Computation and Management of Chemical Properties in CACTVS: An Extensible Networked Approach toward Modularity and Compatibility

Wolf-Dietrich Ihlenfeldt,* Yoshimasa Takahashi, Hidetsugu Abe, and S. Sasaki

Department of Knowledge-Based Information Engineering, Toyohashi University of Technology, Tempaku, Toyohashi 441, Japan

Received June 29, 1993*

The data model of the CACTVS program suite (Chemical Algorithms Construction, Threading, and Verification System) is presented. CACTVS is an open environment which readily integrates new computational modules and data format descriptions in the domain of computational chemistry. These chemical information (property) computation modules are provided with requested input data, typically transparently computed by other modules. Computational results are managed appropriately with regard to file and data base I/O, transport to display servers, input for high-level routines, and report generation. Definitions of property characteristics which include optionally precompiled objects and source plus additional documentation may be stored in networked data bases for global access. This facilitates module reusal and the exchange of modules between interested parties. The elementary data format handling routines and the property computation routines need not to be part of the core program. They are retrieved from and loaded dynamically at run time from local files or data bases which can be reached on the network. This look-up process can be made completely transparent, resulting in a comparably small core program which grows only on demand to incorporate procedures for the evaluation of requested information. A model for asynchronous distributed client–server computations is supported as an alternative to dynamic linking for expensive calculations. CACTVS is intended to make algorithm development in chemistry more productive and to help in collecting the synergetic benefits from the concerted application of compatible algorithm sets which derive chemical information in all its variety.

CHEMICAL PROPERTIES

The core of almost every chemical computations is the evaluation of numerical, textual, or other types of information from molecular structure and auxiliary data. This chemical information of any kind is called a "property" (of ensembles, atoms, etc.) in this paper. Two characteristics of the computational process are important:

(1) Chemical information comes in an astonishing variety of data types and styles. Chemical information can be a simple Boolean flag attached to an ensemble, a string attached to each member of an ensemble of molecules, a GIF[1]-encoded picture of some ray-traced constellation of receptor and ligand, or something very complex like a collection of a variable-length integer and double-precision vectors attached to bonds of a specific type only. Additionally these bonds might be viewed as directional; i.e. there exist different sets of properties when the bond is looked at from the first atom to the last and in the opposite direction. The "bond" might further be some arbitrary nonstandard collection of atoms, which itself may be some artificial construction such as a lone pair substitute.

The point is that no limited set of elementary data types and atom/bond concepts alone will be able to code all this information in its variety. Both compound properties (i.e. those which are a collection of basic elements with simpler data types) and extensibility of the set of elementary data types are mandatory for a system which attempts to handle chemical information in general.

(2) The computation of chemical information typically relies not only on connectivity or 3D atomic coordinates but also on a set of more basic input property data such as the element type or partial charges of the atoms. This input data might be computed by comparatively simple methods. The conversion of an element symbol which was read as a string from some file to a periodic system number or some generic superatom, search list, or electron pair representation is an example. On the other hand, complex methods may be involved which require complex input information itself. The partial charges might be computed by some rather unsophisticated empirical method requiring element numbers and hybridization information as input alone, or by some full-fledged quantum mechanical (QM) procedure, which the local workstation cannot handle itself so some server on a more powerful machine must be consulted.

It is easy to see that the provision of input data for routines which compute some new property is essentially a recursive process. In classical computational programs the final algorithmic routine or the wrapper programs need to know in detail every aspect of the routines called beforehand. This includes data formats and the proper calling sequences and parameters and ultimately requires the complete integration of these routines into one program, often large, sometimes overblown and unmanageable. An alternative is a collection of programs with generally incompatible output formats and a large set of makeshift I/O format conversion tools, tailored more often than not to one very specific subformat of an output option of some program. Many computational routines need similar input information like charge, aromaticity and 3D coordinates, and yet the difficulties involved in the conversion of some freely available program to the local file formats and/or the adaptation of subroutines to local data structuring conventions are often comparable with the efforts of a complete reimplementation from scratch, which is too often the selected procedure due to the problems described. It is consequently difficult to obtain synergetic effects from the cooperation of different programs with the classical isolated approach.

CACTVS

In an attempt to cope with the aforementioned problems, the Chemical Algorithms Construction, Threading, and

* Abstract published in *Advance ACS Abstracts*, January 15, 1994.

0095-2338/94/1634-0109$04.50/0

Table I. Elementary Data Types[a]

name	characteristics	internal type
unknown	unspecified data type	float
Boolean	flags; T/F, on/off, etc., are valid items	long
byte	limited integer value; may be enumerated	long
short	limited integer value; may be enumerated	long
integer	full integer; may be enumerated	long
uint8	unsigned 64-bit integer for hash codes	structure
float	single-precision floating point	double
double	double-precision floating point	double
string	simple string	char*
index	indexing type for "*n*th occurrence in molecule/ens"	structure
bit set	collection of up to 32 independent bits/flags	unsigned long
intpair	integer pair (2D screen coordinate)	structure
floatpair	single precision float pair (2D plot coordinates)	structure
file	external data file with arbitrary contents	structure
blob	internal binary large object	structure
bitvector	arbitrary length bit vector (screens)	structure (u_long*)
bytevector	arbitrary length vector of limited integers	structure (u_char*)
shortvector	arbitrary length vector of limited integers	structure (short*)
intvector	arbitrary length vector of full integers	structure (long*)
floatvector	arbitrary length vector of single precision floats	structure (float*)
doublevector	arbitrary length vector of double precision floats	structure (double*)
stringvector	arbitrary length vector of strings	structure (char**)
compound	wrapper for compound properties	structure

[a] All *structure* internal types are different *structs*. Although some of the types are mapped to common internal representations, they are not equivalent for I/O, especially data base, purposes. They are output as compactly as possible. Vectors are generally coded internally in a packed format.

