TOPOLOGICAL CONSISTENCY CHECKING AND REFINEMENT OF NETWORK-BASED SPACE LAYOUTS

Georg Suter, Associate Professor, suter@iemar.tuwien.ac.at
Department of Digital Architecture and Planning, Vienna University of Technology, Vienna, Austria

ABSTRACT

Network-based space layouts are fine-grained models of architectural spaces. They include rich spatial relationships between layout elements. Such information tends to be limited in existing space representations in building information modeling (BIM) systems. Network-based space layouts could be useful in the space planning and building automation domains. Their main feature is a geometric network of spatial topological relationships between layout elements, which is useful for spatial analysis. A schema for network-based space layouts has been proposed recently. However, the schema does not sufficiently address the issue of topological consistency. Operations on inconsistent layouts may fail or lead to misleading results. This paper examines two topological consistency problems. The first is to check the topological consistency of a layout, and the second to refine a layout, that is, to transform an inconsistent layout into a consistent one.

Keywords: Building information/product models, architectural space models, topological consistency

1. INTRODUCTION

Networks have proved useful in the space planning domain to represent and analyze topological properties of space layouts (see, for example, March and Steadman 1974, Hillier and Hanson 1989). Nodes in such networks typically represent spaces or walls, while weighted, directed edges that connect nodes represent adjacency relationships, dimensions, or distances. Recently, this approach has been applied to more fine-grained layouts. In addition to spaces, these include space boundary and space elements such as space enclosure, furnishing and equipment elements (Suter 2010). A schema for network-based space layouts has been proposed which consists of two complementary subschemas. A layout element network schema represents layout elements and topological relationships between them. Geometric properties of layout elements are defined in the geometry schema. The schemas facilitate queries such as shortest path or nearest neighbor queries to analyze topological and geometric layout properties. This analysis functionality could be particularly useful in space planning and building automation domains. However, it is difficult to implement in most BIM systems due to limited representation of topological relationships.

The schema's design is based on and extends existing architectural space schemas, some of which are (partially) implemented in BIM systems (see, for example, Björk 1992, Eastman and Siabiris 1995, Ekholm and Fridqvist 2000, BuildingSmart 2007). In contrast to these schemas, which incorporate both space-centered and construction-centered views, the network-based space layout schema adopts a space-centered view. A crucial aspect of that view is the notion of architectural space as a volume which enables activities.

This paper examines the topological consistency of network-based space layouts. Examples of topological inconsistencies are a pair of overlapping rooms, or a furnishing element which is not contained in a room. Layout analysis or modification operations may fail or lead to erroneous results if a layout is inconsistent. The first of two problems, which are addressed in the following, is to identify common topological inconsistencies. As will be shown, existing constraints in the network-based space layout schema alone are not sufficient for comprehensive consistency checking. Additional constraints are thus required. The second problem is to transform an inconsistent layout into a consistent one. Such transformation is referred to as refinement. It involves the addition, removal, or modification of layout elements and relationships. In the following sections, a brief overview of the network-based space layout schema is given first (Section 2). The main sections describe constraints for topological consistency checking (Section 3) and a layout refinement procedure (Section 4). The paper concludes with a discussion of future work (Section 5).

2. LAYOUT SCHEMA OVERVIEW

2.1 Layout element network schema

The main data structure of a network-based space layout is a layout element network, which consists of a set of layout elements (*le* s) and a set of layout relationship elements (*lr* s). A *le* network is a geometric network where each *le* node has a position in three-dimensional space and each directed *lr* edge between a *le* node pair has a length. Both *le* s and *lr* s have numeric weights which may be accessed by network solvers during query processing. The *le* network schema (LEN) is structured as follows:

$$\begin{aligned}\mathrm{LEN} = (\mathrm{LE}, \mathrm{LR}) = (&(\mathrm{WS}, \mathrm{SS}, \mathrm{SBE}, \mathrm{SE}),\\ &(\mathrm{WSisAdjacentToWS}, \mathrm{SSisAdjacentToSS}, \mathrm{SSsurroundsSE},\\ &\mathrm{SBEboundsWS}, \mathrm{SBEboundsSS}, \mathrm{SBEisAdjacentToSBE}, \mathrm{SEisAdjacentToSBE}))\end{aligned}$$

where *LE* is a list of sets of *le* s, including
WS, the set of whole spaces (Section 2.2);
SS, the set of subspaces (Section 2.2);
SBE, the set of space boundary elements (Section 2.3);
SE, the set of space elements (Section 2.4);
and *LR* is a list of sets of *lr* s.

Figure 1 shows a diagram of the *le* network schema in UML (Jacobson et al. 1998). Since *lr* s have weight and length attributes, the concrete sub-classes of the LayoutRelationship class are modeled as association classes.

