Consistency grammar for 3D indoor model checking

Shayan Nikoohemat\textsuperscript{1} | Abdoulaye A. Diakité\textsuperscript{2} | Ville Lehtola\textsuperscript{1} | Sisi Zlatanova\textsuperscript{2} | George Vosselman\textsuperscript{1}

\textsuperscript{1}Department of Earth Observation Science, Faculty ITC, University of Twente, Enschede, The Netherlands
\textsuperscript{2}Department of Built Environment, University of New South Wales, Sydney, Australia

Correspondence
Shayan Nikoohemat, Department of Earth Observation Science, Faculty ITC, University of Twente, Enschede, The Netherlands.
Email: s.nikoohemat@utwente.nl

Funding information
Netherlands Organization for Scientific Research (NWO), Grant/Award Number: 13742

Abstract
Indoor 3D models are digital representations of building interiors reconstructed from scanned data acquired by laser scanners, digital depth (RGBD) cameras, and CAD drawings. Consequently, there is noise in the source data and a notable variety in the methods used to treat the noise and to process these data into reconstructed models. Alas, the correctness of these reconstructions and thus their suitability for a given application are uncertain. There is a lack of a robust base logic that would allow for controlling the consistency of these (automatically) generated models. Fortunately, correctness criteria are well-defined through existing international standards. Hence, we propose a conceptual framework based on formal grammars to check the semantic, geometric, and topological consistency of a reconstructed 3D model. The proposed method proceeds in three steps to validate the model: (1) correctness checking of individual components; (2) consistency verification of instances’ interactions; and (3) model consistency check for targeted applications. Our method identifies the components in the model that violate the given rules derived from the current standards and expert knowledge. Ultimately, we propose a quantified formulation of our method that may be straightforwardly integrated into industrial-level model checkers. The approach is independent of level of details and reconstruction method.

This is an open access article under the terms of the Creative Commons Attribution NonCommercial License, which permits use, distribution and reproduction in any medium, provided the original work is properly cited and is not used for commercial purposes.
© 2020 The Authors. Transactions in GIS published by John Wiley & Sons Ltd
In recent years there has been impressive progress in indoor data collection technologies thanks to advanced remote sensing devices. Such sensor systems that provide high-quality images, point cloud and depth information enable us to reconstruct and update indoor 3D models through semi-automatic and automatic methods. These models are digital representations of the buildings' interior and are useful for architects and engineers in planning and construction. Indoor 3D models are reconstructed from different sources such as floor plans, images, LiDAR and RGB-depth data using automatic approaches. Exploiting multiple sources and automatic approaches makes such models inconsistent in terms of current standards. Although existing standards and data models such as industrial foundation classes (IFC), ISO 16739 (ISO, 2018) and ISO 19107, CityGML and IndoorGML (Gröger, Kolbe, Nagel, & Häfele, 2012; ISO, 2019; Lee et al., 2016) provide a suitable guideline to generate 3D models, the complexity of the indoor environment and noise in the scanned data obstruct the creation of a consistent 3D model in an automatic workflow. The question is, how we can have a systemic way to check the consistency of our model without the presence of ground truth and with the minimal interaction of humans.

The generated model should comply with standards of current indoor models (IndoorGML, CityGML, IFC) considering the geometry, topology, and data models of entities to provide a reliable platform for applications such as evacuation and safety management in large buildings. Obviously, some of the consistency checks could be inherited from standards, but sometimes the generated 3D models do not follow any specific standard yet they are giving a thorough overview of the building and interiors, such as using grammar (Ikehata, Yang, & Furukawa, 2015) or constructive solid geometry (CSG, Xiao & Furukawa, 2014). However, a correct and consistent indoor 3D model can be used for many applications. Indoor 3D models can differ regarding the applications or standards which they comply with; for example, for navigation applications and considering IndoorGML standards, reconstructing indoor spaces and doors should suffice and reconstruction of walls is not necessary. In contrast, for applications aiming to reconstruct buildings according to the BIM standard (IFC), walls are important components of the resulting 3D model and should therefore be reconstructed in volumetric shapes (Bassier, Klein, Van Genechten, & Vergauwen, 2018; buildingSMART International, 2013; Ochmann, Vock & Klein, 2019) to meet the IfcWall properties. Similarly, when considering the energy consumption of buildings for heating purposes or thermal comfort-related applications, a model with the presence of windows and air outlets is necessary (Kim & Anderson, 2013).

A 3D model can have different levels of complexity regarding the application, the reconstruction method, and the input data. Using a 3D model for 3D GIS or BIM applications can lead to different representations of the geometry primitives. Integration of models between CAD and GIS has been studied in the literature (de Laat & van Berlo, 2011; Pu & Zlatanova, 2006). Similarly, standards (e.g., IFC, CityGML, IndoorGML) provide a guideline on how a 3D model should be in terms of geometry and semantics for more specific applications such as construction or navigation. However, it is not trivial to check the consistency of a complex 3D model against a specific standard, because all the components should be present in the model otherwise the comparison is rejected in the first place. For example, a 3D model which represents walls as surfaces and does not provide spaces is not compatible with both IFC and IndoorGML, yet could be a consistent model for specific applications. Therefore, we need a system which checks 3D models against some common knowledge present in the building language as well as existing standards. Such a system should not be sensitive to the type of geometry and should be scalable and adaptable to different applications. The tool that we choose for this purpose is a type of formal grammar called consistency grammar, where the components of a 3D model compose the symbols in the grammar and the constraint rules control the consistency of the model.

Previous works verify a model by comparing it against ground truth, either with a point-to-point comparison (Lehtola et al., 2017) or a plane-to-plane comparison (Tran, Khoshelham, & Kealy, 2019a). The consistency of an indoor 3D model consists of completeness and correctness and should be checked in terms of topology, geometry, and semantics when there is no ground truth. While completeness of the model is dependent on external information, for example knowing the correct number of rooms, the correctness of components can be controlled individually and in relation with each other. From a mathematical point of view, geometries and their topology can
be validated. Staircases and floors, for instance, are disjoint or connected. Each floor is covered by rooms and each room has at least one door. For example, if the accessibility of rooms and interiors to an (emergency) exit is defective because of topological errors in the model, then the indoor navigation for evacuation fails.

In this article, we introduce a consistency grammar which allows one to use different geometric representation and source of data for 3D model reconstruction and yet be able to identify, formalize, and verify the consistency of the model. Our main contribution is the introduction of a three-stage consistency checking workflow, empowered by grammar rules and constraints, that facilitate the automation of reconstructed 3D model validation. The first step consists of checking all the individual components of the 3D model, the second step checks the interactions between those components, and the third step checks application-specific constraints. The method is highly adaptable to any domain of expertise that may need a 3D building model, as the constraints can be customized to fit specific domain expectations. We provide examples of use of the approach considering boundary representations (B-Rep), geometry type, and indoor navigation application.

Although we focus on B-Rep, all types of explicit geometric representation can be handled by our approach (e.g., B-splines, octree, voxel, etc.). Indeed, all the introduced conceptual framework can still be used as long as the proper adaptations of the grammar’s constraints are provided. However, implicit geometric representations such as CSGs or sweep geometry cannot be the direct input to our approach, as their primitives do not correspond to the final shapes of the reconstructed 3D model that needs to be validated. Therefore, they can be used with the approach as soon as their corresponding Boolean or parametric operations are performed to obtain explicit geometric representations (B-Rep, splines, etc.). Moreover, we propose a quantitative formulation for this consistency grammar in the form of an energy function that, we believe, may straightforwardly be used to build automatic model checkers. Our formalization is independent of standard specifications and can be extended to a variety of applications.

