ZEOBUILDER: A GUI Toolkit for the Construction of Complex Molecular Structures on the Nanoscale with Building Blocks

T. Verstraelen, V. Van Speybroeck, and M. Waroquier*

Center for Molecular Modeling, Gent University, Proeftuinstraat 86, B-9000 Gent, Belgium

Received February 29, 2008

In this paper, a new graphical toolkit, ZEOBUILDER, is presented for the construction of the most complex zeolite structures based on building blocks. Molecular simulations starting from these model structures give novel insights in the synthesis mechanisms of micro- and mesoporous materials. ZEOBUILDER is presented as an open-source code with easy plug-in facilities. This architecture offers an ideal platform for further development of new features. Another specific aspect in the architecture of ZEOBUILDER is the data structure with multiple reference frames in which molecules and molecular building blocks are placed and which are hierarchically ordered. The main properties of ZEOBUILDER are the feasibility for constructing complex structures, extensibility, and transferability. The application field of ZEOBUILDER is not limited to zeolite science but easily extended to the construction of other complex (bio)molecular systems. ZEOBUILDER is a unique user-friendly GUI toolkit with advanced plug-ins allowing the construction of the most complex molecular structures, which can be used as input for all ab initio and molecular mechanics program packages.

1. INTRODUCTION

Specialized graphical computer programs are indispensable for the construction of atomic models in computational chemistry research. The advances in the development of nanoscale materials and the modeling of increasingly complex biological systems imply new demands on the software toolkits that are used to construct and manipulate the molecular models in these challenging research fields. Computational chemistry is playing an increasingly important role in all aspects of zeolite science.

There is large demand for graphical tools that build large frameworks with different compositions and topologies. All existing commercial (Cerius2,[1] Materials Studio,[2] CrystalMaker,[3] Diamond,[4] etc.) software packages are of high quality and are indispensable in most molecular modeling applications using well-validated techniques. A survey of all available open-source tools (Gamgi,[5] PyMol,[6] JMol,[7] Avogadro,[8] AGM,[9] ect.) reveals that these GUIs have rather limited features and are not widely used due to a lack of applicability, feasibility, or transferability. We could consider the extension of existing graphical toolkits, but even the most adequate package poses serious problems when implementing new features such as the growth of a zeolite framework starting from elementary building blocks. An essential ingredient for a molecular building toolkit is a suitable data structure with multiple reference frames in which molecules and molecular building blocks are placed and are hierarchically ordered. A powerful plug-in framework for the extension of a GUI toolkit with new features is equally important and unfortunately not present in many existing codes.

Inorganic crystals such as zeolites are conventionally classified on the basis of subunits in their framework structure.[10] The basic building unit for a zeolite structure is the tetrahedron formed by a central T atom and four oxygen atoms at the corners, where the T atom is typically silicon or aluminum. All zeolite frameworks can be constructed from a small set of larger secondary building units (SBUs), although the list of SBUs has been extended over the years when new zeolite frameworks were developed.[10] It is common practice to describe the structure of a zeolite in terms of even larger composite building units (CBUs) such as the sodalite cage, the double six ring, and so on.[11] At an even larger scale, one recognizes periodic building units (PerBU's), for example, chains or channels formed by an infinite series of SBUs and CBUs. An overview of the standard secondary, composite, and periodic building units for zeolites is given in the Atlas of Zeolite Framework types.[12] The concept of building units is a recurring theme in many fields: for example, proteins are built from peptides; metal−organic frameworks consist of metal centers and ligands in between them, and so on. From this point of view, a graphical molecular editor must fully embrace the concept of molecular building blocks.

Next to the graphical toolkits, a distinct category of nongraphical software tools is used for the manipulation and study of inorganic structures. Zebedde[13,14] is a powerful computational tool focusing on the design of suitable template molecules for a given pore structure but does not address the construction of the framework itself. There are also many popular structure refinement and predicition tools, for example, DLS-76,[15] GRINSP,[16] SHELX,[17] and so on. We are convinced that a graphical toolkit should not duplicate all of the features in the existing nongraphical software. Instead, a GUI toolkit must be able to interoperate with these programs through open standards for file formats and communication protocols.[18]

Taking into account all of these considerations, the authors preferred to build a new GUI toolkit—called "ZEOBUILDER"—from scratch, which involves all of the ingredients

* To whom all correspondence should be addressed. Tel.: 32 (0)9 264 65 59. E-mail: michel.waroquier@UGent.be.

10.1021/ci8000748 CCC: $40.75
Published on Web 06/11/2008

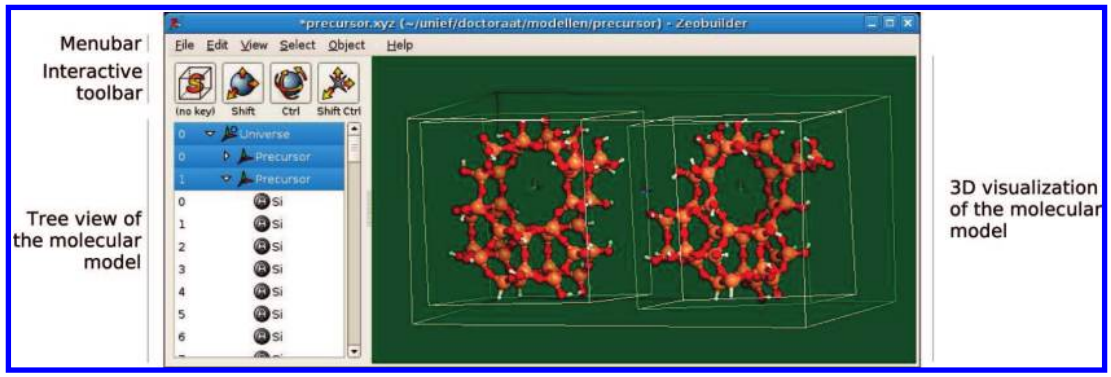

Figure 1. The graphical user interface of ZEOBUILDER consists of four major parts: a menu bar, an interactive toolbar, a tree view of the model, and a 3D visualization of the molecular system.

required for the construction of complex hierarchical zeolite models and which offers an ideal platform for the further development of new features in an open-source facility.

The two key-advantages of ZEOBUILDER can be summarized as follows:

(i) ZEOBUILDER has a user-friendly graphical interface that presents all its features in a toolkit fashion. With an appropriate choice of well-selected features and creativity, the user should be able to manage the most complex tasks.

(ii) Additionally, the program is also developer-friendly. The toolkit approach guarantees that a minimum of coding effort results in maximal functionality. Further, it will be distributed as an open-source cross-platform tool, and new features are easily implemented as plug-ins.

The plug-in architecture is the main technical advantage of ZEOBUILDER, when compared to the open-source alternatives. To encourage new developers, a ZEOBUILDER plug-in is a simple self-contained text file: it is cross-platform by definition, and no compilation is required. The program is written in the popular and highly portable Python programming language. ZEOBUILDER as presented in this paper is not restricted to building large zeolite structures but is easily extended to the construction of other complex (bio)molecular systems. ZEOBUILDER has all of the features needed to act as a generic molecular editor that is also capable of manipulating organic species, for example, ligand-substrate coordination, internal rotations, pseudorotations, and so forth. Transferability is a key property of the new toolkit.

The structure of the paper is as follows: Section 2 outlines the main features implemented in ZEOBUILDER that make it a unique tool from the viewpoint of the user. It offers the reader a first impression of the wealth of possibilities embedded in the code when building and manipulating the most complex molecular systems. Section 3 is devoted to the methodology of some key operations like condensation reactions of zeolite blocks. The next section shows how user-friendly ZEOBUILDER is. With the help of three examples the reader gets started with a list of step-by-step instructions. Finally, in section 5, some practical instructions are given on how to obtain the latest version of ZEOBUILDER and how to get support.

2. MAIN FEATURES

A screenshot of the user interface of ZEOBUILDER is displayed in Figure 1. The large display window in the bottom right gives a 3D visualization of the molecular system. At the left, a tree structure is displayed with a list of all objects constituting the molecular model. The menu bar on top of the display gives access to all noninteractive tools to modify the displayed molecular model. The interactive tools that operate directly on the 3D view are configured with the help of the four toolbar buttons in the top left of the window. Each toolbar button associates an interactive tool with a combination of modifier keys (shift and ctrl) on the keyboard. For example, when the user drags with the mouse button while holding the shift key pressed, (a part of) the model is translated. The default settings are as follows: when no keys are pressed, the selection tool is active. When the shift key is pressed, one performs translations in the 3D view. Similarly, the ctrl key is associated with interactive rotations. When both the shift and the ctrl key are pressed, one translates the global rotation center, or one changes the scale of the 3D display by zooming in or out. These interactive functions are not fixed. Other interactive tools, such as a measurement tool and a sketch tool, are configured by clicking on one of the four toolbar buttons.

