This is a pre-print for personal use only.
The paper will be published in Springer's Lecture Notes in Geoinformation and Cartography (LNG&C) series in a book titled “Recent Advances in 3D Geoinformation Science - Proceedings of the 18th 3D GeoInfo Conference”.
This article was selected based on the results of a double-blind review of the full paper.

Link to the original publication
https://link.springer.com/chapter/10.1007/978-3-031-43699-4_30

Identification and Interpretation of Change Patterns in Semantic 3D City Models

Son H. Nguyen and Thomas H. Kolbe

Technical University of Munich, Germany
(son.nguyen, thomas.kolbe)@tum.de

Abstract. Urban Digital Twins have received significant attention in recent years due to their economic and research importance. Although many definitions exist, the general consensus agrees on a continuous two-way data flow between a physical entity and its virtual counterpart in a digital twin. In the context of smart cities and semantic 3D city models, however, no major breakthrough in realizing such complex change detection and analysis systems has yet been achieved. While several methods for change detection in semantic 3D city models have been proposed, the analysis of found changes, especially the identification of patterns among a large number of changes, has not been given as much attention. Without a proper handling of patterns, it is difficult to provide useful interpretation of changes with respect to stakeholders. Therefore, this research proposes a framework to define, detect and decipher complex semantic change patterns in semantic 3D city models. The approach provides a central rule network to describe aggregation relations between changes as well as methods to identify and capture detected change patterns directly in the graph representation of a city model.

Keywords: semantic networks, change patterns, urban digital twins

Pre-print

1 Introduction

Digital Twins have in recent years become a major driving force behind many technological and economic progresses worldwide. In the context of smart cities and urban development, a digital twin of a city - an Urban Digital Twin - is a comprehensive framework for organizing and harnessing the many diverse aspects of a city, ranging from physical components and logical structure to partaking actors and processes. Urban Digital Twins are created for specific purposes. The goal is to gain essential insights into the state of the city and its development by observing and analyzing the information available in its corresponding digital twin, thereby supporting both regular operations and critical urban planning and decision-making. Despite the many definitions of digital twins, the general consensus agrees that a digital twin must involve a *physical entity*, a corresponding *digital representation* and a *continuous feedback loop* between the physical and digital entity. This means that changes in the real world must be reflected on the digital side, and vice versa, as illustrated in Figure 1. Such

This is a pre-print for personal use only.
The paper will be published in Springer's Lecture Notes in Geoinformation and Cartography (LNG&C)
series in a book titled “Recent Advances in 3D Geoinformation Science - Proceedings of the 18th 3D GeoInfo Conference”.
This article was selected based on the results of a double-blind review of the full paper.

systematic two-way synchronization is however difficult to implement and scale efficiently [5, 15]. According to a recent survey among international experts [7], updating (including change detection, version management and efficiency) was identified as one of the most commonly cited technical challenges of Urban Digital Twins, for which no known complete solution has yet been achieved. As a result, many current smart city deployments, especially in 3D city modelling, often replace old datasets with newer versions, which not only wastes time and computational resources, but also ignores any meaningful development that may have materialized in the datasets during the recorded time period.

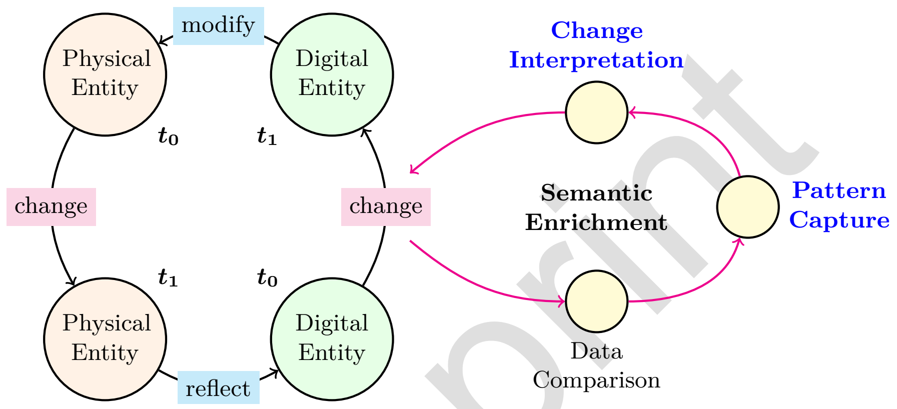


Fig. 1: An overview of an Urban Digital Twin (left) and its Semantic Enrichment process (right) between two temporal versions of the digital entity. The focus of this research is on the Pattern Capture and Change Interpretation (blue).

Pre-print

Therefore, acquiring efficient methods not only to detect changes but also to assess and understand the results is of significant advantage. Moreover, the gained knowledge provides a valuable insight into the semantic interrelations among changes. Thus, the general goal is to increase the semantic usefulness and readiness of the model. The modified digital entity can therefore be further enriched with semantic contents in a multi-levelled process called the *Semantic Enrichment*, as illustrated on the right-hand side of Figure 1. This process consists of three consecutive levels listed in ascending semantic order as follows:

1. **Data Comparison**: Snapshots recorded at different timestamps are matched and compared. This step is directly linked to the data storage and has the lowest level of semantic detail.
2. **Pattern Capture**: Based on the changes detected in the previous level, patterns of changes are captured to provide additional semantic context on the data. The semantic detail of this level is thus increased.
3. **Change Interpretation**: Combining the results from previous levels, comprehensive semantic interpretation of changes can be produced. This level has the highest concentration of semantic contents.

This is a pre-print for personal use only.
The paper will be published in Springer's Lecture Notes in Geoinformation and Cartography (LNG&C) series in a book titled “Recent Advances in 3D Geoinformation Science - Proceedings of the 18th 3D GeoInfo Conference”.
This article was selected based on the results of a double-blind review of the full paper.

Change detection and version control in general is not new and has been discussed by several studies in recent years, such as in the field of semantic 3D city modelling [10, 14]. The majority of these studies however focused solely on the detection of literal modifications in the data (Semantic Enrichment, level 1), without further considering their semantic context. Most changes however also have a meaning and purpose relevant to specific groups of stakeholders [8]. A graph-based framework was introduced [9] to better model and understand the interrelations between changes and stakeholders, as well as the correlations and reasoning of changes in semantic 3D city models (Semantic Enrichment, level 3). An open question, however, remains as to how patterns of changes can be efficiently captured (Semantic Enrichment, level 2). The current approach in many smart city deployments is to create database queries for each pattern and execute them in any necessary order on an ad hoc basis. This not only requires expert knowledge on the structure of the underlying databases, but may further lead to unwanted scheduling and efficiency problems, especially if rules are dependent on each other (forming a “pattern” of patterns).

