Boolean operations on network-based space layouts

Georg Suter, Filip Petrushevksi, and Milos Sipetic
Design Computing Group, Vienna University of Technology, Austria
georg.suter@tuwien.ac.at

Abstract. Network-based space layouts are schematic models of architectural spaces. In a network-based space layout, selected spatial relations between whole spaces, subspaces, space boundary elements, and space elements (such as walls, openings, or furnishing elements) are modeled as a geometric network. Applications may query such a network to determine, for example, whole spaces or subspaces that are adjacent to a building's perimeter. This paper introduces Boolean operations on network-based space layouts. These include *union*, *intersect*, and *subtract* operations. Specifications of Boolean operations are provided and their processing is described. Examples illustrate operation behaviour as well as how operations may be composed into expressions.

1. Introduction

Space layouts are used in architectural design to model the spatial configuration of buildings. Space information in layouts is relevant for applications in various domains. For example, building services designers reuse space layouts created by architectural designers to develop heating, ventilation, and air conditioning (HVAC), lighting, access control, and security systems. Several layout representation methods exist to support the analysis of space properties, or the generation of alternative layouts with respect to functional and spatial requirements (Steadman, 1983; Hillier and Hanson, 1989; Liggett, 2000; Borkowski, et al. 2002). These methods typically explicitly model space proximity, adjacency, or access relations. This modeling approach has been extended to more fine-grained layouts which are referred to as network-based space layouts (Suter 2010a). In addition to regular spaces or whole spaces, a network-based space layout models subspaces. Selected spatial relations between layout elements are represented explicitly in a directed, weighted graph or network.

The structure of network-based space layouts may be analyzed with graph algorithms (Bondy and Murty, 2010), e.g. to determine the shortest path between two circulation subspaces, or to classify whole spaces and subspaces as adjacent to a building's perimeter (Suter 2010a). Similarly, query languages such as SQL (ISO/IEC, 2008) may be used to extract data from layouts. Although sufficient for simple queries, such layout operations are not closed, that is, they do not generally result in consistent layouts (Suter 2010b). Closed layout operations may be composed into expressions. In previous work, layout operations were introduced, including layout selection, aggregation, and decomposition operations (Suter 2011). Boolean layout operations are the focus of this paper. These are instances of binary operations which accept two argument layouts. Boolean layout operations include *union*, *intersect*, and *subtract* operations. Specifications for these operations are provided and their processing is described. Examples illustrate operation behavior as well as how operations may be composed into expressions.

2. Network-based space layout concepts

2.1 Layout elements and spatial relations

Layout concepts that are relevant for the definition of layout operations are reviewed in this section. Detailed descriptions are provided in Suter (2010a) and Suter (2010b). Network-based space layouts incorporate aspects of existing architectural space models (see, for example, Bjoerk, 1992; Eastman and Siabiris, 1995; Ekholm, 2000; BuildingSmart, 2010). A layout consists of four types of layout elements (*le*): whole spaces (*ws*), subspaces (*ss*), space boundary elements (*sbe*), and space elements (*se*) (Figure 1). A whole space is a space which is bounded on all sides by *sbe* elements. An *sbe* is part of an immaterial layer with zero thickness that bounds a whole space. A partial space or subspace is a space which is contained in a whole space. It may or may not be bounded on all sides by *sbe* elements and may surround space elements. Different types of subspace volumes are supported. In the example layout in Figure 1, subspace volumes correspond to geodesic Voronoi cells (Aurenhammer and Klein, 2001) that are derived from whole space boundaries (used as obstacles) and