Verification System (CACTVS)[2,3] has been designed and implemented. The ultimate aim of the system is to provide a complete and reasonably portable system for the development, testing, and production runs of algorithms in chemical data processing. This includes a number of high-level capabilities such as an object-oriented graphical environment which supports mouse-driven interaction of molecular ensembles with structure editing and analysis tools, property requesters, I/O objects, portals to data bases, and numerous other gadgets. A general mechanism for the bidirectional information exchange with display servers is part of the concept. The user environment will be described in detail elsewhere. Furthermore, CACTVS uses experimental data base technology[4] for the storage and retrieval of the wide information variety it is capable of handling. The supported data base system provides interesting features like historical access and dynamic definition of search operators. Other parts of the complete system are a new approach toward a general-purpose scripting control language and its relation to visual desktop objects. The dynamic linking of property computation routines and data type handling functionality will be described in detail below. In a similar style the system also supports dynamic I/O format definitions and command language extensions. This paper focuses on the very essence of the system, namely, the handling and evaluation of chemical properties.

PROPERTY DEFINITION

The core of CACTVS is the framework to handle arbitrarily complex chemical information. An isolated piece of information about a chemical structure is called a "property". Chemical properties in CACTVS are either atomic in the sense of using a single elementary data type, such as a float vector for every bond, or are compounds. In a compound, a collection of more basic data types forms an inseparable piece of information. A good example is a catalog entry consisting of a manufacturer name, stored as a string, and the catalog number of the molecule. Both parts are useful only in combination.

Before a property can be handled, the essential characteristics of this property must have been defined. CACTVS currently supports some 170 internal predefined properties, which can be augmented by an unlimited number of auxiliary properties whose definitions may be loaded from some file or data base or are entered at run time. The description of a property is typically entered only once with the aid of an interactive editor and then stored at some publicly accessible place in a file or data base record. The definition of a property comprises a variety of field which describe its characteristics in considerable detail. Some of the more interesting items in the property definition form are as follows:

(1) Data Type. This is a simple elementary type (for example integer, Boolean, floating point pair [*coordinate*], string vector, bit vector, binary large object [*blob*]) or a combination of those for complex objects. See Table 1 for a list of the built-in elementary types, which can be augmented by user-defined elementary types.

(2) Attachment Point. Data can be attached to ensembles, molecules of an ensemble, atoms, bonds, or rings. Note that bonds or atoms do not necessarily correspond to the classical concepts. Torsional angles for example are a property of collections of four atoms, which are internally treated as a special class of bonds. If molecules or atoms migrate to another ensemble or molecule, respectively, the attached data are preserved (but see the invalidation concept below).

(3) Property Invalidation. This describes the circumstances under which the property is invalidated; i.e. if some other property is recomputed or edited or structural changes (atoms or bonds are added or deleted, molecules are merged to ensembles) take place, the validity status of the property is automatically adjusted. Properties can be marked for automatic update if they are going to become invalid due to those influences.

(4) Validity Filtering. Often properties are defined only for a specific subset of structural entities. For example, many topological indices are defined only for non-hydrogen atoms. In the context of CACTVS with its flexible and unusual concept of "atoms" and "bonds" most properties are valid only for certain classes of bonds and atoms (i.e. often only for

classical definitions). Filters which control the validity range of a property consist of a reference property, one or two reference values, and a comparison operator. Reference properties do not need to be part of the core set. A typical simple filter consists of the property "A_ELEMENT" (the number in the periodic system of elements, itself valid only for traditional atoms), "6" as a reference value, and "=" as a comparison operator to restrict the validity of a property to classical carbon atoms.

(5) Type Enumerations. For interaction with humans it is generally a good idea to use symbolic names for property values which are coded internally as some flag or integer value. An example are *R*/*S* and *E*/*Z* stereo descriptors. These are conveniently handled internally as integers but should be read and output as strings. Conversion proceeds with a table of recognized strings.

(6) Computational Methods. If properties are computable, the type of computation involved is described here. It can be either a built-in function, a dynamically loaded object, a server, or a command language script. Servers may run on remote hosts, and they operate asynchronously. This means the system does not block if some lengthy QM calculation has been submitted. The program is rather notified when the computation has finished or something has gone wrong and acquires the results at notification time. Script computation routines are handy to incorporate results from traditional programs which can be provided with input data and started as external processes. Other possible applications of scripts are the presentation of interactive forms for user input in the case of properties which are not computable and finally quick hacks for computationally inexpensive operations like the combination of some numeric input values to another property using coefficients derived from multivariate linear regression or similar methods. Dynamically loaded objects have the advantage that they run with the full speed of a built-in subroutine but do not clutter the core program. They can be unloaded and replaced at runtime, so there is no linking overhead during the development and debugging phase of some new algorithm. The system with the current set of test operators and test molecules remains up and running while new versions of algorithmic modules under development are loaded.

(7) Parameters for the Computational Routines. Often tuning parameters have to be passed to computational routines (for example damping factors, integration method selectors, etc.). If some result is obtained from those routines and attached to a chemical entity, the parameters used are copied to the property description of the ensemble, so the exact procedures followed to obtain a result can be reconstructed, even if these parameters have been changed in the meantime for the processing of additional structures.

(8) Documentation. Multiple textual documentation files of different formats can be attached to the definition. Additionally short columns for descriptive texts, comments, and literature references suitable for data base searching are provided.

