DEFINITION OF OPERATIONS ON NETWORK-BASED SPACE LAYOUTS

Georg Suter, PhD, Associate Professor
Department of Digital Architecture and Planning, Vienna University of Technology
georg.suter@tuwien.ac.at, http://www.iemar.tuwien.ac.at

ABSTRACT: *Network-based space layouts are schematic models of architectural spaces for building design. They adopt a space-centered view and incorporate aspects of existing architectural space models. A layout has a geometric network that represents certain spatial relations between layout elements. A network may be analyzed using graph algorithms, such as nearest neighbor or shortest path algorithms. Operations on network-based space layouts are defined that are closed, that is, they derive consistent layouts from given consistent layouts. Closed layout operations may be composed into expressions. This facilitates the derivation of multiple views of buildings. An example of a layout operation expression is presented in which a lighting-specific layout is derived from an existing layout.*

KEYWORDS: *Architectural space models, Building information models, Query languages*

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). 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 and space elements, such as walls, openings, furnishing, or technical equipment elements. 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. The objective of this paper is to define layout operations that are closed and appear particularly useful for building design.

2. REVIEW OF NETWORK-BASED SPACE LAYOUT CONCEPTS

2.1 Layout elements and layout element 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 are schematic models of architectural spaces. They adopt a space-centered (as opposed to an enclosure-centered) view and 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*). A whole space is a space which is bounded on all sides by space boundary elements. A space boundary element 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 space boundary elements and may surround space elements. The latter are (physical) objects, including windows, tables,

or luminaires that are either contained in or enclose a whole space. Space elements have attributes that indicate if they are whole space contained or whole space enclosing space elements (*cse* or *ese*). A desk is an example for a *cse*, a door for an *ese*. The distinction of *cse*s and *ese*s matters because they participate in different spatial relations.

A layout has a layout element network, which is a directed, weighted graph with *le*s as nodes and spatial layout element relations (*ler*) as edges. *Le*s and *ler*s have weights, which facilitates layout analysis with graph algorithms. Spatial relations in an *le* network include certain adjacency, boundary, surrounds, and proximity relations between *le*s. These relations are illustrated with an example layout (Fig. 1). As they are difficult to visualize together, relations are shown as partial views of the same layout. Moreover, space volumes are offset in Fig. 1b.-d. for improved visualization. Volumes may actually touch, as in Figure 1a.

In the layout in Fig. 1, subspace volumes are disjoint and fill the volumes of containing whole spaces. These subspace volumes are geodesic Voronoi cells that are generated with subspace positions and whole space volumes used as, respectively, sites and obstructions (Aurenhammer and Klein, 2000). Each point inside a subspace volume is nearer to the subspace node than to any other subspace node. Given a position (e.g. of a person), one can thus easily find its nearest subspace node. Other subspace volume derivation methods are supported, but these are not present in the layout.