ZEOBUILDER can be started up without any preprogrammed input file. Molecules can be built on a free basis. If desirable, any XYZ file can be used as input facilitating the construction of complex molecular structures. However, the default file format is ZML, which is an internal format based on the XML standard. It is flexible toward future extensions, and it is capable of storing all aspects of a ZEOBUILDER model. Other formats such as PDB and XYZ are also supported, but these formats cannot include all aspects of a molecular structure constructed with ZEOBUILDER. In addition to loading and saving models with different file formats, one can also import a file into the current model or one can export a part of the current model into a file. For the user's convenience, a library is added containing all preprogrammed ZML files of unit cells for all commonly used zeolite frameworks (IZA database[12]). At any time, molecules or parts of molecular systems can be saved in the desired format (XYZ, PDB, or ZML).

The molecular structure is visualized by balls with their size and color adapted to commonly used conventions. An algorithm is built in to optionally display bonds between atoms. The procedure on how to carry out this operation is outlined in section 4. Sometimes, it can be useful to use a less-cluttered representation omitting the oxygens, and straight lines are drawn connecting the tetrahedral atoms. An illustration is given in Figure 2, where a sodalite cage unit is displayed in an FAU framework.

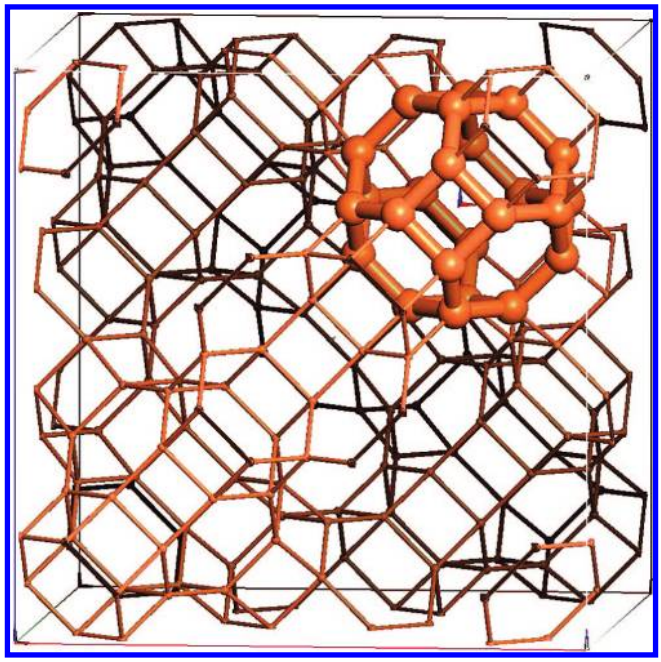

Figure 2. The sodalite cage unit in the FAU framework. Oxygen atoms are omitted, and straight lines are drawn connecting the tetrahedral (T) atoms.

One of the essential concepts of ZEOBUILDER is the novel data structure that represents the molecular model. The user can define a hierarchical structure of reference frames in which molecules or molecular building blocks are placed. This feature is especially useful for constructing models based on building blocks. On the top of the hierarchy stands a space-fixed reference frame. Molecules and other body-fixed reference frames can be inserted freely. They have subhierarchical character. The number of hierarchical levels of the reference frames is unlimited. They all can again serve as containers of molecular structures. When one body-fixed reference frame is submitted to a geometrical transformation (rotation, translation, and inversion), all of the objects contained in the frame will undergo the same transformation.

The hierarchical structure of the reference frames is also practical for other complex models like ligand−receptor systems and coordination complexes. This tranferability is one of the main advantages associated with ZEOBUILDER.

For the convenience of the user, a series of features that are common to usual editor tools has been implemented (be it a word processor or drawing software). Each operation in ZEOBUILDER can be undone, redone, and repeated without limitations. One can cut, copy, and paste parts of molecules or any other group of objects. Additionally, all of the attributes of each object or group of objects can be inspected and modified.

ZEOBUILDER also incorporates some special basic features allowing complex transformations not present in current graphical toolkits. Unique is the ability to insert arbitrary points, vectors, planes, and so forth at any place in space and to associate a specific function to it. A point in space can be regarded as a rotation center or an inverse center, a vector as a rotation axis, a plane for a reflection operation, and so forth. One selects the object (it can be part of the molecular system), and all geometrical operations can be executed. To illustrate with an example, we demonstrate an internal rotation of part of a molecule about an axis. First, that part of the molecule is selected and stored in a separate frame. Next, the rotation axis is determined by a vector defined by two selected atoms. By means of the right interactive tools, the rotation is easily performed.

The most extended and most innovative feature of ZEOBUILDER, which makes it unique and different from other available graphical GUIs, is the multiple builder tools built into the code. Two zeolite building blocks can merge to one larger system with the help of condensation reactions. The mechanism is illustrated in Figure 3. The building blocks are assumed to be rigid structures, and a connection between two building blocks is obtained by rotating and translating one of the two blocks in an attempt to search for a possible set of oxygen pairs that fulfill requirements for a successful condensation process. Terminating hydrogens are hidden in order to make the figure more transparent. The selected oxygen pairs are then connected with a spring, as shown in Figure 3. Introducing springs in a structure always precedes some optimization procedure. In this condensation reaction, the rigid body optimization brings together the two merging blocks until a situation is obtained with partially overlapping oxygens (those connected with a spring). The algorithm applied to perform this condensation will be described in the next section. Finally, each pair of partially overlapping oxygens is replaced by one bridging oxygen atom that connects the two building blocks. In this way, one of the multiple conformations is constructed by merging the two

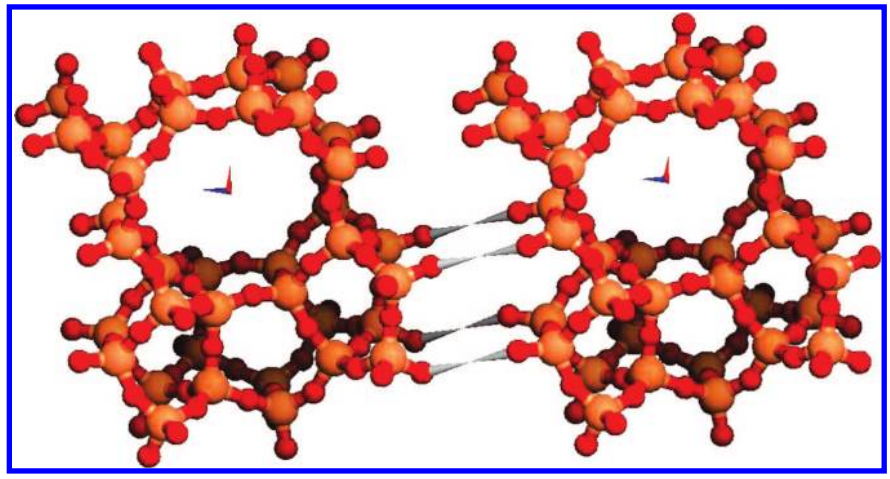

Figure 3. Illustration of how the condensation takes place after choosing manually some pairs of merging oxygens in the two building blocks.

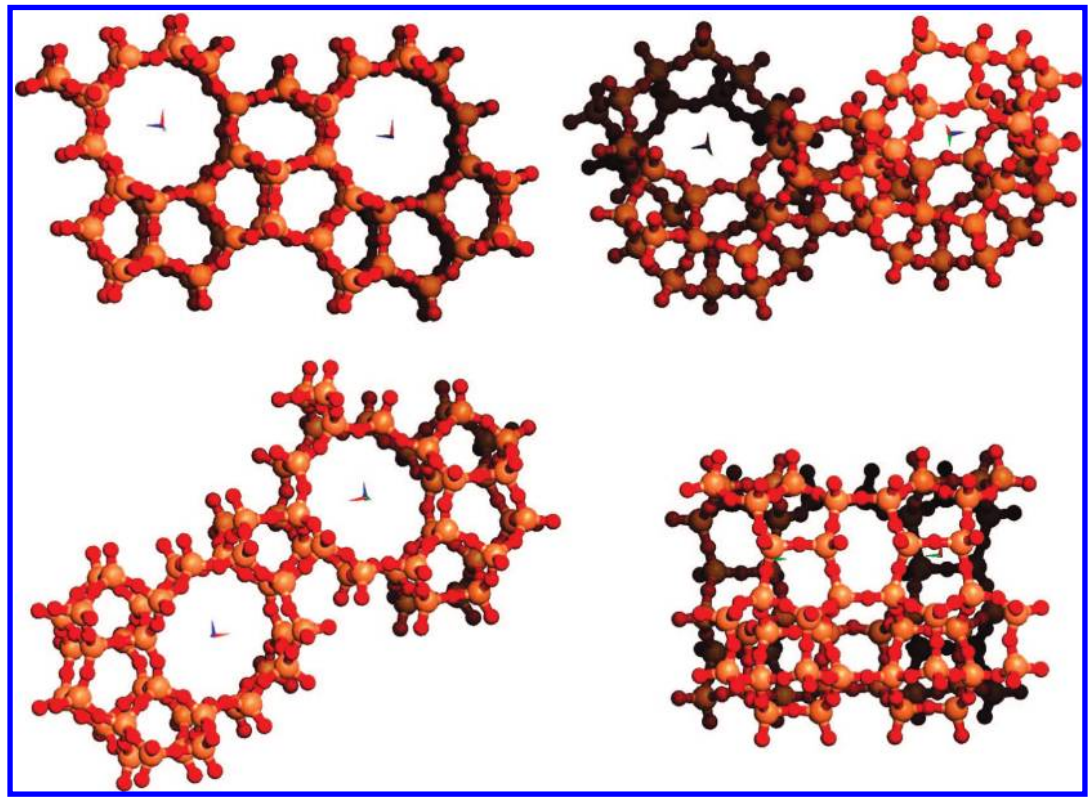

Figure 4. Four plausible conformations after a condensation scan of two zeolite building blocks (MFI precursors[19–21]).

zeolite blocks. The manual search for possible pairs of oxygens appropriate for condensation can be very time-consuming. So, an automatic scanner of all plausible condensation reactions is built in and leads to a series of conformations ordered following a quality factor, which depends on a series of parameters mostly related to steric hindrance and other geometry effects resulting from the merging process. This quality factor reflects the plausibility of formation of the particular conformation.