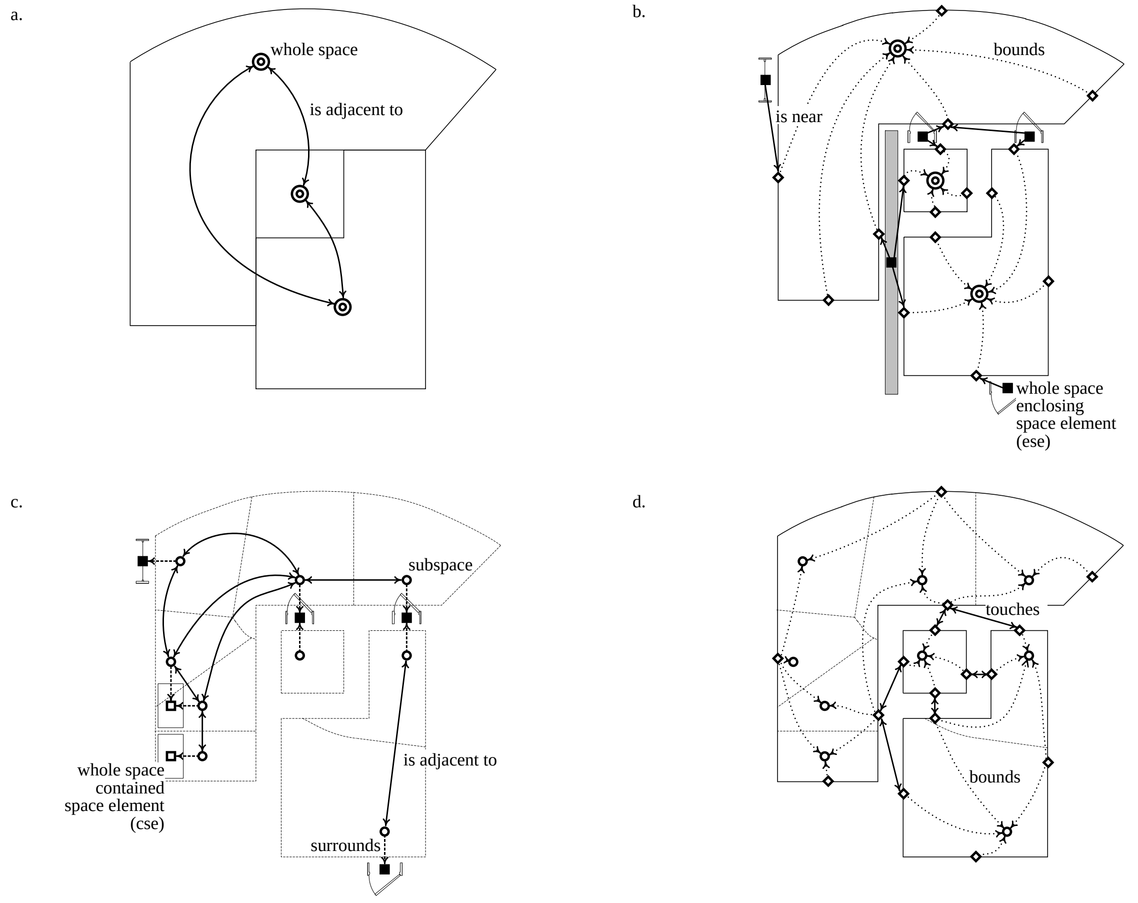

Figure 1: Layout elements and spatial relations in a network-based space layout. a. Adjacency relation between whole spaces (A_{WS}), b. Boundary relation between *sbe* elements and whole spaces ($B_{SBE,WS}$) and proximity relation between *se* and *sbe* elements ($N_{SE.SBE}$), c. Adjacency relation between subspaces (A_{SS}) and surround relation between subspaces and *se* elements ($S_{SS,SE}$), d. Boundary relation between *sbe* elements and subspaces ($B_{SBE,SS}$) and touch relation between *sbe* elements (T_{SBE}). Space boundaries are offset in b. – d. for improved visualization.

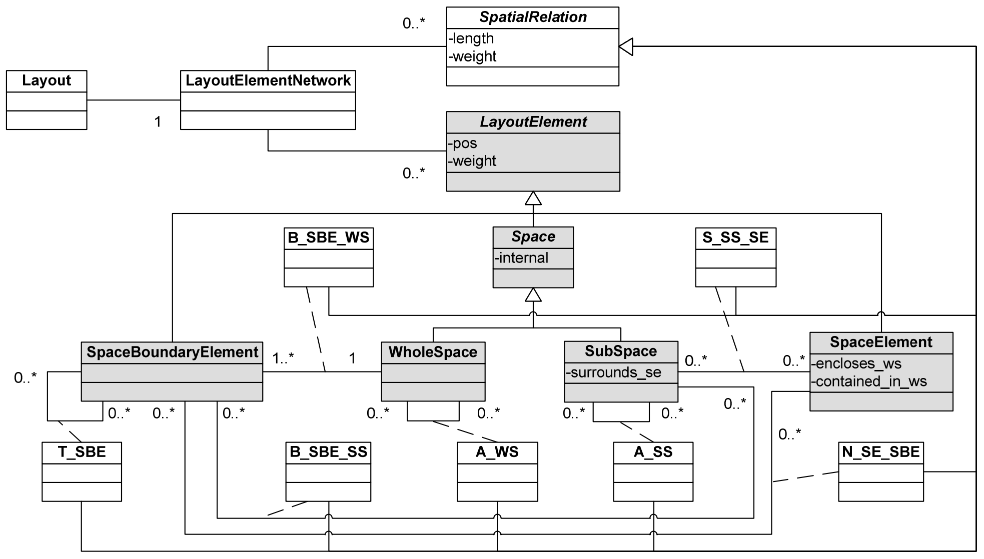

Figure 2: Layout element network schema.

subspace positions (used as sites). Other types of subspaces such as spherical subspaces are supported as well. Space elements (*se*) are (physical) objects, including windows, tables, or luminaires that are either contained in or enclose a whole space. *Se* elements have attributes that indicate if they are contained in or enclose a whole space (se_c or se_e). A desk is an example for an se_c, a door for an se_e. The distinction of se_c and se_e elements matters because they participate in different spatial relations.

A layout has a layout element network, which is a directed, weighted graph with layout elements as nodes and spatial relation (*sr*) elements as edges. *Le* and *sr* elements have weights, which facilitates layout analysis with graph algorithms. Spatial relations in an *le* network include certain adjacency, boundary, surround, touch and proximity relations between *le* elements that are useful for layout analysis and from which other relations may be derived (Figure 1). For example, the enclosure relation between se_e elements and whole spaces may be derived from $N_{SE.SBE}$ and $B_{SBE,WS}$ relations. Figure 2 shows a UML diagram of the layout element network schema (Jacobson, et al. 1998). As *sr* elements have attributes, *SpatialRelation* classes are modelled as association classes.