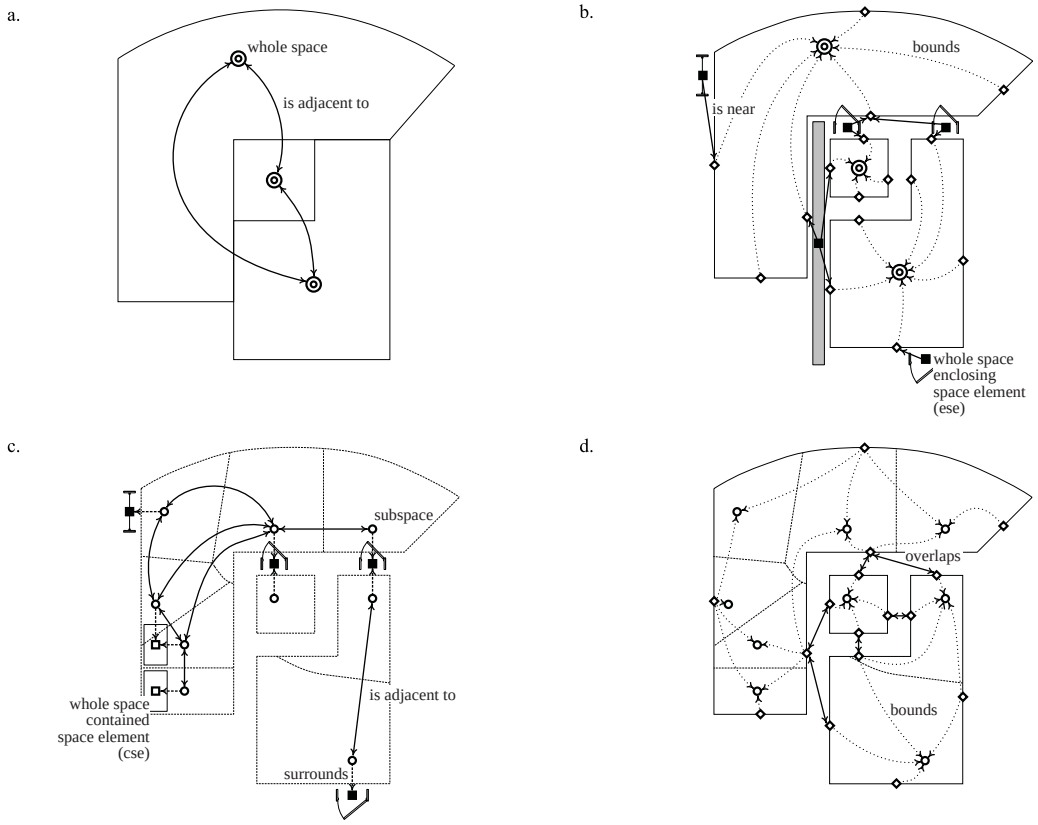

Fig. 1: Spatial relations in a network-based space layout. a. Adjacency relation between whole spaces (A_{WS}), b. Boundary relation between *sbe*s and whole spaces ($B_{SBE,WS}$) and proximity relation between *se*s and *sbe*s ($N_{SE.SBE}$), c. Adjacency relation between subspaces (A_{SS}) and surround relation between subspaces and *se*s ($S_{SS,SE}$), d. Boundary relation between *sbe*s and subspaces ($B_{SBE,SS}$) and overlap relation between *sbe*s (O_{SBE}).

2.2 Layout refinement

Spatial consistency of layouts is relevant for layout operations. A spatially consistent layout meets certain constraints on explicit and implicit spatial relations between *le*s (Suter, 2010b). As an example for an implicit 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 uses constraints to identify and resolve spatial inconsistencies in layouts has been outlined in Suter (2010b).

Fig. 2 illustrates spatial inconsistencies in a layout and how they are resolved by the refinement routine. For example, there is a desk in the unrefined layout (Fig. 2, left) which is not contained in a whole space and is thus removed. There is another desk in the down right whole space, which, according to its type definition (template), is surrounded by at most four subspaces. Two subspaces are feasible but not present. These missing subspaces are inserted (Fig. 2, right). As a consequence of the insertions, volumes of subspaces in the down right whole space are inconsistent and must be (re)generated together with subspace adjacencies. These and other inconsistencies of *le*s and *ler*s are resolved in the refined layout.

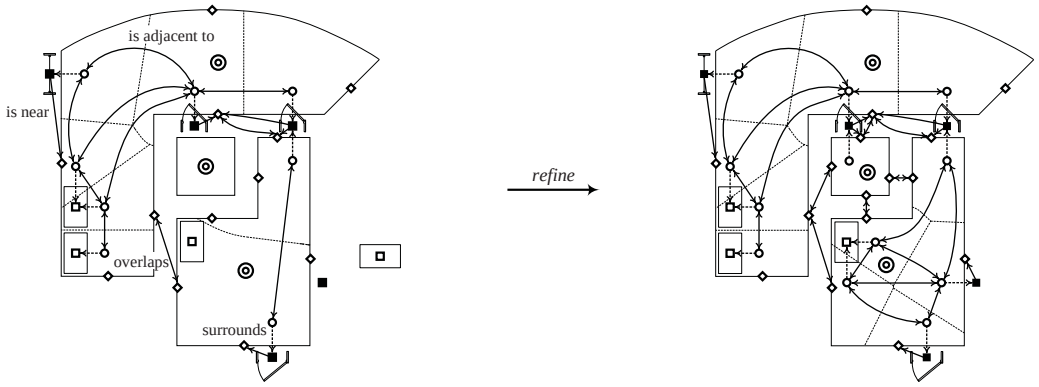

Fig. 2: Illustration of the layout refinement routine. A_{WS}, $B_{SBE,WS}$, and $B_{SBE,SS}$ relations are not shown.

3. LAYOUT OPERATIONS

3.1 Overview

Layout operations are introduced, including *select*, *aggregate*, *decompose*, and $update_{WEIGHT}$ operations. Their signatures and processing are defined and illustrated. Variants with different signatures are conceivable for each operation but not described here for space reasons. 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 the processing of *select*, *aggregate*, and *decompose* operations. Refinement is not required in case of the $update_{WEIGHT}$ operation as it does not modify structure or geometry of argument layouts.