As an example, in Figure 4, we show four plausible compositions (conformations) resulting from a scan of condensation reactions between two MFI precursors.[19–21] This procedure can be repeated without any limitation and opens perspectives to construct new zeotype materials with different microporosities and topologies of the framework. Obviously, the applicability of this building ability can be easily extended to all types of inorganic nanosized building blocks.

The concept of springs, as introduced in the building process of molecular blocks, can be extended to other applications. A spring can be added to keep a particular (organic) molecule or cations localized in a channel or cage of a zeolite framework, after which an optimization can be performed. There are also other applications outside the realm of zeolite science: for example, in building complexes of molecules where some geometrical constraints are imposed pushing the complex in a well-directed local minimum of the PES, the concept of springs can be very useful. In addition, these tools are very helpful in creating input files for molecular modeling program packages such as Gaussian 03,[22] ADF,[23,24] CPMD,[26] CP2K,[27,28] GULP,[29] CHARMM,[30] GROMACS,[31] GROMOS,[32] LAMMPS,[33] NAMD,[34] and so on. During the past few decades, many force-field models have been developed that are applicable to zeolite systems. In particular, the Catlow library[35] and the parameters derived by Sauer et al.[36,37] are widely used.

In addition to these advanced model manipulations, ZEOBUILDER also contains a series of suitable tools to incorporate elements such as B, Ge, Zn, P, and transition elements into the framework creating different molecular sieves. Additionally, ZEOBUILDER also supports one-, two-, and three-dimensional periodic systems and the tools to transform a periodic system into a cluster, to create super cells, or to wrap a cluster model into a periodic box.

All of the features above focus on model manipulation; that is, given one or more input structures, one can modify and compose these structures into any reasonable model that serves as input for computational chemistry packages. ZEOBUILDER also contains tools to extract (nontrivial) geometrical or topology-related information from a molecular model. With one of the menu options, one can also make some statistical analysis on bond lengths, bending, dihedral angles, and so forth in a molecule or parts of a molecule. One can take into account all internal coordinates in such an analysis, or one can impose a specific subset of internal coordinates. Another interesting plug-in module is the counter tool. To illustrate with an example: in a complex zeolite structure, an interesting piece of information could be the number of TO_4 tetrahedra in the model. Due to the finite size of zeolite models, the T-centra are not all connected with four bridging oxygens, and different silicon environments are present. It is not obvious to extract the distribution of the various Q1, Q2, Q3, and Q4 centers.

A plug-in is also implemented in ZEOBUILDER to count the number of rings in the framework, which is not a trivial task especially in large structures. Pore sizes are also easily measured.

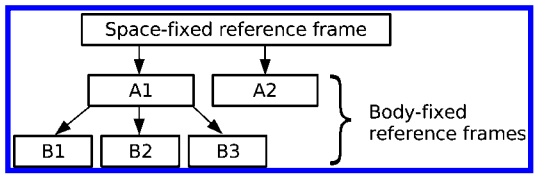

Figure 5. A schematic example of a hierarchical structure of reference frames.

3. CONCEPTS AND ALGORITHMS

3.1. Hierarchical Structure—Multiple Reference Frames. The internal data structure, employed by ZEOBUILDER to represent the molecular model, is based on a hierarchical concept and enables the development of diverse builder algorithms in the program.

In most builder programs, the standard approach is to define all atom coordinates with respect to one space-fixed reference frame. ZEOBUILDER abandons this concept by introducing multiple body-fixed reference frames to be defined in a hierarchical structure. Each frame contains a building block or an aggregation of building blocks. The builder algorithms in ZEOBUILDER will manipulate these reference frames instead of the individual atoms. A schematic example is given in Figure 5. In this example, the space-fixed frame contains two body-fixed frames (A1 and A2), and A1 is again composed of three other body-fixed reference frames (B1, B2, and B3).

Each frame contains a list of objects such as points, subframes, and so on. Within the terminology of ZEOBUILDER, these objects are called children as elements of the frame (the parent). Each object has a coordinate, t, with respect to the parent frame. Body-fixed reference frames also store a rotation matrix, **R**, that describes the orientation relative to the parent frame. The absolute position of an atom C1 in reference frame B2 is then given by $t_{\mathrm{C1,\,abs}} = R_{\mathrm{A1}}(R_{\mathrm{B2}}t_{\mathrm{C1}} + t_{\mathrm{B2}}) + t_{\mathrm{A1}}$. The notation and the implementation of this concept is largely facilitated by merging the rotation matrix and translation vector of each reference frame into a single transformation matrix, **U**, with the following structure:

$$U_{\mathrm{A1}} = \begin{bmatrix} R_{\mathrm{A1}} & t_{\mathrm{A1}} \\ 0 & 1 \end{bmatrix} \tag{1}$$

This reduces the expression for the absolute position of an atom C1 in reference frame B2 to

$$\begin{bmatrix} t_{\mathrm{C1,abs}} \\ 1 \end{bmatrix} = U_{\mathrm{A1}}U_{\mathrm{B2}} \begin{bmatrix} t_{\mathrm{C1}} \\ 1 \end{bmatrix} \tag{2}$$

Similarly, one can express the position of atom C1 relative to the reference frame A2:

$$\begin{bmatrix} t_{\mathrm{C1,A2}} \\ 1 \end{bmatrix} = U_{\mathrm{A2}}^{-1}U_{\mathrm{A1}}U_{\mathrm{B2}} \begin{bmatrix} t_{\mathrm{C1}} \\ 1 \end{bmatrix} \tag{3}$$

This data structure also has a major technical advantage; that is, the accelerated graphics hardware that is responsible for the 3D visualization of the model can efficiently process a ZEOBUILDER data structure because the hardware implementation is also based on algebra of four-dimensional vectors and a hierarchical structure of reference frames.

3.2. Condensation Algorithm. One of the main operations in building nanosized zeolite structures is the polycondensation reaction of two zeolite blocks. The mechanism is shown in Figure 6. Two or more terminating oxygen atoms are selected in both blocks A and B. By rotating and translating the rigid body B, one can find a situation where sets of oxygen pairs fulfill all requirements for a successful condensation process. The two blocks are connected by the newly formed oxygen bridges (see Figure 6a). The algorithm consists of fine-tuning the position of the bridging oxygen atoms so that they are exactly coinciding. Condensation takes place with a release of water molecules. Hydrogens are hidden in order to make the visualization of the process more transparent. In the case of a condensation reaction with only two oxygens, a degree of freedom is left (rotation of block B about the O−O axis). For three or more coinciding oxygens, the position of block B is definitely determined. Due to the fact that the building blocks are handled as rigid structures, the overlap will only be approximate. An algorithm is developed that minimizes the cost function J defined as the sum of the squared O−O distances that are expected to coincide. An example of two building units with two oxygen pairs logically connected is illustrated in Figure 6b. The algorithm as implemented in ZEOBUILDER joins the two blocks by minimizing the cost function J. At the end of the operation, a minimum value of J remains. This minimum value is displayed, and the user is free to decide whether the condensation reaction is accepted. Next, a fine-tuning mechanism is put into operation in order to induce exact overlap of the siloxane bridges. This condensation mechanism creates a new conformation consisting of the two originally separate building blocks.

We have illustrated the condensation algorithm in a specific polycondensation reaction of zeolite blocks, but it should be stressed that the built-in algorithm can be applied to any kind of elimination reaction.