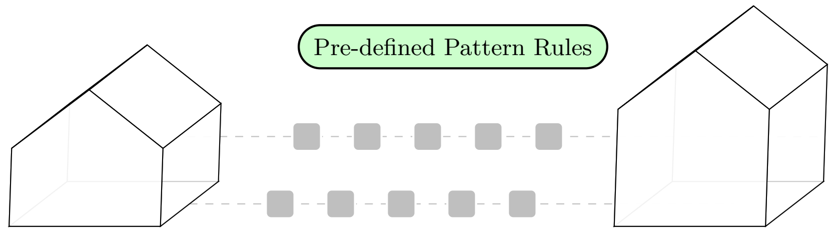
This research explores the changes that occurred in the digital representation of a city within a digital twin. The goal is to uncover patterns and underlying meanings or reasoning that these changes have on the real city. To achieve this, this paper proposes a framework to define, detect and decipher complex change patterns in semantic 3D city models, namely (1) a single rule network to define all aggregation semantic rules, and (2) methods to detect patterns based on given rules. The proposed methods were developed for Urban Digital Twins represented by semantic 3D city models in CityGML as one of their core components but can also be applied to other fields with similar use cases and semantic object representations, such as in the BIM field.

Pre-print

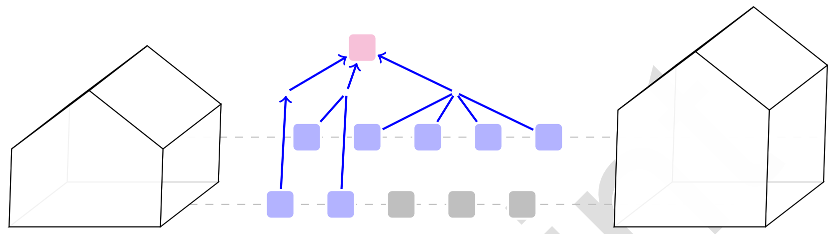
2 Foundations and Related Work

Since semantic 3D city models are structured as graphs [1, 3, 9], most matching methods also naturally store changes in compatible data structures. This leads to the pattern matching problem of graphs. Figure 2 gives an overview of the methods introduced in this research, where (1) changes detected between graph representations of two semantic 3D city models are first matched against given pattern rules (Figure 2a), (2) additional interpretation nodes are then created and attached to the source nodes in the graph database (Figure 2b), which (3) can ultimately be utilized to derive meaningful interpretations about the changes in the data (Figure 2c). To achieve this, four key technical requirements of matching change patterns in semantic 3D city models must be fulfilled, namely Dynamic Aggregation, On-the-fly Typing, Origin Handling and Memory Efficiency, as described in Table 1. Thus, this research proposes several concepts, which, to a degree, have their roots in Rete networks, Petri nets and graph transformation systems. This section shall therefore provide a brief introduction to some the most relevant components of each concept.

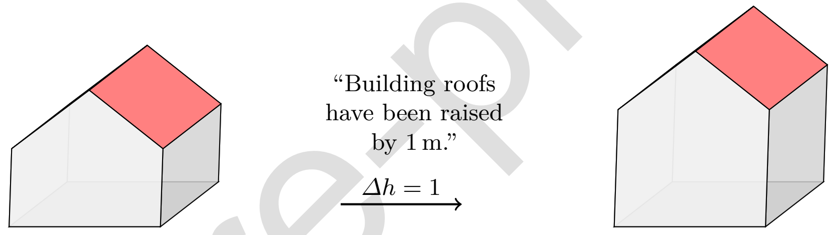
This is a pre-print for personal use only.
The paper will be published in Springer's Lecture Notes in Geoinformation and Cartography (LNG&C)
series in a book titled "Recent Advances in 3D Geoinformation Science - Proceedings of the 18th 3D GeoInfo Conference".
This article was selected based on the results of a double-blind review of the full paper.



(a) Detected changes (gray) between graph representations of two given buildings are matched against pre-defined pattern rules (green).



(b) A pattern (red) indicating a possible roof raise of a building has been detected with moved roofs and resized walls upwards by 1 m.



(c) Interpretations are derived from the detected pattern.

Fig. 2: An illustration of the pattern matching process. Input changes (gray) belonging to a pattern (blue) are aggregated into new interpretation nodes (red).

Pre-print

2.1 Rete Networks

The Rete match algorithm is an efficient method for matching a large number of patterns against a large number of objects [4]. The algorithm was originally developed for production system interpreters. A typical production system consists of (1) an unordered collection of conditional statements, (2) a global **working memory** (or database) holding temporary data, and (3) a rule interpreter that can assess rule conditions. To avoid iteration over input, the Rete algorithm stores matched objects in each corresponding rule. When a new element is inserted or an existing element is removed from the shared working element, affected rules are notified and their list of stored objects is updated accordingly.

This is a pre-print for personal use only.
The paper will be published in Springer's Lecture Notes in Geoinformation and Cartography (LNG&C) series in a book titled "Recent Advances in 3D Geoinformation Science - Proceedings of the 18th 3D GeoInfo Conference".
This article was selected based on the results of a double-blind review of the full paper.

Table 1: Key requirements for change patterns in semantic 3D city models.

Requirement	Description
Dynamic Aggregation	Most aggregation rules specify a static number of input objects. In many important use cases, however, input quantity is not known before execution. Therefore, dynamic aggregation must be enabled.
On-the-fly Typing	As input for pattern matching, changes of city objects may be given in any order. To avoid repeated iteration, the system must be able to identify changes based on their types and attributes in real-time.
Origin Handling	In addition to types, changes must be distinguished by their semantic context. For example, when interpreting changes to a building, only changes relevant to that specific building are considered.
Memory Efficiency	Graph representations of cities may become very large, leading to a potentially overwhelming number of produced changes. This requires efficient algorithms with regard to memory consumption.

To avoid repeated iteration over rules, a directed acyclic graph representation of rules is used for pattern matching. This graph is called the Rete network [4].

One major advantage of the Rete match algorithm is its processing speed due to the employed working memory containing temporary data of each pattern during execution. The working memory "memorizes" previously read input objects as well as intermediate results in "buckets", which, if full, shall trigger corresponding actions. Thus, the working memory allows both on-the-fly type checking and on-demand reactivation of pending rules. Moreover, the Rete algorithm excels in use cases, where input is a stream consisting of randomly ordered and differently typed objects, on which a large number of insertion and deletion operations are performed. The performance of the Rete algorithm largely depends on the implementation of its working memory. In general however, in worst-case scenarios, the original Rete algorithm may store all temporary data in main memory, degenerating memory efficiency. The methods proposed in this research employ an extended version of the afore-mentioned working memory while avoiding excessive memory consumption.

Pre-print

2.2 Petri Nets

Petri nets were first proposed to model parallel and distributed systems [11]. A Petri net is a bipartite graph consisting of places and transitions. The partitions are connected via directed arcs. Places may contain **tokens**, which can travel between neighbouring places. In rule-based systems, places semantically represent the current state or conditions of rules, while transitions represent actions. If a sufficient number of tokens exist at a place, its outgoing transition(s) shall be triggered, consuming input tokens and producing new ones. The number of tokens consumed and produced is dictated by the transitions' weights.