Numerous other fields control I/O characteristics, numerical precision, algorithm debugging, possible directionality, and other minor features. See Figure 1 for a view of the property editor which is part of the core system.

PROPERTY EVALUATION

CACTVS uses delayed evaluation whereever possible. This means that property computation routines are invoked only when a property is actively requested. This is also true for very basic operations like converting between element symbol and element number. No property is considered inherently more elementary than others,[5] and no standard minimum property set is required. A simple example gives an idea of the processes involved:

I/O routines which have to write results in a specific format may for example request the element symbols of the atoms. Only at this moment the system checks whether it knows anything about a property called "A_SYMBOL", which denotes the element symbols. In this case the property and its computation routines are in the basic built-in set, so no external consultation of files and data bases is performed. In case the data originated from some file with a format which stores atoms as element symbols, these data are already available and valid and a handle will be passed to the requesting I/O function directly. Otherwise, the computation function is called which derives the symbols. The computation routine itself, which uses atom type flags and periodic system numbers, requests these data in precisely the same way the I/O routine did for the symbol. This process is recursive.[6] The I/O function does not need to know anything about the methods involved to derive the element symbol, about the input data for this function, and whether an actual calculation takes place or the data are already available. All it is informed about by default is success or failure. A number of modifiers for the property requests come in handy for I/O formats with varying data contents. Property requests may be detailed by specifying that some data are delivered only if they are already available or do not exceed a certain computational cost level. Properties may be marked for output under all circumstances or for active suppression.

The simple example of element symbols is easily extended to more complex situations. The key points of this chapter are recursion in the information gathering process, lazy evaluation, and information encapsulation. Complex algorithms dealing with chemical information can be built step by step by requesting a set of input parameters in a module and computing a new property. All those modules are guaranteed to be compatible to each other and can be developed and optimized independently. Because they are compatible, they can be reused easily without any rewriting. If some routine is enhanced, all routines which rely on data computed by this routine automatically benefit from the improvements, without recompilation or any other effort.

PROPERTY DEFINITION RETRIEVAL

Property definitions can be stored in files or data base records. CACTVS supports a networked data base system[4] which allows worldwide access if no access control mechanisms are employed. This approach facilitates the free and unbureaucratic exchange of modules between researchers. A data base can be configured to allow access by all interested parties from any Internet host or it can be operated as a repository for a work group or even a single user. Fine grain control of the amount of information made public is available:

(1) The basic property definition is the core information. If no source and no binaries of the actual computational routines are made available to the public, this still can serve as an advertisement and invitation for direct contacts to the developing research group or company.

(2) Multiple binaries for different architectures can be deposited. When a property description is retrieved, the system automatically selects a suitable binary if it is available. The binary is either an object for dynamic linking or a complete