3.3. Multiple Scan of Various Conformations. The manual selection of possible condensation reactions can be of some utility but is not of practical use when multiple condensation reactions belong to the possibilities. The search for all possible conformations formed by merging two blocks A and B is automated in ZEOBUILDER. A conformation scanner has been built in that finds out all possible sets of oxygen pairs c_{ik} ($k = 1, K$) that fulfill requirements for a successful condensation process (label i defines the specific conformation under study). It is not the intention of the authors to go into detail how the algorithm is set up, but the main strategy is based on the search for equivalent or nearly equivalent triangles formed by three oxygen atoms in both blocks A and B. The triangles should overlap with each other within a fixed threshold value (cost function J represents a highly suitable measurement instrument). Other constraints are also built in to avoid unphysical constructions. For example, the merging of triangles formed by oxygens at large distances from each other generates spurious situations with tetrahedra in an unrealistic spatial ordering in most of the cases. All of these spurious constructions should be removed, and ZEOBUILDER only retains N suitable conformations, completely determined by the set of three or more oxygen pairs c_{ik} ($k = 1$, K). Each conformation is stored by means of the transformation matrix belonging to the rotation and translation of block B to form the actual conformation (see subsection 3.1). A search is performed to remove all duplicates leading to similar conformations. A course quality

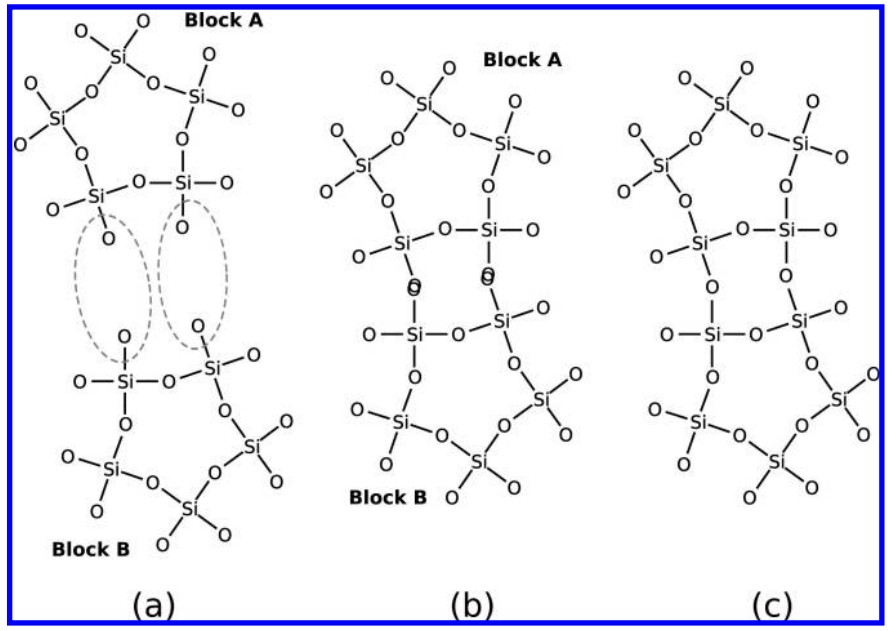

Figure 6. A schematic illustration of the condensation reaction.

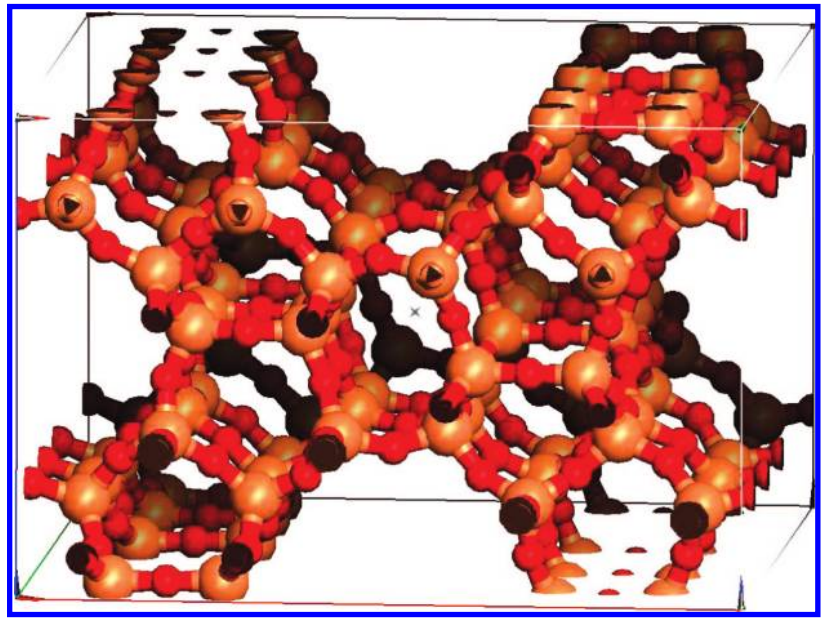

Figure 7. The unit cell of silicalite-1 (MFI).

factor is calculated for each conformation, and the results are sorted by this quality factor.

This procedure can be repeated with a third block, C, merging with one of the retained conformations $[AB]_i$. In this way, ZEOBUILDER is able to build nanoscale zeolite structures.

4. HOW TO WORK WITH ZEOBUILDER

4.1. Example 1: A Model for Zeotiles. In the first example, we focus on zeotiles, which are crystalline silica materials with two levels of porosity and structural order, and which are built from nanoslabs of uniform size with the Silicalite-1 zeolite framework generated by the TPA template.[38] Different tiling patterns have been imaged by high-resolution electron microscopy,[39] and they form a suitable and advanced application of ZEOBUILDER in an attempt to model the observed tiling patterns. The initial building block of an MFI nanoslab is the MFI precursor described in the work of Kirschhock et al.[19–21] In these papers, the mechanism of formation of the nanoslabs from the precursor is discussed.

First, we demonstrate how the MFI precursor as a building block is derived from the silicalite-1 unit cell. Next, nanoslabs are built from these building blocks. Finally, nanoslabs are tiled into the characteristic hexagonal pattern of Zeotile-1.[39]

Construction of the MFI Precursor. With the menu function "File" > "Open", one loads the Silicalite-1 unit cell from the IZA library[12] (see Figure 7). The space-fixed reference frame is represented by a right-hand coordinate system in the lower-left corner of the unit cell. By convention, the *x* axis is colored red, the *y* axis green, and the *z* axis blue. As this unit cell is rectangular, the *x*, *y*, and *z* axes coincide with the crystal directions *a*, *b*, and *c*. In the initial 3D view on the structure, the *z* axis lies perpendicular to the screen. In the tree display, all atoms of the unit cell are labeled. The next step is the transformation of the unit cell

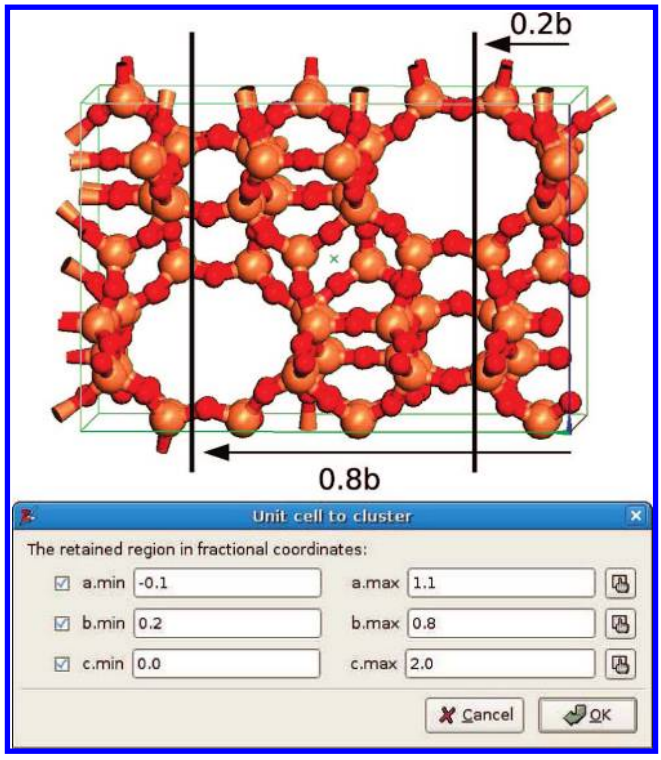

Figure 8. The Sodalite-1 unit cell viewed along the 100 direction. The selected interval along the 010 direction is indicated by vertical lines.

into a cluster that is large enough to contain at least the desired MFI precursor, as suggested by Kirschhock et al.[21,40] From this cluster model, one can remove the redundant atoms with the interactive selection tool. For the transformation of a periodic model into a cluster, one must specify the fractional coordinates of the region in the unit cell that should be retained. In order to estimate this region, it is instructive to rotate the unit cell in all directions with the interactive rotation tool and to examine in particular the crystal views along the three crystal directions. One proceeds as follows:

1. Select the global reference frame, that is, the top item in the tree structure.

2. Click "Object" > "Unit cell" > "To cluster".

3. A dialogue box appears, prompting the user to specify the cutoff values in fractional coordinates in the three crystal directions (see Figure 8). Click OK.

4. A transformation of the space-fixed frame to the center of mass is desirable for further operations, and this is carried out by selecting the frame and clicking "Object" > "Transform" > "Center of mass frame". This menu option keeps the crystal axes in the original directions.

5. The redundant atoms can now be removed with the interactive selection tool. Use the left mouse button to select a part of the model. All objects in the drawn rectangle become selected. Alternatively, one can also point and click at objects. Additionally, one can also use the middle mouse button to extend the current selection. Similarly, the right mouse button can be used to remove parts from the selected region. The function "Edit" > "Delete" will remove the selected objects from the model. Once the MFI precursor is constructed, it can be saved for further applications.

