lastly, one can note that problem-solving approaches involving part–whole relations can be found in the context of scene analysis and object recognition (Havens & Mackworth, 1987; Matsuyama & Hwang, 1990), and in the modeling of topic hierarchies in semantic networks (Schubert, Papalaskaris & Taugher, 1987).

4. Conclusion

In this article, we have presented an object-based architecture for organic synthesis planning. Regarding classical organic synthesis planning systems, the basic ideas presented here are quite different and new, e.g. object-based representation of chemical objects and classification-based reasoning. We have explained the tactical point of view of organic synthesis planning, i.e. a transform can be applied to a molecular substructure if the substructure lies in a favorable context. Important questions related to the use of synthetic strategies still remain open (Napoli & Lieber, 1993), e.g. when more than one transform is available, as is the case in the example of Section 3.5.3, which transform must be chosen, and more generally, what is the best strategy that the system must follow?

In parallel, object-based representations of transforms and retrosynthetic trees are currently being investigated. A transform may decrease or increase molecular complexity and for this reason may play quite different roles in the retrosynthetic approach, i.e. exchanges, removals and additions. The O-subsumption relation can then be used to categorize transforms according to viewpoints describing these roles, making transforms accessible for a variety of strategic reasons.

Currently, the authors are working on the design of knowledge-based systems for organic synthesis planning within the research group GDR 1093 of the CNRS (Centre National de la Recherche Scientifique). Two object-based systems, including RESYN (Vismara *et al.*, 1992) and the system described in this paper, have been built by members of the group and will be used as a basis for further developments.

The goal of the research group GDR 1093 is to model analogical reasoning, tactical and strategic knowledge in organic chemistry, in order to implement a new kind of organic synthesis planning system, able to handle strategies as well as tactics.

The author would like to thank the anonymous referees whose comments and suggestions have greatly contributed to improve the final version of the paper.

References

ATTARDI, G. (1991). An analysis of taxonomic reasoning. In M. LENZERINI, D. NARDI & M. SIMI Eds. *Inheritance Hierarchies in Knowledge Representation and Programming Languages*, pp. 29–49. Chichester: John Wiley.

ATTARDI, G., CORRADINI, A., DIOMEDI, S. & SIMI, M. (1986). Taxonomic Reasoning. *Proceedings of the 7th European Conference on Artificial Intelligence (ECAI'86) Brighton, England*, pp. 236–245.

BAADER, F. HOLLUNDER, B., NEBEL, B., PROFITLICH, H.-J. & FRANCONI, E. (1992). An empirical analysis of optimization techniques for terminological representation systems. *Proceedings of the Third International Conference on Principles of Knowledge Representation and Reasoning (KR'92)*, Cambridge, MA, USA, pp. 270–281.

BARONE, R. & CHANON, M. (1986). Computer-Aided Organic Synthesis (CAOS). In G. VERNIN & M. CHANON, Eds. *Computer Aids to Chemistry*, pp. 19–102. Chichester: Ellis Horwood.

BOBROW, D. & WINOGRAD, T. (1977). An Overview of KRL, a Knowledge Representation Language. *Cognitive Science*, **1**(1), 3–46.

CHANDRASEKARAN, B. & GOEL, A. (1988). From numbers to symbols to knowledge structures: artificial intelligence perspectives on the classification task. *IEEE Transactions on Systems, Man and Cybernetics*, **18**(3), 415–424.

CLANCEY, W. (1985). Heuristic classification. *Artificial Intelligence*, **27**, 289–350.

COREY, E. (1971). Computer-assisted analysis of complex synthetic problems. *Quarterly Review of the Chemical Society*, **25**, 455–482.

COREY, E. & CHENG, X. (1989). *The Logic of Chemical Synthesis*. New York: John Wiley.

COREY, E., LONG, A. & RUBENSTEIN, S. (1985). Computer-assisted analysis in organic synthesis. *Science*, **228**, 408–418.

DUBOIS, J. (1976). Ordered chromatic graph and limited environment concept. In A. BALABAN, Ed. *Chemical Applications of Graph Theory*, pp. 333–370. London: Academic Press.

DUCOURNAU, R. & HABIB, M. (1991). Masking and conflicts, or to inherit is not to own! In M. LENZERINI, D. NARDI & M. SIMI, Eds. *Inheritance Hierarchies in Knowledge Representation and Programming Languages*, pp. 223–244. Chichester: John Wiley.