2.2 Layout refinement

Spatial consistency of layouts is relevant for layout operations. A spatially consistent layout meets certain constraints on spatial relations between *le* elements (Suter, 2010b). As an example for a spatial constraint, no pair of whole spaces in a layout may overlap. The conversion of a possibly inconsistent layout to a consistent one is termed as refinement. A refinement routine that evaluates constraints to identify and resolve spatial inconsistencies in layouts has been outlined in Suter (2010b).

Figure 3 illustrates how the refinement routine works. A_{WS}, $B_{SBE,WS}$, and $B_{SBE,SS}$ relations are not shown in the figure for improved visualization. The layout on the left-hand side features several spatial inconsistencies. For example, a desk se_c is not contained in a whole space. It is thus not included in the refined, consistent layout on the right-hand side. Four subspaces are inserted to ensure that *se* elements are surrounded by all subspaces that are feasible in their contexts. Missing subspaces are instantiated based on *se* templates. For example, the door *se* template has a subspace on its front, and one on its back.

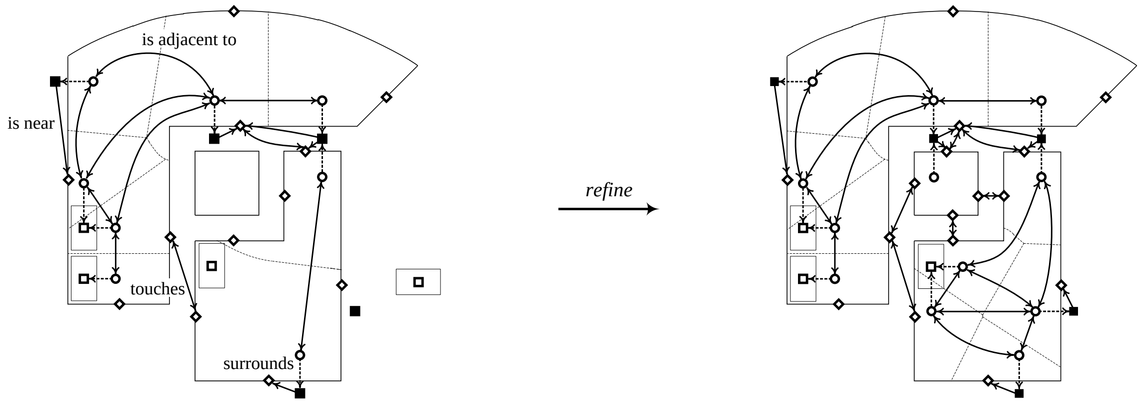

Figure 3: Illustration of the layout refinement routine. Whole space nodes, A_{WS}, $B_{SBE,WS}$, and $B_{SBE,SS}$ relations are not shown.

2.3 Layout operations

Operations are desirable to derive new layouts from existing layouts. In previous work, *select*, *aggregate*, and *decompose* operations have been introduced (Suter, 2011). Each operation is closed, that is, it accepts a consistent layout as an argument and returns a consistent layout. Spatial consistency of result layouts is ensured by layout refinement (Section 2.2), which is the last step in processing *select*, *aggregate*, and *decompose* operations. The *select* operation, for example, may be used to select *le* elements from an argument layout *E* based on predicates on *le* elements in *E*. With this operation, a layout consisting of whole spaces that are offices may be selected from an argument layout that also includes service rooms and circulation spaces. The operation has the signature

$$select_{P_{LE}}(E)$$

where

$P_{LE} = ({}_{p_1}(LE_1), {}_{p_2}(LE_2), \dots {}_{p_n}(LE_n))$ is a list of predicates on attributes of *le* elements in E – targeted *le* elements are selected, and

E is a layout (a layout operation expression).

If there is a predicate on whole spaces in P_{LE}, then whole spaces are selected explicitly. If not, then whole spaces are selected implicitly by selected *le* elements that are dependent on them. The operation does not support predicates on attributes of *sr* elements because these are derived automatically from *le* elements during layout refinement (Section 2.2).

Examples of the *select* operation are shown in Figure 4. In the first example (Figure 4, top right), all whole spaces are selected from the argument layout (Figure 4, top left). *Sbe* and *sr* elements are derived when the intermediate layout is refined. In the second example (Figure 3, down left), whole spaces that are *WORK* spaces and all *se* elements are selected from the same argument layout. The cabinet in the *CIRCULATION* whole space is initially selected but not contained in a selected whole space. It is therefore removed when the intermediate layout is refined. Subspaces are selected implicitly if they surround selected *se* elements. In the third example (Figure 4, down right), there is no whole space predicate. That is, whole spaces are selected implicitly if they are related to selected doors or subspaces. The predicate ${}_{width<0.9}(SE \bowtie DoorT)$ targets space elements that are doors (instances of type *DoorT*) and less than 0.9 m wide. The ⋈ symbol stands for the natural join operation in relational algebra. A stand-alone subspace in the left whole space in the argument layout which does not surround an *se* is selected as well.