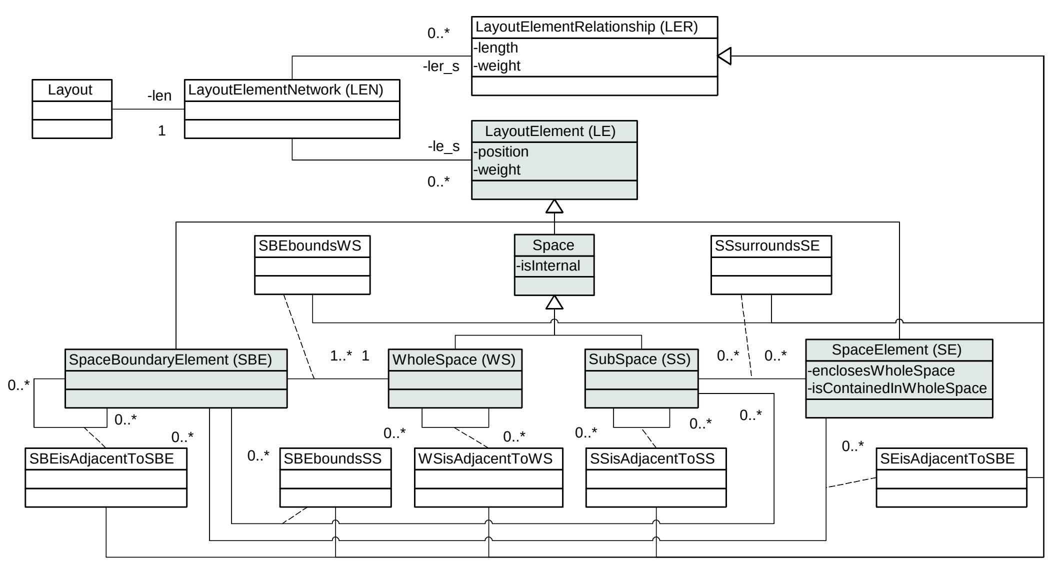

Figure 1: Layout element network schema.

2.2 Spaces

The concept of architectural space is a generalization of whole space and subspace concepts. A space has a volume. There are two major differences between whole spaces and subspaces:

1. A whole space must be completely bounded. That is, there must be a space boundary element (as defined below) for each whole space volume face. Conversely, a subspace boundary may be incomplete. That is, there may be subspace volume faces which do not have a subspace boundary element.
2. A subspace must be contained in a whole space and may overlap with other subspaces in that whole space. Conversely, a whole space must not be overlap with any other whole space.

Recursive space decomposition within a layout (e.g. of buildings into floors, zones, and rooms) is not supported. Instead, this could be addressed by multiple layouts with aggregation relationships between spaces in different layouts. The schema would need to be extended to include such inter-layout topological relationships.

2.3 Space boundary elements

A space boundary element (*sbe*) is part of a space boundary, which is an immaterial layer which delimits a space. An *sbe* does not only bound a whole space, but may bound multiple subspaces at the same time and be adjacent to other *sbe* s and space elements.

2.4 Space elements

Space enclosure, equipment, or furnishing elements are examples of space elements (*se* s). In contrast to *sbe* s, which are immaterial, *se* s are typically physical objects. The respective SpaceElement class is an abstract class - its subclasses are not shown in Figure 1. An *se* may be surrounded by subspaces. For example, a window can be viewed as being surrounded by a subspace. Such a subspace can be thought of as a natural lighting zone. Default subspaces may be defined relative to *se* s in *se* product libraries, which facilitates the creation as well as the refinement of layouts (Section 4).

There are two attributes which designate an *se*'s role as, respectively, contained in or enclosing a whole space. Although there are no corresponding topological relationships in the schema, values for these attributes influence which other relationships a *se* participates in and what constraints are applied to them (Section 3). For instance, an *se* which is designated as whole space enclosing must be adjacent to at least one *sbe*. Whereas the role of an *se* is typically unambiguous, a cabinet *se* may enclose a whole space in one context and be contained in a whole space in another context. Role attributes provide the flexibility to account for *se* s which can assume both roles.

2.5 Geometry schema

Geometry data associated with *le* s complements the *le* network and facilitates the derivation of topological relationships, among other things (Suter 2010). An existing solid boundary representation (Brep) schema is used to represent space, *sbe*, and *se* shapes (Figure 2). Breps are widely used in solid modeling systems. Ideally, a Brep schema covers linear and curved as well as simple and non-simple polygons and polyhedra. The ISO 10303-42:2003 standard is chosen because it defines a comprehensive solid Brep schema (ISO 2003). It is also used by IFC (BuildingSmart 2007).

