Hindawi Publishing Corporation
Journal of Nanomaterials
Volume 2015, Article ID 736943, 26 pages
http://dx.doi.org/10.1155/2015/736943

Research Article

Meshing Highly Regular Structures: The Case of Super Carbon Nanotubes of Arbitrary Order

Christian Schröppel and Jens Wackerfuß

Emmy Noether Research Group MISMO (Mechanical Instabilities in Self-Similar Molecular Structures of Higher Order), Institute of Structural Analysis, University of Kassel, Mönchebergstraße 7, 34125 Kassel, Germany

Correspondence should be addressed to Jens Wackerfuß; wackerfuss@uni-kassel.de

Received 16 January 2015; Accepted 12 July 2015

Academic Editor: Xiao-Yu Yang

Mesh generation is an important step in many numerical methods. We present the "Hierarchical Graph Meshing" (HGM) method as a novel approach to mesh generation, based on algebraic graph theory. The HGM method can be used to systematically construct configurations exhibiting multiple hierarchies and complex symmetry characteristics. The hierarchical description of structures provided by the HGM method can be exploited to increase the efficiency of multiscale and multigrid methods. In this paper, the HGM method is employed for the systematic construction of super carbon nanotubes of arbitrary order, which present a pertinent example of structurally and geometrically complex, yet highly regular, structures. The HGM algorithm is computationally efficient and exhibits good scaling characteristics. In particular, it scales linearly for super carbon nanotube structures and is working much faster than geometry-based methods employing neighborhood search algorithms. Its modular character makes it conducive to automatization. For the generation of a mesh, the information about the geometry of the structure in a given configuration is added in a way that relates geometric symmetries to structural symmetries. The intrinsically hierarchic description of the resulting mesh greatly reduces the effort of determining mesh hierarchies for multigrid and multiscale applications and helps to exploit symmetry-related methods in the mechanical analysis of complex structures.

1. Introduction

In recent years, mesh generation, which is an important part of most numerical analyses, has emerged as a research subject in its own right [1]. Various methods for the creation of meshes exist, yet many concepts are not formalized, and different approaches exist in various fields in both research and applications.

We propose a new method, the "Hierarchical Graph Meshing" (HGM) method, to generate meshes of complex, yet highly regular, structures. We apply this method to the construction of super carbon nanotubes (SCNTs) of arbitrary order. SCNTs are derived from carbon nanotubes (CNTs). CNTs are graphitic microtubules, first described in greater detail in [2]. Graphene and many graphitic compounds exhibit exceptional mechanical properties, such as high strength and tensile modulus. Figures 1 and 2 show the hierarchy and symmetry characteristics of a super carbon nanotube, an example of a highly hierarchical structure exhibiting multiple symmetries. The hierarchy of the SCNT results from the fact that a SCNT of a given order is always composed of a collection of constituent structures derived from the preceding order; that is, a SCNT of order 1 consists of carbon nanotubes ("SCNTs of order 0"), which in turn consist of carbon atoms and their bonds. The SCNT also exhibits multiple intertwining symmetry characteristics: translational, reflectional, and rotational symmetry. The combination of multiple symmetry characteristics is also being shown in Figure 3.

The geometric aspect of obtaining a monolithic mesh of a super carbon nanotube of higher order is a nontrivial task, involving nonlinear transformations of the constituent elements at each hierarchy order. We would like to stress,

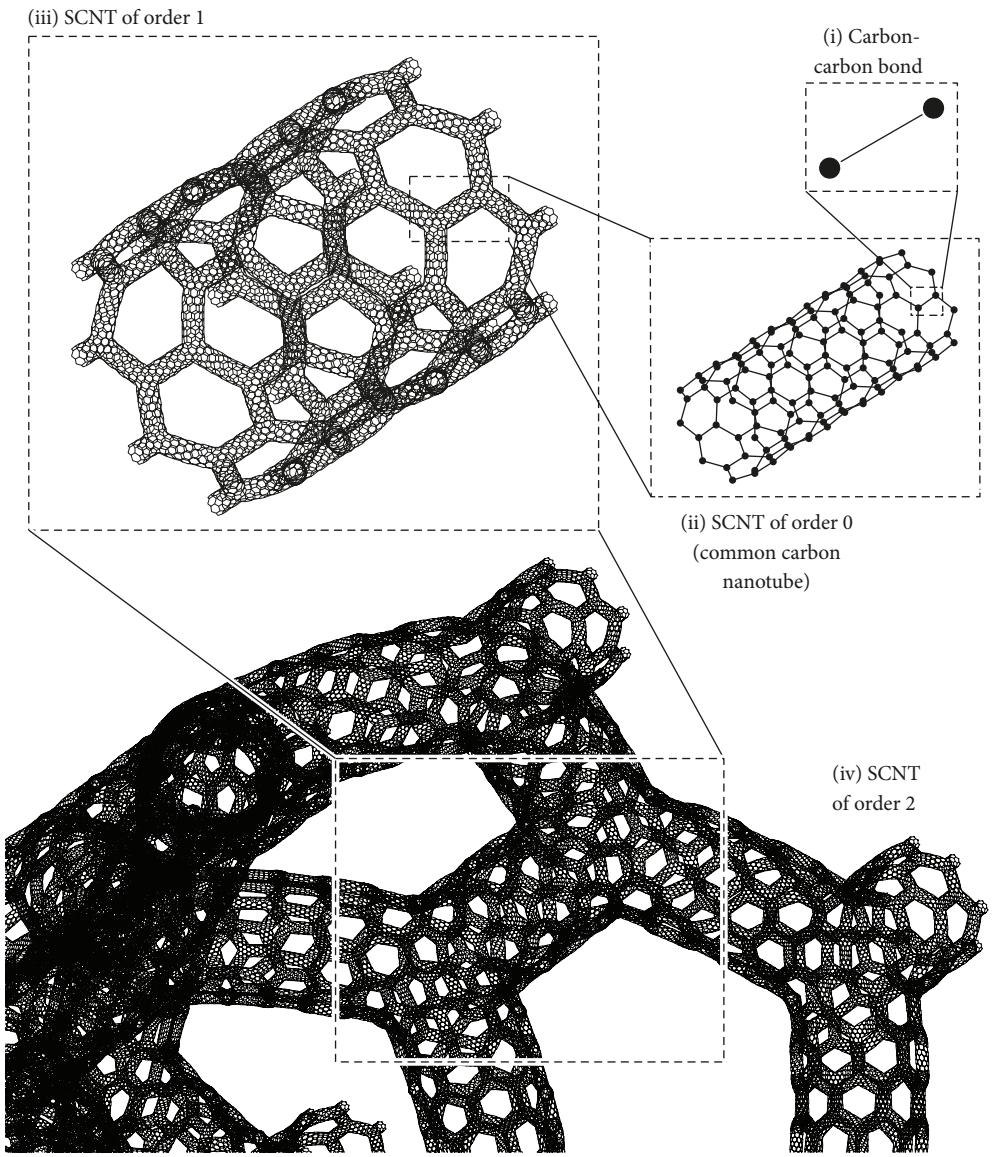


Figure 1: Hierarchy of a super carbon nanotube. Each hierarchy level is composed of elements of the preceding hierarchy level. Starting with the carbon-carbon bond (i), it is thus possible to obtain super carbon nanotubes (SCNTs) of arbitrary order. The image is based on data obtained by the graph-based HGM method.

however, that this text focuses almost exclusively on the structural aspect of the construction of super carbon nanotubes and does not offer any detailed description of their geometry.

The HGM method is applicable to all situations in which a mesh characterized by regularities such as hierarchy and symmetry either reflects the actual physical characteristics of a structure or emerges as part of the modeling process. It can be used both to build a structure based on certain regularities and to describe a given hierarchically symmetric structure. In particular, as an example of the latter case, geometrically irregular structures may be represented by meshes that exhibit certain structural regularities. Hierarchy and symmetry characteristics may also emerge as a result of design decisions. For example, frameworks of trusses are often characterized by multiple symmetries (see Figure 3). It is important to distinguish structural symmetry and geometric symmetry, as the former does not depend on the existence of the latter and, in the absence of fractures or other topological changes, is preserved in deformed structures that are no longer geometrically symmetric.

Regular meshes are often used for academic examples and for the analysis of structures that exhibit simple forms of symmetry. In contrast, more complex structures that, in principle, would lend themselves to a description by meshes that preserve their hierarchy and symmetry characteristics are often described by irregular meshes. As a consequence, important hierarchy and symmetry information is lost and can no longer be exploited in the subsequent analysis. In contrast, the HGM method proposed in this paper results in a tuple-based indexing system which preserves such

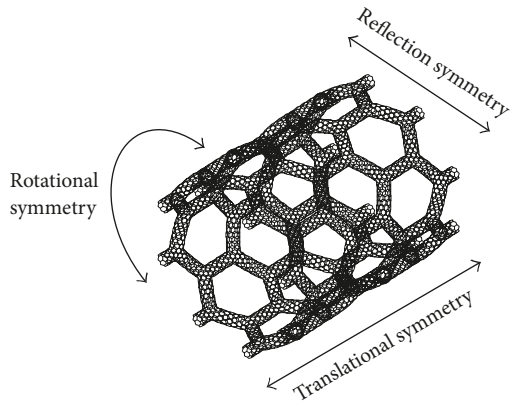


Figure 2: Symmetry characteristics. A super carbon nanotube exhibits multiple symmetries at each hierarchy level.

regularities and provides a systematic and formalized way to describe and to generate meshes (see Figure 4).

Many, if not all, structures that may be generated by the Hierarchical Graph Meshing (HGM) method presented here can as well be produced on an ad hoc basis, applying elementary transformations on indices representing the nodes and interdependencies of the mesh. However, such ad hoc methods lead to descriptions of the resulting configurations that are highly dependent on the specific operations chosen to generate the mesh.

In this situation, it is difficult to communicate the actual meaning of the description of the resulting structure, as one has to refer to elementary operations rather than abstract, but exactly defined, steps of the generation process. Such abstract steps include translational, reflectional, and rotational symmetries and hierarchy characteristics, including self-similarity. A systematic and modularized method of generating hierarchical and symmetric meshes, such as the graph-theoretic method presented here, is also a precondition for the development of computer algorithms and programs that can be used for the production of large classes of meshes, providing defined interfaces for both input and output of mesh-related information.

Seen from this perspective, one of the major advantages of the HGM method is precisely the fact that it enables generic program modules to do most of the basic work associated with the meshing process. As a result, researchers and practitioners can focus on the specific characteristics of their respective problems and task instead of coping with the often tedious work associated with elementary indexing systems and operations.

(1) The HGM Method in the Context of Multigrid Algorithms. The HGM method results in tuple-based descriptions of configurations that contain the hierarchy-related information in an explicit and readily accessible form. This information can in turn be used in the subsequent mechanical analysis of the structure.

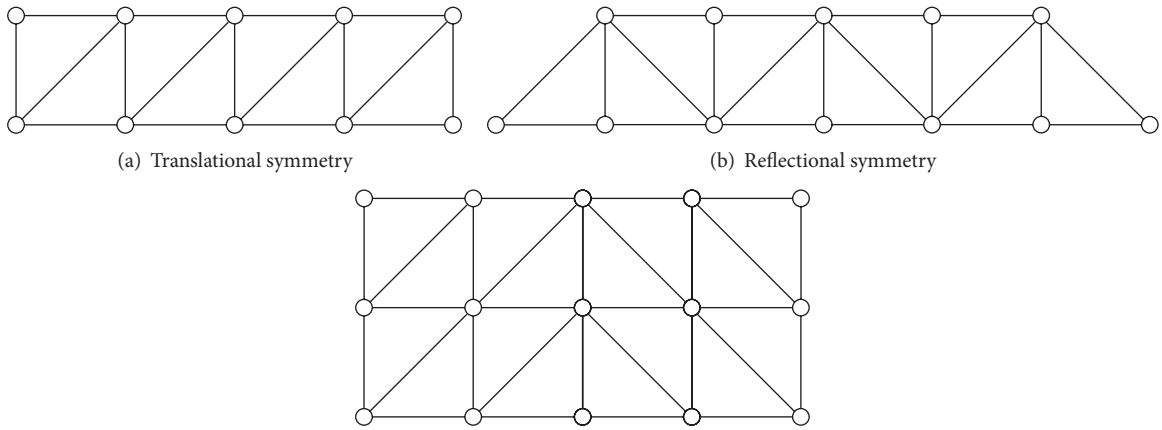
Multigrid methods are usually based on coarse meshes that are derived from the fine mesh often by heuristic methods. With the HGM method, coarse meshes, which can be based on the leading indices of the tuples identifying the nodes, can be built in parallel with the generation of the fine mesh. With regard to complex, highly regular structures, multigrid and multiscale methods can therefore benefit greatly from the application of the HGM method.

Both multiscale and multigrid methods have become important tools in the analysis of a wide range of physical phenomena, including problems in solid and fluid mechanics [3–5]. Increases in computational power and refinements in programming tools and concepts have enabled researchers to develop conceptually complex but highly efficient algorithms in a systematic way, providing diverse methods with a potentially wide range of applications [6].

Generic multigrid methods, such as the algebraic multigrid method (AMG), can be applied to a wide range of problems and lead to algorithms generally characterized by good accuracy, robustness, and convergence. For unstructured grids, such generic methods, in combination with heuristic algorithms for coarsening and interpolation, are often the methods of choice. Coarsening methods include the Ruge-Stüben (RS) algorithm [7], parallel independent set algorithms [8, 9], and the multilevel incomplete lower-upper triagonal (MLILU) decomposition [10]. The Falgout coarsening scheme, which combines aspects of the RS and the parallel independent set algorithms, is described in [11, 12]. A challenge in the development of such algorithms is to achieve computational efficiency and good scalability in parallel implementations and to construct meshes that have predictable local characteristics and optimize speed and convergence rates in the subsequent calculations. (See also [13].) In the context of the algebraic multigrid method, it has been proven difficult to model configurations characterized by rapid and oscillating changes in the element-related coefficients [14].

Meshes may result from the discretization of a continuous configuration, or they may be essentially determined by the underlying physical model as in atomistic simulations. Regularities may be given by exact or approximate geometric periodicity, symmetry, or hierarchy of the structure itself, or they may be introduced by the meshing process as a result of the specific discretization chosen to model an otherwise irregular structure. In the latter case, irregular configurations may be discretized by regular meshes, or topological as well as approximate geometric regularities may be described by meshes that are regular from a structural, or graph-theoretic, viewpoint.

In the presence of such regularities, the HGM method proposed in this paper preserves these characteristics as an integral part of the description of the mesh, thus enabling the systematic construction of mesh hierarchies for multigrid and multiscale methods. While regularities at the finer grid levels may be most interesting in the context of the variational multiscale method [15], applications of the heterogeneous multiscale method [16] may focus more on regularities at coarser grid levels. Reference [17] shows that multigrid transfer operators that do not take the specific characteristics of the finer grid into account, such as injection and projection and linear interpolation, and even transfer operators based on

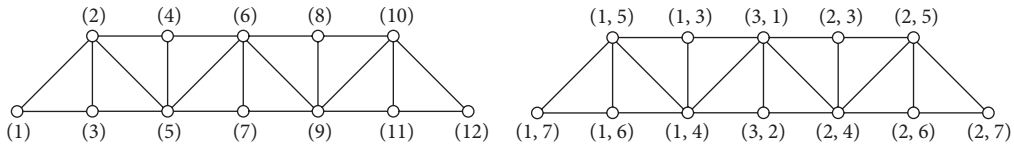


(a) Translational symmetry

(b) Reflectional symmetry

(c) Combined translational symmetry (vertically) and reflectional symmetry (horizontally)

Figure 3: Combining symmetry characteristics in a hierarchical structure.



(a) Sequential indexing: the information about the reflection symmetry cannot be retrieved from the indices

(b) Tuple-based indexing: the leading index identifies the symmetric parts of the graphs as well as the invariant nodes. The trailing index identifies mutually symmetric nodes

Figure 4: Commonly used sequential indexing and tuple-based indexing of the HGM method.

Schur complements are less efficient than transfer operations derived from the application of a homogenization method adapted to the respective problem.

Algebraic multigrid (AMG) methods [18, 19] do not depend on a predefined hierarchy of meshes. Thus, direct input from the HGM method, which focuses on the description of structured meshes, is not strictly necessary to implement AMG methods. However, it is possible to refine structured meshes at a coarser level, which may be constructed by the HGM method, by unstructured meshes at the finer level, and a systematic description of the existing regularities of the mesh helps to implement efficient mesh adaptation and parallelization strategies. In addition, the HGM method is based on structural regularities, not geometric regularities. Thus, the ability of AMG methods to process meshes without geometric regularities may be combined with input from the HGM method based on structural regularities. Such regularities may be simply based on patterns of a stronger and weaker linkage between variables.

Thus, HGM and AMG, as well as other multigrid algorithms, can be combined to obtain efficient, modular, and widely applicable strategies for the solution of engineering problems.

(2) Basic Characteristics of the HGM Method. The Hierarchical Graph Meshing (HGM) method can be understood as a sequence of modules, each of which creates a particular hierarchy or symmetry characteristic of the resulting graph. In the most basic case, a set of graphs is combined to form a single, hierarchically structured graph. In this case, each node is identified by a two-index tuple, with the leading index indicating the graph from which the node originated. If two or more of the original graphs are mutually symmetric, then this symmetry is being retained in the resulting graph, which thus becomes a hierarchically symmetric graph.

Such hierarchically symmetric graphs may exhibit more than one hierarchy level and more than one symmetry. In particular, symmetries may be of different kind, that is, reflectional as well as translational and rotational symmetries, and may have one or more invariant node sets. The modularized nature of the HGM method allows setting up the parameters for each hierarchy and/or symmetry characteristic separately. Thus, all information of a structure, such as information related to the geometry, the topology, or the connectivity, as well as the symmetries at different hierarchy levels of the structure, is being introduced in well-defined steps of the algorithm, avoiding a patchwork of ad hoc solutions that are often being applied in the generation of actual meshes for finite element analyses or other purposes.

Retaining the information related to the symmetry characteristics opens new and efficient methods for the mechanical analysis of hierarchically symmetric structures. For

example, the stiffness matrix of a geometrically symmetrical configuration can be block-diagonalized using symmetry group operations [20–22]. Information about the hierarchy of a configuration can also be used to efficiently compute important characteristics of a structure, such as its eigenvalues [23]. Preserving the information on the hierarchy of a structure is also important with regard to domain decomposition techniques. Exploiting the symmetry characteristics of a configuration can greatly reduce the computational effort involved in finite element analysis [24]. However, concepts related to symmetry group operations have not been incorporated into general-purpose finite element programmes in a way that allows for an automatized processing of symmetric configurations [25, 26]. To the knowledge of the authors, no implementations of any algorithms that deliver similar results exist, thus precluding the possibility of performing quantitative comparisons against established alternative approaches.