Construction of an MFI Nanoslab. Following the lines given in refs 19–21 for the construction of a half-nanoslab, we start with joining three building blocks (MFI precursors) along the *c* direction of the MFI structure. This is achieved with the following steps:

1. It is desirable to arrange the MFI precursor in a separate body-fixed reference frame. This is extremely useful when introducing multiple precursors in a building process. They can each individually be submitted to all geometrical manipulations. To carry out this operation, first select the global reference frame. Then, choose the menu option "Select" > "Children" followed by "Object" > "Arrange" > "Frame".

2. Then, this frame (and its contents) is duplicated. Select the body-fixed reference frame, and click "Edit" > "Duplicate".

3. Select the duplicated frame, and click "Object" > "Transform" > "Translate". A dialogue box prompts the user to specify the desired translation. Enter the 001 cell vector ($t.x = 0$ Å, $t.y = 0$ Å, and $t.z = 13.42$ Å), and click OK.

4. A third duplicate of the precursor is created in a similar way, but now the duplicate is translated in the opposite direction. With a translation over the exact periodicity in the *c* direction, a situation is created with physically distinct but completely overlapping oxygens. It is obvious that one oxygen atom should be removed from each duo, but we will postpone this operation until the end of this example.

If necessary, the scale of the 3D view can be modified with the interactive zoom function, that is, keep the Ctrl and the Shift keys pressed while dragging with the left mouse button in the 3D view.

In a second step, this row of three precursors is again arranged in a separate reference frame, and this new frame is duplicated. The duplicate is reflected with respect to a plane orthogonal to the *b* direction. Carry out following operations:

1. Select the three precursor frames, and activate the menu function "Object" > "Arrange" > "Frame".

2. Place the view of the three-precursor ensemble orthogonal to the *b* axis. In this position, the atoms at both the left and right edges are aligned.

3. Select all of the atoms in one of the two edges, and click "Object" > "Add" > "Plane". The reflection plane is created and is visualized on the display.

4. Select the frame that contains the three MFI precursors.

5. Duplicate this frame with "Edit" > "Duplicate".

6. Select the duplicated frame, and add one of the two reflection planes to the selection.

7. Perform the reflection by choosing "Object" > "Transform" > "Reflection". A dialogue window with the reflection parameters pops up. The numbers are filled in automatically, on the basis of the selected reflection plane. Click OK.

8. Select the reflection planes, and delete them with "Edit" > "Delete".

9. Select both the original and the reflected reference frame, and click "Object" > "Arrange" > "Frame". As indicated above and shown in Figure 9, there are two possible reflection planes, and both options result in different (pure MFI) geometries for the half-nanoslab.

Construction of Zeotile-1. Following the description as given in the work of Kremer et al.,[39] two half-nanoslabs are now joined into face-sharing double units, measuring 2.6 nm × 2.0 nm × 4.0 nm. As we succeeded in building two types of half-nanoslabs, the construction of a double unit is not unique, even when this double unit must exhibit a proper MFI topology. In

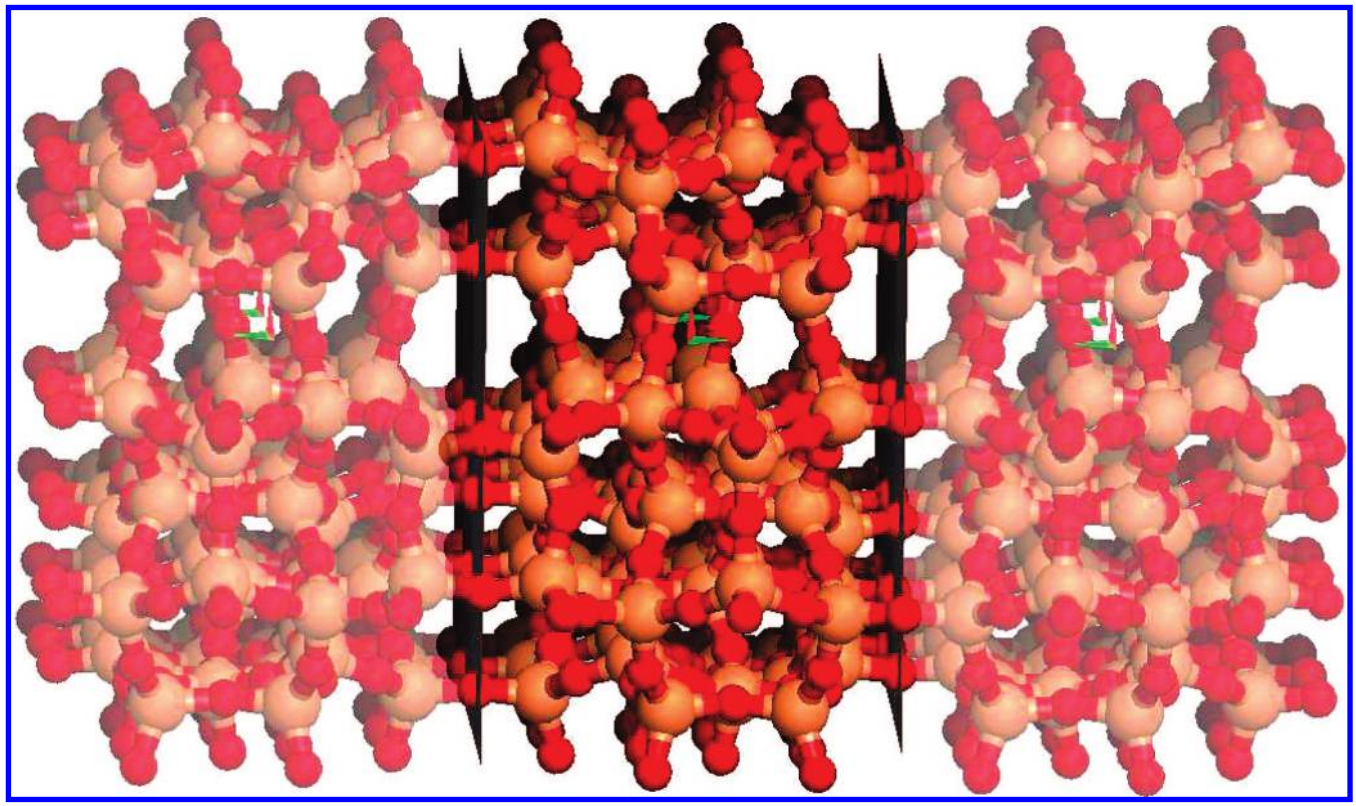

Figure 9. The two possible mirror planes for duplicating the frame with three precursors.

this example, we restrict ourselves to a double unit constructed by identical half-nanoslabs. In this configuration, the two dense edges of both half-nanoslabs—edges orthogonal to the *a* direction of the MFI structure—are shared. This can be performed by the following steps:

1. One chooses and creates a reflection plane containing all terminating oxygens at the dense side of the half-nanoslab.

2. Duplicate the whole construction. Carry out a reflection of the duplicated frame with respect to the reflection plane.

3. Eliminate the reflection planes.

4. The final structure is stored in a separate reference frame. Finally, in order to create a hexagonal pattern based on the double units, some more advanced techniques are required. First, two duplicates of the double unit are rotated +120° and −120°, respectively, along the long direction (*c* axis) of the double unit. The rotation of one double unit is carried out with the following steps:

1. One first defines the rotation axis parallel to the *c* direction of the MFI structure by connecting two atoms with a vector. On the basis of the structure of the MFI precursor, a pair of atoms that defines the correct rotation axis can be easily selected.

2. In order to rotate a double unit, one first selects the reference frame that contains the double unit, and then one adds the rotation vector from the previous step to the selection. The menu function "Rotate about axis" first shows a popup dialogue that contains all of the rotation parameters (rotation axis, rotation center, and rotation angle). One only needs to give an input value for the rotation angle since the other parameters are filled in automatically. The rotation axis is completely determined by the selected vector. After clicking the "OK" button, the double unit is rotated.

3. A similar operation is carried out with the third double unit, but now a rotation over −120° is performed. The three double units can now be moved into a triangle in such a way that only at the corners are oxygen atoms overlapping. This is illustrated in Figure 10. A condensation reaction then makes a new structure. The conformation shown in the figure is not the unique one. There are probably others, but we gave preference to a regular triangle structure with an edge of approximately 2.0 nm.

The next operation is a periodic extension of the triangle of three double units to create the ultime hexagonal/triangular pattern of Zeotile-1. One can simply define the new unit cell vectors by drawing them in the model. Two half-unit cell vectors can be obtained by connecting the centers of the double units with a vector (see Figure 10). The lengths of these vectors can be doubled. The third unit cell vector is orthogonal to the first two, that is, parallel to the *c* direction in the MFI structure or the rotation axis defined above. It can by obtained by drawing a vector from the origin of the reference frame of one MFI precursor to the origin of the reference frame of a second MFI precursor that is translated along the *c* direction with respect to the first one. To finalize the model, one takes the following steps:

1. Select the three vectors that define the unit cell, and click "Object" > "Unit cell" > "Define unit cell vector(s)". They are indicated as "Arrow" in the tree display.

2. The initial building blocks (MFI precursors) are still stored in a hierarchy of separate reference frames, which is impractical for further computational applications. By choosing the menu option "Object" > "Arrange" > "Unframe" for all of these reference frames, all of the atoms are directly positioned in the global reference frame.