FININ, T. (1986). Interactive classification: a technique for acquiring and maintaining knowledge bases. *Proceedings of the IEEE*, **74**(10), 1414–1421.

GELERNTER, H., ROSE, J. & CHEN, C. (1990). Building and refining a knowledge base for synthesic organic chemistry via the methodology of inductive and deductive machine learning. *Journal of Chemical Information and Computer Sciences*, **30**, 492–504.

GELERNTER, H., SRIDHARAN, N., HART, A., YEN, S., FOWLER, F. & SHUE, H. (1973). The discovery of organic synthetic routes by computer. *Topics in Current Chemistry*, **41**, 113–150.

GOMEZ, F. & SEGAMI, C. (1991). Classification-based reasoning. *IEEE Transactions on Systems, Man and Cybernetics*, **21**(3), 644–659.

GOTTLOB, G. (1987). Subsumption and implication. *Information Processing Letters*, **24**, 109–111.

HAVENS, W. & MACKWORTH, A. (1987). Representing and using knowledge of the visual world. In N. CERCONE & G. MCGALLA, Eds. *The Knowledge Frontier: Essays in the Representation of Knowledge*, pp. 429–450. New York: Springer-Verlag.

KACZMAREK, T., BATES, R. & ROBINS, G. (1986). Recent developments in NIKL. *Proceedings of AAAI'86,* Philadelphia, PA, USA, pp. 978–985.

KAINDL, H. (1994). Object-oriented approach in software engineering and artificial intelligence. *Journal of Object-Oriented Programming,* **6**(8), 38–45.

KARP, P. (1993). *The Design Space of Frame Knowledge Representation Systems.* Technical Note **520,** SRI International, Menlo Park, CA, USA.

LEVESQUE, H. & BRACHMAN, R. (1987). Expressiveness and tractability in knowledge representation and reasoning. *Computational Intelligence,* **3**(2), 78–93.

LEVINSON, R. (1985). *A self-organizing retrieval system for graphs.* Technical report AI-85-05, Artificial Intelligence Laboratory, The University of Texas at Austin, TX, USA.

LIPKIS, T. (1982). A KL-ONE classifier. In J. SCHMOLZE & R. BRACHMAN, Eds. *Proceedings of the 1981 KL-ONE Workshop,* pp. 126–143.

MARKOWITZ, J., NUTTER, J. & EVENS, M. (1992). Beyond is-a and part-whole: more semantic network links. *Computers & Mathematics with Applications,* **23**(6–9), 377–390.

MASINI, G., NAPOLI, A., COLNET, D., LÉONARD, D. & TOMBRE, K. (1991). *Object-Oriented Languages.* London: Academic Press.

MATSUYAMA, T. & HWANG, V. S.-S. (1990). *SIGMA: A Knowledge-Based Aerial Image Understanding System.* New York: Plenum Press.

NAPOLI, A. (1990). Using frame-based representation languages to describe chemical objects. *New Journal of Chemistry,* **14**(12), 913–919.

NAPOLI, A. (1992). *An object-based approach to computer-aided planning of organic syntheses.* Rapport de recherche 92-R-101, Centre de Recherche en Informatique de Nancy.

NAPOLI, A., LAURENÇO, C. & HEITZ, M. (1986). Synthèse automatique optimale en chimie organique. Actes des 62emes Journ1ees Internationales sur les Syst2emes Expert et leurs „organique. *Actes des 6èmes Journées Internationales sur les Systèmes Expert et leurs Applications, Avignon,* pp. 1219–1234.

NAPOLI, A. & LIEBER, J. (1993). Finding strategies in organic synthesis planning with case-based reasoning. In M. RICHTER, S. WESS, K.-D. ALTHOFF & F. MAURER, Eds. *Proceedings of the First European Workshop on Case-Based Reasoning (EWCBR'93), Kaiserslautern,* pp. 264–269.

NEBEL, B. (1990). *Reasoning and Revision in Hybrid Representation Systems.* Lecture Notes in Computer Science **422**. Berlin: Springer-Verlag.

NILSSON, N. (1980). *Principles of Artificial Intelligence.* Palo Alto, CA: Tioga.

PATEL-SCHNEIDER, P. (1990). Practical, object-based knowledge representation for knowledge-based systems. *Information Systems,* **15**(1), 9–19.