3.2 *Select* operation

The *select* operation selects *le*s from the argument layout *E* based on predicates on *le*s in *E*. For example, a layout consisting of whole spaces that are offices may be selected with the *select* operation from an argument layout that also includes service rooms and circulation spaces. The operation has the signature

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

where

- $P_{LE} = ({}_{p_1}(LE_1), {}_{p_2}(LE_2), \ldots {}_{p_n}(LE_n))$ is a list of predicates on attributes of *le*s in *E* – targeted les 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 if they are related to selected *le*s. This is because *le*s that are not whole spaces are dependent on whole spaces. For example, a furnishing element that is not contained in a whole space is considered as inconsistent (Section 2.2). Similarly, if a space element is selected explicitly but not its surrounding subspaces, then these subspaces are selected implicitly (and vice versa). The operation does not support predicates on attributes of *ler*s because *ler*s are derived from *le*s. Operation processing involves these steps:

1. A relational algebra selection operation (Silberschatz, Korth, and Sudarshan, 2006) is created and executed for each $_{p_i}(LE_i)$ in P_{LE} to select *le*s from the argument layout *E*.

2. If there is no predicate on whole spaces in P_{LE}, then whole spaces that are related to selected *le*s are selected. A whole space *ws* is related to a layout element *le* if
   ws contains *le* (*le* is a *cse* or a subspace),
   ws is bounded by *le* (*le* is an *sbe*), or
   ws is bounded by an *sbe* that is near *le* (*le* is an *ese*).
3. If there are selected space elements that are surrounded by unselected subspaces, then these subspaces are selected (and vice versa).
4. The intermediate layout is refined into the result layout (Section 2.2).

Examples of the *select* operation are shown in Fig. 3. In the first example (Fig. 3, top right), all whole spaces are selected from the argument layout (Fig. 3, top left). *Sbe*s and *ler*s are derived when the intermediate layout is refined.

In the second example (Fig. 3, down left), whole spaces that are *WORK* spaces and all *se*s 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*s.

In the third example (Fig. 3, 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 $\bowtie$ 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 a space element is selected as well.

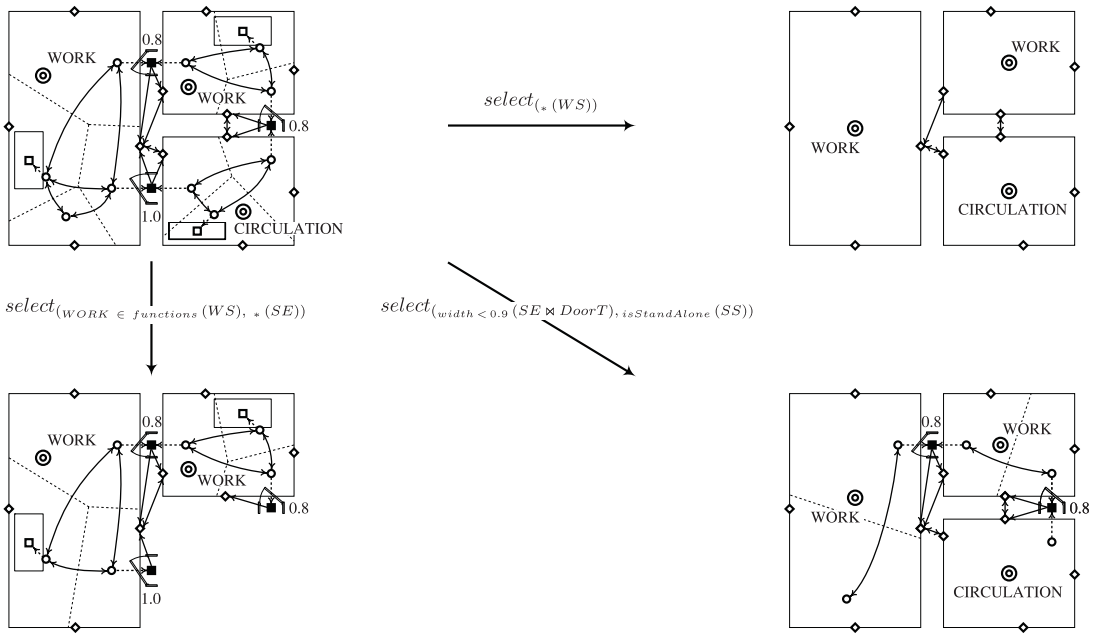

Fig. 3: Examples of the *select* operation. A_{WS}, $B_{SBE,WS}$, and $B_{SBE,SS}$ relations are not shown.

3.3 *Aggregate* operation

The *aggregate* operation merges sets of whole spaces in the argument layout *E* into larger whole spaces. Whole spaces in a whole space set belong to the same group (as specified by given whole space attributes) and are connected in a subnetwork of the *le* network of *E* (as specified by given predicates on barrier *le*s and *ler*s). For example, the *aggregate* operation may be used to derive a layout with whole spaces that result from merging whole

spaces with the same function in an argument layout. The operation has the signature

$$aggregate_{G_{WS},\, P_{LE_b},\, P_{LER_b}}(E)$$

where

- $G_{WS} = (g_{ws_1}, g_{ws_2}, \ldots, g_{ws_k})$ is a list of whole space attributes that are used to group whole spaces in *E*,
- $P_{LE_b} = (_{p_1}(LE_1), _{p_2}(LE_2), \ldots, _{p_n}(LE_n),)$ is a list of predicates on attributes of *le*s in *E* – targeted *le*s are used as barriers,
- $P_{LER_b} = (_{p_1}(LER_1), _{p_2}(LER_2), \ldots, _{p_m}(LER_m),)$ is a list of predicates on attributes of *ler*s in *E* – targeted *ler*s are used as barriers, and
- *E* is a layout (a layout operation expression).

