Theor Chem Acc (2003) 109:229–232
DOI 10.1007/s00214-002-0402-6

Theoretical
Chemistry Accounts

Regular article

Atom-type description language: a universal language to recognize atom types implemented in the VEGA program

A. Pedretti, L. Villa, G. Vistoli

Istituto di Chimica Farmaceutica e Tossicologica, University of Milan, Viale Abruzzi 42, 20131, Milan, Italy

Received: 26 April 2002 / Accepted: 12 September 2002 / Published online: 19 March 2003

Abstract. The atom-type description language (ATDL) is a universal language used to describe and recognize the atom types from chemical connectivity. In this paper the ATDL approach specifications are reported with several examples. To date, this language is implemented in VEGA (http://www.ddl.unimi.it), a multipurpose program able to convert and manage several molecular file formats. This software uses the ATDL to assign the correct atom types in order to help several functions (file format conversion, molecular properties calculation, surface mapping and interaction energy analysis).

Keywords: Force fields – Atom type assignment – Atomic parametrization

Introduction

One of the fundamental quests in computational chemistry is to continuously facilitate the interchange of data. This requirement is becoming increasingly important owing to the increased number of file formats which have invaded the world of molecular modelling.

Indeed, each software has its own specific file format and more often than not the molecular modelling packages cannot directly exchange information as their molecular files are incompatible. Moreover, owing to the diffusion of the force fields, one may want to preserve the same format while modifying the force field by re-assigning the atom types with a new atom classification.

Several classifications (force fields) have been proposed to classify the atoms into atom types on the basis of atomic number, hybridization, bond order and bonded partners [1, 2]. Every classification uses specific symbolic codes to represent its atom types.

Furthermore the PDB [3] file format has become the leading format for the storage of biomacromolecules, owing to the explosion of experimentally resolved structures deposited in the Brookhaven Data Bank [4]. This format may be the only type that is recognized by almost all molecular modelling packages. Unfortunately PDB files do not include any information on atom types or atomic charges and it this is a limitation for structure modelling and for analysis of receptor–ligand interactions.

The atom typing is centred around the assumption that when considering analogous chemical environments, atoms will usually have local properties that are transferable between molecules. This concept plays a pivotal role not only in the parameterization of the potential-energy functions (force field), but also in the calculation of all molecular properties based on an atomic parameterization or on a surface projection of local values (as for log$P$ prediction).

For this reason we are searching for a method with which to describe the atom typing of both force fields and atomic parameterizations. This method should be independent of all schemes already reported in the literature, but able to represent all of them in a clear, easy and general way. We propose the atom-type description language (ATDL) as a tool to achieve these goals as it is able to define any atom type and allows switching from one atom-type classification to another while referring to the corresponding ATDL definition.

The ATDL is based on molecular connectivity and each atom is defined by describing not only the directly bonded atoms but also by the theoretically infinite

Contribution to the 8th Electronic Computational Chemistry Conference, 2002

Electronic Supplementary Material to this paper (full text of the lecture in html as given at the ECCC8 conference) can be obtained by using the SpringerLink server located at http://dx.doi.org/10.1007/s00214-002-0402-6

Correspondence to: Giulio Vistoli
e-mail: giulio.vistoli@unimi.it

sublevels of connected atoms, using the recursive schemes within the C programming language. This characteristic allows a more accurate definition of atom types and reduces the conflicting assignments. The user can expand VEGA capabilities, by writing new force field templates with the ATDL.

ATDL specifications

Each atom is defined by a five-character string (Fig. 1). The first two characters are the element symbol of the atom. If the element symbol is one character only, the second character must be a dash (–). Special characters are also included: X for any atom and # for any heavy atom. The third character is the bond order: one can use values from 1 to 6 for real bond order, 0 for a nonbonded atom and 9 for a bonded atom without a specified bond order. The fourth character is the ring indicator: ranging from 3 to 6 if the atom is a 3 to 6-ring member, 0 for a nonring member atom and 9 for a nonspecified ring atom. The fifth character is the aromatic indicator: 0 for a nonaromatic atom, 1 for an aromatic atom and 9 if the aromaticity is not specified.

Examples: C-300 is an sp^2 aliphatic carbon, Si900 is a generic silicon with any bond order and C-361 is an aromatic carbon included in a six-member ring.

After the definition of the atom you may add the description of bonded atoms (also in ATDL) limited by parentheses as the description of connected atoms is crucial to correctly define the atom properties. For example, O-200 indicates an oxygen atom not in a ring and bonded to two other atoms. This specification can be used both for a hydroxyl group and for an ether function. In order to resolve this conflict you can add the description of connected atoms in ATDL: O-200 (C-400 H-100) for a hydroxyl group and O-200 (C-400 C-400) for ether.