Algebraic graph theory is a powerful tool for the systematic generation of meshes, and the elementary steps of the construction algorithm rely on algebraic operations of directed and undirected graphs. In this paper, a particular formulation of graphs, based on directed graphs, is being used. Reference [27] has demonstrated that directed graphs allow for the generation of a larger set of hierarchically structured configurations compared to undirected graph.

The graph-algebraic approach allows for a computationally highly efficient implementation of the algorithm. As complex steps can be described as a well-defined sequence of elementary operations, and operations can be arranged in a way that minimizes computational cost, code optimizations can focus on a small subset of the overall program. In addition, as a result of the identification of arcs as pairs of tuples of indices, basic operations such as intersections, unions, and compositions of graphs can be calculated in a very efficient way, especially for highly hierarchical and symmetric graphs. Generally, the exploitation of hierarchy and symmetry characteristics, if present, significantly reduces the computational cost. For carbon nanotubes (CNTs) and super carbon nanotubes (SCNTs) [28, 29], which can be seen as a reference case of hierarchically symmetric structures, a MATLAB-based implementation developed by the authors scales linearly with the number of nodes in the respective mesh.

The atomic-scale finite element method can be used to explore the mechanical properties of molecular structures [30]. Reference [31] developed a computationally efficient algorithm based on the atomic finite element method for the analysis of large-scale carbon structures. Using continuum mechanics methods, CNTs and tube-like connections in SCNTs have been modeled as cylinders as well as structures composed of Euler and Timoshenko beams [32]. In this context, the HGM method proposed in this paper provides a description of configurations such as SCNTs that help to integrate such approaches into a multiscale concept for analyzing large-scale molecular structures.

(3) Outline. In the section immediately following this introduction, we provide a brief outline of the geometry and the common structural properties of super carbon nanotubes (SCNTs). In SCNTs, multiple hierarchy and symmetry properties are intertwined in a complex way. Thus, such structures are excellent examples that illustrate the characteristics of the HGM method presented in this paper.

Section 3 presents the graph-theoretic foundation of the algorithm. It extends commonly used graph-theoretic operations by introducing unary and binary operations on hierarchical graphs, such as the composition of graphs and the categorical product. Section 4 introduces the concept of graph symmetry and its relation to the description of structures by hierarchical graphs.

Section 5 applies the methods and operations developed up to this point to construct SCNTs of arbitrary hierarchical order. The hierarchical nature of the SCNT structure is reflected in the similarity of the different steps of the construction algorithm, which can be grouped into sequences, each comprising specific steps based on a common template. Section 6 reviews a MATLAB-based implementation of the HGM method, compares the method to a geometry-based algorithm that employs a neighborhood search algorithm, and analyzes the performance of the HGM algorithm for super carbon nanotubes of different sizes and hierarchical orders.

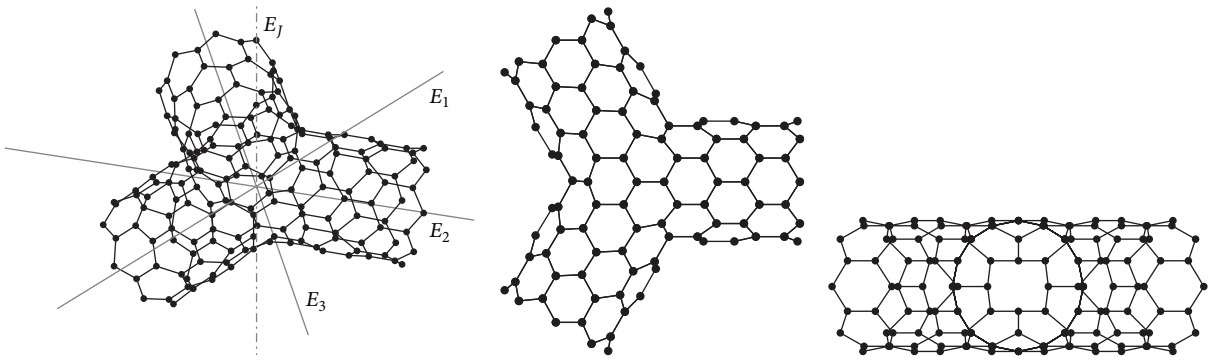
The main results are summarized in the concluding section. Some proofs related to algebraic graph operations are given in Appendix.

2. Geometric and Structural Properties of Super Carbon Nanotubes

2.1. Hierarchy and Symmetry Characteristics. The structure of super carbon nanotubes (SCNTs) can be thought of as a network of carbon nanotubes (CNTs) connected by Y-shaped structural elements, that is, CNT junctions. Conceptually, a SCNT can be generated by replacing the carbon-carbon bonds in a single-walled carbon nanotube by CNTs and the carbon atoms by Y-shaped junctions. A number of recent publications on SCNTs, such as [28, 29], take this interpretation of the structure of super carbon nanotubes as the starting point.

This process can be recursively applied to generate SCNT structures of arbitrary order of hierarchy. Following [28], we identify the single-walled CNT with the SCNT of order 0. The SCNT of order 0 is the first step in the iterative process of generating SCNTs of arbitrary order, as shown in Figure 1.

Figure 5 shows a SCNT junction, a constituent part of the SCNT of order 1, and its projection onto the horizontal plane of symmetry and one of the vertical planes of symmetry. The starting point of the construction of these symmetries is the rotational axes of the branches of the junction. In a fully symmetric junction, these rotational axes coincide in a single point. A vertical rotation axis exists, which contains this point of intersection and is perpendicular to all rotational axes of the adjacent CNTs. We denote this axis as E_J and the three rotational axes of the CNTs as E_k, $k \in \{1, 2, 3\}$. In the following, we write the plane that includes all points of two lines, A and B, as $P(A, B)$. A regular junction has four planes of symmetry:



(a) Axonometric projection (b) Projection on the horizontal plane (c) Projection along an axis E_k, $k \in \{1, 2, 3\}$

Figure 5: Symmetries of a CNT junction.

(i) The plane $P(E_1, E_2) = P(E_2, E_3) = P(E_3, E_1)$, which we will call the *horizontal plane of symmetry*.

(ii) The planes $P(E_J, E_k)$, $k \in \{1, 2, 3\}$, which will be referred to as the *vertical planes of symmetry*.

In addition, a regular junction exhibits a 3-fold rotational symmetry with respect to E_J axis, the *symmetry axis*.

In order to retain the multiple symmetries of SCNTs in mesh generation, it is helpful to abandon the traditional interpretation of SCNTs as collections of tube elements and junction elements. Instead, we interpret a SCNT as a CNT in which the carbon atoms are being replaced by Y-shaped junctions that comprise both the "traditional" junction elements and half of the adjacent carbon nanotube elements. These Y-shaped junctions are being connected at their ends by carbon-carbon bonds, in the case of the zigzag orientation, or, alternatively, share a "ring" of carbon atoms, in the case of the armchair orientation.

2.2. Super Carbon Nanotubes as Embedded Surfaces. Super carbon nanotubes, as well as CNTs, are graphene-based structures. A graphene sheet can be viewed as a hexagonal lattice of carbon atoms or as a trigonal lattice of hexagons that are being formed by six carbon atoms, respectively. Following the latter interpretation, a graphene sheet in the Euclidean space can be thought of as a 2-dimensional surface embedded in a 3-dimensional space.

Similarly, a CNT, as a graphene-based structure, can be thought of as a 2-dimensional surface embedded in a 3-dimensional space. Therefore, the theory of hypersurfaces, a branch of differential geometry, can be used to explore its geometric properties. The following exposition of such basic properties draws on concepts and terms originating from differential geometry.

In the case of the simple CNT, the surface, when interpreted as the surface of a cylinder, has zero Gaussian curvature and can thus be unrolled onto a plane. Therefore, a CNT may consist entirely of hexagons. The respective surface of a SCNT, however, is a surface of higher genus (i.e., it necessarily has "holes") with a negative Gaussian curvature. This implies that this surface cannot be unrolled onto a plane and that it is not possible to construct such a surface based exclusively on hexagons.

Differential geometric observations show that the number of different types of polygons in each junction element is not arbitrary. While different combinations of nonhexagonal shapes, for example, pentagons, heptagons, or octagons, are possible, and these "defects" may be located at different positions on the junction element, the resulting structure's Euler characteristic, defined as the sum of atoms and surface elements minus the number of bonds, must match the total curvature of its surface, divided by 2π. For closed surfaces, the total curvature is determined by the genus of the respective surface. It can thus be shown that a junction element with three ends, independent of its specific geometric shape, has a total curvature of $K = -2\pi$ and that the number n_k of k-edged surface elements in a junction element must fulfill the condition $-(1/3)\pi \sum_{k=3}^{\infty}(k-6)n_k = K$. Thus, a junction element, in addition to an arbitrary number of hexagons, may contain, for example, three octagons or six heptagons or a combination of six octagons and six pentagons. This finding has been presented in more detail in [33–35].

3. Graph Algebra

The elementary operations employed in the HGM method are based on graph-theoretic concepts. Therefore, in order to lay the groundwork for the subsequent presentation of hierarchically symmetric graph and the description of the construction of super carbon nanotubes, we develop a graph algebra based on elementary unary and binary graph operations.

The following exposition, in part, draws on, and in some cases expands on, the algebraic approach to graph theory as presented in [36–39]. A very accessible introduction to graph theory can be found in [40]. In particular, the union, intersection, and product operations are well known, although they generally have been defined for undirected, loopless graphs [38, 41–43]. The composition of graphs can be regarded as a graph morphism, and the theory of graph morphisms can be readily applied. If we identify graphs with graph morphisms, however, we can regard them as elements of a half-ring, thus enabling us to use the properties of half-rings, for

example, distributive laws, in order to streamline and simplify calculations. To the knowledge of the authors, no attempt to use these properties of graph morphisms in order to construct large, highly regular structures has been made in the existing literature.

Graphs are an abstract concept of relations between elements of a set. The elements of such a set may be actual physical items, or they may be abstract entities, such as the nodes derived from the discretization of a structure. Hierarchies and symmetries can be defined on graphs, and such properties of a graph can be used to characterize the structure that it represents. Generally, graph-based hierarchies and symmetries are not dependent on geometric hierarchies and symmetries, although structures that exhibit regularities from a graph-based perspective often show related regularities from a geometric viewpoint.

3.1. Directed Graphs. In many practical applications, graphs are primarily interpreted as a systematic collection of data. In graph theory, the structure of graphs and of the operations that can be performed on graphs is more formalized. However, different concepts exist to describe graphs.

In textbooks on graph theory, graphs are generally introduced by defining *undirected graphs*, and *directed graphs* are often described as structures that are generated by transforming undirected graphs (e.g., in [36, 44]). In contrast, the graph algebra presented in this paper takes directed graphs as its starting point. This follows the approach taken in [39]. Reference [27] has also employed directed graphs to construct large structural models, using different graph products to describe specific graph operations.[1]

Various definitions and notations of graphs are being used in the literature. In order to build a more comprehensive algebraic structure, we identify *directed graphs* $\Gamma(\mathscr{W}, \mathscr{A})$ with ordered pairs consisting of a *node corridor* $\mathscr{W}$ and a set of *arcs* $\mathscr{A}$.

A *graph space* of dimension n is denoted as $\mathfrak{G}^{(n)}$. If the nodes of the graphs belonging to the graph space are given by a one-element index, then $n = 1$, and the *space of directed graphs* of dimension 1 is denoted as $\mathfrak{G}^{(1)}$. If the nodes are given by a multi-index, and n is the number of elements in a tuple that identifies a node in the graph space, the space of directed graphs is a space of *hierarchical graphs* of dimension n and is denoted as $\mathfrak{G}^{(n)}$.

In order to simplify the following exposition, we introduce two basic operations for tuples, the natural projection, denoted by p, and the n-ary Cartesian product, denoted by cart. Let $(M_k)_{k=1}^n = (M_1, M_2, \ldots, M_q, \ldots, M_n)$ denote a tuple composed of n sets M_k. Then, the natural projection is given by

$$p_q : (M_k)_{k=1}^n \longmapsto M_q, \tag{1}$$

and the n-ary Cartesian product, which maps a tuple of sets to a set of elements, is given by

$$\text{cart} : (M_k)_{k=1}^n \longmapsto M_1 \times M_2 \times \cdots \times M_n. \tag{2}$$

For tuples of sets, we define set operations based on the representation of such tuples as disjoint unions. Thus, for tuples of sets given by $T \coloneqq (M_k)_{k=1}^n$ and $T' \coloneqq (M_k')_{k=1}^n$, we have $T \cup T' \coloneqq (M_k \cup M_k')_{k=1}^n$, $T \cap T' \coloneqq (M_k \cap M_k')_{k=1}^n$.

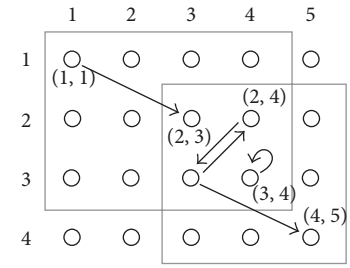


Figure 6: The graph $\Gamma(\mathscr{W}, \mathscr{A})$ with the node corridor $\mathscr{W} = \{(\mathscr{S}, \mathscr{T})\}$, $\mathscr{S} = (\{1, 2, 3\}, \{1, 2, 3, 4\})$, $\mathscr{T} = (\{2, 3, 4\}, \{3, 4, 5\})$, and the arcs $\mathscr{A} = \{(1 \mid 1,\ 2 \mid 3), (2 \mid 4,\ 3 \mid 3), (3 \mid 3,\ 2 \mid 4), (3 \mid 4,\ 3 \mid 4), (3 \mid 3,\ 4 \mid 5)\}$. The discrete coordinates are shown in order to illustrate the definition of the node corridor. Note that the graph itself does not contain any information about its geometric representation.

A nonempty node corridor $\mathscr{W} = \{(\mathscr{S}, \mathscr{T})\}$ can be represented by an ordered pair of tuples, a set of *source nodes* $\mathscr{S}$ and a set of *target nodes* $\mathscr{T}$. Operations on graphs may, however, result in an empty node corridor that would be represented by an empty set. For this reason, the node corridor is defined as a set of tuples, containing at most one element, not as a single tuple.

The *nodes* $\mathscr{V}$ of a nonempty graph are given by the union of the source nodes and the target nodes; that is, $\mathscr{V} \coloneqq \mathscr{S} \cup \mathscr{T}$. Figure 6 shows an example of a graph in the graph space $\mathfrak{G}^{(2)}$. In this figure, as well as elsewhere, some tuples are denoted as $t_1 \mid t_2 \mid \cdots \mid t_n$, which is equivalent to $(t_1, t_2, \ldots, t_n)$.

The arcs of a graph can be understood as directed connections between source nodes and target nodes; that is, an arc connects a node $v \in \text{cart}(\mathscr{S})$ to a node $v' \in \text{cart}(\mathscr{T})$.[2] Arcs can thus be represented by ordered pairs of nodes, (v, v'). As a result, the arcs belonging to a graph, $\mathscr{A}$, are a subset of the *arc space*, $\mathfrak{A}^{(n)}$, which itself is constructed from the node space $\mathfrak{V}^{(n)}$:

$$\mathfrak{A}^{(n)} \cong \left(\text{cart}\left(\mathfrak{V}^{(n)}\right)\right)^2 = \left(\mathfrak{V}_1 \times \mathfrak{V}_2 \times \cdots \times \mathfrak{V}_n\right)^2. \tag{3}$$

We introduce two functions that can be used to identify the source nodes and the target nodes, respectively, of an arc, a set of arcs, or a node corridor. The *source* of an arc a, that is, the domain, indicated by the first element of the tuple representing the arc, is given by $d(a) \coloneqq p_1(a)$, and its *tail*, or the range, indicated by the tuple's second element, is given by $r(a) \coloneqq p_2(a)$. Thus, $a \equiv (d(a), r(a))$. For sets of arcs $A \in \mathfrak{A}$, we also define the functions $d(A) \coloneqq \{d(a) \mid a \in A\}$, and $r(A) \coloneqq \{r(a) \mid a \in A\}$. For a graph $G = \Gamma(\mathscr{W}, \mathscr{A})$, $\mathscr{W} \neq \{\}$, we define $d_{\mathscr{W}}(G) \coloneqq d(\mathscr{W}) = \mathscr{S}$, $r_{\mathscr{W}}(G) \coloneqq r(\mathscr{W}) = \mathscr{T}$, $d_{\mathscr{A}}(G) \coloneqq d(\mathscr{A})$, and $r_{\mathscr{A}}(G) \coloneqq r(\mathscr{A})$. For $\mathscr{W} = \{\}$, these functions result in the empty set.

If $d(a) = r(a)$, the arc a connects a node with itself and is also called a *loop*. We call nodes that are connected to other nodes, or to themselves, that is, the elements of $\mathscr{V}_c \coloneqq d(\mathscr{A}) \cup r(\mathscr{A})$, *connected nodes*. Nodes that are not connected, that is, the elements of $\mathscr{V}_i \coloneqq \mathscr{V} \setminus \mathscr{V}_c$, are called *isolated nodes*.

Table 1: Graph operations.

Operation	Notation	Remark
Opposite	G^*	See Section 3.1
Union	$G_1 \cup G_2$	See Section 3.1
Intersection	$G_1 \cap G_2$	See Section 3.1
Composition	$G_2 \circ G_1$ or $G_2 G_1$	See Section 3.3
Conjugation	$G_1 \diamond G_2$	See Section 3.4
Categorical product	$G_1 \otimes G_2$	See Section 3.5

As a unary operation in $\mathfrak{G}^{(n)}$, we define the opposite of an arc as $a^* := (r(a), d(a))$ and the opposite of a set of arcs as $A^* := \{a^* \mid a \in A\}$. The opposite of a node corridor $\mathscr{W} = \{(\mathcal{S}, \mathcal{T})\}$ is $\mathscr{W}^* := \{(\mathcal{T}, \mathcal{S})\}$. The *opposite of a graph* is given by $\Gamma(\mathscr{W}, \mathscr{A})^* := \Gamma(\mathscr{W}^*, \mathscr{A}^*)$.

The union and the intersection of two sets of arcs, A and A', are given by the common set-theoretic operations; that is, $A \cup A' := \{a \mid a \in A \vee a \in A'\}$ and $A \cap A' := \{a \mid a \in A \wedge a \in A'\}$. The union of two nonempty node corridors $\mathscr{W} = \{(\mathcal{S}, \mathcal{T})\}$ and $\mathscr{W}' = \{(\mathcal{S}', \mathcal{T}')\}$ is defined as $\mathscr{W} \cup \mathscr{W}' := \{(\mathcal{S} \cup \mathcal{S}', \mathcal{T} \cup \mathcal{T}')\}$. Similarly, the intersection is defined as $\mathscr{W} \cap \{\mathscr{W}' := (\mathcal{S} \cap \mathcal{S}', \mathcal{T} \cap \mathcal{T}')\}$. For $\mathscr{W} = \{\}$, $\mathscr{W} \cup \mathscr{W}' = \mathscr{W}'$ and $\mathscr{W} \cap \mathscr{W}' = \{\}$. Both operations are commutative.