Whole space grouping is analogous to grouping in aggregation operations in relational algebra (Silberschatz, Korth, and Sudarshan, 2006). Barrier *le*s and *ler*s define a subnetwork of the argument layout's *le* network. Barriers are not passable when whole space connectivity is determined.

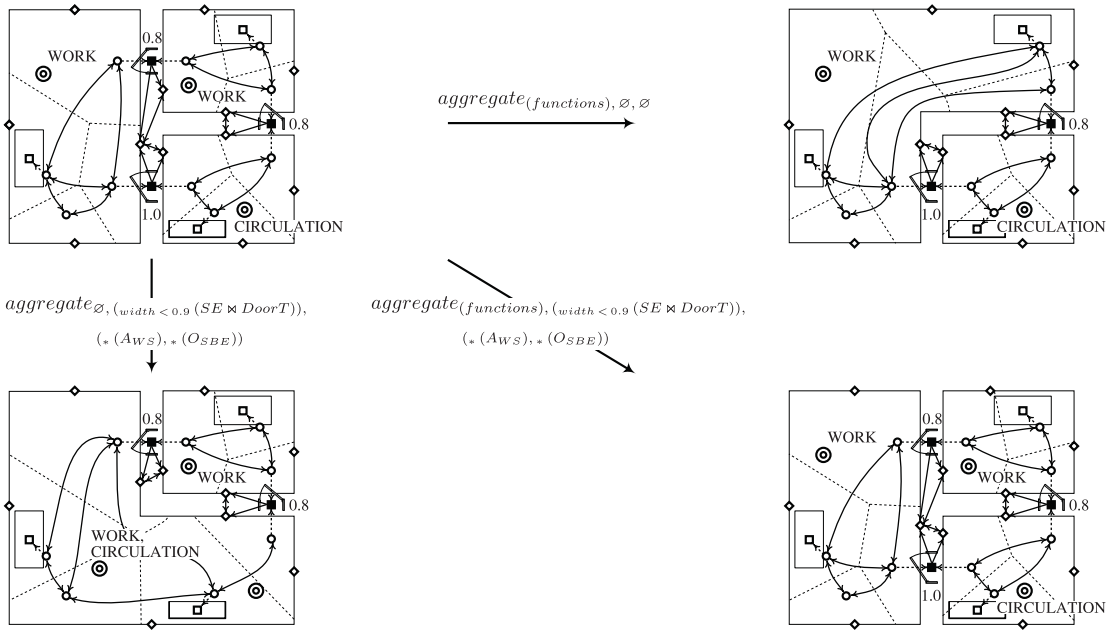

Fig. 4: Examples of the *aggregate* operation. A_{WS}, $B_{SBE,WS}$, and $B_{SBE,SS}$ relations are not shown.

Operation processing involves these steps:

1. The set of whole spaces *WS* in the argument layout *E* is partitioned into whole space subsets $\{WS_1 \subset WS, WS_2 \subset WS, \ldots WS_n \subset WS\}$ such that all whole spaces in a subset WS_i belong to the same whole space group and are connected. Moreover, no two whole spaces in different subsets belong to the same group and are connected. Whole space groups are determined based on whole space attributes (as specified by G_{WS}). Whole space connectivity is determined based on the argument layout's *le* subnetwork (as specified by P_{LE_b} and P_{LER_b}). Directions of *ler*s are ignored to determine whole space connectivity.
2. Whole spaces in a subset WS_i with more than one whole space are replaced by a new whole space *ws*. The volume of *ws* equals the union of the volumes of the whole spaces in WS_i. Certain attribute values of whole spaces in WS_i (such as area or volume) are aggregated and assigned to *ws* attributes.

3. The intermediate layout is refined into the result layout (Section 2.2).

Examples of the *aggregate* operation are shown in Fig. 4. In the first example (Fig. 4, top right), grouping of whole spaces in the argument layout (Fig. 4, top left) is done based on the functions attribute. Since no barriers are defined ($P_{LE_b} = \emptyset$ and $P_{LER_b} = \emptyset$), the full *le* network of the argument layout is used to determine whole space connectivity.