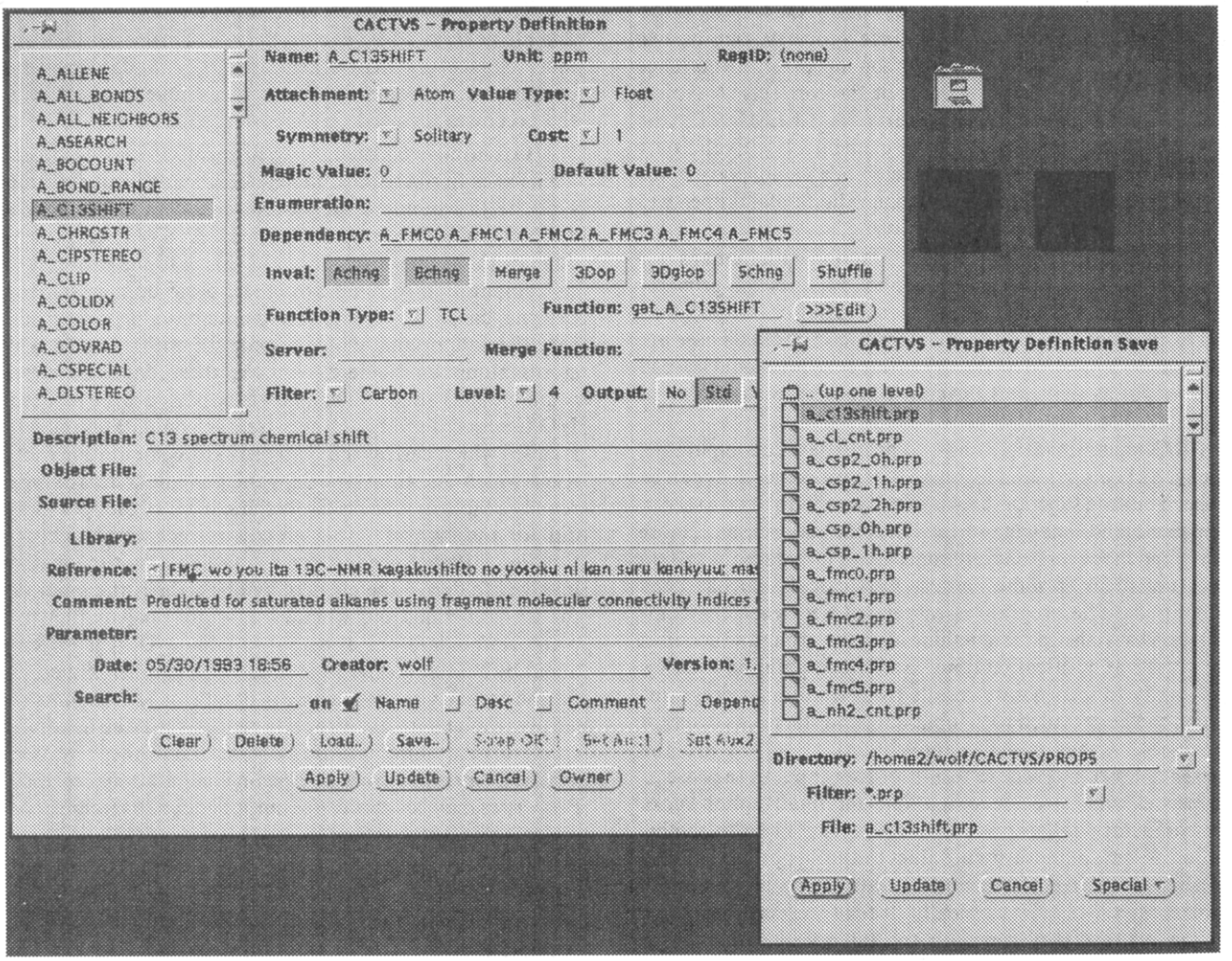

Figure 1. Interactive property editor. This is already the final stage after the automatically generated evaluation function has been added. The property definition including the evaluation script is going to be saved now for future projects. The properties in the save window list named "a_fmc..." are the definitions of the fragment molecular connectivity properties.

executable server. Scripts, a third way for property computations, are portable between architectures and operating systems.

(3) In case somebody is not willing to release even a binary but agrees to share results and donate computer time, the definition entered in the data base may contain the address of a server local to the data base (probably started automatically by *inetd* on an incoming call) which accepts computation requests from foreign hosts.

(4) Source files of the computational routines or servers can be archived. Ideally there is a single source which can be compiled on different operating systems, but provisions are made to allow for system-dependent source distributions.

(5) The last part of a property definition is documentation. Multiple documents and multiple instances of a document in different formats (plain ASCII, PostScript, FrameMaker binary, etc.) can be attached to a property definition. A property definition without documentation suffers a big loss of usability, so documentation is considered an integral part of the property description.

Two basic methods for the retrieval of a property definition are supported. The first method is manual search. Data bases can be connected to and scanned for property names, keywords, literature references, author names, or any other field of the definition. Searching is not limited to the basic definition relation but can be extended to documentation and function relations. The second method is an implicit search. The system can be configured to consult known data bases and scan directories which hold property definition files whenever the name or registry identification of an unknown property is encountered. This is a further step of information hiding. A computational routine does not have to rely on knowledge about some input data somewhere in the local system. If the used property is registered somewhere in one of the data bases in the list of the known and accessible depositories, it will be found and utilized properly. In an extreme case the system downloads automatically the newest version of an algorithm as an object for dynamic loading or as a complete server from the data base of a group on the opposite side of the globe. In such cases the downloaded definition will probably be copied to a local data base for efficiency reasons—hopefully into one which is not accessible from the outside in order to avoid swamping the community with unofficial and outdated releases.

ELEMENTARY DATA TYPES

Currently, CACTVS supports some 23 elementary data types by built-in functions (see Table 1). Besides basic types found also as variable types in programming languages, several kinds of vectors, large binary objects, memory-mapped files, and indexing types are part of this set. It should be remembered that these types are linked to a single chemical entity (atom, etc.), so for example a 3D distance matrix is

conveniently coded as a floating point vector attached to each atom. This collection certainly covers a wide selection of chemical information, especially if it is remembered that properties may be defined as compounds of elementary data types. However, it has already been explained that this set is not necessarily sufficient to handle all information imaginable. Therefore, CACTVS allows new elementary data types to be created. All that is necessary to add such functionality is to provide a set of functions which are dynamically loaded into the core system. The logical structure of such a set of handling functions is not complicated. Property computation functions which fill these custom structures with data and display servers which visualize the information are naturally expected to have detailed knowledge about the structures they are working on. The rest of the system needs only simple functions for freeing, duplication, storing, and I/O to strings, files, and data bases, XDR communication with servers plus some information about naming and sizes [XDR = external data representation, a standard to code binary data in a portable way so computers with different internal byte orders and other representation incompatibilities can share non-ASCII data]. If a CACTVS data base is expected to store properties of nonstandard data types, I/O functions for the data base and search operators working on these types may be provided and stored permanently in the data base. The set of functions needed to handle a custom data type in the CACTVS core may also be stored as source or binary in a data base and distributed via the same mechanisms and with the same options as the modules for property computations.

APPLICATION EXAMPLE

A practical application example will help to understand the process of property computation routine invocation and property definition look-up better. Although the data structures are simple in this example, the basic methods scale up to data of arbitrary complexity.

The problem selected as an instructive example is the derivation of a predictive formula for ^{13}C NMR chemical shifts. A data file with suitable information (molecular structure and experimental shifts) was located. The precise contents of the file are not known. The student who input the structures has left, and his documentation is difficult to read.[7] It has been decided to find a regression formula for the prediction of ^{13}C NMR chemical shifts of saturated hydrocarbons first to prove the general feasibility of the approach. The properties selected for multivariate regression analysis are fragment molecular connectivities (FMC) of different sphere ranges, a kind of molecular connectivity index.[8-10] Modules for the FMC computations have been programmed for this study and definitions set up. The FMC connectivity indices are named "A_FMC0" to "A_FMC5", spanning 0–5 spheres around an atom. They are single precision floating point numbers.