We can thus define two binary operations in the graph space $\mathfrak{G}^{(n)}$: the *union* of two graphs is given by $\Gamma(\mathscr{W}, \mathscr{A}) \cup \Gamma(\mathscr{W}', \mathscr{A}') := \Gamma(\mathscr{W} \cup \mathscr{W}', \mathscr{A} \cup \mathscr{A}')$ and their *intersection* is given by $\Gamma(\mathscr{W}, \mathscr{A}) \cap \Gamma(\mathscr{W}', \mathscr{A}') := \Gamma(\mathscr{W} \cap \mathscr{W}', \mathscr{A} \cap \mathscr{A}')$. The union and the intersection of two graphs are illustrated in Figures 7(a) and 7(b). The unary graph operation resulting in the opposite graph, as well as several fundamental binary graph operations, is collected in Table 1.

3.2. Undirected Graphs. While an arc is a directed connection between two nodes, an *edge* is an undirected connection between two nodes. We can therefore represent arcs by ordered pairs of the nodes incident to them and edges by the sets of the nodes incident to them. Graphically, edges may be represented by two arcs pointing from a node v_i to a node v_j, and vice versa, or by a line between the two nodes. A loop is an arc that is identical to its opposite; thus loop edges are represented as arcs pointing from a node v_i to itself.

We introduce a function υ that maps an arc $a \in \mathfrak{A}^{(n)}$ to an edge $e \in \mathfrak{E}^{(n)}$, with $\mathfrak{E}^{(n)}$ being the *edge space*:

$$\begin{aligned} \upsilon : \mathfrak{A}^{(n)} &\longrightarrow \mathfrak{E}^{(n)}, \\ a &\longmapsto \upsilon(a) := \{d(a), r(a)\}. \end{aligned} \tag{4}$$

We can identify edges and arcs as follows: $\mathfrak{E}^{(n)} \ni \upsilon(a) \cong \{a, a^*\} \subseteq \mathfrak{A}^{(n)}$. Note that the set $\{a, a^*\}$ contains only one element if $d(a) = r(a)$, as this condition implies $a = a^*$. Similarly, the function υ shall map a set of arcs A to a set of edges E:

$$\begin{aligned} \upsilon : \mathscr{P}\left(\mathfrak{A}^{(n)}\right) &\longrightarrow \mathscr{P}\left(\mathfrak{E}^{(n)}\right), \\ A &\longmapsto \upsilon(A) := \{\upsilon(a) \mid a \in A\}. \end{aligned} \tag{5}$$

With regard to a nonempty node corridor $\mathscr{W} = \{(\mathcal{S}, \mathcal{T})\}$, we define the result of the function υ as $\upsilon(\mathscr{W}) := \mathscr{V} = \mathcal{S} \cup \mathcal{T} \cong \mathscr{W} \cup \mathscr{W}^* = \{(\mathcal{S} \cup \mathcal{T}, \mathcal{T} \cup \mathcal{S})\} = \{(\mathscr{V}, \mathscr{V})\}$. For $\mathscr{W} = \{\}$, $\upsilon(\mathscr{W}) = \{\}$.

When applied to graphs, the function υ shall map a directed graph $\Gamma(\mathscr{W}, \mathscr{A})$ to an *undirected graph* $\Upsilon(\upsilon(\mathscr{W}), \upsilon(\mathscr{A}))$. With the nodes $\mathscr{V} = \upsilon(\mathscr{W})$, edges $\mathscr{E} := \upsilon(\mathscr{A})$, and the *space of undirected graphs* of order n denoted as $\mathfrak{U}^{(n)}$, we have

$$\begin{aligned} \upsilon : \mathfrak{G}^{(n)} &\longrightarrow \mathfrak{U}^{(n)}, \\ \Gamma(\mathscr{W}, \mathscr{A}) &\longmapsto \upsilon(\Gamma(\mathscr{W}, \mathscr{A})) =: \Upsilon(\upsilon(\mathscr{W}), \upsilon(\mathscr{A})) \\ &= \Upsilon(\mathscr{V}, \mathscr{E}). \end{aligned} \tag{6}$$

We can identify undirected graphs with directed graphs as follows:

$$\begin{aligned} \mathfrak{U}^{(n)} &\ni \Upsilon(\upsilon(\mathscr{W}), \upsilon(\mathscr{A})) \\ &\cong \Gamma\left(\mathscr{W} \cup \mathscr{W}^*, \mathscr{A} \cup \mathscr{A}^*\right) \in \mathfrak{G}^{(n)}. \end{aligned} \tag{7}$$

This allows us to use the operations defined on directed graphs in order to evaluate the respective operations on undirected graphs.

3.3. The Composition of Graphs. In addition to the union and the intersection, we introduce the *composition* on $\mathfrak{G}^{(n)}$. Let $a, a' \in \mathfrak{A}$, as defined in (3). Then,

$$\{a\} \circ \left\{a'\right\} := \begin{cases} \left\{\left(d\left(a'\right), r(a)\right)\right\} & \text{if } d(a) = r\left(a'\right) \\ \{\} & \text{if } d(a) \neq r\left(a'\right). \end{cases} \tag{8}$$

For two sets of arcs, $A \subseteq \mathfrak{A}$ and $A' \subseteq \mathfrak{A}'$, the composition is defined as

$$A \circ A' := \left\{\{a\} \circ \left\{a'\right\} \mid a \in A,\ a' \in A'\right\}, \tag{9}$$

and, for two node corridors, $\mathscr{W} = (\mathcal{S}, \mathcal{T})$ and $\mathscr{W}' = (\mathcal{S}', \mathcal{T}')$, the composition is defined as

$$\mathscr{W} \circ \mathscr{W}' := \begin{cases} \left\{\left(\mathcal{S}', \mathcal{T}\right)\right\} & \text{if } \mathcal{S} \cap \mathcal{T}' \neq \{\} \\ \{\} & \text{if } \mathcal{S} \cap \mathcal{T}' = \{\}. \end{cases} \tag{10}$$

Based on these definitions, the *composition of two graphs* can be defined as

$$\Gamma(\mathscr{W}, \mathscr{A}) \circ \Gamma\left(\mathscr{W}', \mathscr{A}'\right) := \Gamma\left(\mathscr{W} \circ \mathscr{W}', \mathscr{A} \circ \mathscr{A}'\right). \tag{11}$$

The composition of two graphs is illustrated in Figure 7(c).

We define the composition of two graphs of different dimensions as follows: for graphs $G_1 \in \mathfrak{G}^{(m)}$ and $G_2 \in \mathfrak{G}^{(n)}$, $n > m$, the composition is $G_1 \circ G_2 := (G_1 \otimes \bigotimes_{k=1}^{n-m} D_{\mathfrak{V}_k}) \circ G_2$, where $D_{\mathfrak{V}_k}$ is the identity graph in the node space $\mathfrak{V}_k$ and $D_{\mathfrak{V}_k} := \Gamma(\{(\mathfrak{V}_k, \mathfrak{V}_k)\}, \{(v, v)\}_{v \in \mathfrak{V}_k})$.

We denote the n-fold composition of a graph G with itself as G^n. The *closure* of a graph under the composition, which we will also refer to as the closure of the graph, without further

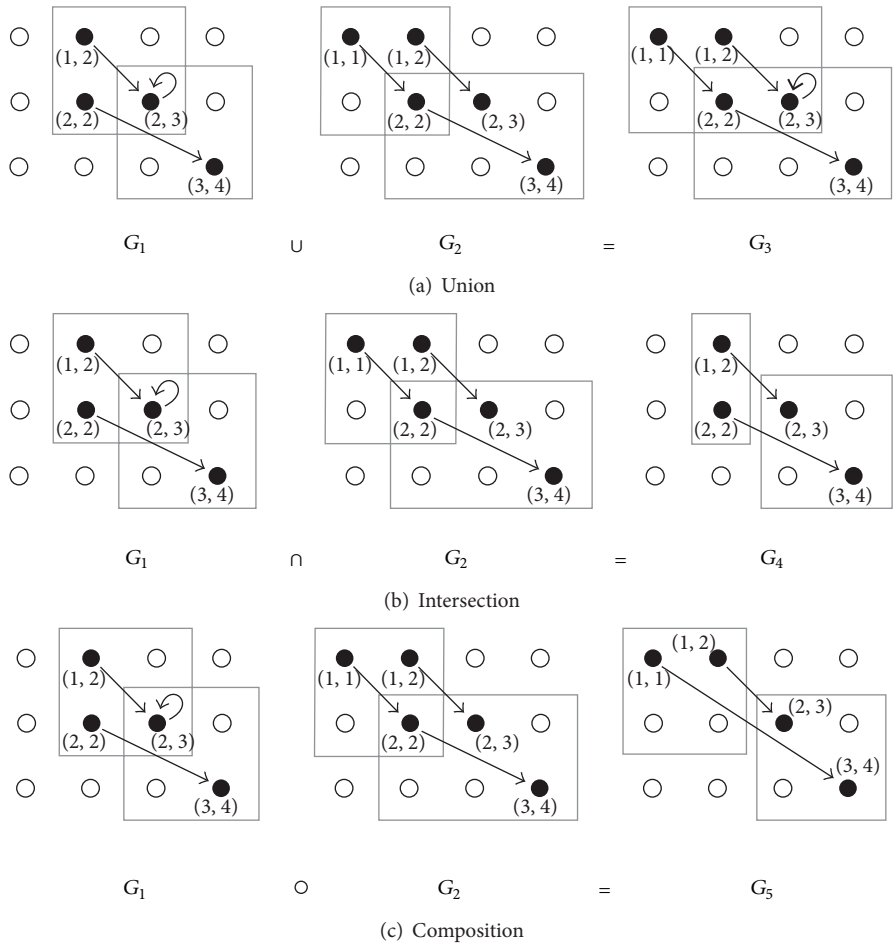


Figure 7: Binary operations.

specifications, is given by the union of the compositions of the graph with itself: $c(G) = \bigcup_{k=1}^{\infty} G^k$. For a finite graph G, there is a finite number N such that $c(G) = \bigcup_{k=1}^{n} G^k$ for $n \geqslant N$.

The empty graph without nodes, which we will also call the *zero graph*, $Z := \Gamma(\{\}, \{\})$, is the neutral element in $\mathfrak{G}^{(n)}$ with regard to the union operation. The union, the intersection, and the composition are associative operations, and the composition distributes over the union.[3]

With regard to the opposite graph, we observe that $(GG')^* = G'^* G^*$ and $(G \cup G')^* = G^* \cup G'^*$ for all $G, G' \in \mathfrak{G}^{(n)}$.

3.4. Special Graphs and Graph Operations. For two graphs $G, G' \in \mathfrak{G}^{(n)}$, we introduce the *conjugation* $G \diamond G' := GG'G^*$.[4] In particular, when applied to an undirected graph, the conjugation results in another undirected graph, with the node corridor given by the source nodes of that graph. Figure 8 illustrates the results of the composition and the conjugation applied to an undirected graph.

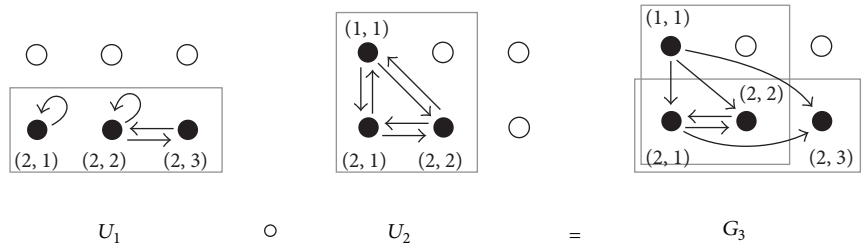
For brevity, we generally omit the composition symbol "" and the juxtaposition of two graphs indicates the composition. Often, we assume that $\mathscr{V}_k$ is isomorphic to the integers $\mathbb{Z}$ and represent a node $v_{k^{(n)}}$, $k^{(n)} \in \mathbb{Z}^n$, with the respective tupel $k^{(n)}$.

If, for a node corridor $\mathscr{W}$, the set of source nodes $\mathscr{S}$ is identical to the set of target nodes $\mathscr{T}$, that is, $\mathscr{S} = \mathscr{V}$ and $\mathscr{T} = \mathscr{V}$, and we define $\mathscr{V}$ by a tuple Ω of k sets of n_k natural numbers ranging from 0 to $n_k - 1$, then, as a shorthand notation, we write $\Gamma_\Omega(\mathscr{A})$ or $\Gamma_\Omega^{\mathscr{A}}$ for $\Gamma(\mathscr{W}, \mathscr{A})$ and $\Upsilon_\Omega(\mathscr{E})$ or $\Upsilon_\Omega^{\mathscr{E}}$ for $\Upsilon(\{\mathscr{V}\}, \mathscr{E})$. The identity graph is denoted as $D_\Omega := \Upsilon(\{\mathscr{V}\}, \{\{v\} \mid v \in \mathscr{V}\})$, and the respective n-node *empty graph* is denoted as $N_\Omega := \Upsilon(\{\mathscr{V}\}, \{\})$. With $v_k \in \mathscr{V}$, we also write J_Ω^k for $\Upsilon(\{\mathscr{V}\}, \{\{v_k\}\})$, and, with $V \subseteq \mathrm{cart}(\mathscr{V})$, we write J_Ω^V for $\Upsilon(\{\mathscr{V}\}, \{\{v\} \mid v \in V\})$. In these shorthand notations, Ω may be replaced by $\mathscr{V}$ itself.

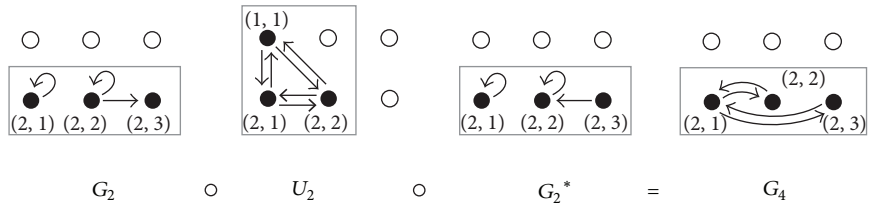
For two graphs $G = \Gamma(\mathscr{W}, \mathscr{A})$ and $G' = \Gamma(\mathscr{W}, \mathscr{A}')$ with identical node corridors $\mathscr{W}$, the *relative complement* is given as $G \setminus G' := \Gamma(\mathscr{W}, \mathscr{A} \setminus \mathscr{A}')$.

We introduce a *transfer graph* $T_\Omega^{\mathscr{A}} := T_\Omega^{\mathscr{A}} \cup (D_\Omega \setminus J_\Omega^V)$, with $V = d(\mathscr{A}) \cup r(\mathscr{A})$. The concept of the transfer graph is particularly important for the algorithm for the construction of hierarchically symmetric graphs, as described in Section 4.

The composition of a graph $\Gamma_\Omega(\mathscr{A}')$ with a transfer graph $T_\Omega^{\mathscr{A}}$, that is, the operation $\Gamma_\Omega(\mathscr{A}') \circ T_\Omega^{\mathscr{A}}$, shifts the domain of each arc $a' \in \mathscr{A}'$, for which $d(a') \in r(\mathscr{A})$, from $d(a')$ to $\{d(a) \mid r(a) = d(a'),\ a \in \mathscr{A}\}$. The conjugation $T_\Omega^{\mathscr{A}} \diamond G$ transfers both the domains and the ranges of the respective arcs of the graph



(a) The composition of two undirected graphs generally results in a graph that is not undirected



(b) The conjugation of an undirected graph results in an undirected graph

Figure 8: Composition and conjugation of undirected graphs.

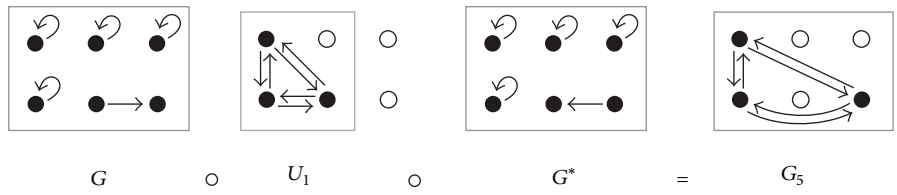


Figure 9: Conjugation of an undirected graph with a transfer graph G, $G = T_{(2,3)}^{\{(2|2,2|3)\}}$.

G. Figure 9 shows how a transfer graph can be used to modify an undirected graph by conjugation.

We denote a directed graph of n nodes that consists of a single *directed path* connecting the nodes sequentially as $\vec{P}_n \coloneqq \Gamma_n(\{(v_k, v_{k+1})\}_{k=1}^{n-1})$. If $\mathcal{S} = \mathcal{T} \cong X$, where X is an ordered set, $\vec{P}_X$ denotes a graph that connects all nodes represented by X sequentially, according to the order of X. For example, $\vec{P}_{\mathbb{Z}} \coloneqq \Gamma_{\mathbb{Z}}(\bigcup_{k\in\mathbb{Z}}\{(v_k, v_{k+1})\})$. We define the respective *directed cycle graph*, which also contains an arc that connects the last node to the first node, by $\vec{C}_n \coloneqq \Gamma_n(\{(v_k, v_{k+1})\}_{k=1}^{n-1} \cup \{(v_n, v_1)\})$, or $\vec{C}_n \coloneqq \Gamma_n(\{(v_k, v_{k+1})\}_{k=1}^{n})$, if k is defined as an element of the cyclic group $\mathbb{Z}/n\mathbb{Z}$. These specific graphs are generally defined as undirected graphs P_n and C_n, with $P_n = \upsilon(\vec{P}_n)$ and $C_n = \upsilon(\vec{C}_n)$.

3.5. The Categorical Product. The categorical product allows combining graphs in a way that models the hierarchical aspects of the underlying structure. From two graphs, $G \in \mathfrak{G}^{(m)}$ and $G' \in \mathfrak{G}^{(n)}$, a graph in the graph space $\mathfrak{G}^{(m\cdot n)}$ can be constructed.

The categorical product of two arcs is given by

$$a \otimes a' \coloneqq \left(\left(d(a), d\left(a'\right)\right), \left(r(a), r\left(a'\right)\right)\right), \tag{12}$$

and the categorical product of two sets of arcs is given by

$$A \otimes A' \coloneqq \left\{a \otimes a' \mid a \in A,\ a' \in A'\right\}. \tag{13}$$

With regard to two node corridors, $\mathscr{W} = (\mathcal{S}, \mathcal{T})$ and $\mathscr{W}' = (\mathcal{S}, \mathcal{T}')$, we define the categorical product as $\mathscr{W} \otimes \mathscr{W}' \coloneqq ((\mathcal{S}, \mathcal{S}'), (\mathcal{T}, \mathcal{T}'))$.