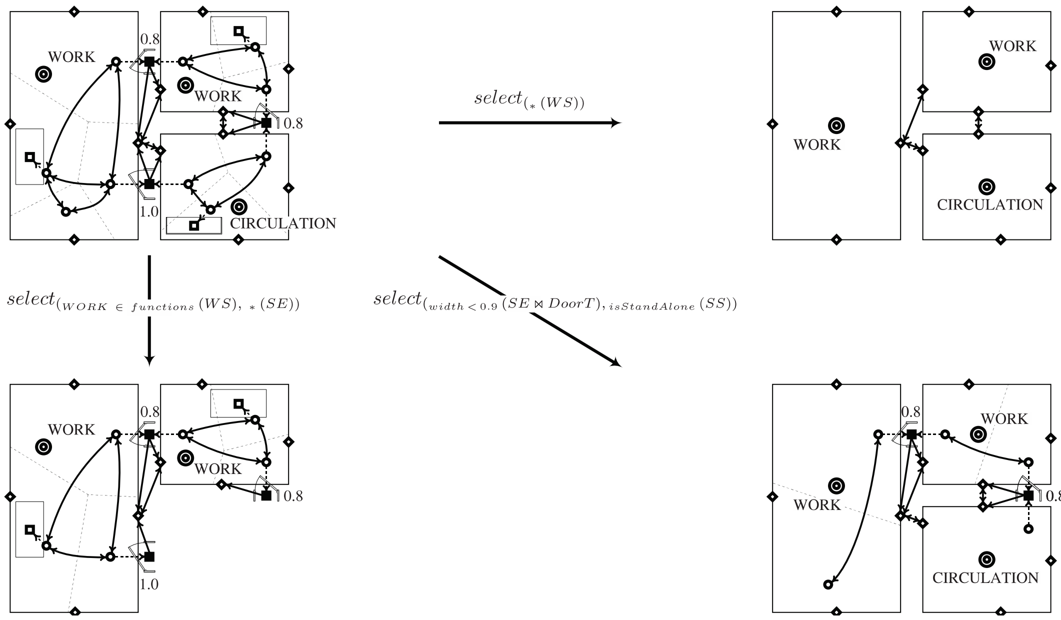

Figure 4: Illustration of *select* operation. A_{WS}, $B_{SBE,WS}$, and $B_{SBE,SS}$ relations are not shown.

3. Boolean layout operations

3.1 Overview

In addition to *select*, *aggregate*, and *decompose* operations, *union*, *intersect*, and *subtract* are desirable operations on network-based space layouts. These operations are conceptually similar to Boolean operations in set theory and solid modelling. Boolean layout operations are instances of binary layout operations, that is, they accept two layouts as arguments. Operations are defined and illustrated in Sections 3.2 - 3.4. Operation processing is described in Sections 3.5-3.10.

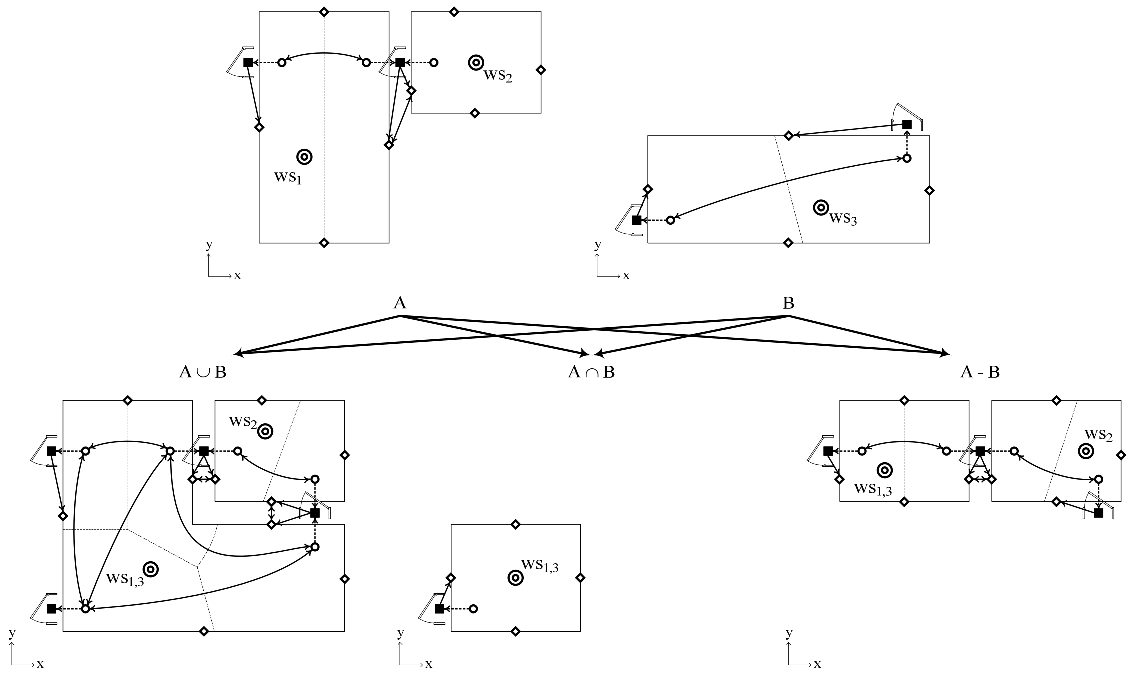

Figure 5: Illustration of *union, intersect,* and *subtract* operations. A_{WS}, $B_{SBE,WS}$, and $B_{SBE,SS}$ relations are not shown.