The consistency grammar does not correct or change the components but identifies, formalizes, and (in-)validates the inconsistency of components in relation to each other and their semantics. A consistency grammar acts as a compiler for 3D models and after parsing the 3D model for components, analyzes their correctness in three phases: (1) individually (instances); (2) in interaction with each other; and (3) in application. Incorrect components are flagged and rejected to previous steps (e.g., semantic labeling or geometric modeling) for further checking. The correction of inconsistent items is based on the judgment of the modeler and can be corrected manually or automatically. However, our system allows some of the constraints to be passed if they are not crucial for the integrity of the 3D model. For example, if a model allows a gap between rooms because the walls are not modeled, then this can be passed to the system and the system continues with further checks.

The main effort in this article is to set the basis for a discussion regarding the consistency validation of a reconstructed model from a point cloud. The full implementation of the framework can be challenging in terms of involving decisions from specialists and testing based on different reconstruction methods, and is part of future work. However, we believe that each step of our framework finds its basis in well-established knowledge (e.g., there is a wide range of literature on checking the geometric/topological validity of 3D objects, which is required in the instance and interaction checks). Therefore, our main contribution is formulating a theoretical framework to organize the steps leading to such validation.

The remainder of the article is organized as follows. Section 2 gives an overview of the current methods for indoor 3D modeling from the literature and compares some of the methods. Section 3 explains the methodology of our work and Section 4 gives some examples to illustrate the problem and solution. Section 5 concludes and discusses future work.

2 | SCIENTIFIC BACKGROUND

Indoor models have been investigated from two main perspectives: (1) indoor model reconstruction and (2) indoor model applications. While the former studies the methods for building 3D models (façade and indoor) from the
data, such as RGBD, LiDAR, and imagery, the latter tries to improve indoor model standards, such as industry foundation classes (IFC), CityGML (LoD4, LoD3), and IndoorGML. However, one important aspect of indoor 3D models is their consistency, which is not addressed sufficiently in the literature. Gröger and Plümer (2009) study how to preserve the geometric/topological consistency of 3D city models. Some of their solutions on primitive shapes geometry and topology can be applied for indoor objects, but they are not suitable for complex indoor scenes. Horna, Meneveaux, Damiand, and Bertrand (2009) use generalized maps (G-maps) as a topological basis to define a set of constraints for keeping the consistency of 3D models generated from 2D floor plans. Their solution is interesting in terms of applying the constraints to keep the production process generic to various types of building, but it is only usable for models generated from floor plans. Gröger and Plümer (2010) introduce a constraints store for the derivation of 3D indoor models by grammars. The constraint store prevents errors contradicting consistency/reachability rules. For instance, by using the grammar, a new split face should not traverse a door inserted with a previous rule. These constraints are generated by rule applications and explicitly formalize the concepts of adjacency, reachability, and semantics. Their proposed constraint store maintains three main rules: (a) equality constraints that mean two boxes share the same face or wall; (b) aggregation constraints that mean face F is the aggregation of face F1 and F2, where F is split with a face or wall; and (c) inside constraints that mean face F1 is inside face F2, for instance a door inside a wall. Their solution is a good example of using grammar to keep the consistency of a model during the production process. However, the proposed approach has no example of a real case study or a data-driven model, and applying their methods on non-Manhattan world buildings because of the box representation for rooms.

In using grammar approaches for 3D modeling, Becker, Peter, and Fritsch (2015) use an L-system combined with a shape grammar to reconstruct interiors from point clouds. The authors define an interior structure as two main subdivisions: rooms and hallways. For each structure, they apply a specific grammar to reconstruct the interior environment. Another example of using grammar is given by Ikehata et al. (2015). They first encode indoor space into a structure graph, and then transform it into a structure grammar. Their work is a practical representation of inverse procedural modeling for indoor environments, but is limited to Manhattan world structures. Split grammar is the famous method proposed in instant architecture by Wonka, Wimmer, Sillion, and Ribarsky (2003) to automate the process of urban design for a variety of building styles. The authors claim that the power of split grammar lies in the restriction of allowed rules, which control the derivation process while keeping it simple. Similarly, procedural modeling is used by several researchers for façade modeling and indoor modeling, but in most cases they produce synthetic models for virtual cities and gaming applications (Aliaga, Demir, Benes, & Wand, 2016; Mathias, Martinović, Weissenberg, & Van Gool, 2011; Müller, Wonka, Haegler, Ulmer, & Van Gool, 2006). There are some recent examples for indoor modeling and using grammar which result in more complex models from point clouds (Dehbi, Hadiji, Gröger, Kersting, & Plümer, 2017; Tran, Khoshelham, Kealy, & Díaz-Vilariño, 2019b; Tran & Khoshelham, 2020) or inferring the semantics of rooms (Hu et al., 2019), but none of these methods offer a complementary solution to check the correctness of the final model. Figure 1 illustrates the evolution of grammar applications from the early stage to this time. The figure shows that shape grammar has increasingly been used in building architecture and modeling. Therefore, we decided to apply grammar for consistency control of the 3D models. Note that our suggested grammar, unlike the mentioned grammars, does not apply production rules to create a 3D model, but it uses rules to verify the components of a model.

Apart from grammar approaches for indoor 3D modeling, there are other methods such as piecewise planar reconstruction (Mura, Mattausch, & Pajarola, 2016), mesh-based methods (Xiao & Furukawa, 2014), and binary space-partitioning methods (Ochmann, Vock, Wessel, & Klein, 2016). Each of the methods result in different types of model and levels of detail. We classify these models into three main categories, based on the type and existence of walls: (a) models with volumetric walls; (b) models with paper-thin walls or surface walls; and (c) models with no walls and just a room representation. Table 1 shows the present layers in each model from the literature, and Figure 2 categorizes them based on the type of wall. Also, we include IndoorGML and IFC examples in Table 1, to compare their differences.
Another important aspect when studying indoor 3D models is the data stream from which the model is created. Most of the models presented in this related work, and those which are the focus of this research, are created from point clouds and RGBD images (Furukawa, Curless, Seitz, & Szeliski, 2009; Mura et al., 2016; Nikoohemat, Diakité, Zlatanova, & Vosselman, 2019; Ochmann and Vock & Klein, 2019). However, our suggested solution can be useful also for models generated from inverse procedural modeling methods and from floor plans (Horna, Damiand, Meneveaux, & Bertrand, 2007; Okorn, Xiong, Akinci, & Huber, 2010). As long as the 3D model can be decomposed into its reconstructing components and can be parsed into our grammar, it can be used in our framework.

3 | METHODOLOGY

The goal of our work is to check the consistency of a 3D indoor model, namely a digital representation of a building’s interior created from point clouds, RGBD images, or floor plans. The process aims at checking the correctness and completeness of the model in the lack of ground truth, by relying on expert knowledge and existing standards.

First, the model should be stored in a (context-free) grammar structure. Here we explain several keywords. For clarification purposes a class is referred to as any component in the 3D model, either abstract or real, such as a permanent structure (a real component), or a space (an abstract component). An instance refers to an object of a given class; for example, each room is an instance of the space class or each piece of furniture is an instance of the furniture class. Symbols are terminals, non-terminals, and the axiom in the grammar. Constraints are rules in the grammar. A 3D model can be represented as a vector model (mesh, polyhedron, etc.) or a raster (e.g. voxel). It can be acquired from digitizing floor plans and converting them to a 3D model, or from depth images or from LiDAR

![Diagram of grammar application domain](image-url)
scans. A 3D model should at least show the layout of spaces and interiors. According to Table 1, each model may have one or some of the below classes. In total, eight classes are considered in our grammar for consistency control:

1. Permanent structure classes including walls, floor, and ceiling are three classes.
2. Openings including doors and windows are considered as two classes.
3. Furniture, obstacles, and objects are considered as one class.
4. Spaces are considered as one class.
5. Stairs or staircases are one class.