The *categorical product* of two graphs [41] is thus given by

$$\Gamma(\mathscr{W}, \mathscr{A}) \otimes \Gamma\left(\mathscr{W}', \mathscr{A}'\right) \coloneqq \Gamma\left(\mathscr{W} \otimes \mathscr{W}', \mathscr{A} \otimes \mathscr{A}'\right) \in \mathfrak{G}^{(m\cdot n)}. \tag{14}$$

The composition distributes over the categorical product, and the categorical product distributes over the union: for $G_1, G_3 \in \mathfrak{G}^{(m)}$ and $G_2, G_4 \in \mathfrak{G}^{(n)}$, we have $(G_1 \otimes G_2)(G_3 \otimes G_4) = G_1G_3 \otimes G_2G_4$. We also observe that $G_1 \otimes G_2 \cup G_3 \otimes G_2 = (G_1 \cup G_3) \otimes G_2$ and $G_1 \otimes G_2 \cup G_1 \otimes G_4 = G_1 \otimes (G_2 \cup G_4)$. Note that $G_1 \otimes G_2 \cup G_3 \otimes G_4 \not\equiv (G_1 \cup G_3) \otimes (G_2 \cup G_4)$, and, in particular, $\upsilon(G_1 \otimes G_2) \not\equiv \upsilon(G_1) \otimes \upsilon(G_2)$. However, for the intersection, we have $G_1 \otimes G_2 \cap G_3 \otimes G_4 = (G_1 \cap G_3) \otimes (G_2 \cap G_4)$.[5]

Figure 10 shows the result of the categorical product of two graphs. We use the notation "$\otimes$" instead of the more commonly used notation "$\times$" to avoid ambiguity with regard to the Cartesian product of sets.

4. Hierarchically Symmetric Graphs

If a structure exhibits both symmetries and hierarchies, then these properties and their relationship should be preserved when describing such a structure as a graph. Often, different

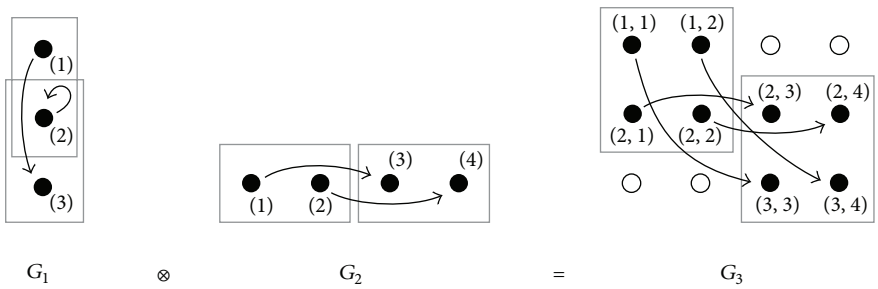
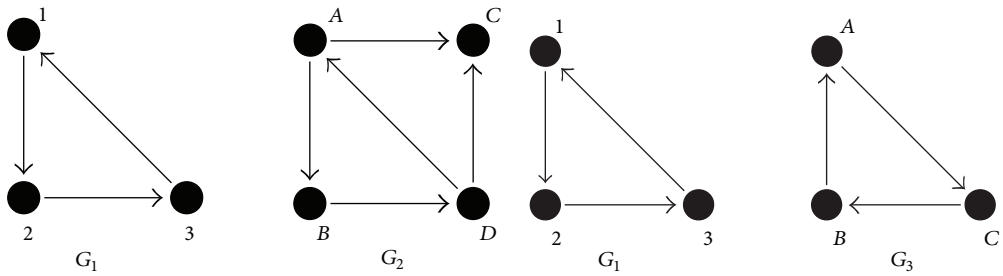


Figure 10: The categorical product.



(a) For the graphs $G_1 = \Gamma(\{1,2,3\}, \mathscr{A}_1)$, $\mathscr{A}_1 = \{(1,2), (2,3), (3,1)\}$, $G_2 = \Gamma(\{A,B,C,D\}, \mathscr{A}_2)$, and $\mathscr{A}_2 = \{(A,B), (A,C), (B,D), (C,D), (D,A)\}$, the map $\varphi : G_1 \to G_2$, given by $\varphi_{\mathfrak{V}} = \{1 \mapsto A,\ 2 \mapsto B,\ 3 \mapsto D\}$ and $\varphi_{\mathfrak{A}} = \{a \mapsto \{\varphi_{\mathfrak{V}}(v) \mid v \in a\}\}$, is a graph homomorphism. In contrast, the maps defined by $\varphi_{\mathfrak{V}} = \{1 \mapsto A,\ 2 \mapsto B,\ 3 \mapsto D\}$, $\varphi_{\mathfrak{A}} = \{a \mapsto \{\varphi_{\mathfrak{V}}(v) \mid v \in a\}\}$, $\varphi_{\mathfrak{V}} = \{1 \mapsto A,\ 2 \mapsto C,\ 3 \mapsto D\}$, and $\varphi_{\mathfrak{A}} = \{a \mapsto \{\varphi_{\mathfrak{V}}(v) \mid v \in a\}\}$ are not graph homomorphisms

(b) The graph G_3 is given by $\Gamma(\{A,B,C\}, \{(A,C), (B,A), (C,B)\})$. The map $\varphi : G_1 \to G_3$, given by $\varphi_{\mathfrak{V}} = \{1 \mapsto C,\ 2 \mapsto B,\ 3 \mapsto A\}$ and $\varphi_{\mathfrak{A}} = \{a \mapsto \{\varphi_{\mathfrak{V}}(v) \mid v \in a\}\}$, is a graph isomorphism. The map $\varphi : G_1 \to G_1$, given by $\varphi_{\mathfrak{V}} = \{1 \mapsto 2,\ 2 \mapsto 3,\ 3 \mapsto 1\}$ and $\varphi_{\mathfrak{A}} = \{a \mapsto \{\varphi_{\mathfrak{V}}(v) \mid v \in a\}\}$, is a graph automorphism

Figure 11: Graph homomorphisms.

symmetry characteristics present in a given structure are hierarchically related, making the hierarchical aspect of the graph-based description the precondition for the precise description of the symmetry characteristics.

Therefore, the concept of the hierarchically symmetric graph is central to an efficient description of such structures. Hierarchically symmetric graphs of order n are a subset of the graph space $\mathfrak{G}^{(n)}$. They are characterized by a hierarchical structure, with nodes identified by tuples, and symmetry relations between parts of the graph that can be identified by the hierarchical description.

The super carbon nanotubes described in the preceding sections are pertinent examples of hierarchically symmetric graphs.

In graphs, symmetries can be characterized by graph automorphisms. We therefore recall the formal definition of a graph automorphism, which is based on the more general concept of a graph homomorphism.

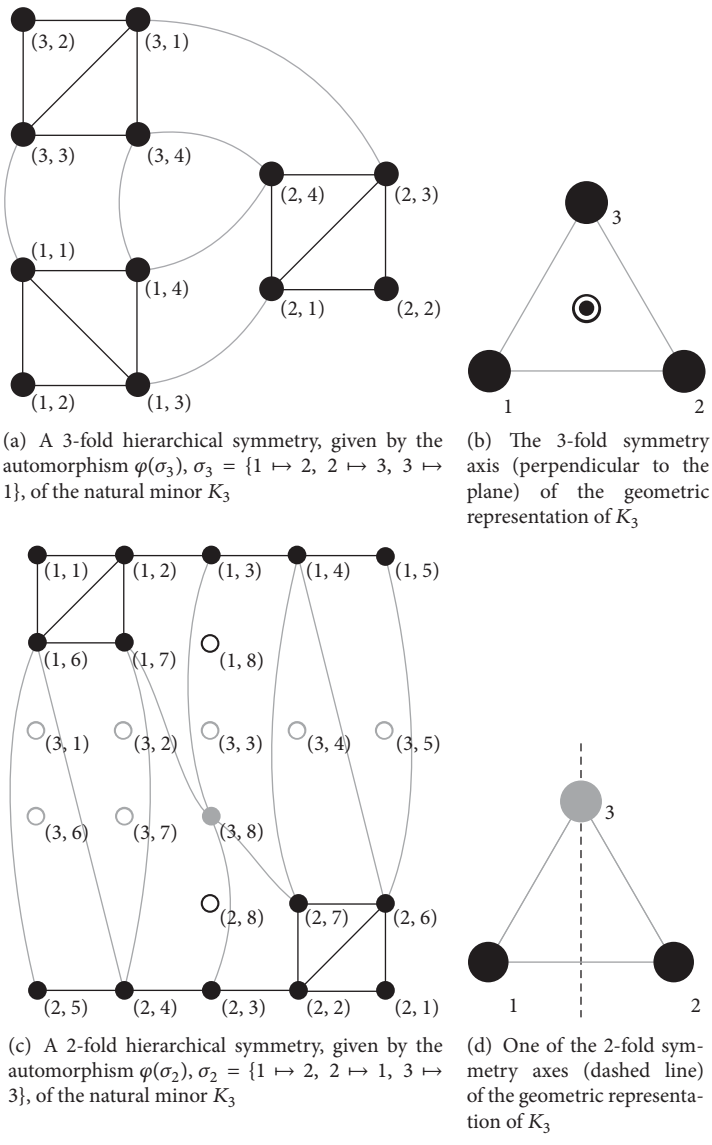
Minors play an important role in graph theory, as they help to describe basic properties of graphs. In the following, we will combine the concept of a graph minor with the concept of a hierarchical graph, introducing the notion of a *natural minor*.

4.1. Identifying Symmetries: Graph Homomorphisms. A *graph homomorphism* is a function $\varphi : \mathfrak{G}^{(n)} \to \mathfrak{G}'^{(n)}$, $\varphi = \varphi_{\mathfrak{V}} \oplus \varphi_{\mathfrak{A}}$, which maps each node $v \in \mathscr{V}$ to a node $\varphi_{\mathfrak{V}}(v) \in \mathscr{V}'$ and each arc $a \in \mathscr{A}$ to an arc $\varphi_{\mathfrak{A}}(a) \in \mathscr{A}$, so that $d(\varphi_{\mathfrak{A}}(a)) = \varphi_{\mathfrak{V}}(d(a))$ and $r(\varphi_{\mathfrak{A}}(a)) = \varphi_{\mathfrak{V}}(r(a))$. Figure 11(a) provides examples of a graph homomorphism.

A *graph isomorphism* is a bijective graph homomorphism. A *graph automorphism* is a graph isomorphism that maps a graph onto itself. Graph automorphisms can be induced by node permutations, as for any node permutation σ there is a unique arc mapping $\varphi_{\mathscr{A}}(\sigma) : a \mapsto (\sigma(d(a)), \sigma(r(a)))$.[6] Figure 11(b) provides examples of a graph isomorphism and a graph automorphism.

A graph is said to have an *n-fold symmetry* if there exists a graph automorphism φ_σ, induced by a node permutation, such that $\sigma^n = \mathrm{Id}$ and $\sigma^k \neq \mathrm{Id}$ for $0 < k < n$. Two graphs are symmetric with regard to each other if they can be mapped onto each other by an isomorphism.[7]

4.2. Describing Hierarchical Graphs: Graph Minors, Projections, and Symmetries. We recall that a hierarchical graph of order n is an element of the graph space $\mathfrak{G}^{(n)}$. Following



(a) A 3-fold hierarchical symmetry, given by the automorphism $\varphi(\sigma_3)$, $\sigma_3 = \{1 \mapsto 2,\ 2 \mapsto 3,\ 3 \mapsto 1\}$, of the natural minor K_3

(b) The 3-fold symmetry axis (perpendicular to the plane) of the geometric representation of K_3

(c) A 2-fold hierarchical symmetry, given by the automorphism $\varphi(\sigma_2)$, $\sigma_2 = \{1 \mapsto 2,\ 2 \mapsto 1,\ 3 \mapsto 3\}$, of the natural minor K_3

(d) One of the 2-fold symmetry axes (dashed line) of the geometric representation of K_3

Figure 12: Hierarchically symmetric graphs. K_3 denotes the complete graph with three nodes.

the established concept of a graph *minor* [36, p. 18–21], we introduce, in the context of hierarchical graphs, the concept of the *natural minor*. The *natural branch sets of order k* of a hierarchical graph are the subsets of source nodes and target nodes that contain nodes whose index tuples are identical except for their last k entries. By contracting those sets of source nodes and target nodes, we can construct a minor of the hierarchical graph, which we will call the *natural minor of order k* and denote as $\pi^k(G)$. Figures 12(b) and 12(d) show the natural minors of order one of the hierarchical graphs shown in Figures 12(a) and 12(c), respectively.

The construction of a natural minor can also be thought of as a natural projection of a hierarchical graph of order n onto its outermost $n-1$ hierarchy levels: to remove the last element from an n-element-tuple $t = (t_k)_{k=1}^n$, we define the function $\pi : t \mapsto (t_k)_{k=1}^{n-1}$. Then, for a graph $G = \Gamma((\mathcal{S}, \mathcal{T}), \mathcal{A}) \in \mathfrak{G}^{(n)}$, the function $\pi : G \mapsto \Gamma((\pi(\mathcal{S}), \pi(\mathcal{T})), \{\pi(a) \mid a \in \mathcal{A}\})$ maps G to its first natural minor. Thus, for every hierarchical graph, there exists a well-quasi-ordered set [36, p. 316-317] of minors given by $G \succcurlyeq \pi(G) \succcurlyeq \pi^2(G) \succcurlyeq \cdots \succcurlyeq \pi^n(G)$, which we call a *hierarchy of natural minors*.

A graph automorphism can be said to be induced by an automorphism of its minor, or be described as a *minor-induced automorphism*, if the automorphism maps all nodes of one branch set to the nodes of another branch set while preserving the relations between them. An automorphism φ of a graph $G \in \mathfrak{G}^{(n)}$, induced by its kth natural minor, thus maps the node v_I, $I = (i_m)_{m=1}^n$, to the node $v_{\tilde{I}}$, $\tilde{I} = \tilde{\varphi}((i_m)_{m=1}^{n-k}) \times (i_m)_{m=n-k+1}^n$, with $\tilde{\varphi}$ given by an automorphism of its kth natural minor $\pi^k(G)$.

We say that a hierarchical graph has an *n-fold hierarchical symmetry of order k* if there exists an automorphism, induced by its kth minor, such that $\varphi^n = \mathrm{Id}$ and $\varphi^m \neq \mathrm{Id}$ for $0 < m < n$. Figure 12 depicts two hierarchically symmetric

graphs, which both share the complete 3-node graph $K_3 = \Upsilon(\{1,2,3\},\{\{1,2\},\{1,3\},\{2,3\}\})$ as a natural minor.

A natural minor of a hierarchical graph can be understood as its natural projection on its outermost hierarchy levels. However, hierarchical graphs may also be projected on arbitrary hierarchy levels. Let σ be a reordering of an n-element-tuple, and, for a graph $G = \Gamma((\mathcal{S},\mathcal{T}),\mathcal{A}) \in \mathfrak{G}^{(n)}$, $\sigma : G \mapsto \Gamma((\sigma(\mathcal{S}),\sigma(\mathcal{T})),\{\sigma(a) \mid a \in \mathcal{A}\})$, such that $\pi^{n-k} \circ \sigma(G)$ is the projection of G onto k hierarchy levels. Then, a *projection-induced automorphism* φ, with $\tilde{\varphi}$ given by an automorphism of its projection on k hierarchy levels, maps the node v_I, $I = (i_m)_{m=1}^{n}$, to the node $v_{\tilde{I}}$, $\tilde{I} = \sigma^{-1}(\tilde{\varphi}((p_m \circ \sigma(I))_{m=1}^{k}) \times (p_m \circ \sigma(I))_{m=k+1}^{n})$. Thus, we first permute the entries of the tuples identifying the nodes, so that all symmetry operations that we want to consider are being induced by the respective natural minor. We then apply the symmetry operations. Finally, we reverse the permutation of the entries of the node tuples.

For a projection-induced automorphism φ_k, let P_k denote the set of invariant hierarchy levels of the natural projection inducing the automorphism. We say that a hierarchical graph has a $(n_k)_{k=1}^{m}$*-fold multiple hierarchical symmetry* if there exists a graph automorphism $\Phi \coloneqq \bigoplus_{k=1}^{m} \varphi_k$, based on k projection-induced automorphisms, such that $P_k \cap P_{k'} = \{\}$ for $k \neq k'$. The graphs representing the structure of symmetric regular super carbon nanotubes exhibit this kind of multiple hierarchical symmetries.

4.3. Construction of Hierarchically Symmetric Graphs. Hierarchically symmetric graphs can be constructed by combining multiple instances of a given graph in a way that preserves graph isomorphisms between these instances. In the simplest case, a graph is duplicated by taking the categorical product: $G \mapsto D_n \otimes G$. This results in n disconnected subgraphs which are pairwise isomorphic. We call the graph G, which constitutes the starting point of the construction process, the *constituent graphs*. For translational and reflectional symmetries, there is often a need to define a *border branch set*, which belongs to two isomorphic subgraphs of the resulting graph and separates the remaining parts of these subgraphs, which are mapped to the *interior branch sets*. For rotational symmetries, an *invariant branch set*, which can be thought of as being situated on the rotational axis, must be defined. Figure 13 depicts examples of hierarchically symmetric graphs, their branch sets, and their constituent graphs.

In most cases, the interior and the border subgraphs of a constituent graph will not be natural branch sets of that graph. In order to modularize the overall construction, it is useful to add another dimension to the graph that separates the subgraphs into natural branch set. In this case, it is much easier to define transfer graphs that can be used to actually construct the hierarchically symmetric graph. Figure 14 shows an example of the steps involved in the construction of a hierarchically symmetric graph.

Note the composition of the graph G', which is derived from the constituent graph G, with the transfer graph T and the symmetry graph S, involves graphs of different dimensions. In this case, the composition is given by $G_1 \circ G_2 \coloneqq (G_1 \otimes \bigotimes_{k=1}^{n-m} D_{\mathfrak{B}_k}) \circ G_2$ for dimension n of G_2 greater than dimension m of G_1 (the definition for $m > n$ is symmetric). The transfer graph only depends on the choice of the natural minor of the hierarchical graph that is being constructed and on the border branch sets assigned to the constituent graph. Thus, the symmetry characteristic of the resulting hierarchically symmetric graph does not depend on the internal structure of the interior or the border subgraph of the constituent graph.