These functions were implemented as dynamically loadable modules which is the natural method for this type of property. On one hand they are fast to compute (a fraction of a second), so the server overhead is wasteful. On the other hand, significant computation is involved, so an interpreted script is too slow. The code size of these routines is small, about 50 lines of C each. The definitions, function source code, and precompiled binaries were put into a local directory accessible by CACTVS.

The following examples uses some high-level CACTVS user environment operations which are not explained in detail in this paper. The only important idea in this context is that there are "tool" gadgets which operate on ensembles flowing through them or dropped onto them. Tools can ask for the computation of properties, and destroy, augment, or edit the passing ensembles. Tools can also read and change property definitions. Another capability of tools used here is the creation and presentation of interactive forms.

All "ensembles" in the following explanation are single molecules, since every file record contained only an isolated molecule.

The steps taken to complete the study including the following:

Step 1: The system has initially absolutely no knowledge about ^{13}C NMR, so we have to define a ^{13}C NMR property first. This is done with the interactive property editor. The property has a simple structure, a single-precision float attached to atoms. It is defined only for carbon atoms. See Figure 1 for a picture of the property editor. No computation routines are defined yet.

The sequence of operations described in the next four points corresponds to the long pipe in the center of Figure 2.

Step 2: The complete file is read. Since we do not know the contained molecule classes in detail, some filtering is required. Duplicates which in this context include stereo isomers are removed with the aid of a molecular structure hash code computed without the inclusion of stereo information—a very useful property. The ensemble hash code of the current ensemble is compared against the hash codes of those molecules which were already encountered in this read cycle. The filter tool requests the hash codes of the molecules passing through it and discards duplicates. The ensemble hash code is a built-in property (64 bit unsigned integer).[11]

Step 3: Molecules which are not saturated alkanes are removed. The CACTVS distribution includes a handy standard tool which filters molecules according to the presence or absence of bond types. This tool requests a property called "E_BCLASSCNT" which is a variable-length integer array attached to ensembles. Each array element contains the count of a specific bond type. The bond types actually counted are defined by the parameter field in the definition of this property. It is parsed by the property computation routine, and strings like "C=C", "N-Cl", and "C-X" are translated into counts of carbon–carbon double bonds, nitrogen–chlorine single bonds, and carbon–heteroatom bonds, respectively. A change of this string results in other bond counts delivered. The filter tool reads back this parameter string from the property definition and builds a panel where the user selects acceptable bond types. Each bond type in the definition is represented by a three-way switch. The property "E_BLCASSCNT" itself is not a built-in property. In order to build the panel, the definition of this property must be known. Once the tool tries to read the parameter string, the system scans the internal register, some local directories, and finally data bases on the network in order to retrieve the definition of "E_BCLASSCNT" and a suitable binary for dynamic loading. Once those molecules surviving the first filter begin to pass through the tool, it reads the panel switches and compares the setting against the value of the requested property. At the time "E_BCLASSCNT" values are accessed first for a molecule, the properly computation binary is dynamically loaded on the arrival of the first molecule and sequentially called for all molecules.

Step 4: After those two property-oriented molecule filters only unique alkanes remain in the data stream. Now it is time to fill the "A_C13SHIFT" property. The *441* format of the source file is one of those abundant semistandardized