ROBERTS, R. & GOLDSTEIN, I. (1977). *The FRL Manual.* AI memo 409, AI LAB, MIT, Cambridge, MA, USA.

ROSE, J. & GELERNTER, H. (1989). ISOLDE: a system for learning organic chemistry through induction. *Proceedings of the Third European Workshop on Knowledge Acquisition for Knowledge-Based Systems (EKAW'89), Paris,* pp. 297–310.

RUSSINOFF, D. (1989). Proteus: a frame-based nonmonotonic inference system. In W. KIM & F. LOCHOVSKY, Eds. *Object-Oriented Concepts, Applications, and Databases,* pp. 127–150. Reading, MA: Addison Wesley.

SCHUBERT, L., PAPALASKARIS, M. & TAUGHER, J. (1987). Accelerating deductive inference: special methods for taxonomies, colour and times. In N. CERCONE & G. MCGALLA, Eds. *The Knowledge Frontier: Essays in the Representation of Knowledge,* pp. 188–220. New York: Springer-Verlag.

SHAPIRO, L. & HARALICK, R. (1981). Structural descriptions and inexact matching. *IEEE Transactions on Pattern Analysis and Machine Intelligence,* **3,** 504–519.

STEFIK, M., BOBROW, D. & KAHN, K. (1986). Integrating access-oriented programming into a multi-paradigm environment. *IEEE Software,* **3**(1), 10–18.

STEPHENS, L. (1990). The classification of semantic relations based on primitive properties. In S. HUMPHREY & B. H. KWASNICK, Eds. *Proceedings of the 1st ASIS SIG/CR Classification Research Workshop, Toronto,* pp. 161–170.

TVERSKY, B. & HEMENWAY, K. (1984). Objects, parts and categories. *Journal of Experimental Psychology (General),* **113**(2), 169–193.

UNGAR, D. & SMITH, R. (1987). Self: the power of simplicity. *Proceedings of the 2nd OOPSLA, Orlando, Florida. ACM SIGPLAN Notices,* **22**(12), 227–242.

VISMARA, P., REGIN, J., QUINQUETON, J., PY, M., LAURENÇO, C. & LAPIED, L. (1992). RESYN—Un système d'aide à la conception de plans de synthèse en chimie organique. *Actes des 12èmes Journeés Internationales Intelligence Artificielle, Systèmes Experts, Langage Naturel, Avignon, Conférence Scientifique,* Volume 1, pp. 305–318.

VOSSELMAN, G. (1990). *Relational matching.* Lecture Notes in Computer Science **628**. Berlin: Springer-Verlag.

WEGNER, P. (1987). Dimensions of object-based language design. *Proceedings of the 2nd OOPSLA, Orlando, Florida. ACM SIGPLAN Notices,* **22**(12), 168–182.

WILCOX, C. & LEVINSON, R. (1986). A self-organized knowledge base for recall, design, and discovery in organic chemistry. In T. PIERCE & B. HOHNE, Eds. *Artificial Intelligence Applications in Chemistry,* pp. 209–230. New-York: ACS Symposium Series **306**.

WINOGRAD, T. (1975). Frame representation and the declarative/procedural controversy. In D. BOBROW & A. COLLINS, Eds. *Representation and Understanding: Studies in Cognitive Science,* pp. 185–210. New York: Academic Press.

WINSTON, M., CHAFFIN, R. & HERRMANN, D. (1987). A taxonomy of part–whole relations. *Cognitive Science,* **11,** 417–444.

WIPKE, W. & DOLATA, D. (1986). A multivalued logic predicate calculus approach to synthesis planning. In T. PIERCE & B. HOHNE, Eds. *Artificial Intelligence Applications in Chemistry,* pp. 188–208. New York: ACS Symposium Series **306**.

WIPKE, W. & ROGERS, D. (1984). Rapid subgraph search using parallelism. *Journal of Chemical Information and Computer Sciences,* **24,** 255–262.

WOODS, W. (1991). Understanding subsumption and taxonomy: a framework for progress. In J. SOWA, Ed. *Principles of Semantic Networks: Explorations in the Representation of Knowledge,* pp. 45–94. San Mateo, CA: Morgan Kaufmann.

WOODS, W. & SCHMOLZE, J. (1992). The KL-ONE family. *Computers & Mathematics with Applications,* **23**(2–5), 133–177.