5. Constructing Super Carbon Nanotubes of Arbitrary Order

The concepts introduced in the preceding sections enable us to model the structure of carbon nanotubes (CNTs) of higher order in a systematic way. The result, a hierarchically symmetric graph, allows one to readily identify hierarchies and symmetries at all hierarchy levels, as well as similarities between the different levels of the hierarchy. In addition, by evaluating the geometry-related functions set up in conjunction with the graph, a mesh in the traditional sense, that is, based on a one-dimensional, sequentially ordered set of nodes, can be derived from the graph at low computational cost.

5.1. Hierarchically Symmetric Graphs and Geometry. Meshes are generally understood as a collection of information of (a) a list of nodes together with their geometric position and (b) a list of elements (which may reduce to simple edges) together with a function that associates each element (or edge) to a list of nodes. The latter information may be provided by a connectivity list, a common approach in FE models.

A graph, however, does not contain any information about the geometry. While geometric representations of graphs are commonly used to illustrate graphs, all operations on graphs are independent of the geometric position assigned to its nodes, and graphs indeed do not necessarily represent objects that do have a reasonable geometric representation.

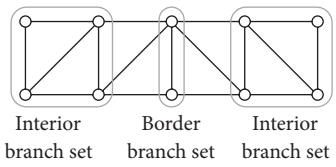
Therefore, if a graph created by the Hierarchical Graph Meshing method presented in this paper is to be used to investigate the mechanical properties of a structure, it must be combined with geometric information about the position of the nodes.[8]

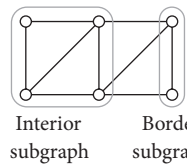
In the most general case, a mesh Ω associated with a hierarchical graph $G \in \mathfrak{G}^{(n)}$ can be denoted as a tuple $\Omega = (G,\Phi)$, in which $\Phi : (\mathfrak{V}, e_V) \to V$ is a function that maps each node $v \in \mathfrak{V} = \mathfrak{V}_1 \times \mathfrak{V}_2 \times \cdots \times \mathfrak{V}_n$ to a position in the vector space V.

For a *geometrically hierarchical mesh*, the function Φ is a composition of functions φ_k, such that

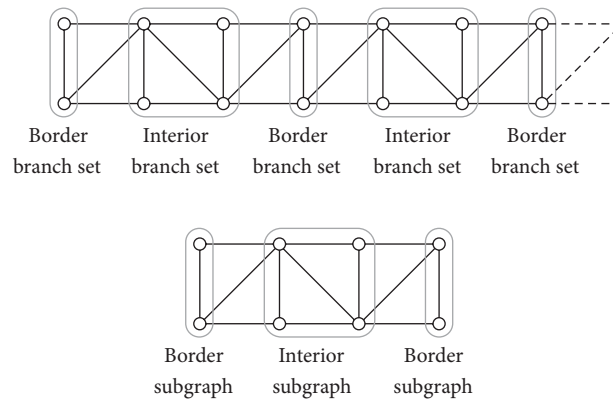
$$\Phi(v, e_V) = \varphi_n(v_n) \circ \varphi_{n-1}(v_{n-1}) \circ \cdots \circ \varphi_1(v_1)(e_V), \quad (15)$$

where each function φ_k, $k > 1$, takes the domain of the preceding function $r(\varphi_{k-1})$ and the kth index of the tuple representing the node as arguments: $\varphi_k : \mathfrak{V}_k \times r(\varphi_{k-1}) \to r(\varphi_k)$. The codomain of the nth function, φ_n, is the vector space V: $r(\varphi_n) = V$, and the domain of the function φ_1 is $(\mathfrak{V}_1, e_V)$.

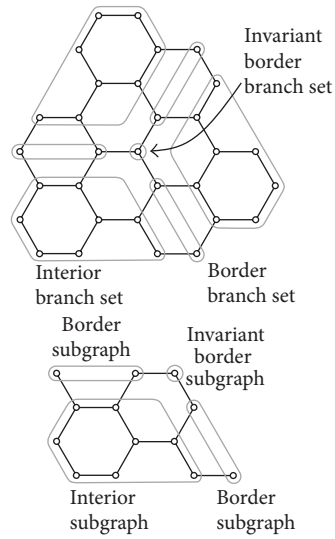




(a) Reflectional symmetry. The node branches of the hierarchically symmetric graph and the subgraphs of the constituent graph



(b) Translational symmetry. The node branches of the hierarchically symmetric graph and the subgraphs of the constituent graph. Border branch sets may be distinguished into interior and boundary border branch sets



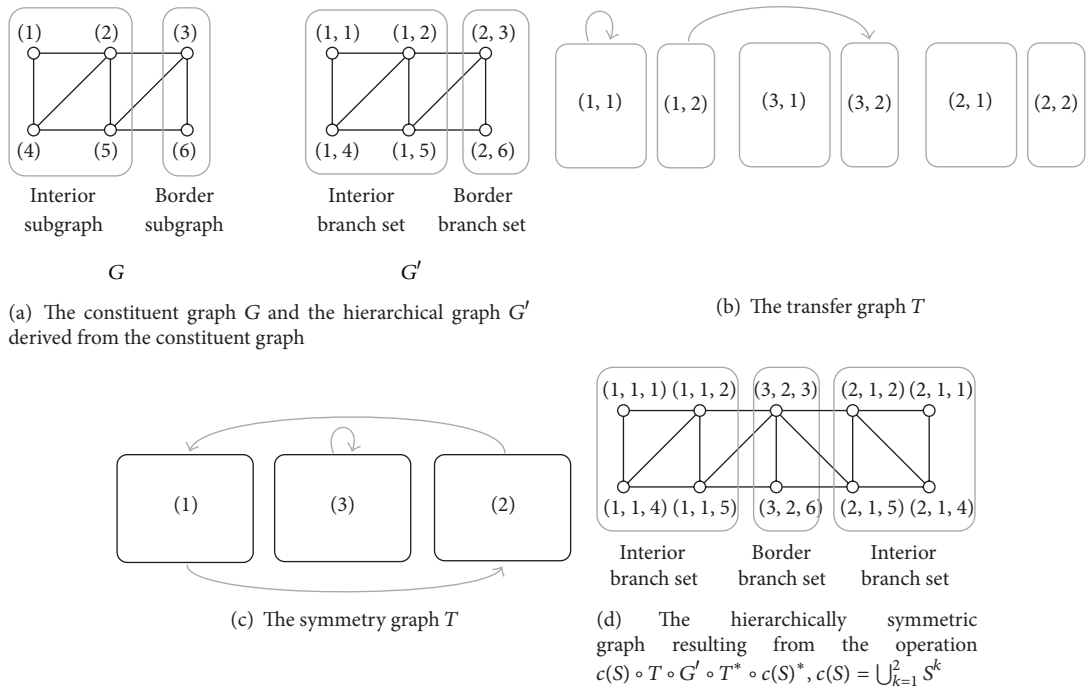
(c) Rotational symmetry. The node branches of the hierarchically symmetric graph and the subgraphs of the constituent graph

Figure 13: Symmetries of hierarchically symmetric graphs. Isolated notes are omitted in the figures.

For a *hierarchically symmetric mesh*, the function $\Phi : \mathfrak{V} \to V$ is a composition of functions ϕ_k, as in the case of the geometrically hierarchical mesh, but with the additional constraint that, for each k, $1 \leqslant k \leqslant n$, there exists a bijection σ_k from the canonical projection of the node space $\mathfrak{V}$, that is, $\mathfrak{V}_k$, to a geometric symmetry group such that $\varphi_k(v_k) = \mu_k \circ \sigma_k(v_k) \circ \rho_k$.[9] The functions μ_k and ρ_k do not depend on the node v_k. This condition ensures not only that the branch sets of the hierarchical graph associated with the mesh are symmetrical as subgraphs but also that their geometric representation is symmetrical in the vector space V (before the application of the function μ_k). A special case of a hierarchically symmetric mesh, which may be called *strongly hierarchically symmetric mesh*, is given for $\mu_k = \mathrm{Id}$ and $\rho_k = \mathrm{Id}$.

The SCNTs that will be constructed in the following sections can be described by hierarchically symmetric meshes. A graphene sheet, before being "rolled" into a tube, can be described as a strongly hierarchically symmetric mesh.

5.2. Iterative Construction. In particular, the construction of each hierarchical order starts with the result of the construction of the preceding order, and the steps of the construction process of each hierarchical order are structurally identical, providing a generalized description of the construction process of the configurations of CNTs of arbitrary hierarchical order. The sequence of hierarchical orders is shown in Table 2. Figure 15 depicts the flow of algorithm for the construction of a super carbon nanotube of arbitrary order.[10]



(a) The constituent graph G and the hierarchical graph G' derived from the constituent graph

(b) The transfer graph T

(c) The symmetry graph T

(d) The hierarchically symmetric graph resulting from the operation $c(S) \circ T \circ G' \circ T^* \circ c(S)^*$, $c(S) = \bigcup_{k=1}^{2} S^k$

Figure 14: Construction of a hierarchically symmetric graph. Isolated notes are omitted in the figures.

Table 2: Hierarchical structures based on carbon nanotubes.

	Order 0	Order 1	⋯
(Initial structure)	Single atom[a]	Junction	⋯
Step 1	Graphene sheet	Sheet of CNTs	⋯
Step 2	Tube	Tube of CNTs	⋯
Step 3	Junction	Junction of CNTs	⋯

[a]The respective graph is actually the zero-dimensional graph.

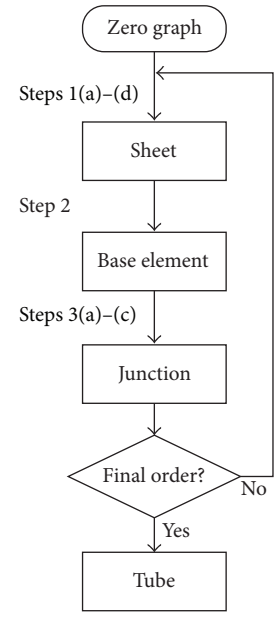


Figure 15: Sequence of the iterative construction of a SCNT of arbitrary order.

Figure 16 illustrates the process of the construction of the mesh of a hierarchically structured CNT. In this example, a zigzag tube is created, which is characterized by a zigzag sequence of carbon atoms (or junctions, at higher orders) along the circumference of the tube. For each order, the starting point is the CNT (or SCNT) junction resulting from the construction of the structure at the preceding order. For the kth level, we denote the respective junction by J_k.

However, as the initial structure for order 0 is the zero graph $J_0 = Z = \Gamma(\{\}, \{\})$, many of the steps of the first iteration level do not explicitly show the full characteristics of the algorithm. For example, the junction J_1 has three separate border node sets, each belonging to a branch of the junction. For the "junction" J_0, all these border node sets collapse to the zero graph. Thus, while Step 1(a) can be expressed as $S_{k,1} = D_2 \otimes J_1 \cup \upsilon(\vec{P}_2 \otimes \partial_4 B_{k-1,3})$ for each order k, connecting two junctions by placing arcs between their respective border node sets, we have $S_{0,1} = D_2 \otimes J_1 \cup \upsilon(\vec{P}_2 \otimes D_7^{(1,1)} \otimes Z) = D_2 \otimes J_1 \cup \upsilon(\vec{P}_2)$ for order 0, in which a pair of arcs (representing an edge) is placed between two isolated nodes.

The following description will therefore generally refer to the situation for order 1 and above and provide some additional remarks concerning the special case of the construction of the CNT of order 0.

We write $\mathcal{S}_{R,n}(r_0, \phi)$ for the n-element group of rotations around r_0 with rotation vector ϕ, $\mathcal{S}_M(r_0, \nu)$ for the 2-element group of reflections about the plane defined by the reference point r_0 and the normal vector ν, and $\mathcal{S}_{T,n}(\tau)$ for the n-element group of translations $\{r \mapsto r + j\tau\}_{j=0}^{n-1}$ with translation

Step	Order 0	Order 1
Initial structure	(Zero graph) (a) $J_0 = Z = \Gamma(\{\}, \{\})$	$\{2\} \times \partial_4 B_{0,3}$, $\{1\} \times \partial_4 B_{0,3}$, $\{3\} \times \partial_4 B_{0,3}$ (Junction) (i) J_1
1(a)	(1) (2) (b) $S_{0,1} = D_2 \otimes J_0 \cup \upsilon(\vec{P}_2)$	$\{(1,1)\} \times \partial_4 B_{0,3}$, $\{(2,1)\} \times \partial_4 B_{0,3}$ (j) $S_{1,1} = D_2 \otimes J_1 \cup \upsilon(\vec{P}_2 \otimes D_7^{(1,1)} \otimes \partial_4 B_{0,3})$
1(b)	(2, 1) (2, 2) (1, 1) (1, 2) (c) $S_{0,2} = D_2 \otimes S_{0,1} \cup \upsilon(\vec{P}_2 \otimes \vec{P}_2{}^*)$	$\{(2,1,3)\} \times \partial_4 B_{0,3}$, $\{(1,2,2)\} \times \partial_4 B_{0,3}$ (k) $S_{1,2} = D_2 \otimes S_{1,1} \cup \upsilon(\vec{P}_2 \otimes \vec{P}_2{}^* \otimes D_7^{(2,3)} \otimes \partial_4 B_{0,3})$
1(c)	(d) $S_{0,3} = D_3 \otimes S_{0,2} \cup \upsilon(L_3)$	(l) $S_{1,3} = D_3 \otimes S_{1,2} \cup \upsilon(L_3 \otimes D_7^{(2,3)} \otimes \partial_4 B_{0,3})$
1(d)	(Graphene sheet) (e) $S_{0,4} = D_2 \otimes S_{0,3} \cup \upsilon(L_4)$	(CNT sheet) (m) $S_{1,4} = D_2 \otimes S_{1,3} \cup \upsilon(L_4 \otimes D_7^{(2,3)} \otimes \partial_4 B_{0,3})$
2	1 2 3 4 5 6 (Base element) (f) $B_{0,1}$	1 2 3 4 5 6 $\{(5,2,3,1,2,5)\} \times \partial_6 B_{0,3}$, $\{(4,2,2,2,1,1)\} \times \partial_4 B_{0,3}$, $\{(4,2,1,2,1,1)\} \times \partial_4 B_{0,3}$ (Base el.) (n) $B_{1,1}$
3(a)	(g) $B_{0,2}$	(o) $B_{1,2}$
3(b)	$\partial_4 B_{0,3}$ (h) $B_{0,3}$	$\partial_4 B_{1,3}$ (p) $B_{1,3}$

FIGURE 16: Iterative construction of a hierarchically structured SCNT of arbitrary order. For arbitrarily large orders, the table can be thought of as extending to an infinite number of columns to the right. For any order k, $k \geq 1$, the initial structure is the result of the last step of order $k-1$. Step 3(b) is followed by Step 3(c), which results in the junction shown in the first row of the second column. The graphs L_3 and L_4 are given as $L_3 := \vec{P}_3 \otimes \vec{P}_2{}^* \otimes \vec{P}_2$ and $L_4 := D_2 \otimes (D_3 \cup \vec{P}_3) \otimes \vec{P}_2{}^* \otimes \vec{P}_2{}^*$.

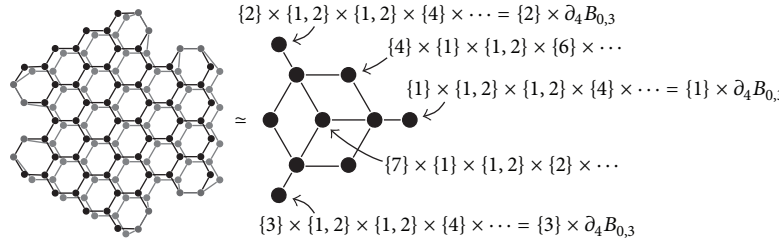


FIGURE 17: The junction J_1.

vector τ. The vector space V is generated by the orthonormal basis vectors $(e_i)_{i=1}^3$. The length of s_0e_1 is the distance between two carbon atoms in a plane graphene sheet, and the length of s_ke_1, $k \geqslant 1$, is the distance between the rotational symmetry axes of two junctions in a plane CNT or SCNT sheet of order k.

Initial Structure. The initial structure for each order k, that is, the junction J_k, is the result of the construction process of the previous order (see Figure 16(i)). In particular, for order k, $k \geqslant 1$, it is the result of combining twelve graphene (or CNT) sheets $B_{k-1,2}$ (see Figures 16(g) and 16(o)). Each of these sheets contains a set of internal nodes, indicated by a leading node index of 1, and several border node sets, indicated by a leading node index greater than 1. However, only the node set indicated by the leading node index 4 remains a border node set after the construction of the junction J_k. As in each of the construction steps, an additional position is added to the node indices; the three border node sets of J_k are identified as $\{j\}\times\partial_4B_{0,3} = \{j\}\times\{1,2\}\times\{1,2\}\times\{4\}\times\cdots$, $j \in \{1,2,3\}$. Figure 17 shows the junction of order 1 and the respective node sets.

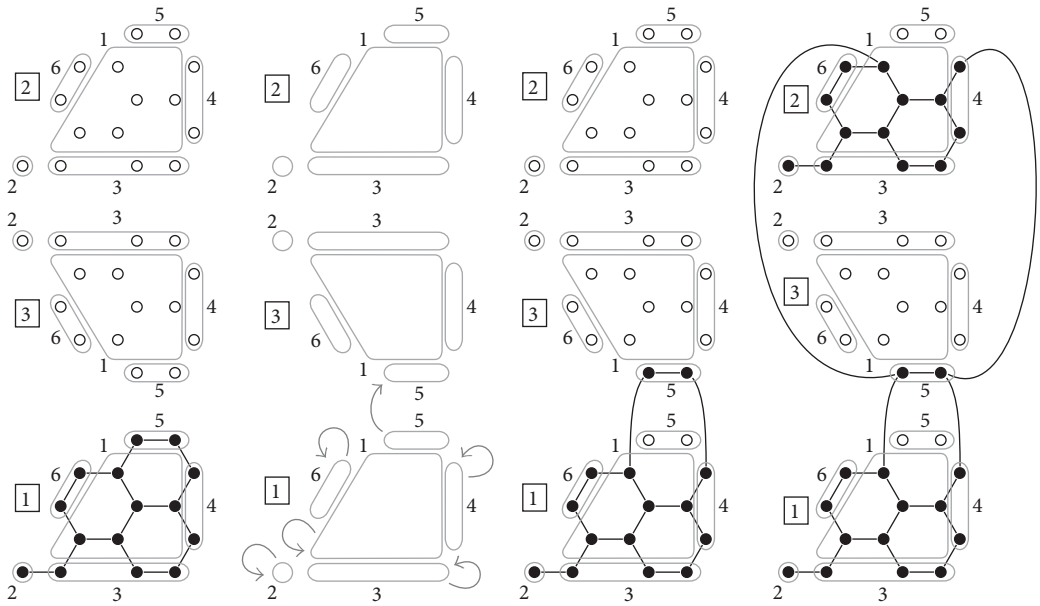
Step 1. (a) In the first step (see Figure 16(j)), the junction J_k is being duplicated, resulting in the graph $D_2\otimes J_k$. Subsequently, arcs are being placed between the respective border node sets of the junctions, given by $(1,1)\times\partial_4B_{k-1,3}$ for the "left" junction and $(2,1)\times\partial_4B_{k-1,3}$ for the "right" junction. The trailing indices of two nodes that are being connected by this operation are identical; thus the additional arcs are given by $\upsilon(\vec{P}_2\otimes D_7^{(1,1)}\otimes\partial_4B_{k-1,3})$. Here, $\vec{P}_2$ indicates that the two junctions are being connected, $D_7^{(1,1)}$ means that the "first" branch of each junction is being connected, and the expression $\partial_4B_{k-1,3}$ indicates that only the nodes belonging to the border node set 4 are included in the operation. For the order 0 (see Figure 16(b)), the operation $D_2\otimes J_0$ results in a one-dimensional graph, and thus the term $\upsilon(\vec{P}_2\otimes D_7^{(1,1)}\otimes\partial_4B_{k-1,3})$ collapses to $\upsilon(\vec{P}_2)$. The symmetry group associated with this dimension of the graph is the reflection group $\mathcal{S}_M((1/2)s_ke_1, e_1)$.