This is a pre-print for personal use only.
The paper will be published in Springer's Lecture Notes in Geoinformation and Cartography (LNG&C)
series in a book titled “Recent Advances in 3D Geoinformation Science - Proceedings of the 18th 3D GeoInfo Conference”.
This article was selected based on the results of a double-blind review of the full paper.

Petri nets have strong scalability and modelling potential in rule-based applications. The use of tokens is well-suited for describing many aggregation rules of changes in semantic 3D city models. Petri nets can be represented mathematically as both graphs and matrices. However, tokens in classical Petri nets have neither an attribute, a type, nor an origin, and are thus indistinguishable. In contrast, changes in semantic 3D city models are typed and attributed. Therefore, this research employs a specialized Petri net capable of distinguishing tokens based on their types, attributes and semantic context.

2.3 Graph Transformation Systems

Due to the graph-based nature of semantic 3D city models, matching their change patterns can also be considered a use case of graph transformation. Graph transformation was first proposed as a graph grammar for rule-based rewriting of non-linear data structures [6,12,13]. In graph transformation, rules are defined using type graphs and instance graphs. A **type graph** defines the conceptual model of object classes, while an **instance graph** is a snapshot containing concrete values and structure prescribed by the corresponding type graph.

Graph transformation systems are a powerful tool for handling complex semantic structures. Type-enabled graph transformation can utilize hidden context information of objects, enabling more complex analyses. However, graph transformation employs graph isomorphism, whose complexity is neither polynomial nor known to be NP-complete [16]. In addition, the structure of the type and instance graphs given by the transformation rules must be known, but this information is often unknown until execution. The methods proposed in this research employ a simplified approach to graph transformation. Instead of relying on graph isomorphism, node types and semantic positions in graphs are used to improve the runtime efficiency of the pattern matching process.

Pre-print

3 Defining Pattern Rules

Based on the strengths and limitations of the concepts discussed in Section 2, this research proposes a compact aggregative rule network to define graph-based rules for change patterns in semantic 3D city models.

3.1 Definitions

A **content network** is a directed and attributed graph representation of a semantic 3D city model, where all information about the city model is stored. Nodes in a content network are called *content nodes*. An example of a content network is shown in Figure 3. A content network contains graph representation of both the old and new dataset. Detected changes are attached to both graphs, but for visual clarity, this study only shows one of these graph representations with connected changes, as shown later in Figure 5.

This is a pre-print for personal use only.
The paper will be published in Springer's Lecture Notes in Geoinformation and Cartography (LNG&C)
series in a book titled "Recent Advances in 3D Geoinformation Science - Proceedings of the 18th 3D GeoInfo Conference".
This article was selected based on the results of a double-blind review of the full paper.

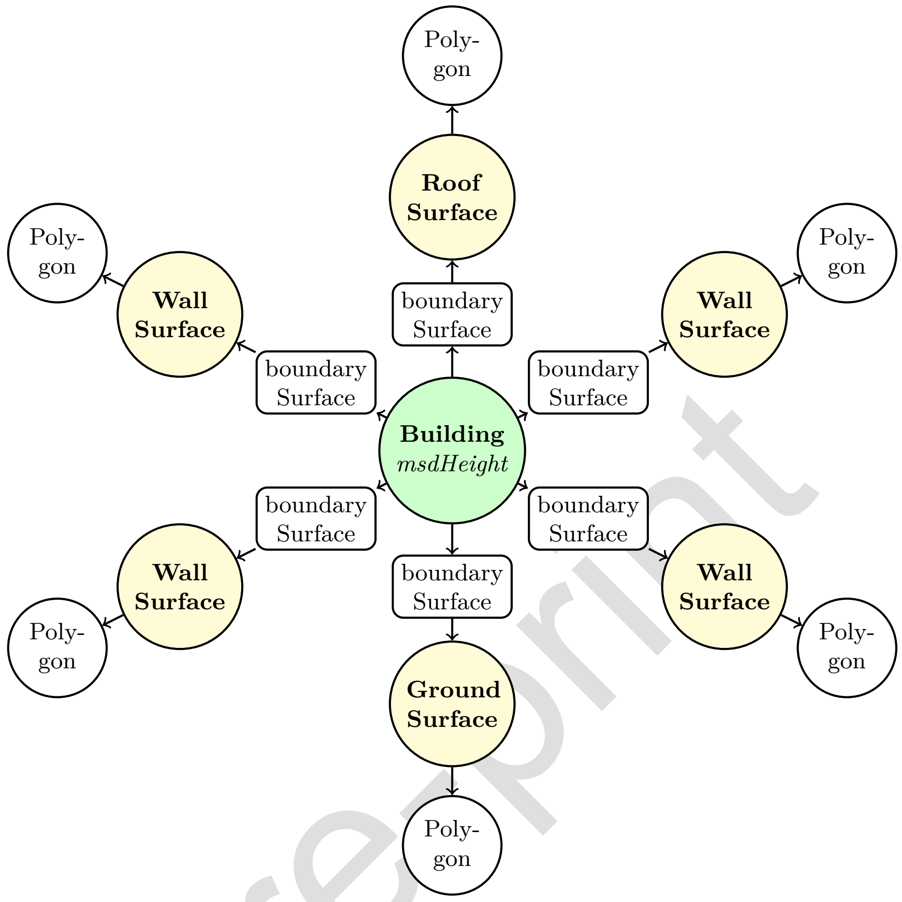


Fig. 3: A content network for a simplified building model, representing multiple 1-to-n relations between a building and its boundary surfaces. This is directly derived from a CityGML dataset as described in previous publications [10].

Preprint

A **rule network** is a directed acyclic and attributed graph that allows all rules for pattern matching to be defined in one place. It is a type graph capable of describing the characteristic behaviours of different types of changes, allowing the dependencies between rules to be explicitly captured. Nodes in a rule network are called *rule nodes* and can represent both literal detected changes in the data and interpreted changes later on. Figure 4 shows an example of a rule network.

Each rule node is assigned a type corresponding to the changes it represents. Rule nodes are connected using directed rule edges. A rule edge has three components: the next content type, a list of conditions and a weight. The content type acts as a "checkpoint" for the rule interpreter to navigate within the content network. For example, the content nodes associated with the rule edge between *PolygonResized* and *WallResized* are wall surfaces in the content network. Con-

This is a pre-print for personal use only.
The paper will be published in Springer's Lecture Notes in Geoinformation and Cartography (LNG&C) series in a book titled "Recent Advances in 3D Geoinformation Science - Proceedings of the 18th 3D GeoInfo Conference".
This article was selected based on the results of a double-blind review of the full paper.