3. The overlapping oxygen atoms that connect the building blocks are merged with the menu option "Merge overlapping atoms".

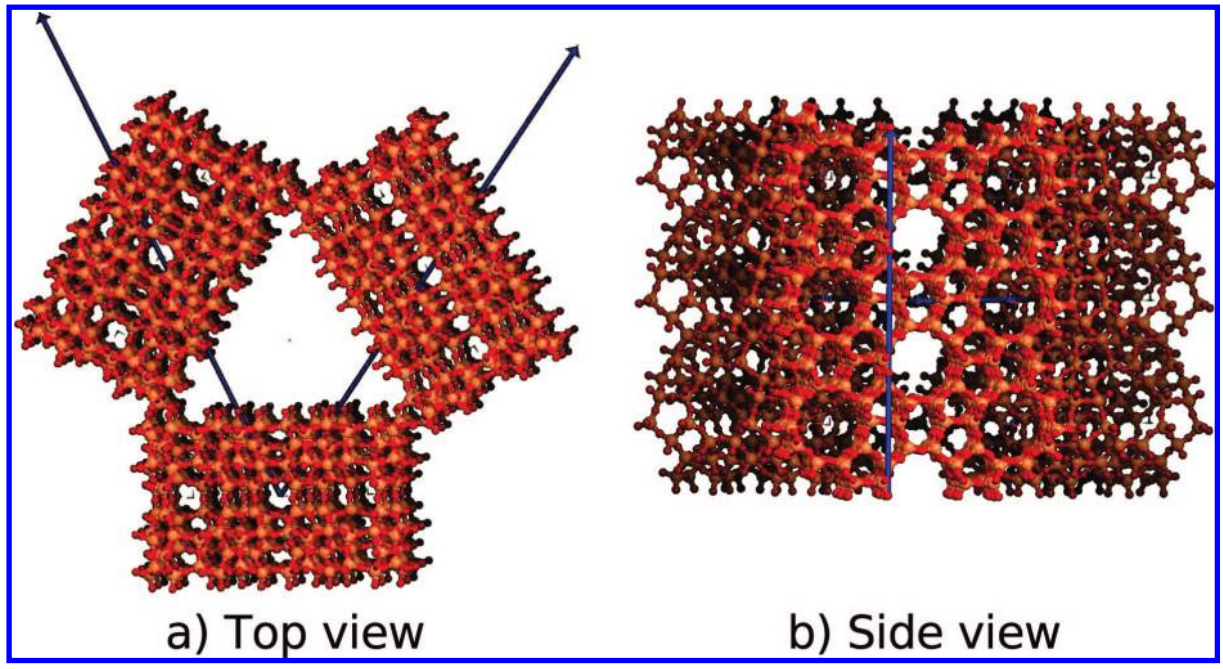

Figure 10. Two visualizations of the triangular structure, which is the basis for the zeotile unit cell. The cell vectors are shown as blue arrows.

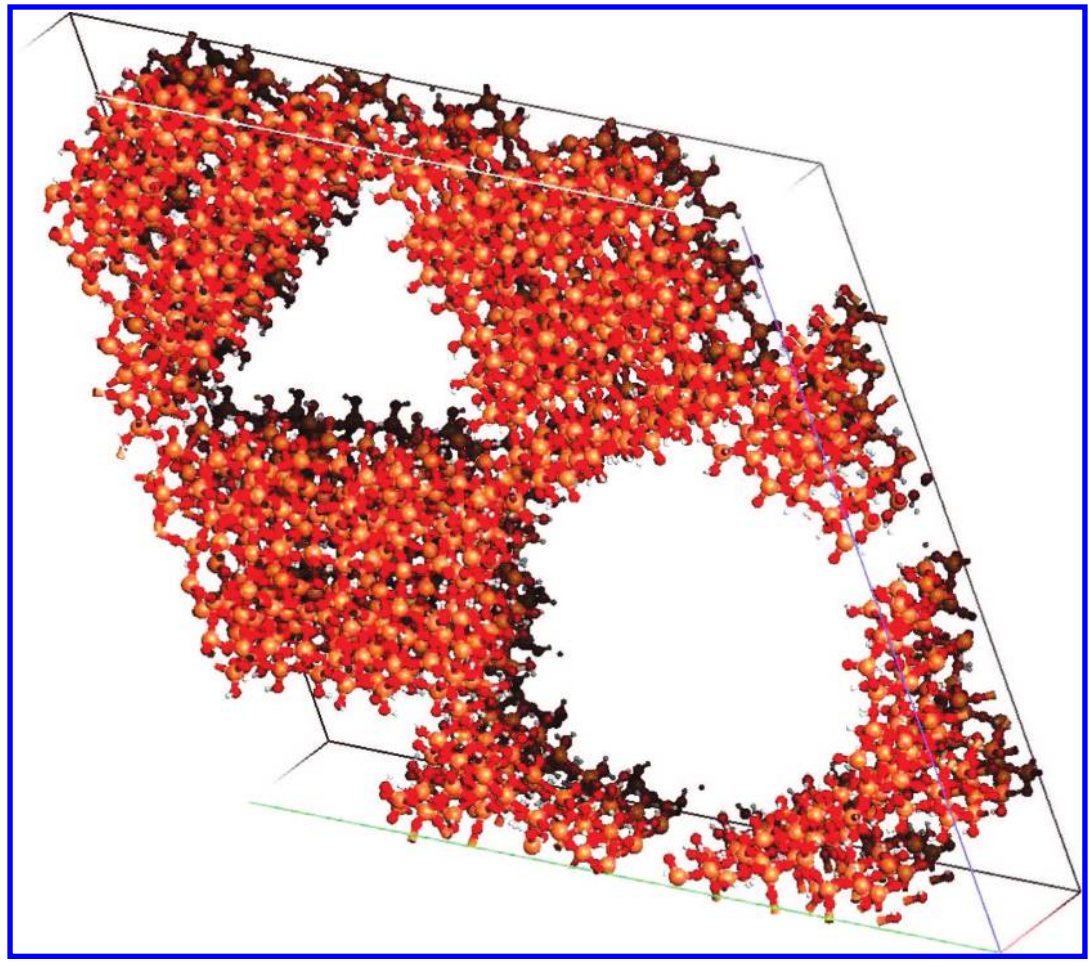

Figure 11. 3D view of Zeotile-1 constructed in example 1.

4. The final structure still contains many dangling oxygens. One can turn the model into a hydroxyl-terminated structure with the function "Saturate with hydrogens". In this way, a zeotile model has been built, and the result is seen in Figure 11.

4.2. Example 2: Aligning a Guest Molecule in a Zeolite Channel. This example will demonstrate how one can add a pentane molecule exactly in the center of the straight channel in a silicalite-1 model, with the pentane molecule aligned along the axis of the straight channel. One

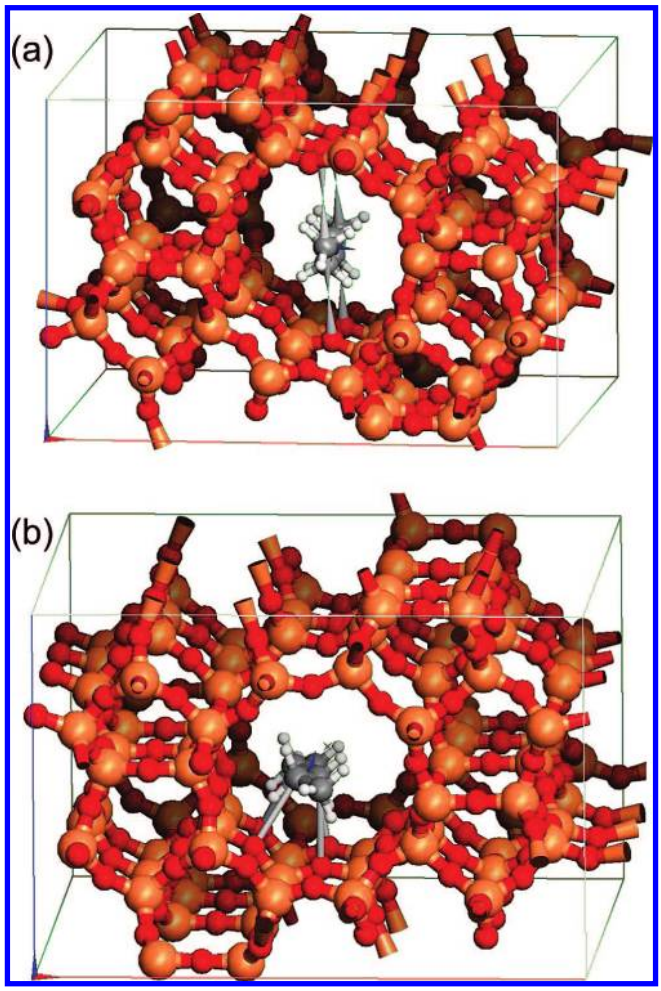

Figure 12. Embedding a pentane molecule in the MFI framework.

first loads the unit cell of Silicalite-1 (MFI) as in the preceding example. As shown in Figure 7, the straight channels of the MFI topology are located at the edges of the periodic box. It is instructive to carry out a translation of the reference frame to another frame with the origin at the center of one of the two straight channels in the Silicalite-1 unit cell.