As mentioned in the Introduction, more than one level of parenthesis can be used for complex descriptions of atom types and each group of atoms, limited by parentheses, contains the descriptions of all atoms bonded to the previous atom.

For instance, the description of an α carbon of a glycine in zwitterionic form is

C-400(N-400(H-100 H100 H-100)
C-300(O-100 O-100) H-100 H-100).

In this way you can indicate an sp^3 carbon bonded to two hydrogen atoms (H-100), a nitrogen atom positively charged (N-400) with three hydrogen atoms and an sp^2 carbon bonded to two nonsubstituted oxygen atoms.

It is important to remember that VEGA reads from left to right and thus the more restrictive atom description must be placed on the furthest left side of the line: C-400 (C-300 X-900 X-900 X-900) and not C-400 (X-900 X-900 C-300 X-900). Indeed with the seond definition, VEGA may recognise the sp^2 carbon as a more generic X-900 and therefore cannot reassign it to the next more specific description.

The description sequence of each atom goes from more to less specific, from the upper to the lower line:

cn C-400 (N-300 X-900 X-900 X-900) more specific,

c C-400 (X-900 X-900 X-900 X-900) less specific.

In the case where these two lines are swapped, if VEGA were to find an sp^3 carbon bonded to an sp^3 nitrogen, the atom type recognized is a generic c type and not a cn one.

For each atom typing scheme (force field or atomic parameterization) VEGA uses a template file, in which the descriptions of atom types are stored together with a symbolic code (for force fields) or a local value (for atomic contributions). Each atom-type definition is stored in a line with three columns. The first column is the atom-type name (or value), the second column is the atom description in ATDL and the third column contains the descriptions of bonded atoms (also in ATDL).

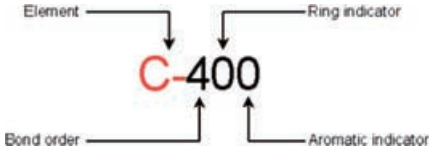

Fig. 1. Atom description scheme in atom-type description language (*ATDL*)

```
h     H-100   (C-900)
h+    H-100   (N-400)
hn    H-100   (N-900)
h*    H-100   (O-200 (H-100 H-100))
ho    H-100   (O-900)
hp    H-100   (P-900)
hs    H-100   (S-900)
dw    D-100   (O-200 (D-100 H-100))
dw    D-100   (O-200 (D-100 D-100))
d     D-100   (X-900)

cg    C-400   (N-400 (H-100 H-100) C-300 H-100 H-100)
cg    C-400   (N-300 (H-100) C-300 H-100 H-100)
ca    C-400   (N-400 C-300 H-100 X-900)
ca    C-400   (N-300 C-300 H-100 X-900)
coh   C-400   (O-200 (C-900) O-200 H-100 X-900)
co    C-400   (O-200 (C-900) O-200 X-900 X-900)
c3h   C-430   (C-430 H-100 H-100 #-930)
c3m   C-430   (C-430 #-930 X-900 X-900)
c4h   C-440   (H-100 H-100 X-940 X-940)
c4m   C-440   (X-940 X-940 X-900 X-900)
c3    C-400   (H-100 H-100 H-100 X-900)
c2    C-400   (H-100 H-100 #-900 #-900)
c1    C-400   (H-100 #-900 #-900 #-900)
cn    C-400   (N-400 X-900 X-900 X-900)
cn    C-400   (N-300 X-900 X-900 X-900)
c     C-400   (X-900 X-900 X-900 X-900)
ci    C-351   (N-351 (H-100) N-351 (H-100) X-900)
```

Fig. 2. Example of ATDL definitions for some atom types of the CVFF force field

As an example, the definitions of CVFF [5] atom types for hydrogen and carbon atoms are reported in Fig. 2.

VEGA in fact supports several force fields, for example, CVFF [5], AMBER [6], CHARMM [7], MMFF [8] and TRIPOS [9]. VEGA can also be expanded with new atom-type descriptions or new template files.

Computational details

The recognition and the assignment of correct atom types using ATDL consists of three steps (Fig. 3). Firstly, VEGA determines the atomic connectivity without the use of existing information (e.g. CONECT records from PDB files) but by calculating all interatomic distances, using a table of van der Waals radii with a tolerance threshold equal to 20%. From the connectivity, VEGA also determines the bond orders and the hybridization types of all atoms. To date VEGA can only handle atoms with filled valencies (i.e. with all hydrogens) and only considers the bond length and not the valence angle.