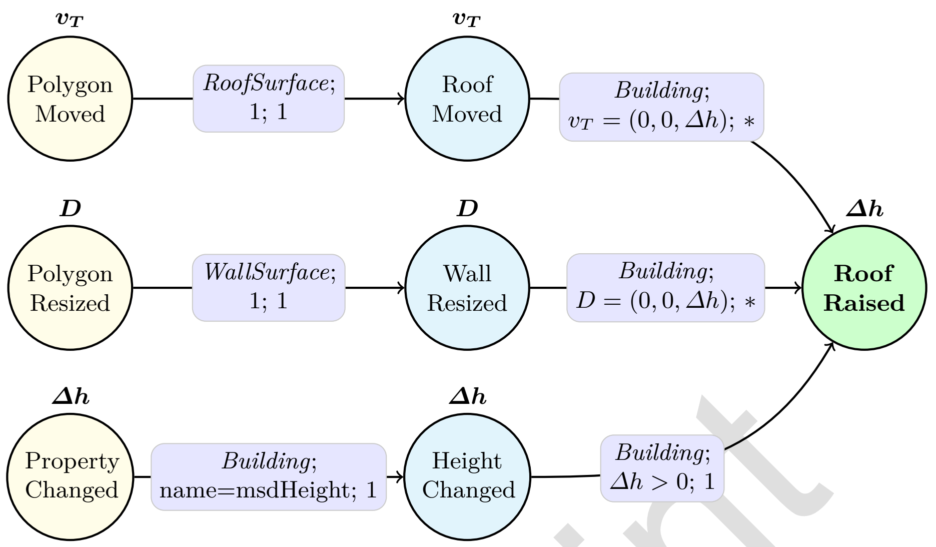


Fig. 4: An example of a rule network for the content network shown in Figure 3. Each edge is notated with ⟨*next content type*; *conditions*; *weight*⟩.

Pre-print

ditions are logical expressions evaluated against properties in the corresponding change nodes. Values can be named and shared as variables across converging rule edges. New properties in previous changes are forwarded to the next one as part of the knowledge gained through the interpretation process. If no condition is needed, the value 1 is used. All conditions must be fulfilled to trigger an aggregation. For example, the rule edge between *RoofMoved* and *RoofRaised* dictates that the translation vector v_T must have no x and y component, while the z value is named Δh to represent any real value. This variable is reused in other rule edges. Moreover, as shown in the rule edge between *HeightChanged* and *RoofRaised*, additional constraints can be introduced to limit property values. Parametrized conditions are evaluated during runtime at the next rule node by matching property values of collected changes, as explained in Section 4.

The weight of a rule edge dictates the number of occurrences of changes corresponding to the previous rule node required for the creation of the next interpretation node. If a rule node has multiple incoming edges, it can only be activated when all previous rule nodes have collected a sufficient number of changes. For example, the rule node *RoofRaised* can only be activated if all required occurrences of *RoofMoved*, *WallResized* and *HeightChanged* exist. The weight can be assigned a specific value or a placeholder $*$ for an unknown value. For instance, since each building can have a different number of wall surfaces, the weight of the rule edge between *WallResized* and *RoofRaised* is first initialized with $*$. This placeholder is updated with a concrete value by the rule interpreter at runtime. The interpreter searches "upwards" in the content network for the next content node that matches the content type given in the rule edge, then

This is a pre-print for personal use only.
The paper will be published in Springer's Lecture Notes in Geoinformation and Cartography (LNG&C)
series in a book titled “Recent Advances in 3D Geoinformation Science - Proceedings of the 18th 3D GeoInfo Conference”.
This article was selected based on the results of a double-blind review of the full paper.

traverses all paths “downwards” until a content node specified by the previous rule node is found. The placeholder is then replaced with the number of reached paths. For instance, while processing the rule edge between *WallResized* and *RoofRaised*, the interpreter searches for a *Building* node, as its type is specified as the next content type of the rule edge. From this *Building* node, all paths to *Wall* nodes are counted. A comparison between this rule network and the concepts previously mentioned in Section 2 is summarized in Table 2.

Table 2: A comparison between the proposed rule network and related concepts with respect to the key requirements described in Table 1.

	Dynamic Aggregation	On-the-fly Typing	Origin Handling	Memory Efficiency
Rete networks[1]	×	●	×	×
Petri nets[1]	×	◐	×	●
Graph transformation[1]	×	◐	●	×
Proposed rule network	●	●	●	●

× Not applicable ◐ Applicable if typing is enabled ● Applicable
[1] Original publication is considered. Some variants may differ.

Pre-print

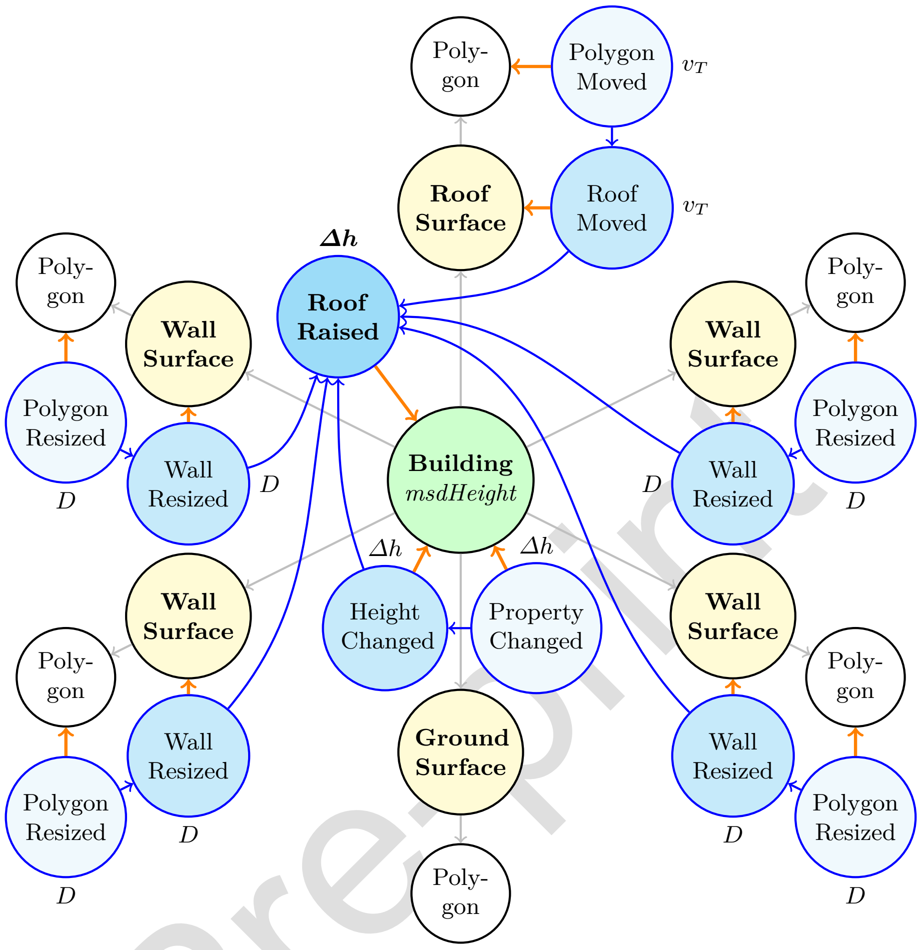
4 Detecting Change Patterns

Given a rule network, change patterns in a content network can be matched. The detected patterns are represented as additional interpretation nodes attached to their corresponding content nodes. For instance, a *PolygonMoved* node is attached to a source *Polygon* node, while a *RoofRaised* is attached to a *Building* object. The results of the pattern matching process are illustrated in Figure 5.