2.6 Layout example

The structure of network-based space layouts is illustrated with an example (Figure 3). As layouts are difficult to visualize, different sublayouts are selected from the same original layout. Numeric weights are not shown for clarity and because they are not relevant for the following sections. In the first example, only whole spaces and the adjacency relationship between whole spaces are selected (Figure 3a). The second example shows a sublayout which includes whole spaces, *sbe* s, *se* s, and certain adjacency and boundary relationships (Figure 3b). Whole

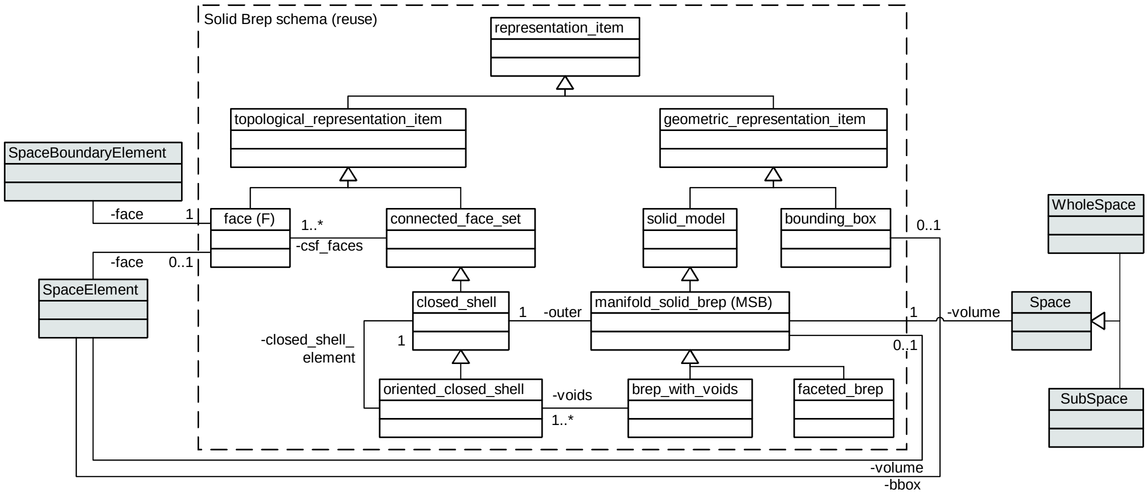

Figure 2: Geometry schema.

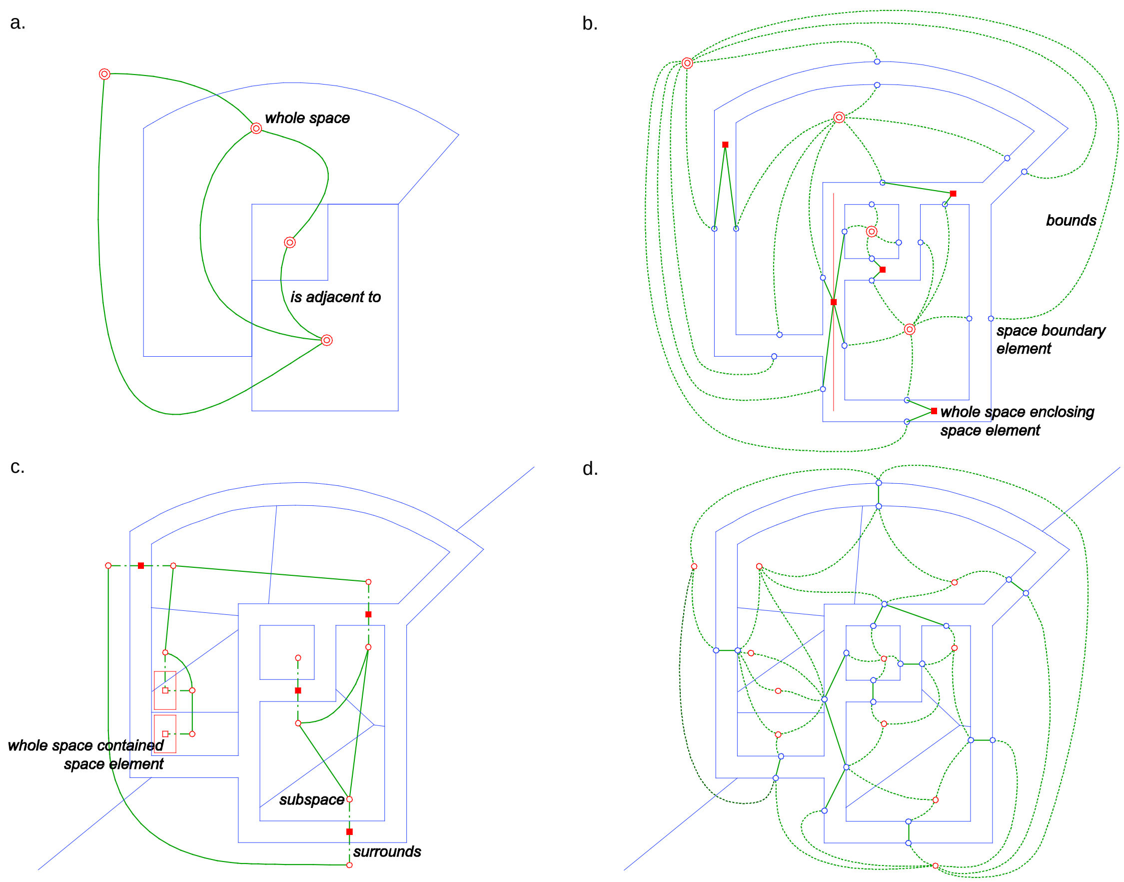

Figure 3: Sublayouts selected from an example layout.

space volumes are offset in the figure to better visualize adjacency relationships involving *sbe* s and *se* s; whole space volumes may actually touch, as in Figure 3a. The next two sublayouts feature subspaces (Figures 3c and 3d). In this example, certain subspace volumes touch and fill the volumes of containing whole spaces in the original layout. Subspace volumes are derived in part from Voronoi cells which in turn are derived from sets of subspace positions that are contained in the same whole space.