In the second example (Fig. 4, down left), aggregation is done only based on whole space connectivity as no whole space attributes are specified ($G_{WS} = \emptyset$). Since A_{WS} and O_{SBE} elements are barriers, feasible paths between whole spaces include doors. Moreover, doors in these paths are more than 0.9 m wide. The new whole space has *WORK* and *CIRCULATION* functions.

In the third example (Fig. 4, down right), grouping is done based on the whole space functions attribute, but tracing of the argument layout's *le* network to determine whole space connectivity is restricted as in the second example. No whole spaces are aggregated because no pair of whole spaces belong to the same whole space group and are connected when barriers are considered.

3.4 *Decompose* operation

The *decompose* operation replaces whole spaces in the argument layout *E* that contain selected subspaces with smaller whole spaces. The volumes of the smaller whole spaces are derived from the volumes of replaced whole spaces and the positions of selected subspaces. For example, the *decompose* operation may be used to derive a layout from an argument layout where subspaces that surround luminaires are used to derive smaller whole spaces. The operation has the signature

$$decompose_{p_{SS}}(E)$$

where

- p_{SS} is a predicate on attributes of subspaces in *E* – targeted subspaces are used to decompose whole spaces, and
- *E* is a layout (a layout operation expression).

Any subspace is selectable for decomposition regardless of its volume type (Section 2.1). The positions of selected subspaces and containing whole space volumes are used to derive geodesic Voronoi cells (Aurenhammer and Klein, 2000, Section 2.1). These cells become the volumes of new whole spaces. Operation processing involves these steps:

1. A relational algebra selection operation is created and executed based on p_{SS} to select subspaces from the argument layout *E* that are used to derive new whole spaces.
2. For each whole space *ws* that contains a set SS_{ws} of selected subspaces, positions of subspaces in SS_{ws} and the *ws* volume are used as, respectively, sites and obstructions to generate geodesic Voronoi cells (Aurenhammer and Klein, 2000).
3. A whole space is created for each selected subspace *ss*. The geodesic Voronoi cell derived for *ss* in the previous step becomes the volume of the new whole space. Attribute values are copied from the whole space that contains *ss* or recomputed.
4. Whole spaces that contain selected subspaces are removed.
5. The intermediate layout is refined into the result layout (Section 2.2).

Examples of the *decompose* operation are shown in Fig. 5. In the first example (Fig. 5, top right), the whole space in the argument layout (Fig. 5, top left) is decomposed based on subspaces that surround luminaires. The weight of these subspaces is 2 or 3. Consequently, each whole space in the result layout contains a luminaire.

In the second example (Fig. 5, down left), the whole space is decomposed based on subspaces that surround luminaires and are near windows. The weight of these subspaces is 2. Each whole space in the result layout contains two luminaires. Decomposition based on subspaces that are distant from windows (whose weight is 3) would result in the same layout.

In the third example (Fig. 5, down right), the whole space is decomposed based on subspaces that surround

windows and whose weight is 1. All luminaires in a new whole space are nearer (by Euclidean distance) to the position of the subspace that surrounds the window in the same whole space than to the position of the same kind of subspace in the other whole space.

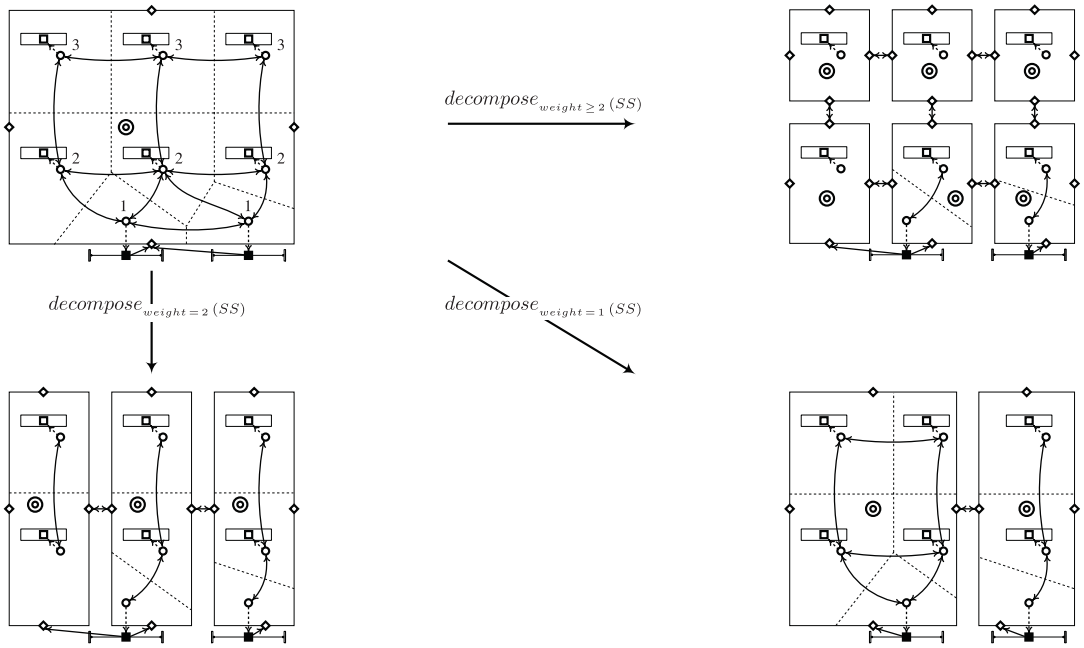

Fig. 5: Examples of the *decompose* operation. A_{WS}, $B_{SBE,WS}$, and $B_{SBE,SS}$ relations are not shown.