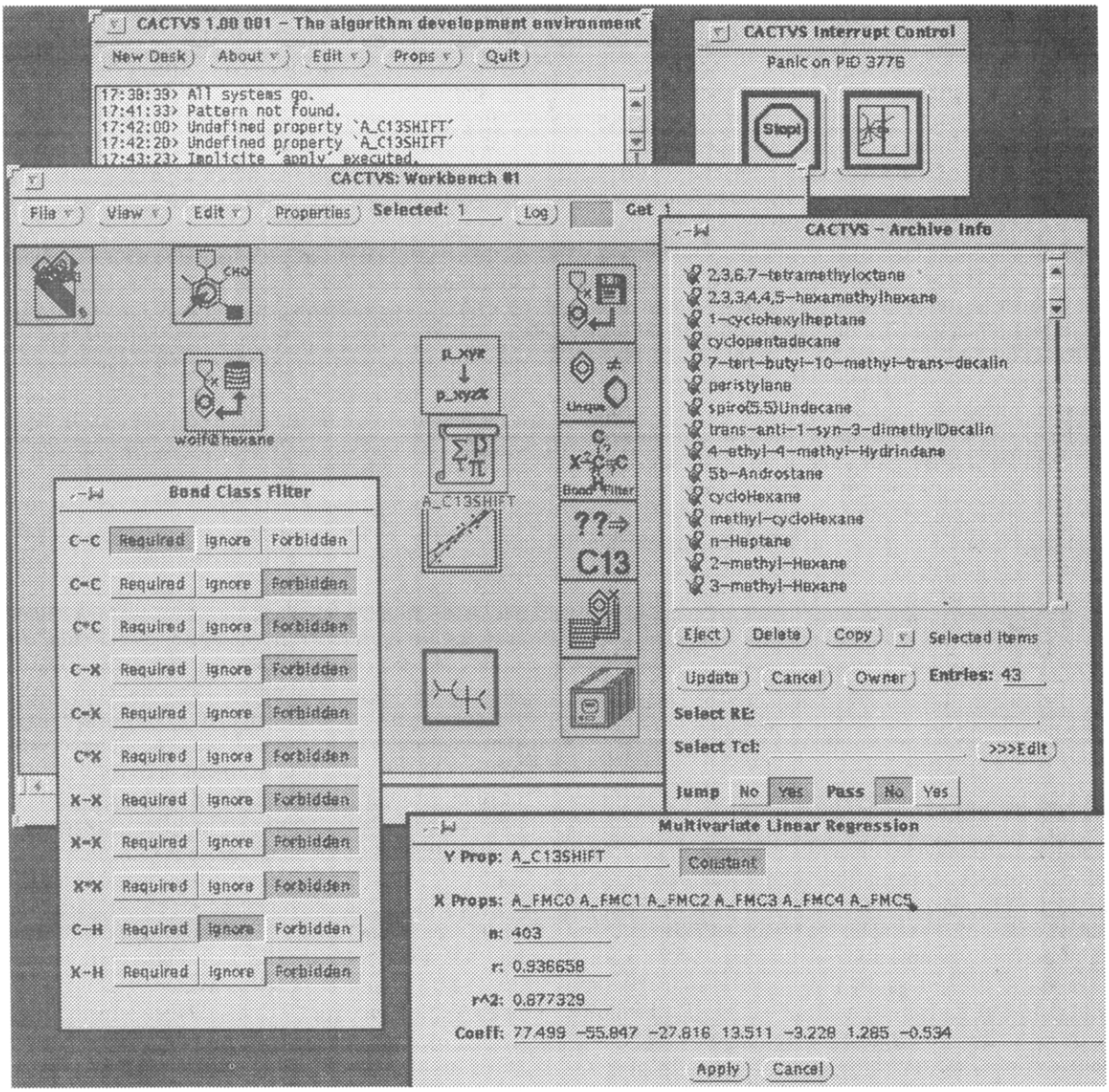

Figure 2. Complete setup for the multivariate regression analysis. The input file is the top right icon on the workbench. It is the head of a pipe which includes a tool for the removal of duplicates, a bond type filter, and an assignment tool which copies the undefined float data read from the file to the property encoding the chemical shift, followed by the multivariate linear regression tool and an archive which acts as a temporary storage facility for the molecules which passed through the pipe and were not filtered out. The settings of the bond class filter are shown in the lower left window. The window was created by the bond class filter tool after analyzing the "E_BCLASSCNT" property definition. The window to the right of the long pipeline shows the names of the filtered compounds stored temporarily in the archive. The names of the properties used in the regression analysis were entered in the lower right window which belongs to the regression tool. Input was requested before the pipe was flooded. After the molecules have drained into the archive, the coefficients and r values are filled in by the regression tool. The shorter three-membered pipe to the left consists of a backup facility, a property requester which recomputes the NMR shifts with the derived formula, and a portal to the display server for statistical X/Y graphs. This second pipe is used in the prediction step. Below is a single molecule which has been omitted from the training set. This molecule is examined in Figure 3. The icon above the bond class filter is a data base portal which was opened to retrieve the tools used in this study and the "E_BLASSCNT" property definition used in the bond class filter.

formats where a limited number of auxiliary fields may hold data of unspecified origin and meaning. When the molecules were read from file, this information was consequently stored in a property "A_UNDEF_FLOAT1" because all that was known about this field to the input routine was that it holds some floating point number. An assignment tool copies these property values to the "A_C13SHIFT" property. The copy process takes into account that the chemical shift is defined only for carbon atoms. Similar filtering information was of course not available for the original undefined float data.

Step 5: This step is the actual statistical analysis. The multivariate linear regression tool invoked for this purpose allows the user to specify the predicted Y property "A_C13SHIFT" in this case) and the properties used as X variable input ("A_FMC0", "A_FMC1", ..., "A_FMC5"). The tool requests the property values linked to the X and Y variables. Since the shifts were assigned in the last step and are still valid, no computation takes place for the Y variable. The X block is composed of the various fragment connectivity indices. The definitions of these properties are located in a local directory or in a data base and loaded automatically. When the computation routines are invoked, elementary properties like the number of non-hydrogen neighbors and bond type information are transparently requested by the FMC routines as input data. Since the statistics tool reads the definitions of the involved properties, it automatically un-