3. TOPOLOGICAL CONSISTENCY CHECKING

3.1 Overview

Topological consistency is a particularly relevant aspect of the consistency of network-based space layouts. On the one hand, topological relationships are implied by *le* geometries (implicit topological relationships). On the other hand, some of these implicit topological relationships are represented explicitly in the *le* network (explicit topological relationships). In this section, constraints are defined on explicit and implicit topological relationships in a layout and the *le* s which participate in them. A layout is considered as topologically consistent if all applicable constraints on its topological relationships are satisfied.

The constraints cover typical topological inconsistencies but not geometric accuracy problems such as self-intersecting volumes or inadvertent gaps between space volumes. Certain constraints are defined specifically on relationships involving *se* s which are designated as whole space contained or whole space enclosing (Section 2.4). Thus *SE* subsets *CSE* and *ESE* are defined, where

$\mathrm{CSE} = \{se | se \in \mathrm{SE} \wedge \mathrm{isContainedInWholeSpace}(se)\}$ is the set of *se* s in a layout that are designated as whole space contained (*cse*), and

$\mathrm{ESE} = \{se | se \in \mathrm{SE} \wedge \mathrm{enclosesWholeSpace}(se)\}$ is the set of *se* s in a layout that are designated as whole space enclosing (*ese* s).

For *se* s which are designated as both whole space contained and whole space enclosing to be considered as consistent, either all constraints on *cse* s or all constraints on *ese* s must be met in addition to those on *se* s.

3.2 Constraints on explicit topological relationships

The *le* network and geometry schema diagrams already include structural constraints on explicit topological relationships (Figures 1 and 2). Structural constraints comprise cardinality ratio and participation constraints (Elmasri and Navathe 2007). These constraints specify, respectively, the maximum and minimum number of relationship elements in which an entity participates. Inconsistencies due to participation constraint violations are illustrated with the sublayouts in Figure 3. (There are no cardinality ratio constraint violations in sublayouts if the original layout is already consistent, which is the case in the example.) A sublayout selected from a layout may or may not be a consistent layout, thus checking the consistency of sublayouts is desirable. In the sublayout in Figure 3a, all whole spaces are inconsistent because, according to the *le* network schema, a whole space must be bounded by at least one *sbe*. However, there are no *sbe* s in this sublayout. By contrast, this total participation (or existence dependency) constraint is met by the sublayout in Figure 3b. In addition to the structural constraints in the *le* network schema, a total participation constraint is defined on *ese* s that are involved in the *SEisAdjacentToSBE* relationship. Participation of *se* s is only partial (or optional) in the *le* network schema diagram.

Constraint on whole space enclosing space elements (ESE.1). *An ese must be adjacent to at least one sbe:*

$$\forall ese \in \mathrm{ESE}, \exists sbe \in \mathrm{SBE}((ese, sbe) \in \mathrm{SEisAdjacentToSBE})$$

where SEisAdjacentToSBE *is the set of adjacency edges between* se s and sbe s *in the layout's* le *network (Section 2.1).*

Without this constraint, *ese* s are feasible which do not enclose any whole space.

3.3 Constraints on explicit and implicit topological relationships

Constraints on explicit topological relationships are necessary but not sufficient to comprehensively check the topological consistency of layouts. For example, in the sublayout in Figure 3b, whole space adjacencies are implied by whole space volumes, but these are not reflected in the sublayout's *le* network. Nevertheless, the *le* network meets structural constraints in the *le* network schema because adjacencies between whole spaces are optional. Additional constraints on explicit and implicit relationships are thus introduced to check the consistency of the *le* network and *le* geometries. For example, the constraint on explicit and implicit whole space adjacency relationships in a given layout is defined as follows.