3.5 *Update*$_{WEIGHT}$ operation

The *update*$_{WEIGHT}$ operation derives weights of shortest paths from selected source *le*s to selected destination *le*s in an argument layout *E* and assigns these weights to the weight attributes of source *le*s. For example, the *update*$_{WEIGHT}$ operation may be used to derive a layout from an argument layout where luminaire weights reflect the weights of shortest paths to nearest windows. The operation has the signature

$$update_{WEIGHT,\, P_{LE_s},\, P_{LE_d},\, P_{LE_b},\, P_{LER_b}}(E)$$

where

- $P_{LE_s} = \left({}_{p_1}(LE_1), {}_{p_2}(LE_2), \ldots, {}_{p_k}(LE_k)\right)$ is a list of predicates on attributes of *le*s in *E* – targeted *le*s are used as source *le*s whose weights are updated,
- $P_{LE_d} = \left({}_{p_1}(LE_1), {}_{p_2}(LE_2), \ldots, {}_{p_l}(LE_l)\right)$ is a list of predicates on attributes of *le*s in *E* – targeted *le*s are used as destination *le*s,
- $P_{LE_b} = \left({}_{p_1}(LE_1), {}_{p_2}(LE_2), \ldots, {}_{p_m}(LE_m)\right)$ is a list of predicates on attributes of *le*s in *E* – targeted *le*s are used as barriers,
- $P_{LER_b} = \left({}_{p_1}(LER_1), {}_{p_2}(LER_2), \ldots, {}_{p_n}(LER_n)\right)$ is a list of predicates on attributes of *ler*s in *E* – targeted *ler*s are used as barriers, and
- *E* is a layout (a layout operation expression).

Only weights of source *le*s targeted by P_{LE_s} are updated by the operation. If there are multiple destination *le*s, then the weight of the shortest path from a source *le* to its nearest destination *le* is determined. The weight of a source *le*

which is also a destination *le* is 0. Operation processing involves these steps:

1. A relational algebra selection operation is created and executed for each $_{p_i}(LE_i)$ in P_{LE_s} to determine source *le*s (le_s) in the argument layout *E*.
2. Similarly, destination *le*s (le_d) are determined based on P_{LE_d}.
3. The nearest le_d is determined for each le_s. Search for the nearest le_d is restricted to a subnetwork of the argument layout's *le* network, as specified by le_bs and ler_bs. Only *ler* weights are considered to determine path weights. *Le* weights and *ler* directions are ignored.
4. For each le_s, the weight of the shortest path to its nearest le_d is assigned to its weight attribute.

Examples of the $update_{WEIGHT}$ operation are shown in Fig. 6. For all examples, *ler* weights are 1. In the first example (Fig. 6, top right), the weight of each subspace in the argument layout (Fig. 6, top left) is updated based on the weight of the shortest path to the nearest window. $B_{SBE,SS}$ elements are barriers for computing shortest paths. Luminaire subspaces with a weight of 2 are closer to the windows than those with a weight of 3.

In the second example (Fig. 6, down left), the weight of each space element is updated based on the weight of the shortest path to a destination luminaire ($id = 101$). $B_{SBE,SS}$ and $N_{SE,SBE}$ elements are barriers for computing shortest paths. Weights suggest that the right window is closer to the destination luminaire than the left one.

In the third example (Fig. 6, down right), the weight of each subspace is updated based on the weight of the shortest path to the nearest *sbe*. The full *le* network is used to compute shortest paths as no barriers are specified ($P_{LE_b} = \emptyset$ and $P_{LER_b} = \emptyset$). There is a luminaire subspace with a weight of 2 which is completely unbounded.