Note that spaces are the only class which can be abstract components and can be represented by the surrounding structure elements such as walls. Real components can be represented by geometric objects such as solids, surfaces, and boundary representations. Note that furniture can have different interpretations in a model. Here by furniture we are referring to objects which are attached to or resting on a permanent structure such as a wall, floor, or ceiling and are detectable in the data; for example, AC power plugs or switches are too small to be considered in the 3D modeling. In the following, we first explain the components of our consistency grammar and then the method for applying the grammar on a 3D model.

### Table 1: Comparison of existing layers in 3D models in the literature and the difference in modeling of walls

| Wall  | Floor/Ceiling | Door | Window | Stairs | Rooms |
|-------|---------------|------|--------|--------|-------|
| INDOORGML | No wall | Yes | Yes | No | Yes |
| IFC | Volumetric | Yes | Yes | Yes | Yes |
| Budroni and Boehm (2010) | Paper-thin | Yes | No | No | No |
| Xiong, Adan, Akinci, and Huber (2013) | Paper-thin | Yes | Yes | Yes | No |
| Mura, Mattausch, Villanueva, Gobbetti, and Pajarola (2014) | Paper-thin | Yes | No | No | Yes |
| Oesau et al. (2014) | Paper-thin | Yes | No | No | No |
| Xiao and Furukawa (2014) | Volumetric | Yes | No | No | No |
| Becker et al. (2015) | Paper-thin | Yes | No | No | No |
| Ikehata et al. (2015) | Paper-thin | Yes | Yes | No | No |
| Turner, Cheng, and Zakhor (2015) | No wall | Yes | No | No | Yes |
| Mura et al. (2016) | No wall | Yes | No | No | Yes |
| Ochmann et al. (2016) | Volumetric | Yes | Yes | Yes | No |
| Ambruş, Claiici, and Wendt (2017) | No wall | Yes | Yes | No | No |
| Macher, Landes, and Grussenmeyer (2017) | Volumetric | Yes | No | No | No |
| Murail, Speciale, Oswald, and Pollefeys (2014) | Paper-thin | Yes | Yes | No | No |
| Wang, Xie, and Chen (2017) | Paper-thin | Yes | No | No | Yes |
| Wang et al. (2018) | Paper-thin | Yes | Yes | Yes | No |
| Nikoohemat et al. (2018) | No wall | Yes | Yes | No | Yes |
| Ochmann et al. (2019) | Volumetric | Yes | No | No | No |
| Tran and Khoshelham, (2020) | Volumetric | Yes | No | No | Yes |
| Nikoohemat et al. (2020) | Volumetric | Yes | Yes | No | Yes |

Note: Note that this comparison is based on the authors’ interpretation of the models. The algorithms in the original literature may produce more details.
Our grammar is a four-tuple $G = (N, T, R, S)$ including a set of non-terminals ($N$), a set of terminals ($T$), a set of constraint rules ($C \in R$), and a start symbol ($S$). Each rule is of the form $a \rightarrow b$, where $(a, b) \in (N, T)$. Terminals and non-terminals are geometric primitives or geometric complexes. For example, a wall can be represented by a solid (GM_Solid, ISO 19107 Standard for geographic information) or a surface. Rules are extracted from standards such as ISO 19107, IFC, and IndoorGML and applied as constraints, meaning that they are not production rules. Each rule only (in)validates the objects in the model against a given constraint. Some of the rules have preconditions, meaning a rule cannot be applied if the precondition is not satisfied. The constraints control the consistency of the 3D model in terms of topology and the integrity of the model in terms of interaction between instances.
3.1.1 | Manhattan-free grammar

Since most of the models in the recent literature are non-Manhattan and could have arbitrary wall layout, our grammar does not enforce orthogonality and parallelism constraints to any of the geometries. A wall can be slanted and have arbitrary intersection angle with adjacent structures. However, one can include these constraints for the aesthetic aspect of the model. For example, it can be included in the grammar that walls should be perpendicular to the building façade and to the floor.

3.2 | Applying the grammar on a 3D model

3D models can be acquired from different sources and by using different methods. First, we explain a common approach for creating 3D models from point clouds. Having a dataset of point clouds from interiors, most of the algorithms extract permanent structures (walls, floors, and ceilings) using methods such as cell complex, adjacency graph, piecewise planar reconstruction followed by a room segmentation approach to create a 3D model. Some related works take it further and add more details such as openings, windows, and stairs. Figure 3 shows a general workflow of extracting geometric primitives from point clouds and then adding semantics to provide a coarse 3D model. Our consistency grammar comes into play at this stage and checks the components of the model concerning their geometry, topology, and semantics. The assumption of our work is that we do not have external sources or ground truth for the consistency check, and it is based on common knowledge, existing standards, and self-control. Any component which is rejected

**FIGURE 3** Flowchart for proposed methodology. Colored shapes represent the main steps in the creation of a consistent 3D model
should be returned to an early stage of reconstruction (e.g., geometric derivation) and can be checked manually or if possible automatically for corrections. The process of correction is not part of the consistency grammar and it is related to the reconstruction algorithm. The objective of a consistency grammar is identifying inconsistency in the model.

The rules check the validity of the component individually and in relation with other components in terms of geometry, topology, and semantics. Before applying the rules, a 3D model parser decomposes the model into components that are stored in grammar (non-)terminals. Then, rules are applied on non-terminals as instances or a pair of interacting instances. When an instance is not valid, it is flagged as invalid and will not be checked further until the issue is resolved and it is returned to the geometric derivation step in the flowchart (Figure 3). However, some of the checks can be flagged as valid if the user (operator of the system) decides to pass the instance. We consider such rules as softer rules that can be relaxed to allow a certain amount of satisfaction (e.g., a door can be disjoint with the floor up to a threshold). This serves the flexibility of the system in terms of applications and the purpose of the modeling.

The consistency grammar is applied in three main steps (Figure 4):

1. **Instance**—meaning a single component in the 3D model is consistency-checked independently.
2. **Interaction**—proximate components which interact with each other are controlled.
3. **Application**—components are controlled based on the application of the model.

Each of the above steps enforce some geometric, topological, and semantic constraints inferred from the standards or which are defined by the experts. In the next subsection we explain how, starting from a 3D model, each of the steps can be applied.

### 3.3 A 3D-model parser

The consistency grammar takes the 3D model as input and decomposes it into the constituting classes. Each class has instances and each instance has attributes such as bounding box, local coordinate, color, and so forth. This is the task of the parser, to recognize each instance in a class and store it as a non-terminal symbol with the class name as the attribute. Note that the parser does not need to infer the type of class for each instance, and this information should be provided in the model. For example, walls and doors have different labels in the original model. Some of the attributes, such as volume, can be calculated from the solids and stored with the instance in the non-terminal set. Other attributes should be provided with the geometry or, if necessary, filled manually in the related fields—for example, the correct number of rooms in the model, if known, or the number of steps in stairs. Attributes include but are not limited to volume, area, width, height, length, color, location, function (kitchen, corridor, ...), type (abstract, real), scope (permanent, temporary), number of levels, number of steps, number of
rooms, and materials. During parsing, the user can add more semantics to the component which would be useful for consistency grammar, for example separating façade walls from internal walls which can define the external shell of the building. Each instance per class is a geometric primitive or a geometric complex and is stored in a set of non-terminals (N), where it is selected and checked with the constrained rules in the grammar.