The method used to detect change patterns during the pattern matching process is summarized in Figure 6 and described in Algorithm 1. The algorithm employs a FIFO (First In, First Out) queue that functions like a conveyor belt in an assembly line. The queue contains all literal and interpreted changes, removes the first element for processing, and stores the aggregated changes at the end of the queue until all elements have been processed. The algorithm aims to aggregate input changes into new, higher-level semantic changes. Changes are aggregated if they satisfy all four criteria: *type*, *origin*, *condition* and *count check* required by given rules. The type and condition check can be determined based on the type and attributes available in each change. To conduct the remaining count and origin check, the method employs two of its key concepts, namely an **aggregative memory** and a graph-based **semantic context**.

When a rule is applied, its associated next content node is initialized with a memory to store crucial aggregation information about the current and maximum number of changes collected per type. Each time the content node encounters a change required by its rule, it increases its count of the object type by one

This is a pre-print for personal use only.
The paper will be published in Springer's Lecture Notes in Geoinformation and Cartography (LNG&C)
series in a book titled "Recent Advances in 3D Geoinformation Science - Proceedings of the 18th 3D GeoInfo Conference".
This article was selected based on the results of a double-blind review of the full paper.



Pre-print

Fig. 5: An example of the results (blue) of the pattern matching process based on the content network shown in Figure 3 and the rule network shown in Figure 4. For visual clarity, *boundarySurface* nodes are omitted. Interpretation connections are shown in orange. In this example, the algorithm starts with the changes *PolygonMoved*, *PolygonResized* and *PropertyChanged* at the lowest level and gradually propagates "upwards" in the content network until a content node with a wanted type is encountered, such as the path *PolygonResized* → *Polygon* → *WallSurface*, where *WallSurface* is required by *WallResized*. Once all criteria have been fulfilled, a new interpretation node *WallResized* is created. The propagation proceeds until a *CityModel* node (not shown) is reached, which is associated with global or systematic change patterns. Thus, the pattern matching algorithm is an aggregation process, where changes of lower semantic levels are aggregated to produce new changes of higher semantic levels.

This is a pre-print for personal use only.
The paper will be published in Springer's Lecture Notes in Geoinformation and Cartography (LNG&C)
series in a book titled “Recent Advances in 3D Geoinformation Science - Proceedings of the 18th 3D GeoInfo Conference”.
This article was selected based on the results of a double-blind review of the full paper.

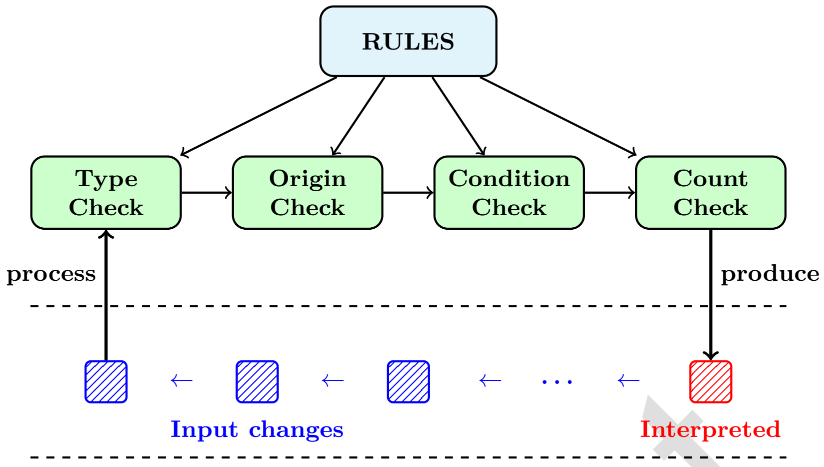


Fig. 6: The pattern matching algorithm uses a queue to process all literal and interpreted changes (blue). Changes that pass all checks are aggregated into a new interpreted change (red) and added to the queue for further processing.

Algorithm 1: Pattern matching algorithm

Data : A content network N_C and a rule network N_R
A queue Q initialized with detected changes connected with N_C
Result: Interpreted changes connected with N_C

```
1  while $Q$ is not empty do
2      $q \leftarrow$ dequeue $Q$
3      $r \leftarrow$ find rule in $N_R$ that accepts $q.type$, or else continue
4      $n \leftarrow$ find node in $N_C$ of $r.type$ starting from $q$, or else continue
5      if $n.memory$ does not exist then
6          $n.memory \leftarrow$ initialize memory
7      end
8      if $r.conditions$ are all fulfilled then
9          if $n$ accepts $q.origin$ then
10             increase $n.memory.count\,(q.type)$ by 1
11             store reference to $q$ in $n.memory.refs$
12         end
13         if $n.memory.count\,(t) = max, \forall t \in n.memory.types$ then
14             $m \leftarrow$ create interpretation node representing the change pattern
15             initialize $m.type$, $m.origin$ and store $q.properties$ in $m$
16             connect $n$ and all stored references in $n.memory.refs$ with $m$
17             enque $m$ into $Q$
18         end
19     end
20 end
```

Pre-print

This is a pre-print for personal use only.
The paper will be published in Springer's Lecture Notes in Geoinformation and Cartography (LNG&C)
series in a book titled "Recent Advances in 3D Geoinformation Science - Proceedings of the 18th 3D GeoInfo Conference".
This article was selected based on the results of a double-blind review of the full paper.

until this value reaches a maximum. The count check is considered complete, if the number of all change occurrences per type reaches a maximum. This memory can be implemented as a collection of key-value pairs. Listing 1 gives an example of the memory of the content node *Building* shown in Figure 5.

```
{
  "variables" : [ "deltaH" ],
  "rules" : [
    {
      "rule_type"    : "RoofMoved",
      "count_value"  : 1,
      "max_value"    : 1,
      "properties"   : { "vT" : "(0, 0, 0.969)" }
    },
    {
      "rule_type"    : "WallResized",
      "count_value"  : 2,
      "max_value"    : 4,
      "properties"   : { "D" : "(0, 0, 0.969)" }
    },
    {
      "rule_type"    : "HeightChanged",
      "count_value"  : 1,
      "max_value"    : 1,
      "properties"   : { "deltaH" : 0.969 }
    }
  ]
}
```

Listing 1: An excerpt from the memory of a *Building* node shown in Figure 5 that has collected 1 instance of *RoofMoved*, 2 *WallResized* and 1 *HeightChanged*.