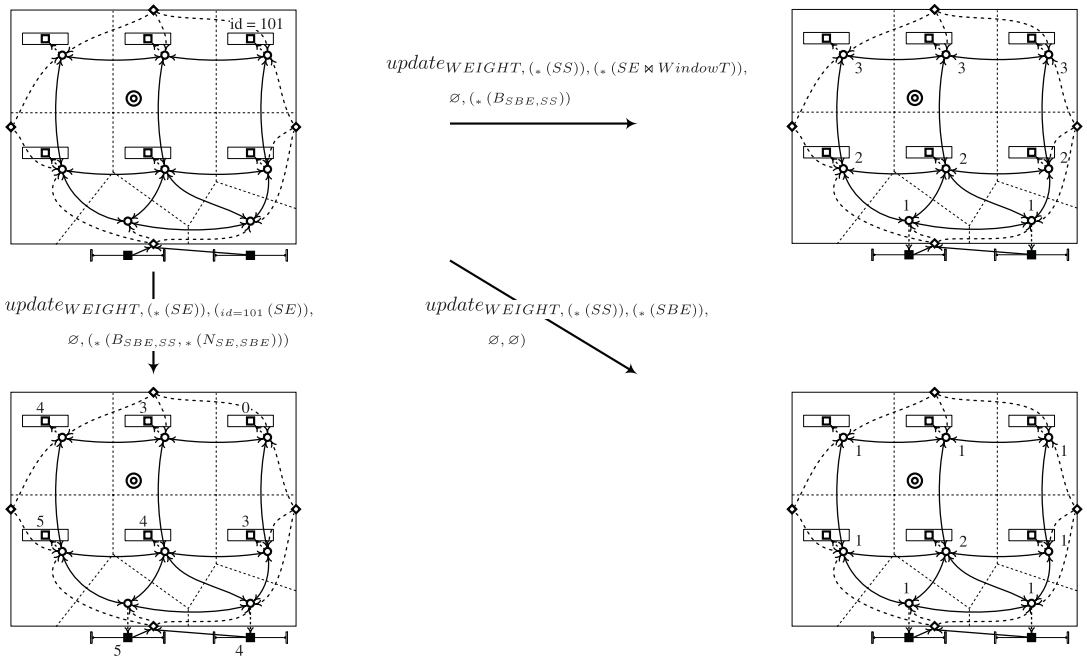

Fig. 6: Examples of the $update_{WEIGHT}$ operation. The $B_{SBE,WS}$ relation is not shown.

4. EXAMPLE OF A LAYOUT OPERATION EXPRESSION

Layout operations may be composed into expressions. This is illustrated with an example in which a lighting-specific layout is derived from an existing layout *L* (Fig. 7). *L* includes walls, doors, windows, desks, and luminaires (Fig. 7, top left). The lighting layout derived from *L* includes luminaires and windows (Fig. 7, down right). While the L-shaped whole space in the lighting layout is enclosed by windows, the rectangular whole space is not. This information could be useful e.g. for a lighting controller to dim luminaires according to daylight availability. The expression uses the four layout operations defined in Section 3:

$$aggregate_{(weight),\ \emptyset,\ \emptyset}($$
$$update_{WEIGHT,(\ *(WS)),\ (\ *(SE \bowtie WindowT)),\ \emptyset,\ \emptyset}($$
$$decompose_{\ *(SS)}($$
$$select_{(\ *(WS),\ \ *(SE \bowtie LuminaireT),\ \ *(SE \bowtie WindowT))}(L)$$
$$)))$$

Each operation modifies the layout resulting from the previous operation (respectively, *L* in case of the *select* operation):

1. Whole spaces, luminaires, and windows are selected from *L* (*select* operation).
2. Subspaces that surround luminaires in the layout resulting from step 1 are used to decompose whole spaces (*decompose* operation).
3. The weights of whole spaces in the layout resulting from step 2 are updated with weights of shortest paths to nearest windows (*update*$_{WEIGHT}$ operation). The full *le* network is used to determine shortest paths ($P_{LE_b} = \emptyset$ and $P_{LER_b} = \emptyset$).
4. Whole spaces in the layout resulting from step 3 are aggregated by their weights (*aggregate* operation). The full *le* network is used to determine whole space connectivity ($P_{LE_b} = \emptyset$ and $P_{LER_b} = \emptyset$).