3.4 Consistency control of instances

Since non-terminal instances correspond to 3D geometric entities, their validity should be ensured independently of each other. This can be done from a geometric, topological, or semantic perspective. Geometric and topological validation of 3D objects is a well-explored issue in the literature. This knowledge serves as support for standards commonly used in GIS, such as ISO 19107. The latter specifies conceptual schemas for spatial description and consistent operations for (up to 3D) vector geometry and topology of geographic features. In this work, we rely on the specifications of ISO 19107 to define valid spatial entities (similarly to IFC and CityGML). A comprehensive study on the validation of solids is also presented by Ledoux (2013). It should be noted that the geometric inconsistencies that are discussed in this article do not include the accuracy per se. This is because one could hardly estimate the geometric accuracy of a 3D model without a ground-truth model to compare with. Instead, we address common geometric issues that lead to wrong models independently of the ground truth.

Based on our definition of 3D models, three types of geometry can be expected: (a) surface; (b) volume (solid); and (c) complex. At the beginning of the instance check process, every instance is initially assumed to be invalid. Hence, independent of its semantic nature, checking the geometric and topological consistency of a non-terminal instance consists of applying several geometrical predicates (such as orientation or closure tests, etc.) that should result in Boolean outputs (true or false). In the case of surface elements like polygons, examples of predicates include simplicity check (the boundary of the polygon should not self-intersect or contain duplicated vertices), planarity check (polygons forming building structures are expected to be planar), or degeneracy check (surface area should not be null). In the case of volume elements like polyhedron, they should match the definition of a solid (Gröger & Plümer, 2009; Ledoux, 2013). This includes conditions such as their boundaries should be composed of valid polygons and they should form a full closure of the space. Checks that apply to polygons may also apply to polyhedrons, with one more dimension to be considered (e.g. degeneracy test of a volume to see whether it is flat, with null magnitude). Criteria that define valid 3D polygons and polyhedra can be found in the standards (ISO 19107, OGC, etc.; Ledoux, 2013). Every check that results in a positive output would change the validity state of a non-terminal to true. Otherwise, the invalid instance is flagged and may be considered for a correction process. Complex objects consisting of several primitives, such as an L-shaped wall or stairs, need to be decomposed into simpler geometries in order to validate them.

Finally, the semantic control of an instance includes validating the instance knowing its semantic, class, or function. This is rather more difficult than the two other checks on geometry and topology, because there are fewer semantic standards for objects. By referring to IFC standards (e.g., IFCWall or IFCWindow), it is possible to infer some of the rules for the instances. Similarly, by using IndoorGML standards, we can check the validity of indoor spaces and rooms (cellular space in IndoorGML). Examples such as a wall is a vertical element which bounds rooms and subdivides the space can be interpreted from the IFC standards. Likewise, a room should not be smaller than a certain volume or height are types of rule which can be inferred for semantic control of the instance.

The instance consistency grammar is summarized in Algorithm 1. Note that non-terminals (N) are primitives containing the class type and extra attributes, such as bounding box, color, and height, which can be obtained with getters functions (lines 7 and 8 in the algorithm). The case of a rule can be defined as \( r \in R, X \rightarrow \text{valid} \mid \text{invalid} \prec C \prec E \), where \( C \in R \) applies the constraints and \( E \) returns the type of error in case of invalid results. Rules (R) are a collection of Rs as semantic rules and Rg.t as geometric and topological rules. Note that the selection of a rule from the set of rules is decided based on the geometry and the class of the instances. The result of this step is a set of invalid instances with the type of error which needs to be resolved by expert decision before proceeding to the next step (interaction control).
Consistency control of interactions

The non-terminals from the previous step are validated individually considering their geometric, topological, and semantic information. Here, the grammar controls two instances either from the same class or from different classes in interaction with each other. Although, from a general perspective, all instance interactions correspond to spatial relationships, thus topological relationships, the difference in geometry, topology, and semantics of the instances leads to more specific types of interaction. For example, when checking the intersection between two volumes, geometric and topological information prevails in the interaction. In contrast, a semantic check is related to the interaction of instances considering their class type. For example, furniture cannot be dangled in the model without attachment to a room or a permanent structure. Several topological models allow us to describe extensively the spatial relationships linking instances. Here we rely on the 9-intersection model (Egenhofer & Herring, 1990; Zlatanova, Rahman, & Shi, 2004) (see Figure 5) in our grammar to describe the valid and invalid interactions.

Valid interactions include:

\[
\{\text{disjoint, meet, coveredBy}\} \in R \subset C
\]

where rules \(R\) are defined by constraints \(C\). The remaining relationships will be considered as invalid for interaction of two objects. For example, furniture is coveredBy a room but a wall is met by a door. The meet relation happens when the boundaries of two 3D objects meet but not their interiors (Zlatanova et al., 2004).

A handy way to map the instance interactions is to use an interaction matrix. Table 2 illustrates one that lists a few selected classes and shows their interactions, considering the information of their individual instances. Each row \(i\) and column \(j\) of the matrix refers to an instance in a specific class and each cell \(C_{ij}\) corresponds to the interaction rule of an instance \(i\) with respect to an instance \(j\). While an instance cannot be compared with itself, instances of the same class can interact with each other. Therefore, the diagonal cells on the matrix are not empty.

The colors used in Table 2 allow us to determine the level of interaction that can be expected between two instances. A green cell \(C_{ij}\) means that there must be an interaction between \(i\) and \(j\). This is the highest level of interaction constraint that makes one or both instances invalid if not fulfilled. As an example, a wall instance is necessarily expected to touch other wall and floor instances. This corresponds to the meet relationship in the
Similarly, the red cells represent interactions that should not happen, or said differently, disjoint relations. Otherwise, the instances are flagged as invalid. Yellow cells describe the most flexible constraint as the interaction of the two instances can eventually happen if some conditions are satisfied or may simply not
In both cases, that does not mean there is a problem, unless the identified relationship is an invalid one (e.g., overlap). For example, in some architectural designs a door and a window can meet. Therefore, yellow cells in the matrix can change to either green or red based on expert decision and the building architecture. One should note that the interactions in Table 2 are not necessarily bi-directional. For example, in the case of the interactions between door and wall instances, it is mandatory for door but optional for walls. Such specificities need to be accounted for in the design of the interaction rules.

The interaction matrix is translated into a graph where instances are nodes and their interactions are edges. Nodes are connected by solid lines representing green cells and dashed lines representing yellow cells. Clearly, for red cells no edges exist. Figure 6 shows how the colors are translated into edge types in the graph. Green and yellow cells represent two instances being adjacent, and the line is dashed if the connection is conditional. We included three higher levels of interaction in the grammar rules as a subset of meet in the 9-intersection model:

\[
\text{(support, supported, attachment)} \in \text{<meet>}
\]

A support relation occurs when one object is not only adjacent and meets another object, but also supports the object. A supported object always needs the adjacency of its supporter. For example, a lamp is always supported
by a wall, ceiling, or other furniture. An attachment occurs when a room is an abstract component in the model and is represented by surrounding instances such as walls. For example, furniture should always be attached to a room. Similarly, for adding material and texture, there is no spatial relation and the connection is of attachment type.