Constraint on whole spaces. *If a pair of whole spaces are adjacent based on their volumes, then there must be a corresponding adjacency edge in the le network. Conversely, if a pair of whole spaces are not adjacent based on their volumes, then there must not be a corresponding adjacency edge in the le network:*

$$\begin{aligned}&\forall \mathrm{ws}_1 \in \mathrm{WS}, \forall \mathrm{ws}_2 \in \mathrm{WS}(\\&(\mathrm{isAdjacentTo}(\mathrm{ws}_1, \mathrm{ws}_2) \wedge (\mathrm{ws}_1, \mathrm{ws}_2) \in \mathrm{WSisAdjacentToWS}) \vee\\&(\neg\mathrm{isAdjacentTo}(\mathrm{ws}_1, \mathrm{ws}_2) \wedge (\mathrm{ws}_1, \mathrm{ws}_2) \notin \mathrm{WSisAdjacentToWS}))\end{aligned}$$

where $\mathrm{Boolean\ \ isAdjacentTo}(\mathrm{ws} \in \mathrm{WS}, \mathrm{ws} \in \mathrm{WS})$ *is a function which determines if a given pair of whole spaces are adjacent based on their volumes (Suter 2010), and* WSisAdjacentToWS *is the set of adjacency edges between whole spaces in the layout's* le *network (Section 2.1).*

Similar constraints are defined on other explicit and implicit topological relationships. In case of the *SSsurroundsSE* relationship, *se* product data is used to determine the implicit topological relationship.

3.4 Constraints on implicit topological relationships

Certain implicit topological relationships are not explicitly represented in a *le* network but used either in the derivation of explicit relationships or consistency checking, or both. *WScontainsSS* and *WSFcontainsSBE* are relationships that are used for both purposes. They have been defined in previous work (Suter 2010). *WScontainsSE*, *WSoverlapsWS*, *SEoverlapsSE*, and *SSoverlapsSE* relationships are implicit topological relationships which are not reflected in the *le* network but are used specifically for consistency checking. Formal definitions are provided in Appendix A. In the following constraints are defined on these relationships.

Constraint on whole spaces (WS.1). *Whole spaces must not overlap:*

$$\forall \mathrm{ws}_1 \in \mathrm{WS}, \forall \mathrm{ws}_2 \in \mathrm{WS}(\mathrm{ws}_1 = \mathrm{ws}_2 \vee \neg\mathrm{overlaps}(\mathrm{ws}_1, \mathrm{ws}_2))$$

Constraint on whole space faces (WSF.1). *A whole space face (*wsf*) must contain exactly one* sbe*:*

$$\forall \mathrm{wsf} \in \mathrm{WSF}, \exists ! \mathrm{sbe} \in \mathrm{SBE}(\mathrm{face}(\mathrm{sbe}) = \mathrm{wsf} \wedge \mathrm{contains}(\mathrm{wsf}, \mathrm{sbe}))$$

where $\mathrm{WSF} = \mathrm{faces}(\mathrm{volumes}(\mathrm{WS}))$ *is the set of whole space faces in the layout.*

In the geometry schema diagram, a relationship from *sbe* s to faces is defined, but there is no inverse relationship because an existing solid Brep schema is reused. The additional constraint *WSF.1* ensures that there is a one-to-one relationship between whole space faces and *sbe* s and that each *wsf* contains an *sbe*.

Constraint on space elements (SE.1). Se *s must not overlap:*

$$\forall \mathrm{se}_1 \in \mathrm{SE}, \forall \mathrm{se}_2 \in \mathrm{SE}(\mathrm{se}_1 = \mathrm{se}_2 \vee \neg\mathrm{overlaps}(\mathrm{se}_1, \mathrm{se}_2))$$

Constraint on whole space contained space elements (CSE.1). *A* cse *must be contained in a whole space:*

$$\forall \mathrm{cse} \in \mathrm{CSE}, \exists \mathrm{ws} \in \mathrm{WS}(\mathrm{contains}(\mathrm{ws}, \mathrm{cse}))$$

Constraint on subspaces (SS.1). *A subspace must be contained in a whole space:*

$$\forall \mathrm{ss} \in \mathrm{SS}, \exists \mathrm{ws} \in \mathrm{WS}(\mathrm{contains}(\mathrm{ws}, \mathrm{ss}))$$

Constraint on subspaces (SS.2). *Subspaces must not have the same position:*

$$\forall \mathrm{ss}_1 \in \mathrm{SS}, \forall \mathrm{ss}_2 \in \mathrm{SS}(\mathrm{position}(\mathrm{ss}_1) \neq \mathrm{position}(\mathrm{ss}_2))$$

This constraint is introduced because subspace volumes may be derived in part from Voronoi cells that are based on distinct subspace positions. Other subspace volume derivation methods are conceivable which do not necessarily require distinct subspace positions, however, switching from one subspace volume derivation method to another should not affect subspace consistency.

Constraint on subspaces (SS.3). *A subspace which surrounds an* se *must not overlap any other* se*:*

$$\forall(\mathrm{ss} \in \mathrm{SS}, \mathrm{se}_1 \in \mathrm{SE}) \in \mathrm{SSsurroundsSE}, \forall \mathrm{se}_2 \in \mathrm{SE}(\mathrm{se}_1 = \mathrm{se}_2 \vee (\neg \mathrm{overlaps}(\mathrm{ss}, \mathrm{se}_2)))$$

A narrow overlap definition (Appendix A) is used in this constraint because overlaps in the wider sense are common and not be considered as inconsistencies.