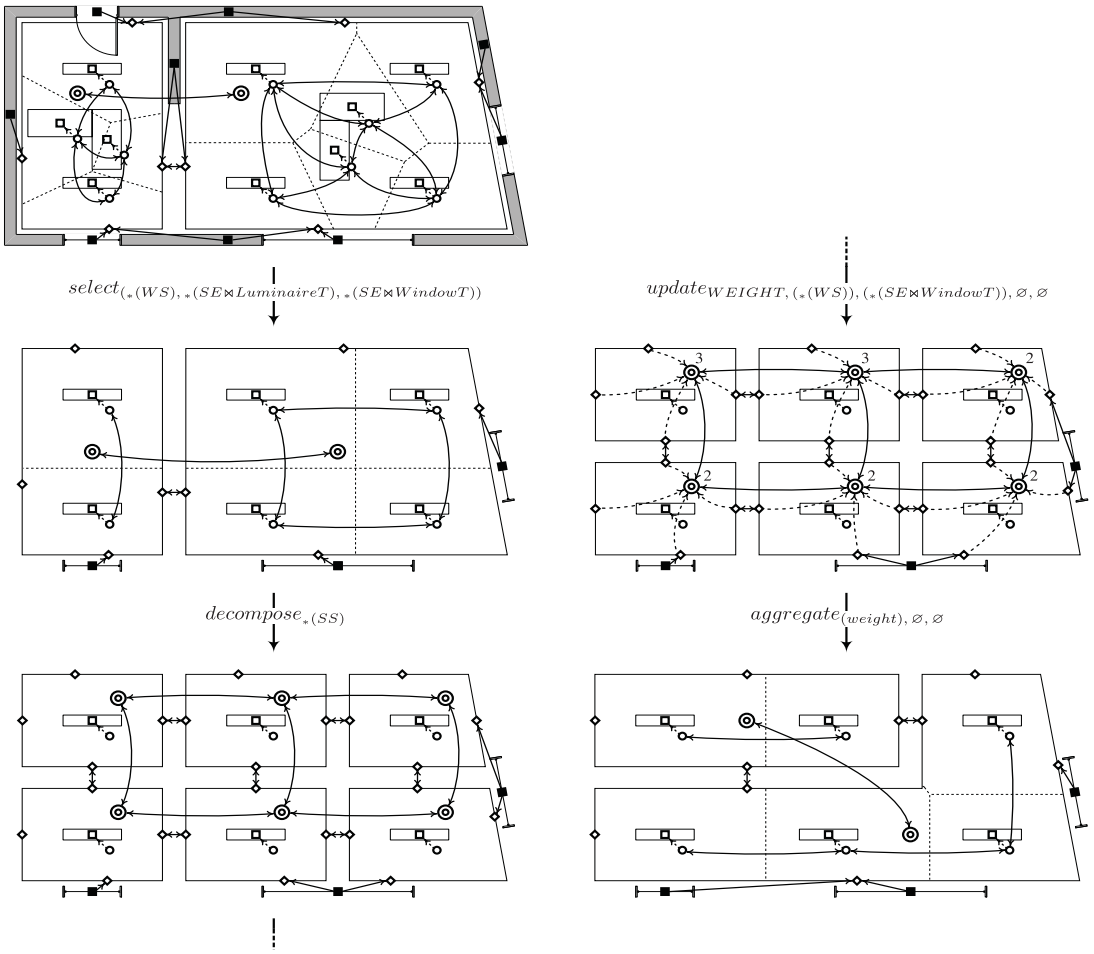

Fig. 7: Layout operation expression example. $B_{SBE,WS}$ and $B_{SBE,SS}$ relations (respectively, the $B_{SBE,SS}$ relation in the layout middle right) are not shown.

5. CONCLUSION

Operations on network-based space layouts have been introduced that are closed and may be composed into operation expressions. Variants of the proposed operations and additional operations may be defined to enhance the expressiveness of an emerging layout algebra. Binary layout operations are conceivable, including *union*, *intersection*, and *difference* operations. Similarly, there may be operations that relate *le*s in different layouts. For example, whole spaces in two layouts may be related by containment. With a rich 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 and to apply those views to individual buildings. Certain views may be sufficiently generic for reuse across buildings.

ACKNOWLEDGEMENT

The author thanks Filip Petrushevski and Milos Sipetic for discussions on the concepts presented in this paper. The author gratefully acknowledges the support by grant Austrian Science Fund (FWF): P22208.

REFERENCES

Aurenhammer F. and Klein R. (2000). Voronoi diagrams, Handbook of Computational Geometry (Sack, J. and Urrutia, G., editors), Elsevier, 201-290.

Bjoerk B.-C. (1992). A conceptual model of spaces, space boundaries and enclosing structures, Automation in Construction, Vol. 1, No. 1, 193-214.

Bondy A. and Murty U. (2010). Graph theory, Springer.

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

Eastman C. and Siabiris A. (1995). A generic building product model incorporating building type information, Automation in Construction, Vol. 3, No. 1, 283-304.

Ekholm A. and Fridqvist S. (2000). A concept of space for building classification, product modelling and design, Automation in Construction, Vol. 9, No. 3, 315-328.

Hillier B. and Hanson J. (1989). The social logic of space, Cambridge University Press.

ISO/IEC (2008). SQL:2008 (ISO/IEC 9075).

Liggett R. S. (2000). Automated facilities layout: past, present and future, Automation in Construction Vol. 9, No. 2, 197 - 215.

Rosenman M. and Gero, J. (1996). Modelling multiple views of design objects in a collaborative environment, Computer-Aided Design, Vol. 28, No. 3, 193-205.

Silberschatz A., Korth H. and Sudarshan S. (2006). Database system concepts, McGraw-Hill.

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

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'.