Pre-print

The use of memory is similar to that of Rete networks [4] and can eliminate repeated iteration by processing changes on the fly. However, in contrast to classical Rete networks, the proposed method does not store entire objects in its memory. Instead, the algorithm first identifies objects based on their types and attributes, then updates the number of their occurrences accordingly. Moreover, at the start, the memory is empty and only expanded as new rules and changes are encountered. This avoids the worst-case memory consumption of Rete networks, where the working memory could hold all input objects at runtime.

Another key concept of the pattern matching algorithm is the ability to distinguish objects based on their semantic context or origin in the network. An origin of a node is a set containing itself and all its ancestors in a directed acyclic network. In an input sequence of changes, such as $(r, w, w, r, r, h, \ldots)$, where the change types are *RoofMoved*, *WallResized* and *HeightChanged*, it is unknown

This is a pre-print for personal use only.
The paper will be published in Springer's Lecture Notes in Geoinformation and Cartography (LNG&C) series in a book titled “Recent Advances in 3D Geoinformation Science - Proceedings of the 18th 3D GeoInfo Conference”.
This article was selected based on the results of a double-blind review of the full paper.

whether these changes refer to the boundary surfaces of the same building or several different ones. Therefore, the origin check is employed as an additional guard to ensure changes are aggregated correctly. This is related to the problem of finding the lowest common ancestor [2], as illustrated in Figure 7.

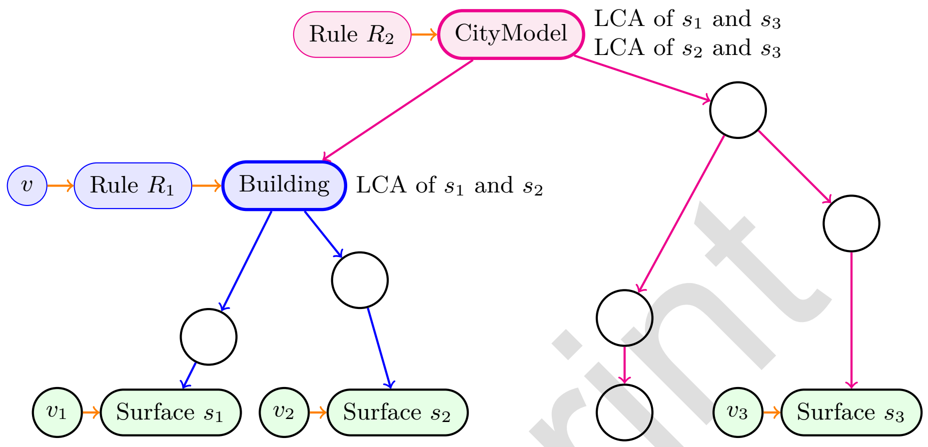


Fig. 7: An example of changes v_1, v_2 and v_3 associated with surfaces s_1, s_2 and s_3 (green). The lowest common ancestor (LCA) of s_1 and s_2 is located at the *Building* node (blue), while the LCA of s_1 and s_3, as well as s_2 and s_3, is located at *CityModel* (red). Thus, rule R_1 accepts v_1 and v_2, while rule R_2 accepts both R_1 and v_3. The interpretation of higher levels only considers the interpretation of objects on the next lower level, or literal changes if no interpretation is available.

Pre-print

5 Application Examples

In the following experiments, change patterns between two CityGML documents in Level of Detail (LoD) 2 are matched. The datasets used are excerpts, each containing 44 buildings from a selected area of Hamburg, Germany. These datasets were recorded in 2016 and 2022 and are provided publicly by the city.[1] All data are managed within a single graph database, which includes the content network of both the city models and their changes, as well as rule networks for detecting change patterns. The graph database Neo4j is used. The rule interpreter shown in Algorithm 1 can be implemented using Neo4j's Cypher query language or its Java API. A Java implementation of the rule interpreter can be exported as a user-defined procedure, which can then be invoked directly from Cypher.

Global patterns have been observed based on a total number of 1049 changes on thematic attributes distributed over roof surfaces, building parts and buildings (see Table 3). A visualization of these patterns can be found in Figure 8.

[1] https://metaver.de

This is a pre-print for personal use only.
The paper will be published in Springer's Lecture Notes in Geoinformation and Cartography (LNG&C)
series in a book titled "Recent Advances in 3D Geoinformation Science - Proceedings of the 18th 3D GeoInfo Conference".
This article was selected based on the results of a double-blind review of the full paper.

Table 3: A summary of thematic changes between 2016 and 2022.

	Property name	Description	B		P		R	
Insert	*Dachhoehe*[1]	Roof height	9	×	25	×	0	×
	Firsthoehe[1]	Ridge height	21	×	63	●	0	×
	Geo.typ2DRef.[1]	Source geometry type	44	●	0	×	0	×
	Grundrissaktualitaet[1]	Ground plan update	44	●	63	●	0	×
	Hauskoordinate[1]	Building coordinates	5	×	0	×	0	×
	Qu.Dacherkennung[1]	Roof detection quality	21	×	63	●	0	×
	Traufhoehe[1]	Eaves height	21	×	63	●	0	×
	tridicon_Dachform[1]	Tridicon roof shape	21	×	63	●	0	×
Update	*gmlid*	Identifier	44	●	63	●	81	●
	creationDate	Modification date	44	●	63	●	0	×
	measuredHeight	Measured height	20	×	63	●	0	×
	function	Function	14	×	0	×	0	×
	roofType	Roof type	1	×	7	×	0	×
	Datenqu.Dachhoehe[1]	Source roof height	2	×	10	×	0	×
	Flaechengroesse[1]	Surface area	0	×	0	×	33	×
	Flaechenneigung[1]	Surface inclination	0	×	0	×	32	×
	Flaechenrichtung[1]	Surface orientation	0	×	0	×	2	×
	Gemeindeschluessel[1]	Municipality key	44	●	0	×	0	×

[1] Generic string attribute **B** Building **P** Building part **R** Roof surface
× Local or clustered change pattern ● Global change pattern