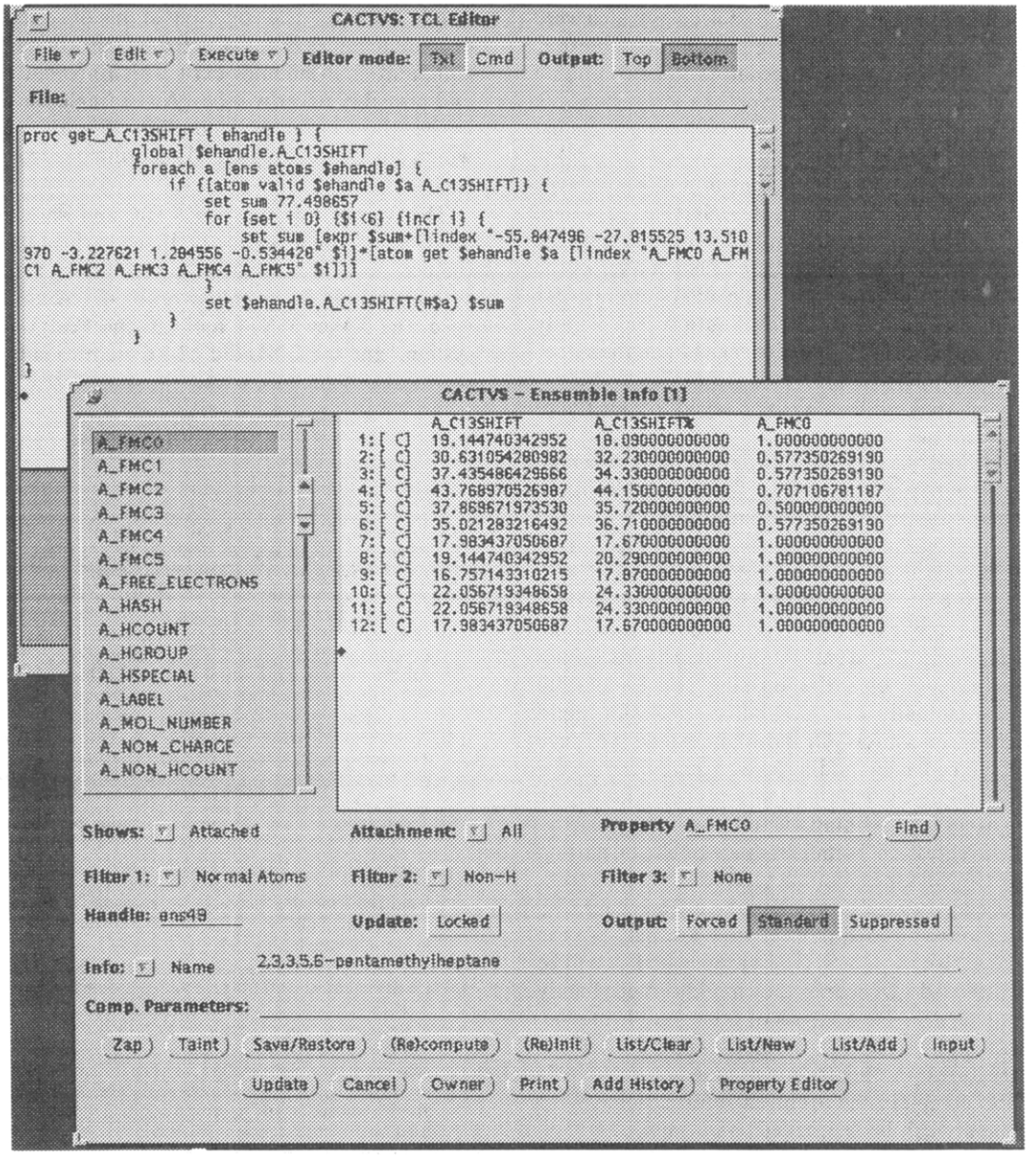

Figure 3. Two windows of the scripting language editor and a molecule data browser. The editor shows the automatically generated evaluation function for the ^{13}C NMR shifts which employs a constant and fragment molecular connectivity index values. A manually coded function might be more readable. This function has become an integral part of the ^{13}C NMR shift property definition and is stored together with it. Whenever now a ^{13}C shift is requested for any molecule, the function is automatically invoked. In the data browser below some prediction results are shown. The predicted shift values and the values read from the file record, which are marked here as a backup property with a percent sign, plus one of the molecular fragment connectivity parameters are listed.

derstands that it has to collect data only for carbon atoms. Data are accumulated until the molecule stream of this operative sequence dries up. At this point the accumulated matrix equation is solved, and the coefficients and *r* values are presented to the researcher. Furthermore, the statistics tool now automatically completes the property definition of "A_C13SHIFT". A property computation function of the "script" type is set up[12] using a generic template. The simple script requests the FMC properties involved in the statistical analysis and multiplies the values with the found regression coefficients in order to compute the predicted chemical shifts. The limitation of this property to carbon atoms is honored. The script is augmented to the original property definition. Figure 3 shows the automatically generated function in an editor window. The molecules passing through the analysis tool are collected in an archive object for the next phase.

The next three steps are a prediction phase designed for testing the new ^{13}C chemical shift prediction routine. The short central pipe in Figure 2 is the graphical representation of this sequence.

Step 6: The collected molecules are submitted from the archive to a second pipe sequence. The prediction step first uses a backup tool to create a copy of the "A_C13-SHIFT" property. The saved property is called "A_C13SHIFT%" and represents the experimental value which was read from file. The property definition is a duplicate of the original; only the name has changed. This duplicate definition is created automatically, without search.

Step 7: The property "A_C13SHIFT" is now recomputed by a property requestor object. A property requestor ignores the validity status and recalculates the property it symbolizes anyway. The automatically generated evaluation script is invoked for this purpose. The FMC properties of the molecules in the training set are still valid, so all that is done here is to multiply the regression coefficients with the FMC values and sum up. The new "A_C13SHIFT" property values are those

predicted by the regression formula. If molecules which were omitted from the training set are dropped on the requestor, it is recognized that their FMC values need to be computed first and the proper action is taken before the script sums up. See Figure 3 for predictive results with a molecule not included in the training set.

Step 8: The complete data set is sent to a display server which plots "A_C13SHIFT%" vs "A_C13SHIFT" for visual examination. The server (not shown here) first requests the plot variable names from the researcher. Then it queries the definitions of those properties from the core system in order to check whether these are representable data types. Finally the property values of the molecules checked in on the server are transferred and plotted.

In the last step, the results need to be saved for documentation and future experiments.

Step 9: The whole "A_C13SHIFT" definition represented by the property requestor is saved in a file or in a data base record. See again Figure 1. The computation script is saved automatically together with the property definition itself. Whenever the "A_C13SHIFT" property is requested now in any context and the storage location is in the search path, it will be found and applied without further user interaction. Note that results are still valid only for saturated hydrocarbons because only those were in the training set. A careful researcher will manually add a validity check for input molecules to the script source. Or she might extend the range of acceptable molecules by adding more parameters to the regression set (simply by entering more property names on the regression panel) and training with other molecule classes. If the generated regression formula is made public by putting the property definition into a publicly accessible data base, it should not be forgotten to supply also the underlying FMC property definitions and modules. If they are put into the same data base as the "A_C13SHIFT" property, they will be retrieved when needed. Of course, they might come in handy as input parameters for the analysis of other problems, too. Textual documentation of the methods applied and the molecular structures in the training data set plus their measured shifts complete the archived information. With this data, the study can be reproduced any time.