3.2 *Union* operation

The $union\ (\cup)$ operation derives a layout by merging overlapping whole spaces in argument layouts E_1 and E_2. Disjoint or touching whole spaces from E_1 and E_2 are also included in the result layout together with spatially consistent *se* elements, subspaces, and *sbe* elements. The operation has the signature

$$union(E_1, E_2)$$

where E_1 and E_2 are layouts. Figure 5 illustrates the $union$ operation. Whole space ws_1 in argument layout *A* overlaps with ws_3 in argument layout *B* (Figure 5, top). The L-shaped, merged $ws_{1,3}$ in the result layout is derived by union of the volumes of ws_1 and ws_3 (Figure 5, bottom left). Ws_2, which touches ws_1 and ws_3, is included in the result layout without modification. All *se* elements and subspaces in the argument layouts are included because they are spatially consistent. However, subspace volumes and certain *sr* elements are modified because they are inconsistent.

3.3 *Intersect* operation

The $intersect\ (\cap)$ operation derives a layout by intersecting overlapping whole spaces in argument layouts E_1 and E_2. Spatially consistent *se*, subspace, and *sbe* elements from E_1 and E_2 are also included in the result layout. The operation has the signature

$$intersect(E_1, E_2)$$

where E_1 and E_2 are layouts. Figure 5 illustrates the $intersect$ operation. The volumes of ws_1 and ws_3 are intersected to create the volume of $ws_{1,3}$ in the result layout (Figure 5, bottom center). Only one *se* and one subspace from argument layouts are spatially consistent and thus included in the result layout.

3.4 *Subtract* operation

The $subtract\ (-)$ operation derives a layout by subtracting whole spaces in argument layout E_2 from whole spaces in argument layout E_1. Spatially consistent *se*, subspace, and *sbe* elements from E_1 and E_2 are also included in the result layout. The operation has the signature

$$subtract(E_1, E_2)$$

where E_1 and E_2 are layouts. Figure 5 illustrates the $subtract$ operation. The volume of *ws₁* is modified by subtracting the volume of *ws₃* (Figure 5, bottom right). *Ws₂* is included in the result layout without modification as it does not overlap with *ws₃*.

3.5 Operation processing

The processing of *union, intersection, and subtract* operations involves three steps (Figure 6):

1. Whole spaces are derived from whole spaces in argument layouts *A* and *B* (Sections 3.6 – 3.8),
2. Layout elements that are not whole spaces (subspaces, *se* and *sbe* elements) and *sr* elements from *A* and *B* are derived (Section 3.9), and
3. The intermediate layout C', which results from steps 1 and 2, is converted to the consistent result layout *C* by refinement (Section 3.10).

While step 1 is operation specific, steps 2 and 3 are the same for all operations.

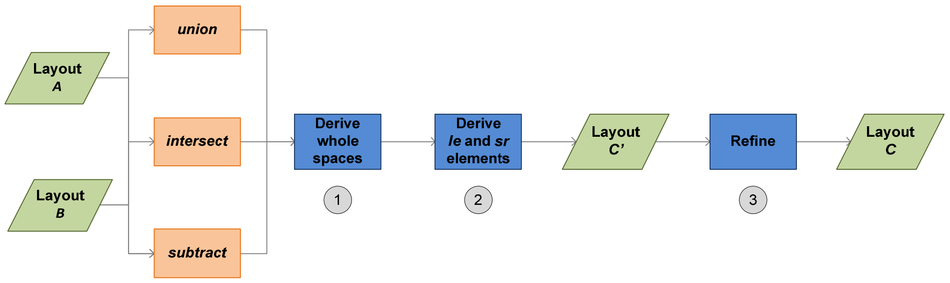

Figure 6: Processing of *union, intersect, and subtract* operations.

3.6 Whole space derivation for the *union* operation

The derivation of whole spaces for the *union* operation is based on the overlap relation O_{ws_A, ws_B} between whole spaces in argument layouts *A* and *B*:

$$O_{ws_A, ws_B} = \{(ws_A, ws_B) \mid ws_A \in WS_A \land ws_B \in WS_B \land i(ws_A.vol) \cap i(ws_B.vol) \neq \emptyset\}$$

where

WS_A is the set of whole spaces in argument layout *A*,

WS_B is the set of whole spaces in argument layout *B*, and

$i(ws_x.vol)$ is the interior of the volume of whole space ws_x.

Let $WS_{C'}$ be the set of whole spaces in the intermediate layout C'. $WS_{C'}$ is derived as follows (see Figure 7 for an example):

1. Each whole space in argument layouts *A* and *B* for which O_{ws_A, ws_B} is not defined (that is, which does not overlap with any whole space in the other layout) is inserted in $WS_{C'}$.
2. Let WS_i be a set of *ws* nodes of the *i*th O_{ws_A, ws_B} component, that is, these nodes are connected in O_{ws_A, ws_B}. Whole spaces in WS_i are merged into a whole space $ws_{C',i}$ which is inserted in $WS_{C'}$. The volume of $ws_{C',i}$ corresponds to the (regularized) solid union (Mortenson 1997) of the volumes of whole spaces in WS_i.