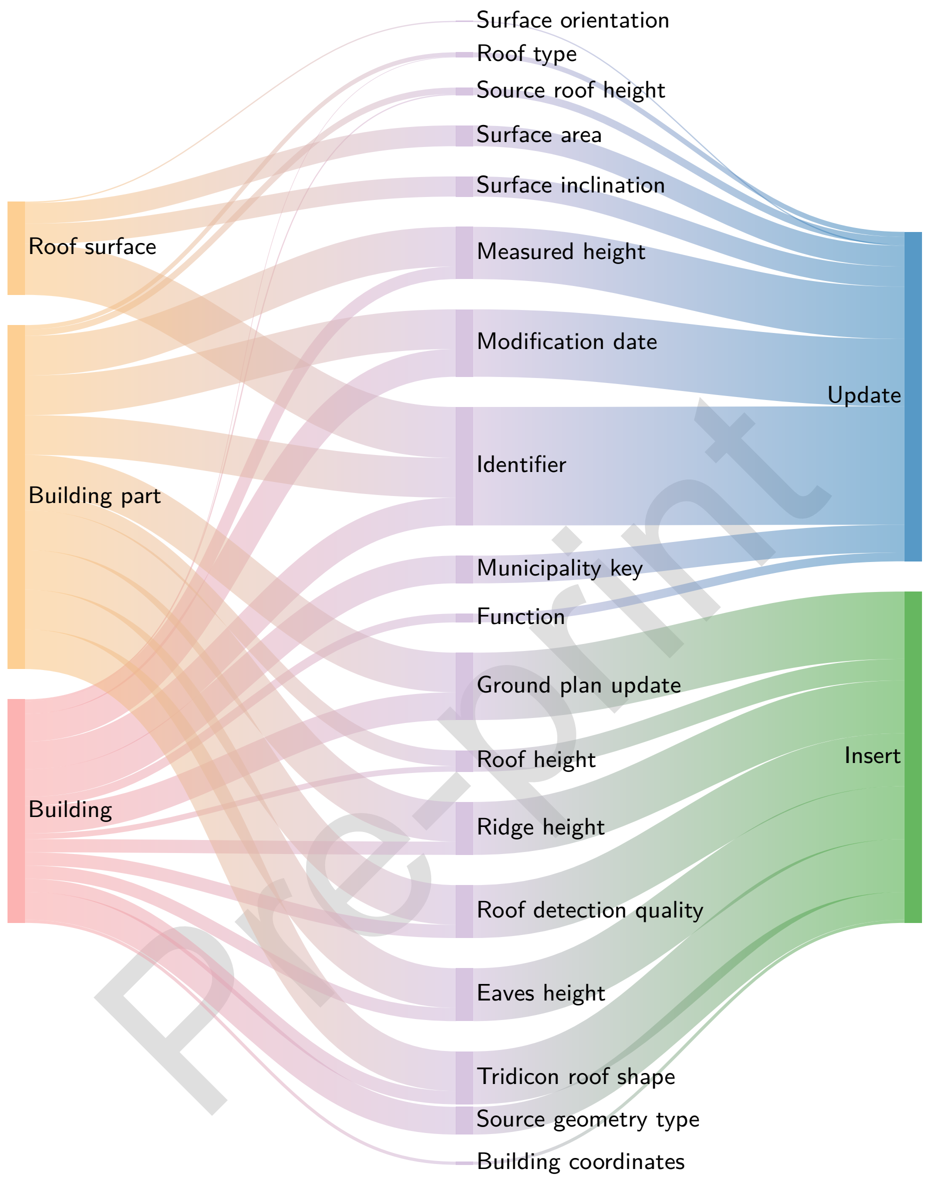
Pre-print

Out of 638 roof, wall, and ground surfaces, 552 have been observed to either be moved or changed in size. Translation is detected by calculating the offset vector between geometries, while size changes are measured by deviations in the surfaces' 3D bounding boxes. Notably, all 134 translation and 394 (94 %) of all size changes occurred vertically, with translation offsets ranging from -0.957 m (downwards) to 1.895 m (upwards), and resize margins between -1.836 m (height decrease) and 2.288 m (height increase). These changes are significantly reduced to a few interpretations in the following three steps.

Firstly, by extending the rule network given in Figure 4, translation and resize changes of the same margin for all roof, wall or ground surfaces of a building are aggregated into interpretation nodes attached to buildings (see Table 4).

Secondly, the aggregated nodes are further combined into tuples. Since each building is bounded by three types of surfaces and each interpretation node indicates a surface translation, resize, or none of the above, there exist 27 com-

This is a pre-print for personal use only.
The paper will be published in Springer's Lecture Notes in Geoinformation and Cartography (LNG&C)
series in a book titled “Recent Advances in 3D Geoinformation Science - Proceedings of the 18th 3D GeoInfo Conference”.
This article was selected based on the results of a double-blind review of the full paper.



Pre-print

Fig. 8: An overview of change patterns detected in the thematic data of the Hamburg datasets between 2016 and 2022. Changes are categorized by their functions shown on the right column (inserted and updated properties). The left column represents the number of change occurrences grouped by feature types, where most modifications occurred. The property names are given in the middle.

This is a pre-print for personal use only.
The paper will be published in Springer's Lecture Notes in Geoinformation and Cartography (LNG&C)
series in a book titled "Recent Advances in 3D Geoinformation Science - Proceedings of the 18th 3D GeoInfo Conference".
This article was selected based on the results of a double-blind review of the full paper.

Table 4: A summary of surface-based changes between 2016 and 2022.

Surface changes	**Aggregated**	**Found patterns of same margin**[1,2]
2 Ground size changes	2 (100%)	0 Building with all grounds resized
6 Wall translations	4 (67%)	1 Building with all walls moved
31 Roof size changes	25 (81%)	5 Buildings with all roofs resized
49 Roof translations	49 (100%)	11 Buildings with all roofs moved
79 Ground translations	79 (100%)	38 Buildings with all grounds moved
385 Wall size changes	204 (53%)	12 Buildings with all walls resized

[1] Only buildings in which *all* surfaces of a given type have been moved or resized by the same amount are considered.
[2] A building may be counted multiple times, up to a maximum of three occurrences, with one count for each boundary surface type.

Pre-print

binations to form an interpretation tuple for each building. A tuple RWG denotes a consistent translation (T), resize (S) or none (X) for all roofs (R, first position), walls (W, second) and grounds (G, third) of a building.

Thirdly, the translation and resize margins stored in the interpretation tuples are studied to reveal correlations between geometric changes. For example:

1. All roof, wall and ground surfaces of one building marked with TTT have been vertically shifted by the same offset -0.049 m, meaning the entire building has been moved downwards by that amount.
2. A common pattern has been observed in all ten buildings marked with TST, where each building's roof and ground surfaces were moved by Δz_r and Δz_g, and all wall surfaces resized by Δz_w, such that $\Delta z_w + \Delta z_g = \Delta z_r$ (see Table 5). For example, the roofs of building B1 have been raised by approximately 1 m, supported by an equivalent increase in wall height and a small downward translation of ground surfaces. This information is useful for stakeholders such as urban planners, energy consultants, and city mayors, as it may indicate that a building has been expanded by an additional storey, potentially increasing the amount of available living space. In contrast, buildings with small deviations, such as B8, may be of interest to data brokers and quality managers.
3. Of the six buildings marked with XXX, two have remained geometrically unchanged, as none of their boundary surfaces has been translated or resized.