(b) The second part of the first step again duplicates the result of Step 1(a) and adds arcs between the respective node sets of the junctions, identified by the leading indices $(1,2,2)$ and $(2,1,3)$. Again, the expression describing the operation for order 0 is much simpler. The symmetry group associated with this dimension of the graph is the translation group $\mathcal{S}_{T,2}((3/2)s_ke_1 + (1/2)\sqrt{3}s_ke_2)$. (See Figures 16(c) and 16(k).)

(c) Subsequently, the junctions along the circumference of the resulting CNT (or super CNT) are created by replicating the result of the previous step. Choosing this direction with regard to the geometry of the structure created in the previous step results in the zigzag configuration, while choosing the orthogonal direction would result in the armchair configuration. This operation can be expressed as $D_m\otimes S_{k,2}$ (shown in Figures 16(d) and 16(l) for $m = 3$). Again, arcs between the respective node sets of the junctions, identified by the leading indices $(j,2,1,2)$ and $(j+1,1,2,3)$, for $1 \leqslant j < m$, are being added. If, instead of D_m, a cycle graph C_m is being used, then the last junction and the first junction are also joined by additional arcs, resulting in a ring that can be replicated by the operation described in Step 1(d) in order to obtain a CNT or a SCNT. The respective symmetry group is $\mathcal{S}_{T,m}(\sqrt{3}s_ke_2)$.

(d) The construction of a rectangular graphene (or CNT) sheet is completed by replicating the graphene (or CNT) ribbons resulting from the previous step along the length axes of the resulting CNT (or SCNT), resulting in n ribbons. As in the previous steps, the border node sets that must be connected by additional arcs are identified by their leading indices, and the expression is much simpler for the order 0. (See Figures 16(e) and 16(m).) The respective symmetry group is $\mathcal{S}_{T,n}(3s_ke_1)$.

Step 2. For the construction of a junction of reasonable geometry, it is necessary to select a trapezoidal area of the graphene (or CNT) sheet produced in the previous step (see Figures 16(f) and 16(n)). A junction is composed of three branches, each of which consists of four base elements. As the branches intersect at an angle of 120°, trapezoidally shaped base elements with an angle of 60° at the rotational axis can be glued together without inducing major deformations of the base elements. Therefore, it is necessary to select the nodes that constitute the graph-based description of this area along with the arcs located between them. This can be done by first constructing a loop graph that connects each of the nodes belonging to the trapezoid area to itself, resulting in the graph $\widetilde{D}_k$, and then performing the conjugation of the graph $S_{k,4}$ with $\widetilde{D}_k$, creating $B_{k,1} = \widetilde{D}_k\circ S_{k,4}\circ\widetilde{D}_k$. The loop graph is obtained as the intersection of four loop graphs that consist of connected nodes on one half space of the node space and isolated nodes on the rest of the node space. As can be seen in Figure 16(n), the index combinations identifying the border node sets can be rather complicated. However, for all orders



(a) $D_3^1 \otimes B_{0,1}$ (b) T_1 (c) $S_{0,1} = T_1{}^*(D_3^1 \otimes B_{0,1})T_1$ (d) $B_{0,2} = \bigcup_{j=1}^{2}(R_1{}^{*j} S_{0,1} R_1{}^j)$

Figure 18: Construction of $B_{0,2}$: $T_1 = D_3^1 \otimes (D_6 \setminus D_6^5) \cup \Gamma_3((1,3)) \otimes D_6^5$ and $R_1 = C_{\{1,2\}} \cup D_3^1 = \Gamma_3(\{(1,2),(2,1),(3,3)\})$.

$k \geqslant 1$, the eight leading indices are sufficient to identify the respective nodes, while for order 0 the first four indices are sufficient.[11] The symmetry group associated with this step is the identity, as the geometric transformations of the positions of the nodes do not depend on the indices assigned to them.

Step 3. (a) In this step, two graphene sheets $B_{k,1}$ are being combined to form the graphene sheet $B_{k,2}$. This step creates a symmetric structure (see Figures 16(g) and 16(o) and 18). The symmetry is described by the minor $X_1 = \Gamma_3(\{(1,2),(2,1),(3,3)\})$, while the use of the transfer graph $T_1 = D_3^1 \otimes (D_6 \setminus D_6^5) \cup \Gamma_3(\{(1,3)\}) \otimes D_6^5$ ensures that the border node set $\partial_5 B_{k,1}$ is transferred to the invariant branch set of the graph, identified by the invariant node 3 of the cyclical rotation graph R_1. Both the transfer graph T_1 and the minor X_1 are being applied for the construction of $B_{k,2}$ at all orders k, while the graph $B_{k,1}$ is of different dimension for each order k. The invariance of the transfer graph T_1 and the rotation graph R_1 across orders is a result of the invariance of the symmetry operations across orders, which itself is a necessary consequence of the self-similarity of the overall structure of SCNTs. The symmetry group associated with this step is the reflection group $\mathcal{S}_M(0, e_3)$.

(b) Again, two configurations of the preceding step, that is, $B_{k,2}$, are being combined to create the junction branch $B_{k,3}$, a cylindrical structure (see Figures 16(h) and 16(p)). This procedure involves the same minor as the previous step and a three-dimensional transfer graph which moves the invariant node set of the resulting configuration, that is, the border node sets $\{1,2\} \times \partial_3 B_{k,1}$, to the invariant branch set, identified by the leading index 3, of the resulting graph. As in Step 3(a), both the minor and the transfer graph are the same for all orders k. The symmetry group associated with this step is the reflection group $\mathcal{S}_M(0, e_2)$.

(c) The construction of a junction of order $k+1$ then concludes with a similar operation that combines the three branches of the junction into a rotationally symmetric configuration. (Note that there is no separate row in Figure 16 for this step, which transforms the result of Step 3(b) into the initial structure used for the construction of the subsequent order.) In addition to the rotation symmetry, the junction also exhibits reflection symmetries at the intersection of its branches. In order to capture all symmetries, the 4-dimensional transfer graph T_3 (see Figure 19) and the rotation graph $R_3 = \vec{C}_{(1,2,3)} \cup \vec{C}_{(4,5,6)} \cup D_7^7$ are being used, resulting in the junction graph

$$\begin{aligned} J_{k+1} &= \left(D_7^1 \otimes B_{k,3}\right) \diamond \left(T_3 c\left(R_3\right)\right) \\ &= c\left(R_3\right) T_3 \left(D_7^1 \otimes B_{k,3}\right) T_3{}^* c\left(R_3\right)^* \\ &= \bigcup_{j=1}^{3} \left(R_3{}^j T_3 \left(D_7^1 \otimes B_{k,3}\right) T_3{}^* R_3{}^{*j}\right), \end{aligned} \tag{16}$$

where $c(G)$ denotes the closure of a graph G. For the construction of the junction, the associated symmetry group is the rotation group $\mathcal{S}_{R,3}(0, e_3)$.

6. Implementation

6.1. Object-Oriented Code. The authors have implemented directed graphs and their functions, following the definitions of the graph algebra presented in this paper, as a set of MATLAB classes. The unary and binary operations for graphs

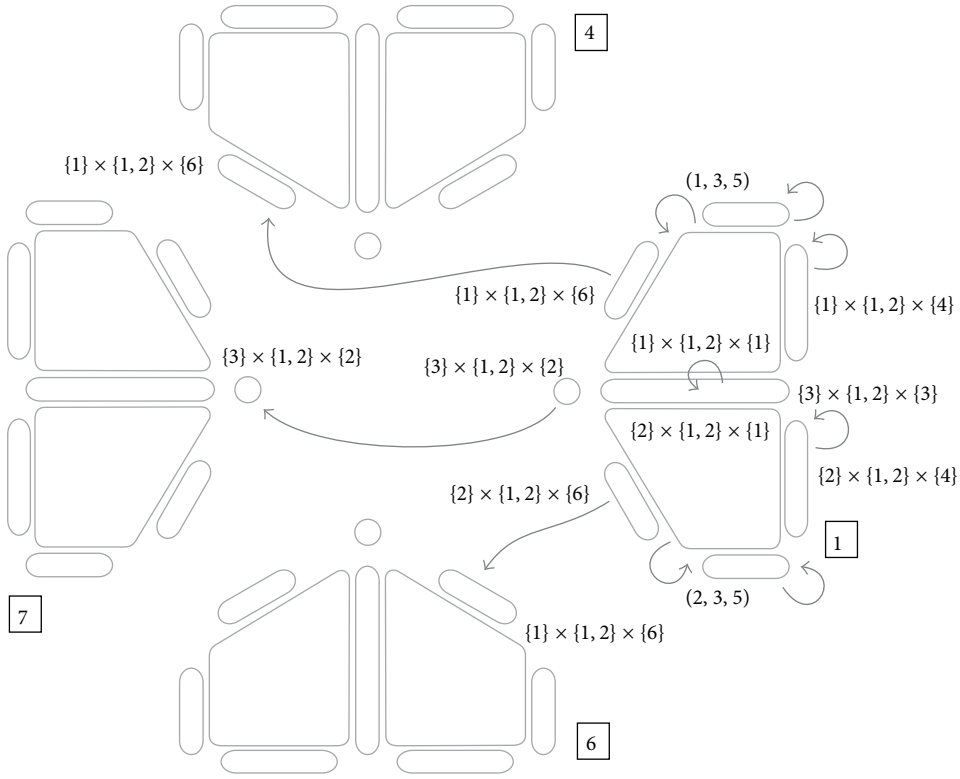


Figure 19: Transfer graph T_3. The branch sets with leading indices 2, 3, and 5 all consist of isolated nodes and are not shown in this figure.

are thus available as methods of a class of graphs. The code of a number of elementary functions associated with the basic graph-algebraic method has been optimized, resulting in a computationally efficient implementation.

Some additional functions, such as the creation of symmetrical graphs, have been implemented as high-level methods, allowing the user to focus on the characteristics of the structure rather than on identifying the relationships of indices. The methods implemented in the MATLAB classes do not presuppose any particular structure of the objects apart from those given by the graph algebra itself. In particular, no assumptions are made with regard to the sequence of elements in an array that stores the elements of a set.

The identification of each node by a tuple of indices, as opposed to a sequential numbering of the nodes, is essential for the efficient implementation of the graph algebra. In particular, the composition of graphs, which includes the identification of matching nodes in different graphs, is much faster if the comparison operation can be subdivided into comparison operations at each hierarchical level, that is, each position in the tuples identifying the nodes.

6.2. Scaling. The speed and scaling characteristics of the method depend on both the size of the structure and the complexity of its hierarchy and symmetry characteristics. As the matching of nodes in the composition of graphs will be the computationally most expensive part of the process in most cases, the construction of configurations with a low relative number of nodes that form invariant subsets with regard to the graph homomorphisms underlying the graph's symmetry characteristics will generally be faster than the construction of configurations that have a large relative number of such nodes.

Hierarchically symmetric graphs form a large class of very diverse items. In addition to differences in the number of nodes, arcs, or edges, graphs may consist of different numbers of hierarchy levels, and they may exhibit few symmetries of large subgraphs or a large number of symmetries between smaller subgraphs. A description of the different characteristics and their influence on the efficiency and scaling of the implementation is thus outside the scope of this paper. In addition, while the MATLAB-based implementation has been optimized to some degree, so that it will probably not constitute the bottleneck in any workflow including mesh construction and numerical analysis, the elementary algorithms can be further optimized, and the functions can be adapted to the specific processor and memory characteristics. Programming languages such as FORTRAN or C would bring more control over elementary steps in the process, resulting in further optimization possibilities.

Super carbon nanotube structures, due to their rather complex hierarchical symmetries, can be used as a reference case to illustrate the basic characteristics of the implementation of the HGM method. We recall the basic sequence in the construction of a SCNT structure (see also Figure 20):

(1) Start with the zero-dimensional graph.

(2) Repeat the following steps for each hierarchy order:

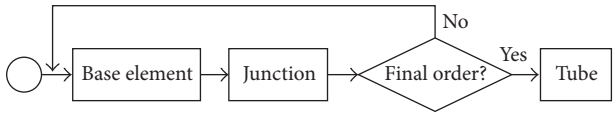


Figure 20: Sequence of the construction of a SCNT.

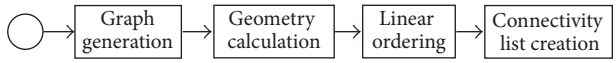


Figure 21: Sequence of the generation of a mesh with the HGM method.

(a) Construct a rectangular graphene (super-graphene) sheet from carbon atoms (CNT junctions) and cut out a trapezoid area, resulting in the *base element* of the respective hierarchy order, that is, the smallest element of the structure that does not have any global (graph-theoretic) symmetries.

(b) Construct a CNT (SCNT) junction by applying symmetry operations to the trapezoid part of the graphene (super-graphene) sheet.

(3) Construct a rectangular graphene (super-graphene) sheet from carbon atoms (CNT junctions) and roll it into a SCNT.

Of these steps, step (2)(b), that is, the construction of the junction, includes graph compositions, which tend to be the most computationally expensive operations. In steps (2)(a) and (3), which generally take less time relative to the size of the configuration at the respective step, the union operation is the computationally most expensive operation. The scaling of the algorithm is also impacted by the presence of an overhead which becomes dominant for smaller configurations and an increase in memory-related operations as information needs to be stored farther away from the processor in the case of larger configurations. Thus, for super carbon nanotubes, the speed of the construction, that is, the calculation time relative to the size of the structure, measured as the number of arcs, reaches a minimum for mid-sized structures.

The following results have been obtained on Xeon E5620 CPU at 2.40 GHz, with 48 GB RAM installed on a Supermicro X8-DA3 mainboard. Each data point reflects the median of the results of six runs.

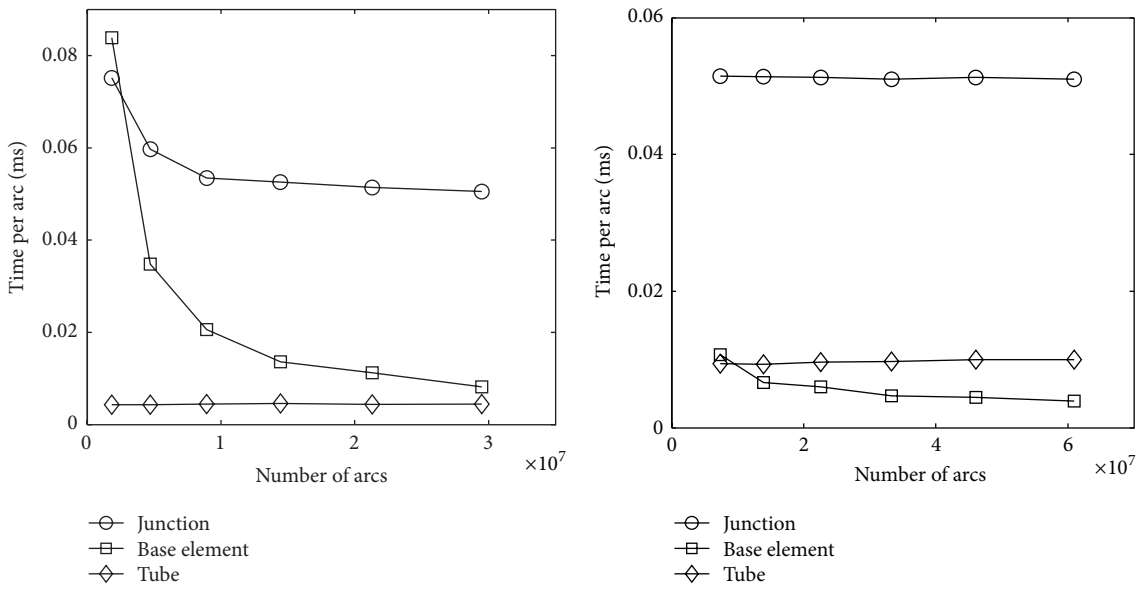
6.3. Mesh Generation. The graph generated by the HGM method contains all information that is needed to describe the structure. However, in order to use the data in commonly used solvers and finite element calculation programs, the hierarchical description must be converted into a format that such programs can process. In most cases, this involves the calculation of an explicit description of the structure's geometry, that is, the coordinates of the nodes, the linear ordering of the nodes, that is, the conversion of their index tuples to a strictly totally ordered set, usually identified with sequential numbers, and the creation of a connectivity list (in this case, up to neighbors of third degree). Figure 21 depicts the sequence of step involved in the creation of a mesh with the HGM method.

Figure 22 depicts the computation speed for the construction of SCNT structures of order 2, comprising a few hundreds of thousands of arcs and about 6 million arcs. The structures vary in the length of the inner tubes, resulting in different counts of arcs in the respective structures. The fall in computation time per node can be attributed to the decreasing impact of the fixed overhead associated with the graph-algebraic functions, as the number of arcs processed in each elementary operation becomes larger. For larger structures, this effect is compensated by other factors, such as matching operations that, in general, do not scale linearly. As the base elements are constructed first, the number of arcs, which enters into the denominator of the computational speed, is considerably lower in that step than in the construction of the junction. The creation of the tube includes, among other steps, the duplication of the junction elements. As a large number of arcs are processed simultaneously in computationally inexpensive elementary graph-algebraic operations, it is the least computationally expensive step in the process.

Figure 23 depicts the complete construction of the mesh for SCNT structures of order 2. In addition to the graph, that is, the node corridor and the set of arcs, the geometry and information needed for the conversion into a traditional mesh are being obtained from the graph.