1. Select the global reference frame.

2. Add a new point to the model with "Object" > "Add" > "Point". The point is automatically selected.

3. Modify the coordinates of the new point to (10 Å, 10 Å, 0 Å) with "Object" > "Properties". A dialogue box opens; choose the tab page Translation, and enter the translation coordinates. These coordinates approximately correspond with the origin of the new desired reference frame.

4. Apply the menu option "Object" > "Transform" > "Define origin". The display of the unit cell is now adapted to the new reference frame. The advantage of introducing a new origin of the reference frame is manifest. With the rotation tool, one easily rotates the model until the view coincides with the *ac* crystal plane. The 10T ring in the straight channel is clearly seen (see Figure 12).

The second step is to import the pentane molecule into the model with the menu function "File" > "Import". The molecule is loaded into a separate reference frame. This frame can be manually placed and oriented at a position where we want to embed the pentane molecule. This can be

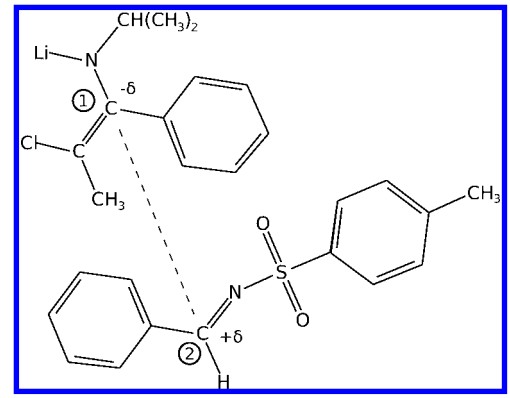

Figure 13. Schematic structure of the reaction wherein the carbon atom bonded to the nitrogen in a tosyl imine conducts an electrophilic attack on the carbon atom next to the chlorine in a lithiated 3-chloro-1-aza-allylic anion

carried out using the interactive rotation and translation tools. The second step makes use of the spring tool, as already outlined: the two ending C atoms of the pentane molecule are connected with oxygen atoms in the wall of the straight channel with springs, as shown in Figure 12a. One can easily introduce springs with the interactive sketch tool. Finally, one optimizes the springs with preset rest lengths. The location of the pentane molecule in the channels of the framework is not unique: one can connect the ending C atoms of the pentane molecule with different oxygen atoms in the zeolite channel, or one can specify different rest lengths, as shown in Figure 12b. All of these settings yield suitable geometrical structures that can be used as input for geometry optimizations in a variety of ab initio packages. This exercise leads to the quantitative description of the adsorption of pentane in the channels of a silica zeolite with MFI topology. One can also use these models as the initial geometry for a molecular dynamics simulation to study the diffusion of pentane in Silicate-1.

4.3. Example 3: Manipulating Organic Structures. The third example demonstrates how ZEOBUILDER can be applied in building molecular constructions beyond zeolite science. We have chosen an application that many computational chemists are confronted with when studying chemical kinetics. The search for the true transition state can be a time-consuming process, not only from the computational viewpoint. Also, the preparation of suitable input files with trial geometrical structures for the near transition state is a difficult task. In chemical reactions with complex reactants, the current graphical tools are not always appropriate for drawing starting geometries. The example below is an aldol-like reaction from the heterocyclic organic chemistry. More specifically, we build a near-transition-state structure of the reaction—displayed in Figure 13—wherein the carbon atom bonded to the nitrogen in a tosyl imine performs an electrophilic attack on the carbon atom next to the chlorine in a lithiated 3-chloro-1-aza-allylic anion. Once this model has been constructed, it can be used as the input geometry for a transition-state optimization in various quantum chemistry programs. The three active species are illustrated in Figure 14. The Li atom can coordinate with a lot of atoms from the two reactants. In principle, all combinations should be tried out. In this example, we take into consideration the situation where the Li atom is coordinated by one oxygen atom of the tosyl imine and by the nitrogen and chlorine

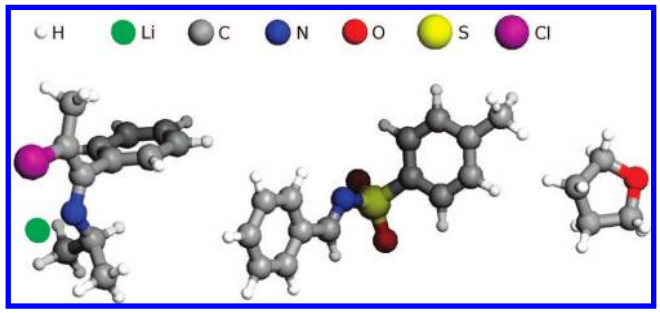

Figure 14. The three active species playing a role in the reaction under study. From left to right: the aza-allylic anion, the thosyl imine, and tetrahydrofuran (THF).

atoms in the aza-allylic anion. As input, we load the optimized structures for the lithiated 3-chloro-1-aza-allylic anion and the tosyl imine. It is essential that both molecules reside in their respective frames, so that they can be submitted to independent rotations/translations—each molecule keeping its internal structure. Good initial guesses for interatomic distances are 2 Å for the Li−O distances and 4 Å for the C−C distance of the forming bond. With the help of the interactive toolbar buttons, the two molecules are transformed in an acceptable position taking into account the estimates of coordination distances. But if too many constraints are imposed, it is more convenient to make use of the spring tools of ZEOBUILDER to facilitate the construction of a proper initial geometry. This is illustrated by the following steps:

1. Using the proper interactive tools, we bring the tosyl imine molecule into position with one of the two oxygens of the imine approximately at a coordination distance with the Li atom.

2. Then, we select the imine frame and consequently add the Li atom to the selection. This selection preserves the Li−O distance during an interactive rotation in the 3D view.

3. During the rotation of the imine frame, we bring the carbon atoms of the forming bonds closer to each other, but we prevent the overlap of the imine molecule with the aza-allylic anion. One gets a structure as displayed in Figure 15a.

4. We now fine-tune the relative orientation of the reacting species. First, we connect the atoms with the prescribed interatomic distances by springs. With the function "Object" > "Properties", we can configure the rest lengths of these springs and carry out a rigid body optimization. This procedure can be repeated until a satisfactory structure is obtained.

5. Other geometrical manipulations can be carried out, keeping all coordination distances unaltered and preventing many reoptimizations, by rotating about an axis formed by the coordinated oxygen and the C atom of the imine (displayed in Figure 15b). This is achieved by introducing an arrow between the two atoms. By rotating the imine about this axis, we can also fine-tune the relative orientation while the lengths of the springs remain constant. In this way, a near transition state is constructed that can serve as an input file in any ab initio molecular modeling package in search of the true transition state.

The case can even become more complex when taking into account solvent molecules. With ZEOBUILDER, the most complex configurations can be handled in an easy way. We first arrange the two reacting species into one new frame. This assures that their relative orientation will not change

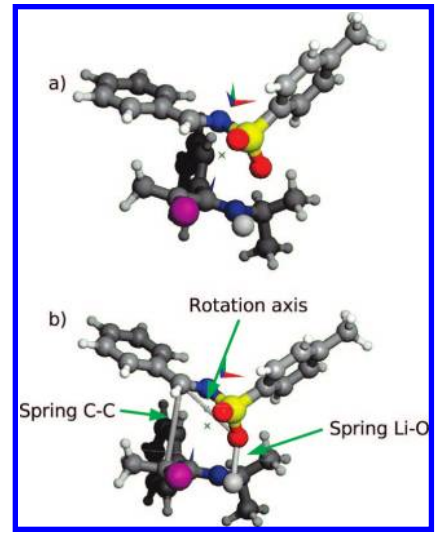

Figure 15. Two intermediate phases in the construction of the input geometry for a transition state optimization. The initial structure (a) is obtained through the interactive functions in ZEOBUILDER. The fine-tuning (b) is based on the spring optimization and a rotation that does not alter the lengths of the springs.

by further spring optimizations. Then, we import a THF molecule and connect the oxygen and the Li atom by a spring. We set the rest length of this spring to 2 Å, and we optimize the spring. The THF molecule can now be oriented with the interactive rotation tool, using the oxygen or the Li atom as a rotation center.

5. PROGRAM AVAILABILITY

Zeobuilder is distributed as open-source software under the conditions of the GNU General Public License, version 3. The software can be downloaded from the code Web site of the Center for Molecular Modeling: http://molmod.ugent.be/code. Documentation, tutorials, example files, installation instructions, and technical support are also available on this Web site.

6. SUMMARY AND OUTLOOK

The continuous advances in computational chemistry and zeolite modeling in particular have increased the demand for a molecular builder toolkit that fulfills all criteria of modern software. ZEOBUILDER is a strong answer to this real need for molecular editors that are used for the preparation of ab initio and molecular mechanics simulations.[43,44]

It is a common practice to describe complex molecular structures, for example, zeolite frameworks, in terms of polyatomic building units. ZEOBUILDER's hierarchical data structure of reference frames largely facilitates the construction and manipulation of complex models based on building blocks.