Therefore, the interpretation nodes produced by the pattern matching process are crucial in providing a deeper understanding of the interrelationships between detected changes in the datasets. Further information on the implementation and additional examples can be found online (work in progress).[2]

[2] https://github.com/tum-gis/citymodel-compare

This is a pre-print for personal use only.
The paper will be published in Springer's Lecture Notes in Geoinformation and Cartography (LNG&C) series in a book titled “Recent Advances in 3D Geoinformation Science - Proceedings of the 18th 3D GeoInfo Conference”.
This article was selected based on the results of a double-blind review of the full paper.

Table 5: An overview of the translation offsets Δz_r and Δz_g of all roof and ground surfaces, resize margins Δz_w of all wall surfaces, and difference in each building's measured heights Δh (in cm). The correlations $\Delta z_w + \Delta z_g = \Delta z_r$ and $\Delta h = \Delta z_w$ apply in all ten buildings (B1-10) marked with TST.

	B1	**B2**	**B3**	**B4**	**B5**	**B6**	**B7**	**B8**	**B9**	**B10**
Δz_r	96.9	16.1	12.0	9.4	−2.8	−4.5	−5.7	−5.8	−6.0	−7.4
Δz_w	102.5	25.1	16.5	12.0	10.8	11.4	−3.2	0.8	5.0	18.8
Δz_g	−5.6	−9.0	−4.5	−2.6	−13.6	−15.9	−2.5	−6.6	−11.0	−26.2
Δh	102.5	25.1	16.5	12.0	10.8	11.4	−3.2	0.8	5.0	18.8

6 Conclusion and Future Work

Based on the strengths and limitations of well-known concepts for rule-based systems, such as those of Rete networks, Petri nets and graph transformation systems, this research proposed a framework to define rules for matching change patterns in semantic 3D city models. The framework employs graph representations of semantic 3D city models, called the content networks, as a basis for all pattern detection processes. Rules are defined in a rule network, which is a type graph that can describe the characteristic behaviours and interrelations of different classes of changes in aggregative semantic patterns. By applying a rule network to a content network, change patterns can be detected and captured during the pattern matching process. The detected change patterns are represented as interpretation nodes connected to the content network, thus enabling faster retrieval and handling of the semantic patterns of changes.

The method employs aggregative rules to effectively condense a large number of changes into a few interpretation nodes that are more comprehensible to various stakeholders. Based on these gained interpretations, more efficient and complex analyses on the city's evolution can be performed. Moreover, the employed rule networks are compact yet expressive, and can be used in highly automated processes. It should be noted, however, that the framework is designed to perform aggregation and logical operations exclusively, thus requiring the provision of detected changes, as presented in our earlier publications [10], including those of complex geometric objects.

Combined with other previous related work [8, 9], this research serves as one of the last missing pieces required for a comprehensive understanding of changes in semantic 3D city models. Further experimentation and optimization of the proposed methods on large-scale real-world datasets are planned. An investigation will be conducted to determine the potential reasons for changes, their impact on city models, and how they can be represented and detected using the proposed framework.

Pre-print

This is a pre-print for personal use only.
The paper will be published in Springer's Lecture Notes in Geoinformation and Cartography (LNG&C)
series in a book titled "Recent Advances in 3D Geoinformation Science - Proceedings of the 18th 3D GeoInfo Conference".
This article was selected based on the results of a double-blind review of the full paper.

References

1. Amgad Agoub, Felix Kunde, and Martin Kada. Potential of Graph Databases in Representing and Enriching Standardized Geodata. *Dreiländertagung der DGPF, der OVG und der SGPF*, 36, 2016.
2. A. V. Aho, J. E. Hopcroft, and J. D. Ullman. On Finding Lowest Common Ancestors in Trees. *SIAM Journal on Computing*, 5(1):115–132, 1976.
3. Kerstin Falkowski and Jürgen Ebert. Graph-based Urban Object Model Processing. *City Models, Roads and Traffic (CMRT'09): Object Extraction for 3D City Models, Road Databases and Traffic Monitoring-Concepts, Algorithms and Evaluation, Paris, France*, 9:115–120, 2009.
4. Charles L. Forgy. Rete: A Fast Algorithm for the Many Pattern/Many Object Pattern Match Problem. *Artificial Intelligence*, 19(1):17–37, 1982.
5. Michael Grieves and John Vickers. *Digital Twin: Mitigating Unpredictable, Undesirable Emergent Behavior in Complex Systems*, pages 85–113. Springer International Publishing, Cham, 2017.
6. Reiko Heckel. Graph Transformation in a Nutshell. *Electronic Notes in Theoretical Computer Science*, 148:187–198, 02 2006.
7. Binyu Lei, Patrick Janssen, Jantien Stoter, and Filip Biljecki. Challenges of Urban Digital Twins: A Systematic Review and a Delphi Expert Survey. *Automation in Construction*, 147:104716, 2023.
8. Son H. Nguyen and Thomas H. Kolbe. Modelling Changes, Stakeholders and their Relations in Semantic 3D City Models. In ISPRS, editor, *Proceedings of the 16th International 3D GeoInfo Conference 2021*, volume VIII-4/W2-2021 of *ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, pages 137–144. New York University, ISPRS, 10 2021.
9. Son H. Nguyen and Thomas H. Kolbe. Path-tracing Semantic Networks to Interpret Changes in Semantic 3D City Models. In *Proceedings of the 17th International 3D GeoInfo Conference 2022*, volume X-4/W2-2022 of *ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences*. UNSW Sydney, ISPRS, 10 2022.
10. Son H. Nguyen, Zhihang Yao, and Thomas H. Kolbe. Spatio-Semantic Comparison of Large 3D City Models in CityGML Using a Graph Database. In ISPRS, editor, *Proceedings of the 12th International 3D GeoInfo Conference 2017*, volume IV-4/W5 of *ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, pages 99–106. University of Melbourne, ISPRS, 2017.
11. Carl Petri. *Kommunikation mit Automaten*. PhD thesis, TU Darmstadt, 1962.
12. John L. Pfaltz and Azriel Rosenfeld. Web Grammars. In *Proceedings of the 1st International Joint Conference on Artificial Intelligence*, IJCAI'69, page 609–619, San Francisco, CA, USA, 1969. Morgan Kaufmann Publishers Inc.
13. Terrence W. Pratt. Pair Grammars, Graph Languages and String-to-Graph Translations. *J. Comput. Syst. Sci.*, 5:560–595, 1971.
14. Richard Redweik and Thomas Becker. *Change Detection in CityGML Documents*, pages 107–121. Springer International Publishing, Cham, 2015.
15. Angira Sharma, Edward Kosasih, Jie Zhang, Alexandra Brintrup, and Anisoara Calinescu. Digital Twins: State of the Art Theory and Practice, Challenges, and Open Research Questions. *Journal of Industrial Information Integration*, 30:100383, 2022.
16. Steven S. Skiena. *The Algorithm Design Manual*. Springer Publishing Company, Incorporated, 2nd edition, 2008.

Pre-print