Constraint on subspaces (SS.4). *If a subspace surrounds a* cse*, then both must be contained in the same whole space:*

$$\forall(\mathrm{ss} \in \mathrm{SS}, \mathrm{cse} \in \mathrm{CSE}) \in \mathrm{SSsurroundsSE}, \exists \mathrm{ws} \in \mathrm{WS}(\mathrm{contains}(\mathrm{ws}, \mathrm{ss}) \wedge \mathrm{contains}(\mathrm{ws}, \mathrm{cse}))$$

4. REFINEMENT

Certain constraints defined in the previous section may also be used to refine layouts, that is, to transform topologically inconsistent layouts to consistent ones. A refinement procedure is outlined in this section (Table 1). The approach is to first reduce a layout to whole spaces, *se* s, and *sbe* s (steps 1-2). Subspaces and, if necessary, missing *sbe* s are added subsequently (steps 3-8). Selected constraints are evaluated in some steps to identify and, typically, remove inconsistent *le* s. Finally, *le* network relationships are derived from *le* geometry data (step 9, Suter 2010).

Step	Constraint evaluation	Layout modification
1.	WS.1	Remove inconsistent whole spaces. Return an empty layout if $\mathrm{WS} = \emptyset$.
2.		Remove *le* network edges and subspaces.
3.	WSF.1	Add or remove *sbe* s to make inconsistent whole space faces consistent.
4.	SE.1, CSE.1, ESE.2	Remove inconsistent *se* s.
5.		Add default subspaces (positions only) and *SSsurroundsSE* relationship.
6.	SS.1, SS.3, SS.4	Remove inconsistent subspaces.
7.	SS.2	Merge inconsistent subspaces.
8.		Add subspace volumes.
9.		Add *le* network relationships and return.

Table 1 Refinement procedure.

Layout refinement always succeeds. If there is no non-overlapping whole space, then an empty layout with $\mathrm{LEN} = (\mathrm{LE} = \emptyset, \mathrm{LR} = \emptyset)$ is returned, which is considered a consistent layout (step 1). For example, the refinement of sublayouts in Figures 3c and 3d would result in empty layouts.

The constraint evaluation order reflects *le* existence dependencies. For example, the constraint WS.1 on whole spaces is evaluated before constraints on *se* s because *se* s are directly or indirectly dependent on whole spaces. Refinement procedure. includes only those constraints and layout modifications which are executed directly by the procedure. For example, the removal of a whole space (step 1) causes the removal of *sbe* s which are indirectly dependent on that whole space. Moreover, whole spaces and *sbe* s which violate total participation constraints in the schema are assumed to be removed from the layout.

All constraints on implicit topological relationships (Section 3.4) are used in refinement. However, one additional constraint on the implicit adjacency relationship between *ese* s and *sbe* s is necessary. It is the counterpart of constraint ESE.1 on the corresponding explicit adjacency relationship in the *le* network.

Constraint on whole space enclosing space elements (ESE.2). *An* ese *must be adjacent to at least one* sbe*:*

$$\forall ese \in ESE, \exists sbe \in SBE(isAdjacentTo(ese, sbe))$$

where Boolean $isAdjacentTo(se \in SE, sbe \in SBE)$ *is a function which determines if a given* se *and* sbe *are adjacent based on their geometry data (Suter 2010).*

Subspaces are added to a layout assuming that a product definition is associated with each *se* which includes surrounding default subspaces. Default subspaces are instantiated (step 5) and constraints on subspaces evaluated. Inconsistent default subspaces are either removed or merged (steps 6 and 7). As subspace volumes may be derived from subspace positions, among other things, their generation is deferred until those subspaces are known which are also present in the refined, consistent layout (step 8).