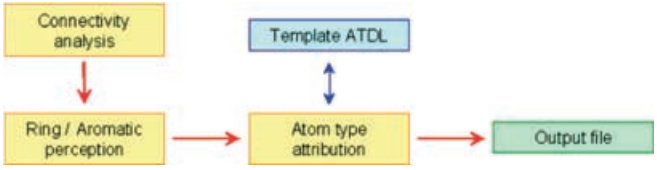

Fig. 3. Main logical component of the ATDL recognition algorithm

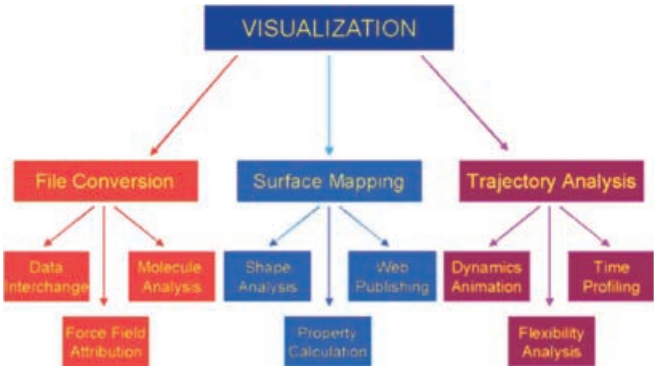

Fig. 4. Logical classification of the VEGA features

In the second assignment step, VEGA identifies the ring systems. VEGA looks for the rings from three to six members by means of an algorithm that recursively analyses the connected atoms for all atoms and determines the closed connections. The perception of aromatic rings is based on Hückel's rule. A ring is classified as aromatic if it obeys the rule of $4n+2\pi$ electrons. All bonds within the ring are typed as aromatic and all hybridization states are sp^2. VEGA recognizes both aromatic and heteroaromatic rings with five or six atoms.

Finally, VEGA checks the correct syntax of the chosen template file with a description parser and matches the atom examined with the ATDL descriptions to find the corresponding atom type.

VEGA features

The features of the VEGA program are summarised in Fig. 4: their detailed description is beyond the scope of this paper. In this section we analyse the most interesting features of our program that take advantage of the ATDL approach and that can be useful in the management of molecular structures.

1. File format conversion. There is a complete list of all the implemented file formats in Table 1. An intelligent algorithm automatically recognizes the input file format, therefore allowing the user to specify the output format only. It is possible to load more than one file at a time with the same or with different formats in order to create molecular assemblies. The calculation of the connectivity is performed separately for each file loaded in order to avoid the connectivity errors of bumping molecules.
   A data de/compressor engine has also been implemented in VEGA to allow the use of compressed

Table 1. Molecular file formats that are recognized by VEGA in I/O operations

Keyword	Description
BIOSYM	New Biosym .car file (archive 3)
CRD	MSI CHARMm text file format.
CSSR	Cambridge data file
FASTA	Fasta is not a real molecular file, because it canonly store the primary structure of proteins or DNA/RNA sequences.
GROMOS	This is the file format of the molecular packageGromos.
GROMOSNM	GROMOS with the coordinates in nanometers.
MOL2	Tripos Sybyl Mol2 file format.
MOPINT	The Mopac internal coordinates file (.dat) is usefulto link Mopac with other software packages. The Mopac keyword CHARGE isautomatically calculated by atomic charges.
MSF	MSI Quanta binary file.
OLDBIOSYM	Old Biosym .car file (archive 1)
PDB	PDB full standard (default)
PDBF	PDB full standard with special records to includeatomic charges and force field parameters (see PDB implementation fordetails)
PDBNOTSTD	Simplified PDB format, more compatible with software packagesthat have a partial implementation of Brookhaven specifications. Specialrecords (HETATM, TER, CONECT and MASTER) are not used.
PDBQ	PDB full standard with atomic charges placed in thelast right column
XYZ	Cartesian coordinates file. The first record is thetotal number of atoms. The atom record contains the element name and the x, y, z Cartesiancoordinates.

files without a preceding un/packing procedure: the packed files can be handled in the same way as unpacked files without the use of an external data decompressor. VEGA supports the following packing formats: Gzip, Bzip2, PowerPacker and UNIX Un/compress.

2. PDB implementation. As mentioned in the Introduction, the PDB files do not include any information on atom types and atomic charges. Furthermore this format uses defined tags and it is not possible to add new user-defined tags to include other information without losing compatibility with other programs. In order to overcome this problem we propose to take advantage of the REMARK 77 tag in order to insert EXTRA lines with atom types and/or atomic charges. Thus the programs able to read this information (like our BioDock [10]) will recognize these EXTRA lines, whereas the other packages will skip the REMARK lines without any compatibility problems. We termed this format PDBFat (PDBF) in order to highlight its compatibility with standard PDB even though it can include more information and VEGA is able to convert any format to PDBFat (and vice versa), assigning the atom types and the atomic charges.