Figure 7 shows the translation of the interaction matrix (Table 2) to the graph. This graph is an abstraction of a 3D model. Such a representation of classes and their instances in the form of a graph is scalable because the experts can add more nodes and can define the connections, such as adding furniture types or beams. A real graph which exactly reflects all the components in a 3D model looks much more complex, as each node represents an instance. Thus, for example, for each wall or door object there should be a separate node in the model. In Section 5 we show an example of such a graph.

It is noteworthy that the interaction matrix provided in Table 2 is not necessarily exhaustive, despite our efforts to illustrate common interaction expectations between semantic instances of building models. In fact, specific building designs could lead to different constraints in such a matrix, leading to a different semantic interaction matrix. While our semantic interaction matrix method is meant to help in the automation of the consistency check process, it is up to the expert user to define their customized interaction constraints based on the expectations from the 3D models.

The interaction consistency grammar is summarized in Algorithm 2. The algorithm traverses all the edges, and for each edge (e) the nodes (u, v) are selected. Similar to instance grammar, the class type and geometric type of the nodes (instances) are derived from the attributes and the corresponding cell in the interaction matrix can be identified. When the interaction type is recognized (green, yellow, condition), the corresponding rule(s) can be applied. Note that the applyRule function (line 7) in the algorithm is not limited to one rule but can sequentially check several constraints for the connection of the nodes. For example, a door and wall connection is verified in terms of geometric, topological, and semantic constraints. The output of the interaction consistency grammar is a set of errors reflecting the edge (and corresponding nodes) and type of errors. This step is passed if the error set is empty, meaning that all the edges and their corresponding nodes are valid.

Algorithm 2 Interaction grammar

R ← \{interaction rules ∈ R\}
2: \(E_{map} \leftarrow \{\text{map of errors and edges} = \emptyset\}\)
   for each edge ∈ Edges do
4:   edge ← getNodes(edge, AdjacencyGraph)
5:     u ← getNodeType(CTypes, GTypes)
6:     v ← getNodeType(CTypes, GTypes)
7:     edge ← applyRule(u, v, R)  \hspace{1cm} \triangleright \text{Applying rules}
8:     if edge ≠ \emptyset then
9:       edgeisValid ← true
10:      else
11:       \(E_{map} \leftarrow \text{insert(error, edge)}\)
12:     end if
13:   end for
14: return \(E_{map}\)

3.6 | Application-specific consistency control

The last step of our methodology consists of controlling the consistency of the model with respect to criteria specific to given applications. While the two previous steps can be considered as generic constraints commonly accepted for 3D models, the rules in this part of the grammar are more specialized to application domains. Thus, the rules in this part
of the process are similar in design to those of instance and interaction checks but remain necessary for specific use cases. Consequently, a 3D model invalidated by this step for one application does not necessarily imply its invalidity for another application, and vice versa. The same applies to validated models. As an application example, we choose indoor navigation and define corresponding rules. We control the consistency of our generated model against a set of defined heuristic constraints (Ci) targeted to specific instances or combinations of instances critical to indoor navigation (e.g., detected doors, stairs, and reconstructed spaces). The correctness of the model regarding those instances can be improved by consistency check. The constraints, which are checked on the data sequentially, are described as follows:

1. Each room should have spatial extents $L$, $W$, and $H$ greater than a given threshold.
2. Each room should be connected to at least one door.
3. There should be a route connecting every pair of rooms in the graph.
4. Two floors are connected via at least one staircase.
5. Each space should have access to the exit door.
6. Emergency doors should not be blocked.

3.7 | Optional constraints:

1. Angular arrangements (e.g., all the corridors have the same turning angle).
2. Wall thickness operations (e.g., all interior walls have the same thickness).
3. Aesthetic operations (e.g., all doors in corridors have the same type).

The verification of C1 is necessary to make sure that rooms are spacious enough for navigating agents to fit in them. This can be seen as an application-specific instance check. In fact, due to the different methods possible for indoor space reconstruction, it can happen that reconstructed volumes are too narrow to be even considered as a space. Thus, this constraint helps in detecting such features and guaranteeing that only reasonable paths will be delivered in the navigation process. While their threshold values can be determined based on input information of the navigation or generic assumptions (e.g., average human size), the estimation of the spatial extents—namely length ($L$), width ($W$), and height ($H$)—is not always trivial, in particular in non-Manhattan configurations, but extent checking on bounding boxes of rooms may often be enough. Once a room is validated, it passes to the next check. Otherwise, it should be flagged for further control. After correction, the pipeline is repeated to create a consistent model (see Figure 3).

**FIGURE 8** Illustration of the constraint C2. A room (emphasized in bold blue lines) for which the node is connected to no other node of the graph characterizes a missing door.
Constraint C2 checks if each room is connected to at least one door. Otherwise, a missing door is flagged for the room. This can be seen as an application-specific interaction check. Door detection is a challenging process in 3D indoor reconstruction from a point cloud. However, although the available algorithms could miss several of them in a scene, a room without any doors necessarily reflects a problem in the reconstruction. This is also an obvious limitation for navigation purposes. As illustrated in Figure 8, by relying on the navigation graph, it is straightforward to identify the problem, as it only requires detecting rooms associated with isolated nodes. Similarly to C1, spaces detected in this check are tagged for further improvement of the model reconstruction.

Constraint C3 controls the connectivity of two spaces. It also relies on the navigation network and covers the cases that cannot be detected with C2 (e.g., two rooms connected by a common door but disconnected from all the other spaces). We assume that every two spaces in a correct 3D model should be reachable and connected through the navigation graph. Therefore, the process checks the existence of a path between every two rooms in the model.

Constraint C4 completes C3 with the necessity of a staircase instance connecting two floor levels. In fact, rooms may be vertically connected due to holes or unreconstructed slabs. In such cases, while it may be assumed that the holes are elevator compartments, there is always a staircase built for use in case of emergency or elevator failure. In contrast, despite the existence of staircase instances in the 3D model, they may not properly connect two levels. This typically happens when the number of steps in the stairs is not reconstructed correctly, breaking C3 (for rooms on different floors) and C4.

Finally, C5 and C6 are constraints important for navigation and related cases (e.g., emergency evacuation). By ensuring a connection to an exit from every room in the model, the consistency check guarantees that at least one of the critical doors of the building (emergency doors) is reconstructed and reachable, and is not blocked. This may not imply that the door is usable, however. Performing C5 comes down to verifying if, after all previous checks, there is at least one exit door that is connected to every space in the same division of the building. Note that using constraints such as C6 implies that the type of door (emergency door, exit door, ...) for the application should be added to the model.

In addition to constraints C1 to C6, experts can define other constraints based on the application. Constraints C7 to C9 are examples of enforcing some geometric controls in the model. In many architecture styles, walls have 90° intersection angles. This can be checked with constraint C7. Similarly, constraint C8 controls whether the interior walls have the same thickness. Likewise, it is possible to control if the same type of door or window is reconstructed for some divisions of the building with constraint C9. For example, in some buildings, doors in a corridor are double doors or sliding doors. Note that although the rules for the application phase of the consistency grammar can be customized for different 3D models, they should not violate any of the previous rules or current geometric and topological standards.