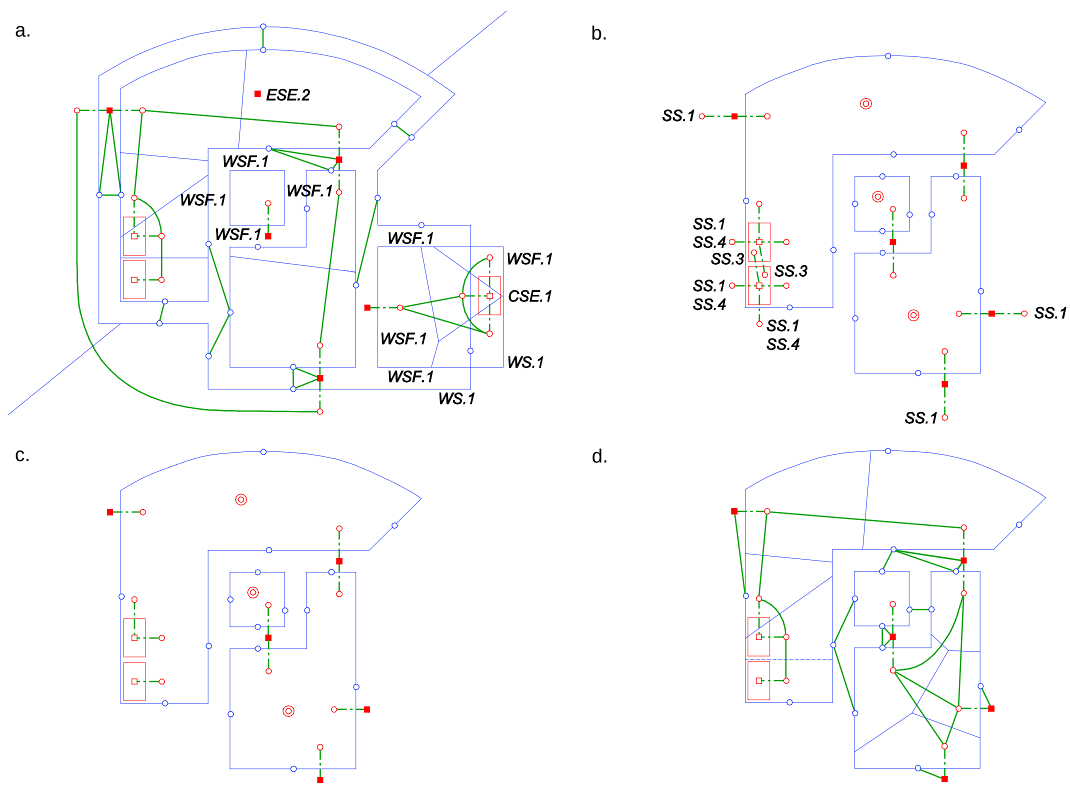

Figure 4: States of a layout undergoing refinement. a. Initial, inconsistent state, b. State after step 5, c. State after step 7, d. Refined, consistent state. (Whole spaces and certain relationships are not shown in a. and d.)

A layout refinement example is given in Figure 4. For clarity, only sublayouts of the actual layouts are shown in Figures 4a and d - the sublayouts do not include whole spaces, space boundary and certain adjacency relationships. Inconsistencies are indicated in Figure 4 by constraints violated. Two whole spaces which are not bounded by *sbe* s (constraint WSF.1) are one reason for the inconsistency of the initial layout (Figure 4a). The whole spaces may have been added to the previously consistent layout together with related *le*s and subspaces. Figure 4b shows the layout state after the addition of default subspaces (step 5). Subspaces which are not contained in a whole space (constraint SS.1), or which are not contained in the same whole space as the *cse* which they surround (constraint SS.4), or overlap with other *se* s (constraint SS.3), are removed in step 6. There is no inconsistency due to constraint SS.2. Figure 4c shows the layout state after evaluation of constraints on subspaces and *se* s (step 7). The refined and consistent layout in Figure 4d includes subspace volumes and a complete *le* network. The described refinement procedure is applied to complete layouts. If inconsistencies are known and limited locally, as, for instance, in case of a *cse* which is added to a particular whole space, then it is feasible to apply the procedure only to immediately affected *le* s instead of a complete layout.

5. DISCUSSION

Network-based space layouts are useful for domains which require information on spatial topological relationships between layout elements. Constraints have been defined which are used to check the topological consistency of network-based space layouts. Some of these constraints are also used in a refinement procedure to transform inconsistent layouts into consistent ones. As a next step in this on-going effort, it is planned to develop a system prototype of the network-based layout schema complemented by topological constraints and the refinement procedure. An obvious choice of implementation environment is a spatial database (see, for example, Oracle 2009). Structural constraints on explicit topological relationships could be implemented in the database's data definition language, and related consistency checking could be delegated to the database. By contrast, the implementation of constraints on implicit topological relationships is less straightforward. This is because these relationships are based on geometric modeling functions such as set operations on Breps of solids. Support for such functions in spatial databases is improving but still limited. Thus, as an alternative, constraints on implicit topological relationships and the refinement procedure could be implemented separately as a dedicated application which would be partially responsible for the consistency of layouts stored in the database.

ACKNOWLEDGEMENTS

The work presented in this paper is in part supported by the grant Austrian Science Fund (FWF): P22208. The author wishes to thank the reviewers for their suggestions.

REFERENCES

Björk, B.-C. (1992) "A Conceptual Model of Spaces, Space Boundaries and Enclosing Structures", *Automation in Construction*, 1 (1), 193-214.

BuildingSmart (2007) "Industry Foundation Classes IFC2x3", http://www.buildingsmart.com, accessed May 18, 2010.

Eastman, C. & Siabiris, A. (1995) "A Generic Building Product Model Incorporating Building Type Information", *Automation in Construction, 3* (1): 283-304.

Ekholm, A. & Fridqvist, S. (2000) "A Concept of Space for Building Classification, Product Modelling and Design", *Automation in Construction,* 9 (3):315-328.

Elmasri, R. & Navathe, S. (2007) "Fundamentals of Database Systems", Pearson.

Hillier, B. & Hanson, J. (1989) "The Social Logic of Space", Cambridge University Press.

ISO TC184/SC4/WG12 N101. (2003) ISO 10303 Part 042.