ZEOBUILDER is also an extensible framework for the development of novel and highly specialized builder algorithms. The spring optimization and the connection scanner, which have been thoroughly demonstrated in this paper, are two practical examples of builder tools that have been developed in the ZEOBUILDER framework. Despite our interest in theoretical zeolite modeling, we have carefully designed this framework with a broad range of applications in mind: no assumptions have been made that

could limit the transferability and the feasibility of ZEOBUILDER.

ZEOBUILDER, as presented in this paper, is only a snapshot from a dynamic development process. We plan diverse updates and extensions to ZEOBUILDER that will address some of the remaining challenges in the construction of sophisticated molecular models. A true geometry optimization, even at a low level of theory (e.g., UFF[41] or PM3[42]), would relieve the current limitation of purely rigid building units. For example, the spring optimization and the scanner assume that the building units are approximately rigid. This rigid-body assumption is adequate for the preparation of most molecular simulations, but it will give an incomplete picture of a synthesis process based on (partially) flexible building blocks. A next generation of scanning algorithms is in preparation to surmount these additional difficulties. Another attractive feature would be the stacking of secondary building units into zeolite crystals, optionally including stacking faults.[11,12] Other items on our schedule include an improved interoperability with external simulation codes and support for more file formats via OpenBabel.[18]

Since ZEOBUILDER is released as an open-source platform for the development of molecular builder algorithms, the authors look forward to collaborations with external researchers and developers in this area.

ACKNOWLEDGMENT

This work is supported by the Fund for Scientific Research - Flanders and the Institute for the Promotion of Innovation through Science and Technology in Flanders (IWT Vlaanderen).

REFERENCES AND NOTES

(1) *Cerius2 Modelling Environment*; Accelrys Software Inc.: San Diego, 2006.
(2) *Materials Studio*; Accelrys Software Inc.: San Diego, 2006.
(3) *CrystalMaker*; CrystalMaker Software Limited: Oxfordshire, UK, 2008.
(4) *Diamond*; Crystal Impact: Bonn, Germany, 2008.
(5) Pereira, C. General Atomistic Modelling Graphic Interface. http://www.gamgi.org/ (accessed Mar 25, 2008).
(6) DeLano, W. *The PyMOL Molecular Graphics System*; DeLano Scientific: Palo Alto, CA, 2002.
(7) Jmol: an open-source Java viewer for chemical structures in 3D. http://www.jmol.org/ (accessed Mar 25, 2008).
(8) Ali, S.; Braithwaite, R.; Bunt, J.; Curtis, D.; Hanwell, M.; Hutchison, G.; Jacob, B.; Margraf, T.; Niehaus, C.; Ochsenreither, S.; Vandermeersch, T. Avogadro. http://avogadro.sourceforge.net (accessed Mar 25, 2008).
(9) Agile Molecule. http://agilemolecule.com (accessed Mar 25, 2008).
(10) Smith, J. *Chem. Rev.* **1988**, *88*, 149–182.
(11) Lobo, R. F. *Introduction to the Structural Chemistry of Zeolites. In Handbook of Zeolite Science and Technology*, Auerbach, S., Carrado, K., Dutta, P., Eds.; Marcel Decker Inc.: New York, 2003; pp 65−89.
(12) Baerlocher, C.; McCusker, L.; Olson, D. *Atlas of Zeolite Framework Types*, 6th ed.; Elsevier: Amsterdam, The Netherlands, 2007; pp 1−398.
(13) Lewis, D. W.; Willock, D.; Catlow, C. R. A.; Thomas, J. M.; Hutchings, G. *Nature* **1996**, *382*, 604–606.
(14) Lewis, D. W.; Catlow, C. R. A.; Thomas, J. M. *Faraday Discuss.* **1997**, *106*, 451–471.
(15) Baerlocher, C.; Hepp, A.; Meier, W. *DLS-76*; Institut für Kristallographie und Petrographie: Zürich, Switzerland, 1976.
(16) Bail, A. L. *J. Appl. Crystallogr.* **2005**, *38*, 389–395.
(17) Sheldrick, G. *Acta Crystallogr.* **2008**, *A64*, 112–122.
(18) Guha, R.; Howard, M.; Hutchison, G.; Murray-Rust, P.; Rzepa, H.; Steinbeck, C.; Wegner, J.; Willighagen, E. *J. Chem. Inf. Model.* **2006**, *46*, 991–998.
(19) Ravishankar, R.; Kirschhock, C.; Knops-Gerrits, P.; Feijen, E.; Grobet, P.; Vanoppen, P.; De Schyver, F.; Miehe, G.; Schoeman, B.; Jacobs, P.; Martens, J. *J. Phys. Chem. B* **1999**, *103*, 4960–4964.
(20) Kirschhock, C.; Ravishankar, R.; Van Looveren, L.; Jacobs, P.; Martens, J. *J. Phys. Chem. B* **1999**, *103*, 4972–4978.
(21) Kirschhock, C.; Kremer, S.; Grobet, P.; Jacobs, P.; Martens, J. *J. Phys. Chem. B* **2002**, *106*, 4897–4900.
(22) Frisch, M. J. et al. *Gaussian 03*; Gaussian, Inc.: Wallingford, CT, 2004.
(23) te Velde, G.; Bickelhaupt, F.; van Gisbergen, C. F. G.; Baerends, E.; Snijders, J.; Ziegler, T. *J. Comput. Chem.* **2001**, *22*, 931–967.
(24) Guerra, C. F.; Snijders, J.; te Velde, G.; Baerends, E. *Theor. Chem. Acc.* **1998**, *99*, 391.
(25) Baerends, E. et al. *ADF*, 2007.01; SCM: Amsterdam, The Netherlands, 2007.
(26) *CPMD*, version 3.11; IBM Corp.: Armonk, NY, 1990−2006; MPI für Festkörperforschung Stuttgart: Stuttgart, Germany, 1997−2001.
(27) CP2K. http://cp2k.berlios.de (accessed on Mar 25, 2008).
(28) Vandevondele, J.; Krack, M.; Mohamed, F.; Parrinello, M.; Chassaing, T.; Hutter, J. *Comput. Phys. Commun.* **2005**, *167*, 103–128.
(29) Gale, J. *JCS Faraday Trans.* **1997**, *93*, 629.
(30) Janezic, D.; Brooks, B. *J. Comput. Chem.* **1995**, *16*, 1543.
(31) Lindahl, E.; Hess, B.; van der Spoel, D. *J. Mol. Modell.* **2001**, *7*, 306–317.
(32) Christen, M.; Hünenberger, P. H.; Bakowies, D.; Baron, R.; Bürgi, R.; Geerke, D. P.; Heinz, T. N.; Kastenholz, M. A.; Kräutler, V.; Oostenbrink, C.; Peter, C.; Trzesniak, D.; van Gunsteren, W. F. *J. Comput. Chem.* **2005**, *26*, 1719–1751.
(33) Plimpton, S. J. *J. Comput. Phys.* **1995**, *117*, 1–19.
(34) Phillips, J. C.; Braun, R.; Wang, W.; Gumbart, J.; Tajkhorshid, E.; Villa, E.; Chipot, C.; Skeel, R. D.; Kale, L.; Schulten, K. *J. Comput. Chem.* **2005**, *26*, 1781–1802.
(35) Lewis, G. V.; Catlow, C. R. A. *J. Phys. C: Solid State Phys.* **1985**, *18*, 1149–1161.
(36) Schröder, K. P.; Sauer, J. J. *Phys. Chem.* **1996**, *100*, 11043–11049.
(37) Sierka, M.; Sauer, J. *Faraday Discuss.* **1997**, *106*, 41–62.
(38) Kirschhock, C.; Buschmann, V.; Kremer, S.; Ravishankar, R.; Houssin, C.; Mojet, B.; Van Santen, R.; Grobet, P.; Jacobs, P.; Martens, J. *Angew. Chem., Int. Ed.* **2001**, *40*, 2637–2640.
(39) Kremer, S.; Kirschhock, C.; Aerts, A.; Vilani, K.; Martens, J.; Lebedev, O.; Van Tendeloo, G. *Adv. Mater.* **2003**, *15*, 1705–1707.
(40) Kirschhock, C.; Ravishankar, R.; Verspeurt, F.; Grobet, P.; Jacobs, P.; Martens, J. *J. Phys. Chem. B* **1999**, *103*, 4965–4971.
(41) Rappe, A.; Casewit, C.; Colwell, K.; Goddard, W.; Skiff, W. *J. Am. Chem. Soc.* **1992**, *114*, 10024–10035.
(42) Stewart, J. J. P. *J. Comput. Chem.* **1989**, *10*, 221–264.
(43) Hocevar, S.; Hodoscek, M.; Penca, M.; Janezic, D. *J. Comput. Chem.* **2008**, *29*, 122–129.
(44) Lesthaeghe, D.; Vansteenkiste, P.; Verstraelen, T.; Ghysels, A.; Kirschhock, C.; Martens, J.; Van Speybroeck, V.; Waroquier, M. *J. Phys. Chem. C* **2008**, *112*, in press.

CI8000748