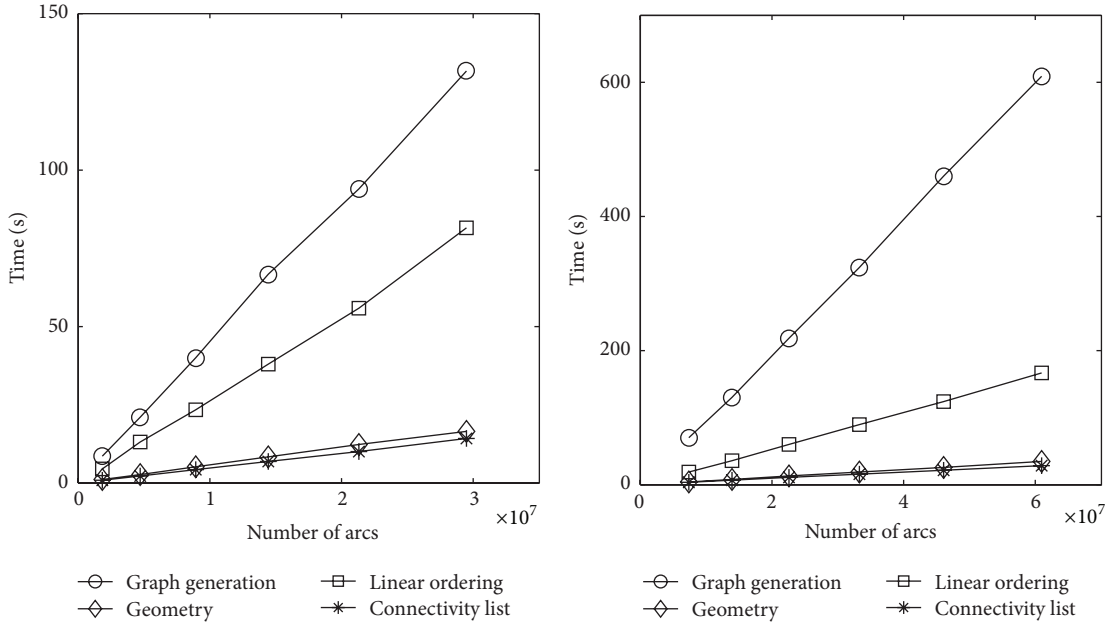
As illustrated in Figure 23, the different parts of the process all display an approximately linear dependence on the total time to the size of the problem, given by the total number of arcs. This indicates that the hierarchical graph method is applicable over a wide range of sizes and is able to process structures characterized by complex hierarchies and symmetries in an efficient way. Nevertheless, one needs to keep in mind that the linear scaling reported here is the result of a number of factors. The actual scaling characteristics of a particular structure will be determined by the extent to which parts of the structure can be replicated at different hierarchy levels and by the share of "border nodes," that is, the nodes that are situated on the border between two parts of the structure that have been glued together during the construction process.

To the knowledge of the authors, there are no benchmark structures for the construction of graphs that could be used for the purpose of comparisons between algorithms and implementations. The large variety of possible structures, which would include different sizes, differences in overall complexity, extent of the hierarchy, and symmetry characteristic, makes it unlikely that a single concept or algorithm would be more efficient than others in all possible cases. The Hierarchical Graph Meshing (HGM) method presented here is capable of exploiting various forms of regularities to the extent that they exist in a given structure, while nonregular structures, or nonregular parts of a structure, may still be described by the common single-number index system, which would constitute a special case in the context of the HGM method.



(a) Inner circumference: 6 hexagons (b) Inner circumference: 18 hexagons

FIGURE 22: Construction of a SCNT of order 2. The outer level circumference is set to six hexagons and the outer length is set to eight zigzag rings (a) or two zigzag rings (b). The inner circumference is also held constant within each series. The inner length varies between two and seven (a) and four to nine (b) zigzag rings, respectively, resulting in different numbers of arcs in the respective structures.



(a) Inner circumference: 6 hexagons (b) Inner circumference: 18 hexagons

FIGURE 23: Construction of the mesh of SCNTs of order 2. See the caption of Figure 22 for details on the characteristics (inner and outer circumferences and lengths) of the structures.

6.4. Computational Efficiency and Complexity: A Comparison. One particular advantage of the HGM method over other approaches, such as eliminating duplicate points, edges, or surfaces after creating a symmetric mesh by replicating the mesh of a smaller part of the structure, is the fact that all relevant information with regard to the connectivity of the structure is preserved at each step of the process. This obviates the need to use special methods that reconstruct such information when processing the resulting mesh. While such methods can be optimized by specific designs, such as divide-and-conquer algorithms, achieving good scaling characteristics is difficult without accessible knowledge about

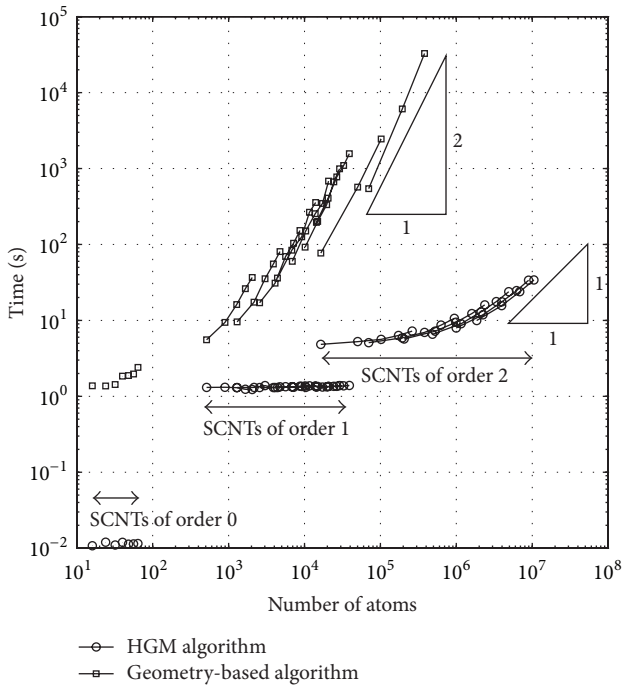


Figure 24: The HGM method (circles) in comparison with a geometry-based algorithm (squares). The triangles indicate the different scaling characteristics of both algorithms. Data connected by straight lines share the same order and circumference parameter; that is, they differ only with regard to their respective length parameters.

the structure, and better scaling often involves the use of complicated algorithms and data management features that in turn slow down the computation.

Concepts related to symmetry group operations have not been incorporated into general-purpose finite element programmes in a way that allows for an automatized processing of symmetric configurations [25, 26]. To the knowledge of the authors, there are no implementations of other algorithms that preserve complex regularities. Thus, there does not exist a benchmark-type implementation against which a quantitative comparison could be made.

In the process of the mechanical analysis of SCNT structures, the authors, prior to the development of the HGM method, have also created a program that treated the symmetry operations, as well as the operations related to the hierarchy characteristics, as separate from the actual creation of the mesh of the structure. Although a powerful neighborhood search algorithm has been employed, this program was far slower compared to the HGM method. Figure 24 shows the scaling characteristics of both algorithms for SCNTs up to order 2. We obtain a speed-up by a factor of about 5 for the smallest SCNTs of order 1, which increases to a factor of 100 and above for most SCNTs of order 2.

Due to overhead cost, the underlying scaling function for the HGM algorithm is visible only for the largest tubes. For these configurations, the HGM method achieves approximately linear scaling. This characteristic is a result of the fact that the number of atoms located on the border of symmetric elements of the configuration, at the respective hierarchical level, is increasing much slower than the total number of atoms. As most of the computation is related to these border nodes, a near linear scaling is obtained for large configurations.

For the geometry-based algorithm, the situation is more complex, as each of the parameters of a SCNT, that is, the hierarchical order, the circumference, and the length, has a different impact on the neighborhood search algorithm (i.e., the part of the program that reconstructs the information that is lost in the geometry-based algorithm while being preserved in the graph-based approach) and thus on the overall scaling characteristics. With regard to the volume V of the configuration, the neighborhood search algorithm has a scaling function $t(V) \in O(V^b \ln V)$, with $b \leqslant 1$. This is due to the fact that the neighborhood search algorithm is able to detect large parts of the overall volume which are completely empty and can be discarded from the computation. The share of these empty volume cells in the overall volume increases as the number of atoms of the structure grows. The length parameter l shows the largest difference between the number of atoms, scaling as $n(l) \in O(l)$, and the volume, scaling as $V(l) \in O(l^3)$. The large impact of the length parameter is indicated by the slope of the graphs connecting the results for SCNTs of the same order and circumference (but of different length parameter) in Figure 24. Thus, $V \circ n(l) \in O(l^3)$, and therefore $t(V \circ n(l)) \in O(l^{3b} \ln l)$. For the circumference factor c, we have $n(c) \in O(c)$ and $V(c) \in O(c^2)$, and thus $t(V \circ n(c)) \in O(c^{2b} \ln c)$. The hierarchical order p is also scaling with an exponent q, which has a value of about 2, resulting in $t(V \circ n(p)) \in O(p^{qb} \ln p)$. Therefore, if the number of atoms is increased by a combination of all parameters, the asymptotic complexity of the algorithm is improved relative to the situation resulting from an increase of the number of atoms solely as a result of an increase of the length parameter. For SCNTs of orders 1 and 2, the scaling exponent reaches a value between 1.5 and 2; that is, the asymptotic complexity of the geometry-based method is clearly superlinear.

Compared to a geometry-based neighborhood search algorithm, the HGM method offers a significant improvement in computational efficiency both with regard to the scaling characteristics and with regard to the absolute time spent for practically relevant computations. The linear scaling of the HGM algorithm is a significant advantage with regard to the construction of large hierarchically symmetric structures.

7. Conclusion

The Hierarchical Graph Meshing (HGM) method presented in this paper provides an efficient and accurate method for the systematic generation of meshes that represent hierarchical and/or symmetric structures. The resulting graph preserves the information about the hierarchy and symmetry characteristics of such structures. This information, in turn, can be used to apply multigrid approaches that exploit such information, for example, in mesh coarsening or domain decomposition methods.

Hierarchically symmetric graphs, due to their use of multi-indexed node labels, can describe structures exhibiting

hierarchy and/or symmetry characteristics in a way that is much more accessible than a linear numbering scheme, the standard approach in mesh generation. This allows identifying nodes with particular properties (e.g., nodes that are located on a symmetry axis of the structure) by their respective combinations of indices. Symmetries are reflected by well-defined graph automorphisms, and, for structures that exhibit geometric symmetries in addition to structural symmetries, such automorphisms correspond to well-defined geometric symmetry groups.

The construction algorithm is based on a concise set of elementary algebraic graph operations, which can easily be organized into a modular process. Each characteristic of a hierarchical or symmetric structure corresponds to a specific step in the algorithm. Therefore, any single characteristic may be changed by modifying the parameters or the operations contained in a specific module. The modular character of the HGM method also makes it conducive to automatization.

The HGM method has been implemented as a MATLAB code. Test runs of the implementation show that the method is computationally efficient and scales clearly better than a geometry-based method employing a neighborhood search algorithm. For super carbon nanotube structures, the code scales linearly with the problem size, that is, the number of nodes and edges contained in the structure, and the speed-up reaches a factor of 10^3 and above for super carbon nanotubes of order 2. Super carbon nanotubes can be regarded as a reference case for hierarchically symmetric structures. The algorithm produces hierarchically symmetric meshes for different types of super carbon nanotubes of arbitrary order in an efficient and easily traceable way.

The conversion from a multi-index graph-based mesh to a mesh based on a linear ordering of nodes is computationally inexpensive, allowing the integration of the HGM method with common mesh-based programs and workflows.

The distributive law of the graph algebra applied in the HGM method allows for the description of graphs as the disjoint union of smaller graphs, which in turn can be written as categorical products. In this way, it is possible to obtain a standardized notation of graphs. As the factorization of the subgraphs allows for the elimination of redundant data in the description of the graph, the graph algebra employed in the HGM method can also be used to significantly reduce memory requirements. The details of this approach, however, must be left for further research at this point.

Appendix

A. Proofs

With $V \subseteq \pi(\mathfrak{V}^{(n)})$, the inverse images of the domain and range functions are given by $d^{-1}(V, A) := \{a \in A \mid d(a) \in V\}$ and $r^{-1}(V, A) := \{a \in A \mid r(a) \in V\}$. As a shorthand notation, we write $d_k^{-1}(A)$ for $d^{-1}(\{v_k\}, A)$ and $r_k^{-1}(A)$ for $r^{-1}(\{v_k\}, A)$ as well as $\alpha(V, A)$ or α_A^V for $d(r^{-1}(V, A))$ and $\omega(V, A)$ or ω_A^V for $r(d^{-1}(V, A))$. The symbol "$\dot{\bigcup}$" denotes the union of disjunct sets.

A.1. Distributivity of the Algebra of Directed Graphs

Proof. Let $G_1 = \Gamma_{\mathscr{W}_1}(\mathscr{A}_1) \in \mathfrak{G}^{(n)}$, $G_2 = \Gamma_{\mathscr{W}_2}(\mathscr{A}_2) \in \mathfrak{G}^{(n)}$, $G_3 = \Gamma_{\mathscr{W}_3}(\mathscr{A}_3) \in \mathfrak{G}^{(n)}$, and $\dot{\bigcup}_{k=1}^{n}\{v_k\} = (d(\mathscr{A}_1) \cup d(\mathscr{A}_2)) \cap r(\mathscr{A}_3)$.

The proposition $(G_1 \cup G_2)G_3 = G_1G_3 \cup G_2G_3$ can be proven as follows: for nonempty node corridors $\mathscr{W}_1$, $\mathscr{W}_2$, and $\mathscr{W}_3$,

$$\begin{aligned}(\mathscr{W}_1 \cup \mathscr{W}_2) \circ \mathscr{W}_3 &= \{(\mathscr{S}_1 \cup \mathscr{S}_2, \mathscr{T}_1 \cup \mathscr{T}_2)\} \\ &\quad \circ \{(\mathscr{S}_3, \mathscr{T}_3)\} \\ &= \{(\mathscr{S}_3, \mathscr{T}_1 \cup \mathscr{T}_2)\} \\ &= \{(\mathscr{S}_3, \mathscr{T}_1)\} \cup \{(\mathscr{S}_3, \mathscr{T}_2)\} \\ &= \mathscr{W}_1 \circ \mathscr{W}_3 \cup \mathscr{W}_2 \circ \mathscr{W}_3.\end{aligned} \tag{A.1}$$

For an empty node corridor $\mathscr{W}_1$, $(\mathscr{W}_1 \cup \mathscr{W}_2)\mathscr{W}_3 = \mathscr{W}_2\mathscr{W}_3 = \{\} \cup \mathscr{W}_2\mathscr{W}_3 = \mathscr{W}_1\mathscr{W}_3 \cup \mathscr{W}_2\mathscr{W}_3$. By the same reasoning, the proposition is true for $\mathscr{W}_2 = \{\}$. For $\mathscr{W}_3 = \{\}$, we have $(\mathscr{W}_1 \cup \mathscr{W}_2)\mathscr{W}_3 = (\mathscr{W}_1 \cup \mathscr{W}_2)\{\} = \{\} = \mathscr{W}_1 \circ \{\} \cup \mathscr{W}_2 \circ \{\} = \mathscr{W}_1\mathscr{W}_3 \cup \mathscr{W}_2\mathscr{W}_3$. Consider

$$\begin{aligned}&(\mathscr{A}_1 \cup \mathscr{A}_2) \circ \mathscr{A}_3 \\ &\quad = \dot{\bigcup}_{k=1}^{n} \left(r_k^{-1}(\mathscr{A}_3) \circ \left(d_k^{-1}(\mathscr{A}_1) \cup d_k^{-1}(\mathscr{A}_2)\right)\right) \\ &\quad = \dot{\bigcup}_{k=1}^{n} \left(\alpha_{\mathscr{A}_3}^{\{v_k\}} \times \left(\omega_{\mathscr{A}_1}^{\{v_k\}} \cup \omega_{\mathscr{A}_2}^{\{v_k\}}\right)\right) \\ &\quad = \dot{\bigcup}_{k=1}^{n} \left(\alpha_{\mathscr{A}_3}^{\{v_k\}} \times \omega_{\mathscr{A}_1}^{\{v_k\}} \cup \alpha_{\mathscr{A}_3}^{\{v_k\}} \times \omega_{\mathscr{A}_2}^{\{v_k\}}\right) \\ &\quad = \mathscr{A}_1 \circ \mathscr{A}_3 \cup \mathscr{A}_2 \circ \mathscr{A}_3,\end{aligned} \tag{A.2}$$

$$\begin{aligned}&(G_1 \cup G_2)\, G_3 \\ &\quad = \Gamma\left((\mathscr{W}_1 \cup \mathscr{W}_2) \circ \mathscr{W}_3, (\mathscr{A}_1 \cup \mathscr{A}_2) \circ \mathscr{A}_3\right) \\ &\quad = \Gamma\left(\mathscr{W}_1 \circ \mathscr{W}_3 \cup \mathscr{W}_2 \circ \mathscr{W}_3, \mathscr{A}_1 \circ \mathscr{A}_3 \cup \mathscr{A}_2 \circ \mathscr{A}_3\right) \\ &\quad = G_1G_3 \cup G_2G_3.\end{aligned}$$

The proposition $G_3 \circ (G_1 \cup G_2) = G_3G_1 \cup G_3G_2$ can be proven in a similar way. □

A.2. Proofs Related to the Categorical Product of Graphs

Proof. Let $G_1 = \Gamma(\mathscr{W}_1, \mathscr{A}_1) \in \mathfrak{G}^{(m)}$, $G_2 = \Gamma(\mathscr{W}_2, \mathscr{A}_2) \in \mathfrak{G}^{(n)}$, $G_3 = \Gamma(\mathscr{W}_3, \mathscr{A}_3) \in \mathfrak{G}^{(m)}$, $G_4 = \Gamma(\mathscr{W}_4, \mathscr{A}_4) \in \mathfrak{G}^{(n)}$, $\dot{\bigcup}_{k=1}^{n}\{v_k\} = r(\mathscr{A}_1) \cap d(\mathscr{A}_3)$, and $\dot{\bigcup}_{k'=1}^{n'}\{v'_{k'}\} = r(\mathscr{A}_2) \cap d(\mathscr{A}_4)$.