Jacobson, I., Booch, G. & Rumbaugh, J. (1998) "The Unified Software Development Process", Addison Wesley.

March, L. & P. Steadman, P. (1974) "The Geometry of Environment: An Introduction to Spatial Organization in Design", MIT Press.

Oracle (2009) "Oracle Database 11g Spatial Option", http://www.oracle.com/technology/products/spatial/index.html, accessed June 11, 2010.

Suter (2010) "Outline of a Schema for Network-based Space Layouts", EG-ICE Invited Paper, 8th European Conference on Product & Process Modelling, Cork, Ireland.

A. DEFINITION OF IMPLICIT TOPOLOGICAL RELATIONSHIPS

Implicit topological relationships are defined in the following which are not already defined in Suter (2010). The context (domain of discourse) for each definition is an individual layout.

WScontainsSE relationship.

$$\mathrm{contains} \subset \mathrm{WS} \times \mathrm{SE} = \{(\mathrm{ws}, \mathrm{se}) | \mathrm{ws} \in \mathrm{WS} \wedge \mathrm{se} \in \mathrm{SE} \wedge \exists \mathrm{wsv} \in \mathrm{WSV} \\ (\mathrm{isContainedInWholeSpace}(\mathrm{se}) \wedge \\ \mathrm{wsv} = \mathrm{volume}(\mathrm{ws}) \wedge \mathrm{contains}(\mathrm{wsv}, \mathrm{position}(\mathrm{se})))\}$$

where $\mathrm{WSV} = \mathrm{volumes}(\mathrm{WS})$ *is the set of whole space volumes in the layout, and* $\mathrm{Boolean}\ \ \mathrm{contains}(\mathrm{solidBrep}, \mathrm{point})$ *is a geometric modeling function.*

This relationship is only defined for *se* s designated as whole space contained (*cse* s).

WSoverlapsWS relationship.

$$\mathrm{overlaps} \subset \mathrm{WS} \times \mathrm{WS} = \{(\mathrm{ws}_1, \mathrm{ws}_2) | \mathrm{ws}_1, \mathrm{ws}_2 \in \mathrm{WS} \wedge \mathrm{ws}_1 \neq \mathrm{ws}_2 \wedge \\ \exists \mathrm{wsv}_1 \in \mathrm{WSV}, \exists \mathrm{wsv}_2 \in \mathrm{WSV}(\mathrm{wsv}_1 = \mathrm{volume}(\mathrm{ws}_1) \wedge \mathrm{wsv}_2 = \mathrm{volume}(\mathrm{ws}_2) \wedge \\ \mathrm{interior}(\mathrm{wsv}_1) \cap \mathrm{interior}(\mathrm{wsv}_2) \neq \varnothing)\}$$

where $\mathrm{solidBrep}\ \ \mathrm{solidBrep} \cap \mathrm{solidBrep}$ *is a geometric modeling function.*

In order to be considered as overlapping, the intersection of the interiors of a pair of whole space volumes must be non-empty.

SEoverlapsSE relationship.

$$\mathrm{overlaps} \subset \mathrm{SE} \times \mathrm{SE} = \{(\mathrm{se}_1, \mathrm{se}_2) | \mathrm{se}_1, \mathrm{se}_2 \in \mathrm{SE} \wedge \mathrm{se}_1 \neq \mathrm{se}_2 \wedge \\ \mathrm{interior}(\mathrm{volume}(\mathrm{se}_1)) \cap \mathrm{interior}(\mathrm{volume}(\mathrm{se}_1)) \neq \varnothing\}$$

This definition addresses all situations but is processing intensive. Alternatively, a definition is conceivable which simply compares *se* positions. However, such a definition does not accurately address all situations.

SSoverlapsSE relationship.

$$\mathrm{overlaps} \subset \mathrm{SS} \times \mathrm{SE} = \{(\mathrm{ss}, \mathrm{se}_1) | \mathrm{ss} \in \mathrm{SS} \wedge \mathrm{se}_1 \in \mathrm{SE} \wedge \exists \mathrm{se}_2 \in \mathrm{SE} \\ (\mathrm{se}_1 \neq \mathrm{se}_2 \wedge (\mathrm{ss}, \mathrm{se}_2) \in \mathrm{SSsurroundsSE} \wedge \\ \mathrm{lineSegment}(\mathrm{position}(\mathrm{se}_2), \mathrm{position}(\mathrm{ss})) \cap \mathrm{volume}(\mathrm{se}_1) \neq \varnothing)\}$$

where $\mathrm{shape}\ \ \mathrm{lineSegment} \cap \mathrm{solidBrep}$ *is a geometric modeling function.*

The definition of the overlap relationship is based on the surround edges in the *le* network involving *se* s and subspaces as well as *se* volumes. It does not require subspace volume data, which is useful in layout refinement. More importantly, it is narrower than an alternative definition based on subspace volumes. Overlapping *se* and subspace volumes are common and not considered as inconsistent.