Additional processing is required to derive attribute values for merged whole spaces. This aspect of operation processing is beyond the paper's scope.

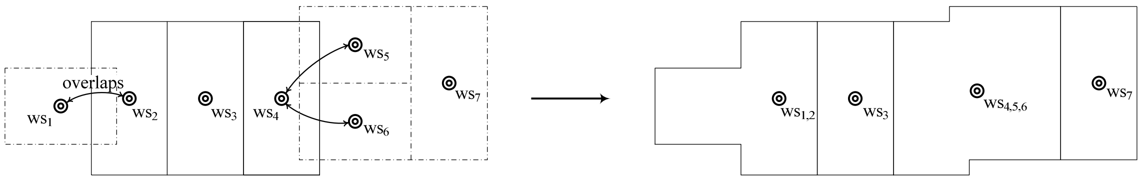

Figure 7: Illustration of whole space derivation for *union*.

3.7 Whole space derivation for the *intersect* operation

For the *intersect* operation, the set $WS_{C'}$ of whole spaces in the intermediate layout C' is derived as follows (see Figure 8 for an example):

1. A whole space $ws_{C'}$ is created for each pair of overlapping whole spaces ws_A and ws_B in A and B. The volume of $ws_{C'}$ corresponds to the solid intersection (Mortenson 1997) of ws_A and ws_B volumes.
2. If the $ws_{C'}$ volume has a single part, then $ws_{C'}$ is inserted in $WS_{C'}$.
3. If the $ws_{C'}$ volume has multiple, disconnected parts, then $ws_{C'}$ is split. Each part becomes the volume of a whole space which is inserted in $WS_{C'}$.

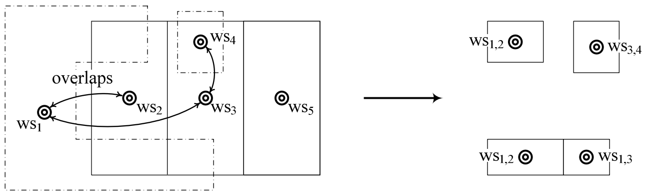

Figure 8: Illustration of whole space derivation for *intersection*.

3.8 Whole space derivation for the *subtract* operation

For the *subtract* operation, the set $WS_{C'}$ of whole spaces in the intermediate layout C' is derived as follows (see Figure 9 for an example):

1. The volume of each whole space ws_B in B which overlaps a whole space ws_A in A is subtracted from the ws_A volume by solid subtraction (Mortenson 1997).
2. If the modified ws_A volume is non-empty and has a single part, then ws_A is inserted in $WS_{C'}$.
3. If the modified ws_A volume has multiple, disconnected parts, then ws_A is split. Each part becomes the volume of a whole space which is inserted in $WS_{C'}$.

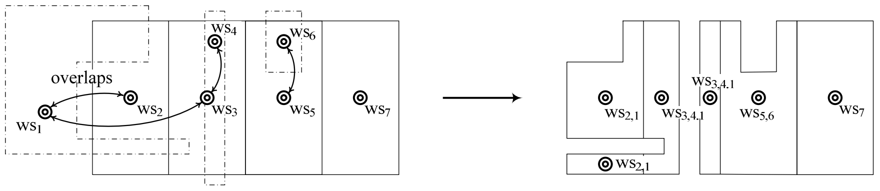

Figure 9: Illustration of whole space derivation for *subtract*.

3.9 *Le* and *sr* element derivation

For all operations, it is desirable to include *le* elements that are not whole spaces in result layouts if they are spatially consistent. The derivation of *le* elements is thus the same for *union*, *intersect*, and *subtract* operations. Subspaces, *sbe* and *se* elements in the intermediate layout C' are derived by union of respective element sets from argument layouts A and B. Duplicate elements, that is, elements with the same unique identifier, are discarded. An *sr* element from A or B is included in C' only if related *le* elements are also included and if it is not a duplicate.

3.10 Refinement

An intermediate layout C' is typically spatially inconsistent. In the *intersect* example in Figure 5, three door se_e elements from layouts A and B are inconsistent in C'. By refining C' (Section 2.2), such inconsistencies are identified and resolved automatically. In case of the *intersect* example, inconsistent door se_e elements are removed. The refinement step concludes the processing of a Boolean layout operation.