   We have also implemented a second variant of PDBFat that include the ATDL definitions of the atoms in the EXTRA lines. This format can be very useful in custom applications in which the user must recognize some atom types for specific purposes (e.g. calculation of the score function in a molecular docking program).
3. Local value attribution. Other than in the file format conversion, the ATDL finds a great number of applications in the attribution of local values or in the parameterization of molecular properties.

VEGA takes advantages of the ATDL in several applications:

1. Calculation and display of MLP surface, assigning to each atom the lipophilic contribution.
2. Calculation of molecular properties such as the hydropathicity profile (ILM) [11], Crippen's log*P* [12] and lipole, Broto's log*P* [13] and lipole, Virtual log*P* [14].
3. Calculation and analysis (with an integrated graph editor) of the aforementioned properties for each frame of a molecular dynamics trajectory, to gain a 4D description.
4. Evaluation of the ligand–biomacromolecule interaction energy, assigning the correct nonbond terms. In this way, VEGA calculates the interaction energy, its components and a table with the receptor residues involved in interaction with a partial interaction energy greater than 1% of the total energy.

Conclusions

The ATDL fits in a more general search of unified and standard languages that are able to represent concepts or classifications that would otherwise require phrases to describe them in a more perceptive and compact way (e.g. the UML language in object-oriented programming).

This language finds interesting applications in all atom-type recognitions, both in force field calculations and in local value attributions. The ATDL would not only be a program application but also a concept. Anyone who can write code can use ATDL specifications to recognize atom types or atomic parameters for custom applications.

Acknowledgements. The authors acknowledge K. Henrich, A.M. Villa and B. Testa for insightful discussion and R. Verma for careful reading of the manuscript. This work is supported by Italian MIUR.

References

1. Hobza P, Kabelac M, Sponer J, Mejzlik P, Vondrasek J (1997) J Comput Chem 18:1136–1150
2. Roterman IK, Lambert MH, Gibson KD, Scheraga HA (1989) J Biomol Struct Dyn 7:421–452
3. Bernstein FC, Koetzle TF, Williams GJD, Meyer EF, Brice MD, Rodgers JR, Kennard O, Shimanochi T, Tasumi M (1977) J Mol Biol 112:535–542
4. Protein Data Bank www.rcsb.org/pdb
5. Dauber-Osguthorpe P, Roberts VA, Osguthorpe DJ, Wolff J, Genest M, Hagler AT (1988) Proteins Struct Funct Genet 4:31–47
6. Cornell WD, Cieplak P, Bayly CI, Gould IR, Merz KM Jr, Ferguson DM, Spellmeyer DC, Fox T, Caldwell JW, Kollman PA (1995) J Am Chem Soc 117:5179–5197
7. (a) Brooks BR, Bruccoleri RE, Olafson BD, States DJ, Swaminathan S, Karplus M (1983) J Comput Chem 4:187–217; (b) MacKerell AD, Bashford D, Bellott M, Dunbrack RL, Evaseck, JD, Field MJ, Fischer S, Gao J, Guo H, Ha S, McCarthy JD Kucnir L, Kuczera K, Lau FTK, Mattos C, Michnick S, Ngo T, Nguyen DT, Prohom B, Reiher WE, Roux B, Schlenkrich M, Smith JC, Stote R, Straub J, Watanabe M, Wiorkiewicz-Kuczera J, Yin D, Karplus M (1998) J Phys Chem B 102:3586–3617
8. (a)Halgren TA (1996) J Comput Chem 17:490–519; Halgren TA (1996) J Comput Chem 17:520–552; Halgren TA (1996) J Comput Chem 17:553–586; (d) Halgren TA, Nachbar RB (1996) J Comput Chem 17:587–615; (e) Halgren TA (1996) J Comput Chem 17:616–641
9. Clark M, Cramer RD III, van Opdenhosch N (1989) J Comput Chem 10:982–1012
10. Pedretti A, Villa L, Vistoli G (2002) J Med Chem 45:1460–1465
11. Pedretti A, Villa AM, Villa L, Vistoli G (30/06/2000) Internet J Chem 3, article 13
12. Ghose AK, Pritchett A, Crippen GM (1988) J Comput Chem 9:80–90
13. Broto P, Moreau G, Vandycke C (1984) Eur J Med Chem 19:71–78
14. Gaillard P, Carrupt PA, Testa B, Boudon A (1994) J Comput-Aided Mol Des 8:83–96