CONCLUSION

CACTVS aims at providing an networked and interoperative environment for computations in chemistry. Computations are encapsulated in modules, which are as independent as possible from the processes responsible for providing input data. The system has been designed with emphasis on global data and program exchange via networks involving data bases with property definitions and computational modules, documentation, and a custom data type handling functionality. Property computation involves state-of-the-art computational techniques such as client–server computing, dynamic loading, and information encapsulation. It is hoped that this system will make algorithm development in chemistry significantly more productive and lead to synergetic effects in the application of different computational methods to chemical problems.

PROJECT STATUS

At the time of print, the first public release of the system should be available. It is intended to put the core into restricted public domain. The copyright status of contributed computational modules is not affected by this. Even now nontrivial computational modules exist which are not public domain and probably never will be. An example for such a module is the 3D coordinate generation algorithm CORINA,[13,14] which quickly builds good 3D coordinates from a connection table.

The system is written in K&R C and was developed under SunOS 4.1.2 on a Sparc II. If the various libraries used by the programs are available, portability to mainstream Unix systems should be no major problem. Among the libraries used, the XView3 X11 tool kit, the Postgres4.1 data base system,[4] and the GNU dld3.2.3 dynamic loader are probably the most problematic ones.

REFERENCES AND NOTES

(1) *G*raphical *I*nterchange *F*ormat. A popular file format for the storage of graphic information.

(2) Ihlenfeldt, W. D.; Takahashi, Y.; Abe, H.; Sasaki, S. CACTVS: A Chemistry Algorithm Development Environment. In *Proceedings of the 15th Symposium on Chemical Information and Computer Sciences/ 20th Symposium on Structure-Activity Relationships*; Machida, K., Nishioka, T., Eds.; Kyoto University: Kyoto, Japan, 1992; pp 102–105.

(3) Ihlenfeldt, W. D.; Takahashi, Y.; Abe, H.; Sasaki, S. Algorithm development in chemistry: The detection of common three-dimensional substructures in large sets of possibly flexible molecules. In *Computer Aided Innovation of New Materials II*; Doyama, M., Kihara, J., Tanaka, M., Yamamoto, R., Eds.; Elsevier: Amsterdam, 1993; pp 1155–1158.

(4) Stonebraker, M. R.; Rowe, L. A. The Design of POSTGRES. *Proceedings of the 1986 ACM-SIGMOD International Conference on the Management of Data*; Association of Computing Machines: Washington, D.C., 1986.

(5) The only exceptions are the classification properties "A_TYPE" and "B_TYPE" for atoms and bonds. They are maintained automatically and are valid at all times because there is no way to derive this information. However, new types of atoms or bonds can be freely added. Current bond types include "classic", "three-center", "hydrogen bond", "bond which must not be present in substructure matching", "quartet of atoms in torsional angle", etc.

(6) The recursion level is limited, so in case there are neither symbols nor element numbers infinite recursion is avoided. Computational routines which can use more than one set of starting data may check the content of the current property set and select their methods depending on what is already available.

(7) This example is not contrived. The data (actually a set of separate files in different directories, only partially saturated with hydrogen atoms—collection and hydrogen addition are two more tasks for tools) used here are the set examined first in: Osada, H. Fragment Molecular Connectivity wo mochiita ^{13}C-NMR kagakushifto no yosoku ni kan suru kenkyuu (in Japanese, translated title: Studies on ^{13}C NMR chemical shift prediction using Fragment Molecular Connectivity). Master's thesis, Toyohashi University of Technology, Toyohashi, Japan, 1993. The coefficients found for alkanes are as follows: 77.499 – 55.847-A_FMC0 – 27.815A_FMC1 + 13.511A_FMC2 –3.228A_FMC3 + 1.285A_FMC4 – 0.534A+FMC5, $r^2 = 0.88$, $n = 403$. The results do not change much without A_FMC3...A_FMC5.

(8) Kier, L. B.; Hall, L. H. *Molecular Connectivity in Chemistry and Drug Research*; Academic Press: New York, 1976.

(9) Kier, L. B.; Hall, L. H. In *Molecular Connectivity in Structure-Activity Analysis*; Research Studies Press (John Wiley & Sons): Letchworth, U.K., 1986.

(10) The precise definitions of the FMC values used in this study are yet unpublished. However, a general idea about the algorithms involved can be found in: Takahashi, Y.; Miyashita, Y.; Tanaka, Y.; Hayasaka, H.; Abe, H.; Sasaki, S. Discriminative Structural Analysis Using Pattern Recognition Techniques in the Structure-Taste Problem of Perillartines. *J. Pharm. Sci.* **1984**, *73*, 737–741.

(11) Ihlenfeldt, W. D.; Gasteiger, J. Hash Codes for the Identification and Classification of Molecular Structure Elements. *J. Comput. Chem.*, in press.

(12) CACTVS uses Tcl/TclX/TclPq as scripting language.

(13) Gasteiger, J.; Rudolph, C.; Sadowsky, J. Automatic Generation of 3D-Atomic Coordinates for Organic Molecules. *Tetrahedron Comput. Method* **1990**, *3*, 537–547.

(14) Sadowsky, J.; Rudolph, C.; Gasteiger, J. The Generation of 3D Models of Host-Guest Complexes. *J. Anal. Chim. Acta* **1992**,*256*, 233–241.