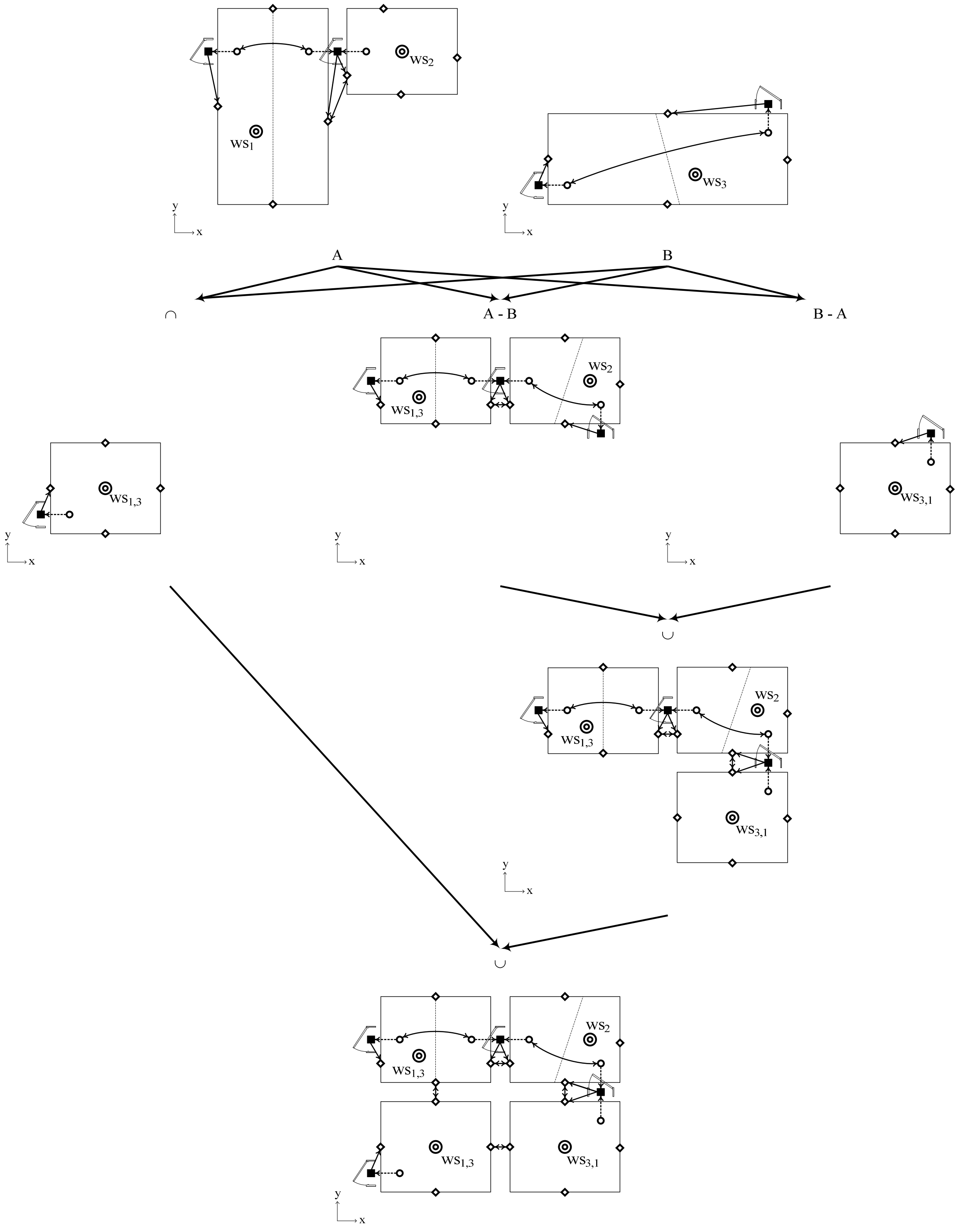

Figure 10: Illustration of the *overlay* operation. A_{WS}, $B_{SBE,WS}$, and $B_{SBE,SS}$ relations are not shown.

4. Layout operation expression examples

Similar to *select*, *decompose*, and *aggregate* operations, Boolean layout operations may be composed into operation expressions. An example is the *symmetric difference* (Δ) operation which is a composition of *subtract* and *union* operations:

$$A \,\Delta\, B = (A - B) \cup (B - A).$$

The *symmetric difference* operation in turn may be used to define an *overlay* operation:

$$A \;\mathit{overlay}\; B = (A \cap B) \cup (A \,\Delta\, B).$$

Figure 10 shows an example of the *overlay* operation. Argument layouts *A* and *B* are at the top of Figure 10, and the result layout at the bottom.

The potential utility of the *overlay* operation is further illustrated with a more realistic example of two layouts of a floor in an existing office building (Figure 11). Layout *A* models elements that are relevant for natural lighting. It includes three large, unpartitioned office zones, several circulation and service spaces, and window se_e elements. Whereas office zone ws_1 receives natural light from one direction only, office zones ws_2 and ws_3 include subspaces which form a zone that receives natural light from two directions. These subspaces are cycle nodes in the A_{SS} relation that is part of the *le* network. Circulation space ws_4 is naturally lit, whereas hallway ws_5 is not. Layout *B* is an artificial lighting layout. It includes individual office spaces and ceiling luminaire se_c elements. Service spaces and circulation spaces that do not provide direct access to office spaces are omitted in *B*. Layout *C* is a combined artificial and natural lighting layout obtained by *overlay* of *A* and *B*. Its *le* network may be traced, for example, to obtain ceiling luminaire se_c elements that are nearest to a given window se_e. Such information is relevant for the design of lighting automation systems that use daylight harvesting control strategies to minimize artificial lighting. The corner office ws_6 and a nearby office ws_7 are the only offices in layout *C* that receive natural light from more than one direction.