4 | ENERGY FUNCTION

We have presented the consistency grammar in a conceptual form. In order to computationally identify the objects that do not fit in the model, we propose a quantitative measure in the form of an energy function. The energy function is a mathematical representation of the conformity of the indoor model to the rules presented above, and is written as:

\[ E_i = E_{self}(i) + \sum_{j \in N(i)} E_{neighbor}(i,j) + \sum_{k \in V} E_{missing}(i,k) \]

\[ E_{neighbor}(i,j) = \begin{cases} 
1, & \text{if } A_{ij} \text{ is red} \\
0, & \text{otherwise} 
\end{cases} \]
where $E_{self}(i)$ is a self-energy describing the instance fit for object $i$, $E_{neighbor}(i,j)$ is the pair energy describing the inappropriateness of the connection between the object $i$ and its neighborhood objects $j, j \in N(i)$, as indicated with red spaces in Table 2, and $E_{missing}(i,k)$ is the pair energy describing possible missing connections of object $i$, as indicated by green spaces in Table 2. Object vector $V$ is the left-hand column of Table 2, namely:

$V = \{\text{wall, floor, ceiling, door, window, stair, room, furniture}\}$

We give suggestions for the parameter values. Table 2 color values can be converted into numerical constants so that, for example, red spaces correspond to number 1 and others to number 0 for the $E_{neighbor}$ calculation, and similarly green spaces correspond to number 1 and others to number 0 for the $E_{missing}$ calculation. This makes the value of the energy function low when the tested indoor model conforms to the interaction rules, and high when it does not. In contrast the instances could, for example, be either accepted or rejected (numerically the energy is zero or infinity).

The total energy for the indoor model would then be computed as the sum of the object energies:

$$E_{total} = \sum_i E_i$$

with the predicament that the more the energy augments from zero, the more inconsistencies the model contains. For example, consider a four-wall room with a roof and a ceiling and with a door in one wall and a window on the opposite wall. Evaluating this model with the energy function yields the result $E = 0$, that is, a score for a valid model. If the model were corrupt, for example, so that the door was labeled as a wall, the $E_{missing}$ energy would augment from 0 to 1 because the room would not have a connection to a door. Also, $E_{missing}$ would also increase, again by 1, if the apparent wall had no connection to the ceiling (that is, the corrupted door object is not as high as the walls that mount to the ceiling). Another example of model corruption would be if one of the physical walls is made largely of glass, making its model object appear to exist only close to the ceiling and not be connected to the floor, which is a problem. This problem is again identified by the energy function term $E_{missing}$, which is augmented by 1 due to the missing wall–floor connection. In contrast, the term $E_{neighbor}(i,j)$ detects invalid connections between floor and ceiling, for example.

Finally, please note that the pair energy notation regarding Table 2 is designed so that the evaluation is done with $E_{column, row}$. Importantly, the matrix is not symmetrical, since, for example, a window must be connected to a wall, but a wall does not need to have a window.

5 | RESULTS

We show some examples for each step of the methodology to exemplify the grammar and implications of the rules. Figure 9 shows an example of a model which has walls, floors, ceilings, doors, stairs, furniture, and rooms. In the illustrated model, each reconstructed component is a volumetric feature, including walls, doors, and rooms represented as solid geometry (polyhedron). Following our framework, the three steps of verification in the methodology (see Figure 4) should be applied to the 3D model. It is therefore decomposed to its constructing components (Figure 9) and, as a first step, the geometric and topological consistency of the solids (e.g. walls) are controlled. The decomposition of the model can be done by 3D-model parser (see Section 3.3).
5.1 | Instances check

We used the model in Figure 10 to illustrate an example of the instance check. Based on what should be expected from 3D solids (Gröger & Plümer, 2009; Ledoux, 2013), the following rules are chosen and described for each single component of the model:

1. Every polygon should be simple (no self-intersection).
2. The solid should be closed and there should be no dangling face or edge.
3. The normal vector of all faces of a volume should point towards the exterior of that volume.
Our 3D model is geometrically described as B-Rep, along with information on its different classes of entities such that each single instance can be distinguished and is provided with its own list of 3D faces and semantic information. That information is used to implement the rules on faces (polygons). The first rule means that no inconsistency on the boundary of the polygon should be accepted (e.g. self-intersecting boundaries, duplicated vertices, etc.). No such issue should be found in the model because the whole model is triangulated, resulting in uniquely simple faces (triangles). The second rule concerns the topological closure of the solids, such that every edge of a given polygon is paired with the same edge of another polygon in the same polyhedron. Figure 10 illustrates a case where a volume is open because one of its sides has not been reconstructed properly and therefore has missing faces and consequently open boundaries. Such an instance should be corrected until a full closure is obtained. Finally, the last rule checks the consistency of the orientation of the volumes. The whole model should follow one convention, which is that either all the face normals point inward or outward of the solid they belong to. We chose the second case, in accordance with ISO 19107. Note that the geometry accuracy depends on the 3D modeling algorithm and the accuracy of the input model, meaning that our method is not checking a model-to-model comparison, because the assumption of our work is based on the lack of ground-truth models.

Once the consistency of the geometric entities is confirmed and the invalid instances are corrected, we go to the second step that checks the interactions between valid instances.

5.2 | Interactions check

As explained in the consistency control of interactions (Section 3.5), we use an interaction matrix (Table 2) to determine the correct interactions in the model. First, an adjacency graph is generated from all the components in the model which have adjacency to each other. Figure 11 shows how a graph represents a 3D model. Nodes are components (e.g. rooms, walls, doors, and windows) and edges are the interactions. To exemplify an interaction problem, Figure 12 shows one issue in the reconstruction of our test model. While in Table 2 it is specified that a door (opening) is expected to be adjacent to a wall, we found a door that does not share a face with its nearest wall instance. This is due to imprecision in the reconstruction, for example the wall thickness is changed without fixing its topology with a door. Other interaction inconsistencies that can be checked include walls not connected to the ceiling or stairs not attached to floors. Such issues are avoided in our model by exaggerating the extents of walls and slabs. However,
this leads to other issues such as intersecting walls. Such interaction would correspond to the overlap relationship from the 9-intersection model, which is not considered valid in our grammar, as it cannot happen in the reality of building elements. Therefore, this should be considered an inconsistency. However, because the main purpose of the reconstruction of the test model was to obtain indoor spaces, and this needed a proper space closure to be extracted (Nikoohemat et al., 2019), we can afford to ignore these interaction inconsistencies in the validation process. This shows the flexibility in the validation framework and makes a link to the next step of the process, as targeted applications may make some consistency rules more relevant or critical than others, which can just be ignored.

Quantitatively, the evaluation of the given model can be done using the energy function formulation of Section 4. If door D is not connected to a wall, we get \( E_{\text{missing}}(\text{door D, wall}) = 1 \), while in the nominal case the result would be zero (see Figure 12). Similarly, the energy function quantifies if, for example, stairs are not attached to the floor and so on.

### 5.3 Application consistency checks

Our example model was made for indoor navigation purposes. Thus, the rules defined in Section 3.6 can be used for our examples as well. While the check of rule C1 is straightforward, inconsistency issues related to C2 can be observed. In fact, as shown in Figure 8, some spaces happen to be connected to no door from the whole model, which means that they are isolated spaces. This reveals missing doors that were not reconstructed or spaces that are wrongly reconstructed. In both cases, the issue needs to be addressed for an indoor navigation usage of the model and for the remainder of the checks to be properly performed.

Indeed, other rules such as C3 depend on C2, because missing doors would logically lead to unconnected pairs of rooms. Thus, the constraint C3 cannot be met as well. Figure 13 illustrates the C4 check, where two floors are connected by at least one staircase. The illustrated path demonstrates a navigation route from a space downstairs to a space upstairs through staircases (here represented by another space connecting both levels). Regarding check C5, it can be performed by defining one door as the main exit door of the building and computing the exit path from each room. Finally, C6 checks if every emergency door has clearance. However, this control can be applied to any door. To perform door clearance, a buffer for the doors should be considered and then the bounding box of the furniture would be intersected with the buffer of the door. If an intersection occurs, this means the door path is blocked (Figure 14).
Again, quantitatively, if room R does not have a door, we get $E_{\text{missing}}(\text{room R, door}) = 1$, while in the nominal case the result would be zero. In addition to missing connections, our method checks for unwanted overlaps. For instance, each time a door is obstructed by a piece of furniture, $E_{\text{neighbour}}(\text{door, furniture})$ is augmented by one. Hence, the unwanted occurrences are quantitively detected.