The proposition $(G_1 \otimes G_2)(G_3 \otimes G_4) = G_1G_3 \otimes G_2G_4$ can be proven as follows: for nonempty node corridors,

$$\begin{aligned}&(\mathscr{W}_1 \otimes \mathscr{W}_2) \circ (\mathscr{W}_3 \otimes \mathscr{W}_4) \\ &\quad = \{((\mathscr{S}_1, \mathscr{S}_2), (\mathscr{T}_1, \mathscr{T}_2))\} \\ &\qquad \circ \{((\mathscr{S}_3, \mathscr{S}_4), (\mathscr{T}_3, \mathscr{T}_4))\}\end{aligned}$$

$$= \{((\mathcal{S}_3, \mathcal{S}_4), (\mathcal{T}_1, \mathcal{T}_2))\}$$
$$= \{(\mathcal{S}_3, \mathcal{T}_1)\} \otimes \{(\mathcal{S}_4, \mathcal{T}_2)\}$$
$$= (\mathcal{W}_1 \circ \mathcal{W}_3) \otimes (\mathcal{W}_2 \circ \mathcal{W}_4), \tag{A.3}$$

while both sides of the equation given in the proposition result in the empty set, if any of the node corridors is empty. Consider

$$\begin{aligned}
&(\mathcal{A}_1 \otimes \mathcal{A}_2) \circ (\mathcal{A}_3 \otimes \mathcal{A}_4) \\
&= \dot{\bigcup}_{k=1}^{n} \dot{\bigcup}_{k'=1}^{n'} \left(\alpha_{\mathcal{A}_3 \otimes \mathcal{A}_4}^{\{(v_k, v'_{k'})\}} \times \omega_{\mathcal{A}_1 \otimes \mathcal{A}_2}^{\{(v_k, v'_{k'})\}} \right) \\
&= \dot{\bigcup}_{k=1}^{n} \dot{\bigcup}_{k'=1}^{n'} \left(\left(\alpha_{\mathcal{A}_3}^{\{v_k\}} \times \alpha_{\mathcal{A}_4}^{\{v'_{k'}\}} \right) \times \left(\omega_{\mathcal{A}_1}^{\{v_k\}} \times \omega_{\mathcal{A}_2}^{\{v'_{k'}\}} \right) \right) \\
&= \dot{\bigcup}_{k=1}^{n} \dot{\bigcup}_{k'=1}^{n'} \left(\left(\alpha_{\mathcal{A}_3}^{\{v_k\}} \times \omega_{\mathcal{A}_1}^{\{v_k\}} \right) \otimes \left(\alpha_{\mathcal{A}_4}^{\{v'_{k'}\}} \times \omega_{\mathcal{A}_2}^{\{v'_{k'}\}} \right) \right) \\
&= (\mathcal{A}_1 \circ \mathcal{A}_2) \otimes (\mathcal{A}_3 \circ \mathcal{A}_4),
\end{aligned} \tag{A.4}$$

$$\begin{aligned}
&(G_1 \otimes G_2)(G_3 \otimes G_4) = \Gamma((\mathcal{W}_1 \otimes \mathcal{W}_2) \circ (\mathcal{W}_3 \otimes \mathcal{W}_4), \\
&(\mathcal{A}_1 \otimes \mathcal{A}_2) \circ (\mathcal{A}_3 \otimes \mathcal{A}_4)) = \Gamma((\mathcal{W}_1 \circ \mathcal{W}_3) \otimes (\mathcal{W}_2 \\
&\circ \mathcal{W}_4), (\mathcal{A}_1 \circ \mathcal{A}_2) \otimes (\mathcal{A}_3 \circ \mathcal{A}_4)) = G_1 G_2 \otimes G_3 G_4.
\end{aligned}$$

□

Proof. The proposition $G_1 \otimes G_2 \cap G_3 \otimes G_4 = (G_1 \cap G_3) \otimes (G_2 \cap G_4)$ can be proven as follows: for nonempty node corridors,

$$\begin{aligned}
&\mathcal{W}_1 \otimes \mathcal{W}_2 \cap \mathcal{W}_3 \otimes \mathcal{W}_4 \\
&= \{((\mathcal{S}_1, \mathcal{S}_2), (\mathcal{T}_1, \mathcal{T}_2))\} \\
&\quad \cap \{((\mathcal{S}_3, \mathcal{S}_4), (\mathcal{T}_3, \mathcal{T}_4))\} \\
&= \{((\mathcal{S}_1 \cap \mathcal{S}_3, \mathcal{S}_2 \cap \mathcal{S}_4), (\mathcal{T}_1 \cap \mathcal{T}_3, \mathcal{T}_2 \cap \mathcal{T}_4))\} \\
&= \{((\mathcal{S}_1 \cap \mathcal{S}_3), (\mathcal{T}_1 \cap \mathcal{T}_3))\} \\
&\quad \otimes \{((\mathcal{S}_2 \cap \mathcal{S}_4), (\mathcal{T}_2 \sqcap \mathcal{T}_4))\} \\
&= \{((\mathcal{S}_1, \mathcal{S}_3) \cap (\mathcal{T}_1, \mathcal{T}_3))\} \\
&\quad \otimes \{((\mathcal{S}_2, \mathcal{S}_4) \cap (\mathcal{T}_2, \mathcal{T}_4))\} \\
&= (\mathcal{W}_1 \cap \mathcal{W}_3) \otimes (\mathcal{W}_2 \cap \mathcal{W}_4),
\end{aligned} \tag{A.5}$$

while both sides of the equation given in the proposition result in the empty set, if any of the node corridors is empty. Consider

$$\begin{aligned}
&\mathcal{A}_1 \otimes \mathcal{A}_2 \cap \mathcal{A}_3 \otimes \mathcal{A}_4 \\
&\quad = \{((d(a_1), d(a_2)), (r(a_1), r(a_2))) \mid (a_1, a_2) \\
&\quad \in \mathcal{A}_1 \times \mathcal{A}_2\} \\
&\quad \cap \{((d(a_3), d(a_4)), (r(a_3), r(a_4))) \mid (a_3, a_4) \\
&\quad \in \mathcal{A}_3 \times \mathcal{A}_4\} \\
&\quad = \Bigg\{ (d(\{a_1\} \cap \{a_3\}) \times d(\{a_2\} \cap \{a_4\})) \\
&\quad \times (r(\{a_1\} \cap \{a_3\}) \times r(\{a_2\} \cap \{a_4\})) \mid (a_k)_{k=1}^{4} \\
&\quad \in \prod_{k=1}^{4} \mathcal{A}_k \Bigg\} = (\mathcal{A}_1 \cap \mathcal{A}_3) \otimes (\mathcal{A}_2 \cap \mathcal{A}_4).
\end{aligned} \tag{A.6}$$

Thus,

$$\begin{aligned}
&G_1 \otimes G_2 \cap G_3 \otimes G_4 = \Gamma(\mathcal{W}_1 \otimes \mathcal{W}_2 \cap \mathcal{W}_3 \otimes \mathcal{W}_4, \mathcal{A}_1 \\
&\otimes \mathcal{A}_2 \cap \mathcal{A}_3 \otimes \mathcal{A}_4) = \Gamma((\mathcal{W}_1 \cap \mathcal{W}_3) \\
&\otimes (\mathcal{W}_2 \cap \mathcal{W}_4), (\mathcal{A}_1 \cap \mathcal{A}_3) \otimes (\mathcal{A}_2 \cap \mathcal{A}_4)) = (G_1 \\
&\cap G_3) \otimes (G_2 \cap G_4).
\end{aligned} \tag{A.7}$$

□

Conflict of Interests

The authors declare that there is no conflict of interests regarding the publication of this paper.

Acknowledgment

Financial support for this research was provided by the Deutsche Forschungsgemeinschaft (DFG) via the Emmy Noether Program under Grant no. Wa 2516/3-1. This support is gratefully acknowledged.

Endnotes

1. In contrast to [27, 39, 44] and others, however, the graph algebra used in the following exposition does not include an incidence function that maps arcs or edges to pairs of vertices. As a result, while vertices and arcs may be indexed, such an index system does not induce any particular ordering on the vertices or on the arcs. As a result, all graph automorphisms that, in the incidence-based graph structure, would simply change the ordering of nodes or edges are indistinguishable from the identity transformation, and any automorphism of a graph based on this structure can be identified with a node permutation. This also implies that the description of graph symmetries must refer to the node permutation itself, as the automorphism that describes a symmetry operation would itself always constitute an identity transformation.
2. The common definition of directed graphs assumes that nodes are elements of a one-dimensional node space; that is, they are not represented by tuples, and it also assumes that the set of source nodes is equal to the set of target nodes; that is, $\mathcal{S} = \mathcal{T} = \widetilde{\mathcal{V}}$. Following

this definition, a simple directed graph is given as $G \cong (\overline{\mathscr{V}}, \mathscr{A})$, with $\mathscr{A} \subseteq \overline{\mathscr{V}} \times \overline{\mathscr{V}}$ [39]. With the isomorphism $(\mathrm{cart}(\mathscr{V}), \mathscr{A}) \cong (\{(\mathscr{V}, \mathscr{V})\}, \mathscr{A})$, we can identify simple directed graphs defined in this way with a subset of $\mathfrak{G}^{(1)}$.

3. For the respective proofs, see Appendix A.1.
4. In general, however, the composition of a graph and its opposite, GG^*, is not the neutral element with regard to the composition of graphs.
5. For the respective proofs, see Appendix A.2.
6. We use the symbol σ for both isomorphisms in general and for the special case of automorphisms, including permutations.
7. The notation σ^n denotes the n-fold consecutive application of the function σ. Symmetric graphs are also called transitive graphs [39, p. 157].
8. In addition to the geometry, a connectivity list can be created from the graph by composing it with itself, as the composition G^n contains the arcs of a graph G that connect neighbors of nth degree. The case of more complicated finite elements is beyond the scope of this paper.
9. For translational symmetries, the isomorphism must exist between $\mathfrak{B}_k$ and a finite symmetry group on a cyclic space or a finite subset $\{s^j\}_{j=0}^{m-1}$, $s \in S_k$, of an (infinite) translation symmetry group on a noncyclic vector space.
10. The construction of a tube is quite similar to the construction of a base element and does not introduce any new elements to the construction method. Therefore, the elementary steps of this part of the construction are not described here.
11. Note that $D_k{}^* = D_k$. This operation may be implemented either by defining infinite graphs, that is, graphs with an infinite number of nodes, or by defining a "selection" operation that eliminates the nodes and arcs outside of the trapezoidal area from the graph $S_{k,4}$.

References

[1] P. J. Frey and P. L. George, *Mesh Generation: Application to Finite Elements*, John Wiley & Sons, Hoboken, NJ, USA, 2nd edition, 2008.

[2] S. Iijima, "Helical microtubules of graphitic carbon," *Nature*, vol. 354, no. 6348, pp. 56–58, 1991.

[3] R. de Borst and E. Ramm, *Multiscale Methods in Computational Mechanics*, Springer, 2011.

[4] W. Hackbusch, *Multi-Grid Methods and Applications*, Springer, Berlin, Germany, 2nd edition, 2003.

[5] T. I. Zohdi and P. Wriggers, *An Introduction to Computational Micromechanics*, Springer, Berlin, Germany, 2nd edition, 2008.

[6] V. Gravemeier, S. Lenz, and W. A. Wall, "Towards a taxonomy for multiscale methods in computational mechanics: building blocks of existing methods," *Computational Mechanics*, vol. 41, no. 2, pp. 279–291, 2008.

[7] J. W. Ruge and K. Stüben, "Algebraic multigrid," in *Multigrid Methods*, S. F. McCormick, Ed., pp. 73–130, SIAM, Philadelphia, Pa, USA, 1987.

[8] A. J. Cleary, R. D. Falgout, H. van Emden, and J. E. Jones, "Coarse-grid selection for parallel Algebraic Multigrid," in *Proceedings of the 5th International Symposium on Solving Irregularly Structured Problems in Parallel*, Lawrence Berkeley National Laboratory, Ed., vol. 1457 of *Lecture Notes in Computer Science*, pp. 104–115, Springer, 1998.

[9] H. de Sterck and U. M. Yang, "Coarsening and interpolation in algebraic multigrid: a balancing act," in *Proceedings of the 12th Copper Mountain Conference on Multigrid Methods*, April 2005.

[10] R. E. Bank and C. Wagner, "Multilevel ILU decomposition," *Numerische Mathematik*, vol. 82, no. 4, pp. 543–576, 1999.

[11] A. J. Cleary, R. D. Falgout, V. E. Henson et al., "Robustness and scalability of algebraic multigrid," *SIAM Journal on Scientific Computing*, vol. 21, no. 5, pp. 1886–1908, 2000.

[12] V. E. Henson and U. M. Yang, "BoomerAMG: a parallel algebraic multigrid solver and preconditioner," *Applied Numerical Mathematics*, vol. 41, no. 1, pp. 155–177, 2002.

[13] J. S. Butler, *Improving coarsening and interpolation for algebraic multigrid [M.S. thesis]*, University of Waterloo, Waterloo, Canada, 2006.

[14] C. G. Bayreuther, *Mehrskalenmodelle in der Festkörpermechanik und Kopplung von Mehrgittermethoden mit Homogenisierungsverfahren [Ph.D. thesis]*, University of Stuttgart, Stuttgart, Germany, 2005.

[15] T. J. R. Hughes, G. R. Feijóo, L. Mazzei, and J.-B. Quincy, "The variational multiscale method—a paradigm for computational mechanics," *Computer Methods in Applied Mechanics and Engineering*, vol. 166, no. 1-2, pp. 3–24, 1998.

[16] E. Weinan, B. Engquist, X. Li, W. Ren, and E. Vanden-Eijnden, "The heterogeneous multiscale method: a review," *Communications in Computational Physics*, vol. 2, no. 3, pp. 367–450, 2007.

[17] C. Miehe and C. G. Bayreuther, "Multilevel FEM for heterogeneous structures: from homogenization to multigrid solvers," in *Multifield Problems in Solid and Fluid Mechanics*, R. Helmig, A. Mielke, and B. I. Wohlmuth, Eds., vol. 28 of *Lecture Notes in Applied and Computational Mechanics*, pp. 361–397, Springer, Berlin, Germany, 2006.

[18] A. Brandt, "Algebraic multigrid theory: the symmetric case," *Applied Mathematics and Computation*, vol. 19, no. 1–4, pp. 23–56, 1986.

[19] K. Stüben, "Algebraic multigrid (AMG): experiences and comparisons," *Applied Mathematics and Computation*, vol. 13, no. 3-4, pp. 419–451, 1983.

[20] R. D. Kangwai, S. D. Guest, and S. Pellegrino, "An introduction to the analysis of symmetric structures," *Computers and Structures*, vol. 71, no. 6, pp. 671–688, 1999.

[21] A. Kaveh and M. Nikbakht, "Block diagonalization of Laplacian matrices of symmetric graphs via group theory," *International Journal for Numerical Methods in Engineering*, vol. 69, no. 5, pp. 908–947, 2007.

[22] K. Koohestani, "Exploitation of symmetry in graphs with applications to finite and boundary elements analysis," *International Journal for Numerical Methods in Engineering*, vol. 90, no. 2, pp. 152–176, 2012.

[23] A. Kaveh and H. Rahami, "An efficient analysis of repetitive structures generated by graph products," *International Journal for Numerical Methods in Engineering*, vol. 84, no. 1, pp. 108–126, 2010.

[24] A. Zingoni, "Group-theoretic exploitations of symmetry in computational solid and structural mechanics," *International Journal for Numerical Methods in Engineering*, vol. 79, no. 3, pp. 253–289, 2009.

[25] K. Suresh and A. Sirpotdar, "Automated symmetry exploitation in engineering analysis," *Engineering with Computers*, vol. 21, no. 4, pp. 304–311, 2006.

[26] J. C. Wohlever, "Some computational aspects of a group theoretic finite element approach to the buckling and postbuckling analyses of plates and shells-of-revolution," *Computer Methods in Applied Mechanics and Engineering*, vol. 170, no. 3-4, pp. 373–406, 1999.

[27] A. Kaveh and K. Koohestani, "Graph products for configuration processing of space structures," *Computers and Structures*, vol. 86, no. 11-12, pp. 1219–1231, 2008.

[28] V. R. Coluci, D. S. Galvao, and A. Jorio, "Geometric and electronic structure of carbon nanotube networks: 'super'-carbon nanotubes," *Nanotechnology*, vol. 17, no. 3, pp. 617–621, 2006.

[29] Y. Li, X. Qiu, F. Yang, X.-S. Wang, Y. Yin, and Q. Fan, "A comprehensive study on the mechanical properties of super carbon nanotubes," *Journal of Physics D: Applied Physics*, vol. 41, no. 15, Article ID 155423, 2008.

[30] B. Liu, Y. Huang, H. Jiang, S. Qu, and K.-C. Hwang, "The atomic-scale finite element method," *Computer Methods in Applied Mechanics and Engineering*, vol. 193, no. 17–20, pp. 1849–1864, 2004.

[31] J. Wackerfuß, "Molecular mechanics in the context of the finite element method," *International Journal for Numerical Methods in Engineering*, vol. 77, no. 7, pp. 969–997, 2009.

[32] M. Wang, X. Qiu, X. Zhang, and Y. Yin, "Equivalent parameter study of the mechanical properties of super carbon nanotubes," *Nanotechnology*, vol. 18, no. 29, Article ID 295708, 2007.

[33] M. V. Diudea, B. Parv, and E. C. Kirby, "Azulenic tori," *MATCH Communications in Mathematical and in Computer Chemistry*, no. 47, pp. 53–70, 2003.

[34] S. M. Ferrer, N. V. Khokhriakov, and S. S. Savinskii, "Geometry of multi-tube carbon clusters and electronic transmission in nanotube contacts," *Molecular Engineering*, vol. 8, no. 4, pp. 315–344, 1999.

[35] I. László, "Topological description and construction of single wall carbon nanotube junctions," *Croatica Chemica Acta*, vol. 78, no. 2, pp. 217–221, 2005.

[36] R. Diestel, *Graph Theory*, vol. 173 of *Graduate Texts in Mathematics*, Springer, New York, NY, USA, 4th edition, 2010.

[37] R. Diestel, *Graphentheorie*, Springer, Heidelberg, Germany, 4th edition, 2010.

[38] C. Godsil and G. Royle, *Algebraic Graph Theory*, Springer, Berlin, Germany, 2001.

[39] U. Knauer, *Algebraic Graph Theory: Morphisms, Monoids, and Matrices*, De Gruyter, Berlin, Germany, 2011.

[40] J. A. Bondy and U. S. R. Murty, *Graph Theory*, Springer, London, UK, 2008.

[41] G. Hahn and C. Tardif, "Graph homomorphisms: structure and symmetry," in *Graph Symmetry*, G. Hahn and G. Sabidussi, Eds., pp. 107–166, Kluwer Academic Publishers, Dordrecht, The Netherlands, 1997.

[42] R. Hammack, W. Imrich, and S. Klavžar, *Handbook of Product Graphs*, Discrete Mathematics and Its Applications, CRC Press, Boca Raton, Fla, USA, 2nd edition, 2011.

[43] D. J. Miller, "The categorical product of graphs," *Canadian Journal of Mathematics*, vol. 20, pp. 1511–1521, 1968.

[44] J. A. Bondy and U. S. R. Murty, *Graph Theory with Applications*, Macmillan, London, UK, 1976.