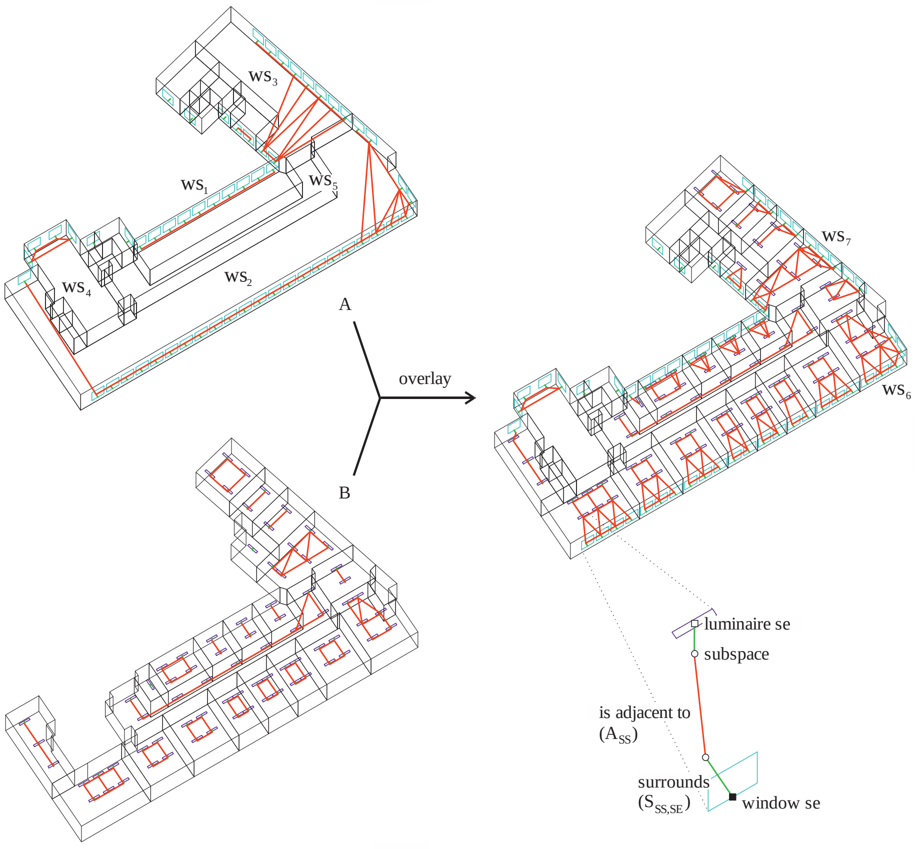

Figure 11: *Overlay* example. Subspace volumes, A_{WS}, $N_{SE,SBE}$, T_{SBE}, $B_{SBE,WS}$, and $B_{SBE,SS}$ relations are not shown.

5. Conclusion

Boolean operations on network-based space layouts have been introduced. These operations are closed, that is, they accept consistent layouts as arguments and return consistent layouts. Thus they may be composed into expressions. The *overlay* operation is an example of an operation that is a composition of Boolean operations. In addition to operations on individual layouts, there may be operations that relate *le* elements in different layouts. For example, whole spaces in two layouts may be related by containment. With such a comprehensive set of layout operations, it may be feasible to define multiple, domain-specific space views of buildings (Rosenman and Gero, 1996) in a compact manner. Certain views may be sufficiently generic for reuse across buildings.

Acknowledgements

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

References

Aurenhammer, F. & Klein, R. (2000), Voronoi diagrams, *in* J. Sack & G. Urrutia, ed., 'Handbook of Computational Geometry, Chapter V', Elsevier Science Publishing, pp. 201-290.

Bjoerk, B.-C. (1992), 'A conceptual model of spaces, space boundaries and enclosing structures', Automation in Construction 1(1), 193-214.

Bondy, A. & Murty, U. (2010), Graph Theory, Springer.

Borkowski, A.; Grabska, E. & Szuba, J. (2002), On graph-based knowledge representation in design, in T. Songer, ed., 'Computing in Civil Engineering, Int. Workshop on IT in CEA', pp. 232-238.

BuildingSmart (2010), 'Industry Foundation Classes IFC2x4', BuildingSmart, Technical report, BuildingSmart.

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.

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

ISO/IEC (2008), 'SQL:2008 (ISO/IEC 9075)'(ISO/IEC 9075), Technical report, ISO/IEC.

Jacobson, I.; Booch, G. & Rumbaugh, J. (1998), The unified software development process, Addison Wesley Longman.

Liggett, R. S. (2000), 'Automated facilities layout: past, present and future', Automation in Construction 9(2), 197 - 215.

Mortenson, M. (1997), Geometric modeling, John Wiley & Sons.

Rosenman, M. & Gero, J. (1996), 'Modelling multiple views of design objects in a collaborative environment', Computer-Aided Design 28(3), 193-205.

Steadman, P. (1983), Architectural morphology, Pion.

Suter, G. (2011), Operations on network-based space layouts, *in* H.-J. Bargstädt & K. Ailland, ed., '11th Conference on Construction Applications of Virtual Reality', Bauhaus University, Weimar, Germany, pp. 567-578.

Suter, G. (2010a), Outline of a schema for network-based space layouts, *in* K. Menzel & R. Scherer, ed., '8th European Conference on Product and Process Modelling, European Group for Intelligent Computing in Engineering (EG-ICE)', Taylor & Francis, Cork, Ireland.

Suter, G. (2010b), Topological constraints for consistency checking of network-based space layouts, *in* W. Thabet, ed., '27th International Conference on Applications of IT in the AEC Industry', Cairo, Egypt.