Adding external sources, such as the footprint of the building or number of rooms, can help in invalidating some of the errors and checking the completeness of the model.

We proposed a framework to validate the consistency of indoor 3D models in three steps. The consistency control incorporates the indoor standard data models and ISO geometry standards to apply constraints in a formal grammar setting. Our consistency grammar does not fix the errors but detects them. The integration of correction...
routines is part of future work. One limitation of the proposed method is detecting objects which are misclassified. This is difficult without user intervention; for example, if a large screen is misclassified as a wall, it is not trivial to identify this misclassification. Similarly, if a room has two doors, and one door is detected but the other door is missed—because it was closed and coplanar with the wall—our consistency grammar cannot identify it. Although our framework is scalable and flexible, similar to most of the grammar approaches, it needs expert knowledge for assembling suitable rules in each step and this can be time consuming. However, after this is done once, the method can be applied for the same type of model repeatedly. The energy function of Section 4 is limited to instance and interaction terms for the consistency and does not contain evaluation of the applications defined in Section 3.6. This development is part of future work.

6 | CONCLUSIONS AND FUTURE WORK

The uncertainties in automatically generated 3D indoor models call for a basic logic for model checking. In this article, we have introduced a three-step conceptual framework for checking the consistency of indoor 3D models. Moreover, we also propose a quantitative method in the form of an energy function that, we believe, can be straightforwardly integrated into (industrial) model checkers. Our method is independent of ground-truth data, because such data is rarely available. Instead, we promote the use of expert input and indoor standards, such as IFC, IndoorGML, and ISO 19107. Our framework and method are generic, in that they are suitable for different types of 3D models reconstructed from point clouds, RGBD images, or floor plans.

The final output of our grammar, after several iterations of the model (flowchart in Figure 3), is a satisfactory 3D model which has passed several consistency tests. Our proposed solution does not correct primitives and 3D objects but it (in)validates them in three steps: first, the individual instances are controlled for geometric, topological, and semantic consistency. Then, every two adjacent objects are controlled using an interaction matrix and 9-intersection model schema. Finally, the model is verified for specific applications such as evacuation simulation. After each rejection, the flagged object should be fixed and the grammar needs the corrected component to continue the process. The correction of the errors, which may have several reasons, is the responsibility of the user. Therefore, we do not offer a correction process. However, it is possible to fix minor geometry errors such as a missing closure of a solid or intrusion of volumes. Because of the scalability and flexibility of our system, it is possible to add more nodes and define new rules. Similar to any grammar, defining new rules needs expert knowledge. The suggested adjacency matrix and its corresponding graph help the users of the system to keep the integrity of the grammar when adding new rules. The consistency grammar is scalable to more complex architectures by adding more classes (pillars, furniture types) to the interaction matrix and the corresponding adjacency graph. The energy function is versatile in that it can be adapted for different structures. We present an example version that we believe is rather generic for office and residential buildings. The key idea is that such an intuitive mathematical formulation can be presented and minimized to obtain results; the result is not the absolute minimum energy per se, but it is the minimum (i.e. optimum) result.

This work proposes a concept of model checking using a grammar framework. Section 5 is not based on a full implementation of the proposed methodology, but only examples of how a model can be checked at each step, thus no computational result is presented in this work. The implementation can vary based on the type of model (see Table 2: volumetric walls, no walls, surface representation).

As future work, we will test the proposed methodology on two types of model—volumetric and surface representation—to investigate the further development of the grammar. Our consistency grammar could be implemented for international standards such as IndoorGML and IFC for reconstruction algorithms specifically targeting them. Then it is feasible to study the weaknesses of the framework for further improvements. As an extension of our consistency grammar, it is logical to add automatic corrections instead of rejecting objects; for example, face-on-face or closeness of a polygon can be fixed automatically instead of rejection. One question is whether we
could use the repetitiveness and symmetrical characteristics of building interiors to validate or correct some other parts of the model which suffer from missing data. This is something that can be integrated into the consistency grammar by a learning mechanism. Further work could incorporate texture and material, room functions, and agents (building occupants, wheelchair, drones) to demonstrate challenging scenarios.

ACKNOWLEDGMENTS
This work is part of the TTW Maps4Society project Smart 3D indoor models to support crisis management in large public buildings (13742), which is (partly) financed by the Netherlands Organization for Scientific Research (NWO).

ORCID
Shayan Nikoohemat https://orcid.org/0000-0001-7102-4446
Abdoulaye A. Diakité https://orcid.org/0000-0003-0119-6124
Ville Lehtola https://orcid.org/0000-0001-8856-5919

REFERENCES
Allaga, D. G., Demir, I., Benes, B., & Wand, M. (2016). Inverse procedural modeling of 3D models for virtual worlds. In Proceedings of the 43rd International Conference and Exhibition on Computer Graphics and Interactive Techniques, Anaheim, CA (pp. 1–316). New York, NY: ACM.

Ambruş, R., Claiici, S., & Wendt, A. (2017). Automatic room segmentation from unstructured 3-D data of indoor environments. IEEE Robotics & Automation Letters, 2(2), 749–756.

Bassier, M., Klein, R., Van Genechten, B., & Vergauwen, M. (2018). IFC wall reconstruction from unstructured point clouds. ISPRS Annals of Photogrammetry, Remote Sensing & Spatial Information Sciences, 4(2), 33–39.

Becker, S., Peter, M., & Fritsch, D. (2015). Grammar-supported 3D indoor reconstruction from point clouds for "as-built" BIM. ISPRS Annals of Photogrammetry, Remote Sensing & Spatial Information Science, 1, 17–24.

Budroni, A., & Boehm, J. (2010). Automated 3D reconstruction of interiors from point clouds. International Journal of Architectural Computing, 8(1), 55–73.

buildingSMART International. (2013). Industry Foundation Classes Release 4 (IFC4). Retrieved from https://standards.buildingsmart.org/IFC/RELEASE/IFC4/FINAL/HTML/

de Laat, R., & van Berlo, L. (2011). Integration of BIM and GIS: The development of the CityGML GeoBIM extension. In T. H. Kolbe, G. König, & C. Nagel (Eds.), Advances in 3D geo-information sciences (pp. 211–225), Berlin, Germany: Springer.

Dehbi, Y., Hadlji, F., Gröger, G., Kersting, K., & Plümer, L. (2017). Statistical relational learning of grammar rules for 3D building reconstruction. Transactions in GIS, 21(1), 134–150.

Egenhofer, M. J., & Herring, J. (1990). A mathematical framework for the definition of topological relationships. In Proceedings of the Fourth International Symposium on Spatial Data Handling, Zurich, Switzerland (pp. 803–813).

Furukawa, Y., Curless, B., Setz, S. M., & Szeliski, R. (2009). Reconstructing building interiors from images. In Proceedings of the 12th International Conference on Computer Vision, Kyoto, Japan (pp. 80–87), Piscataway, NJ: IEEE.

Gröger, G., Kolbe, T. H., Nagel, C., & Häfele, K.-H. (2012). OGC City Geography Markup Language (CityGML) encoding standard. Retrieved from www.ogc.org/standards/citygml

Gröger, G., & Plümer, L. (2009). How to achieve consistency for 3D city models. Geoinformatica, 15, 137–165.

Gröger, G., & Plümer, L. (2010). Derivation of 3D indoor models by grammars for route planning. Photogrammetrie–Fernerkundung–Geoinformation, 2010(3), 191–206.

Horna, S., Damiani, G., Meneveauux, D., & Bertrand, Y. (2007). Building 3D indoor scenes topology from 2D architectural plans. In Proceedings of the Second International Conference on Computer Graphics Theory and Applications, Barcelona, Spain (pp. 37–44), Setubal, Portugal: INSTICC.

Horna, S., Meneveauux, D., Damiani, G., & Bertrand, Y. (2009). Consistency constraints and 3D building reconstruction. Computer-Aided Design, 41, 13–27.

Hu, X., Fan, H., Noskov, A., Zipf, A., Wang, Z., & Shang, J. (2019). Feasibility of using grammars to infer room semantics. Remote Sensing, 11, 1535.

Ikehata, S., Yang, H., & Furukawa, Y. (2015). Structured indoor modeling. In Proceedings of the IEEE International Conference on Computer Vision, Santiago, Chile (pp. 1323–1331), Piscataway, NJ: IEEE.

ISO. (2018). ISO 16739–1. Retrieved from http://www.iso.org/cms/render/live/en/sites/isoorg/contents/data/standard/07/03/70303.html

ISO. (2019). ISO 19107. Retrieved from http://www.iso.org/cms/render/live/en/sites/isoorg/contents/data/standard/06/61/66175.html
Kim, H., & Anderson, K. (2013). Energy modeling system using building modeling open standards. *Journal of Computing in Civil Engineering*, 27, 203–211.

Ledoux, H. (2013). On the validation of solids represented with the international standards for geographic information. *Computer-Aided Civil & Infrastructure Engineering*, 28, 693–706.

Lee, J., Li, K.-J., Zlatanova, S., Kolbe, T. H., Nagel, C., & Becker, T. (2016). OGC IndoorGML Standard. OGC® IndoorGML. Retrieved from https://www.ogc.org/standards/indoorgml

Lehtola, V., Kaartinen, H., Nüchter, A., Kajälouoto, R., Kukko, A., Litkey, P., ... Hyypä, J. (2017). Comparison of the selected state-of-the-art 3D indoor scanning and point cloud generation methods. *Remote Sensing*, 9, 796.

Macher, H., Landes, T., & Grussenmeyer, P. (2017). From point clouds to building information models: 3D semi-automatic reconstruction of indoors of existing buildings. *Applied Sciences*, 7(10), 1030.

Mathias, M., Martinović, A., Weissenberg, J., & Van Gool, L. (2011). Procedural 3D building reconstruction using shape grammars and detectors. In *Proceedings of the 2011 International Conference on 3D Imaging, Modeling, Processing, Visualization and Transmission*, Hangzhou, China (pp. 304–311). Piscataway, NJ: IEEE.

Müller, P., Wonka, P., Haegler, S., Ulmer, A., & Van Gool, L. (2006). Procedural modeling of buildings. *ACM Transactions on Graphics*, 25(3), 614–623.

Mura, C., Mattausch, O., & Pajarola, R. (2016). Piecewise-planar reconstruction of multi-room interiors with arbitrary wall arrangements. *Computer Graphics Forum*, 35, 179–188.

Mura, C., Mattausch, O., Villanueva, A. J., Gobbetti, E., & Pajarola, R. (2014). Automatic room detection and reconstruction in cluttered indoor environments with complex room layouts. *Computers & Graphics*, 44, 20–32.

Murai, S., Speciale, P., Oswald, M. R., & Pollefeys, M. (2017). Indoor Scan2BIM: Building information models of house interiors. In *Proceedings of the 2017 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Vancouver, BC, Canada (pp. 6126–6133). Piscataway, NJ: IEEE.

Nikoohemat, S., Diakité, A., Zlatanova, S., & Vosselman, G. (2019). Indoor 3D modeling and flexible space subdivision from point clouds. *ISPRS Annals of Photogrammetry, Remote Sensing & Spatial Information Sciences*, 4(2/WS), 285–292.

Ochmann, S., Vock, R., & Klein, R. (2019). Automatic reconstruction of fully volumetric 3D building models from oriented point clouds. *ISPRS journal of photogrammetry and remote sensing*, 151, 251–262.

Ochmann, S., Vock, R., Wessel, R., & Klein, R. (2016). Automatic reconstruction of parametric building models from indoor point clouds. *Computer Graphics*, 54, 94–103.

Oesau, S., Lafarge, F., & Alliez, P. (2014). Indoor scene reconstruction using feature sensitive primitive extraction and graph-cut. *ISPRS Journal of Photogrammetry & Remote Sensing*, 90, 68–82.

Okorn, B., Xiong, X., Akinci, B., & Huber, D. (2010). Toward automated modeling of floor plans. In *Proceedings of the Fifth Symposium on 3D Data Processing, Visualization and Transmission*, Paris, France.

Pu, S., & Zlatanova, S. (2006). Integration of GIS and CAD at DBMS level. In *Proceedings of the 25th Urban Data Management Symposium*, Aalborg, Denmark (pp. 9.61–9.72). Delft, the Netherlands: UDMS.

Tran, H., & Khoshelham, K. (2020). Procedural reconstruction of 3D indoor models from LiDAR data using reversible jump Markov Chain Monte Carlo. *Remote Sensing*, 12(5), 838.

Tran, H., Khoshelham, K., & Kealy, A. (2019a). Geometric comparison and quality evaluation of 3D models of indoor environments. *ISPRS Journal of Photogrammetry & Remote Sensing*, 149, 29–39.

Tran, H., Khoshelham, K., Kealy, A., & Díaz-Vilarriño, L. (2019b). Shape grammar approach to 3D modeling of indoor environments using point clouds. *Journal of Computing in Civil Engineering*, 33, 04018055.

Turner, E., Cheng, P., & Zakhor, A. (2015). Fast, automated, scalable generation of textured 3D models of indoor environments. *IEEE Journal of Selected Topics in Signal Processing*, 9(3), 409–421.

Wang, C., Hou, S., Wen, C., Gong, Z., Li, Q., Sun, X., & Li, J. (2018). Semantic line framework-based indoor building modeling using backpacked laser scanning point cloud. *ISPRS Journal of Photogrammetry & Remote Sensing*, 143, 150–166.

Wang, R., Xie, L., & Chen, D. (2017). Modeling indoor spaces using decomposition and reconstruction of structural elements. *Photogrammetric Engineering & Remote Sensing*, 83(12), 827–841.

Wonka, P., Wimmer, M., Sillion, F., & Ribarsky, W. (2003). Instant architecture. *ACM Transactions on Graphics*, 22(3), 669–677.

Xiao, J., & Furukawa, Y. (2014). Reconstructing the world’s museums. *International Journal of Computer Vision*, 110, 243–258.

Xiong, X., Adan, A., Akinci, B., & Huber, D. (2013). Automatic creation of semantically rich 3D building models from laser scanner data. *Automation in Construction*, 31, 325–337.

Zlatanova, S., Rahman, A. A., & Shi, W. (2004). Topological models and frameworks for 3D spatial objects. *Computers & Geosciences*, 30(4), 419–428.

**How to cite this article**: Nikoohemat S, Diakité AA, Lehtola V, Zlatanova S, Vosselman G. Consistency grammar for 3D indoor model checking. *Transactions in GIS*. 2021;25:189–212. [https://doi.org/10.1111/tgis.12686](https://doi.org/10.1111/tgis.12686)