Educational Facility Design with Graph Grammar Systems

E. Grabska, B. Strug, G. Ślusarczyk
Jagiellonian University, Poland
barbara.strug@uj.edu.pl

Abstract. This paper deals with graph grammar systems which are used to generate representations of complex designs. Space layouts of designs have their internal representations in the form of attributed graphs representing both topological structures and semantic properties. The system of graph grammars, which work in parallel on the common graph independently generating layouts of different design components, is proposed. The grammars of the system jointly generate a set of graphs representing design task solutions with required properties. The presented approach is illustrated through the example of designing kindergarten facilities.

1. Introduction

This paper deals with graph grammar systems that are a useful approach to coordinate the actions among generative subsystems involved in the generation of complex designs. The conceptual stage of the design related to layouts and arrangement of spaces in educational facilities are studied to illustrate the power of this approach in design automation.

Educational facilities are composed of many different design components based on various functional and organizational requirements. Designing such a facility requires thus integrating this expectations into one project. Hence there is a need for a formal model that would allow both to represent the structure of the whole design in an integrated way and to describe separately diverse aspects of the design process.

In this paper the externalization of early design solutions is made with the use of various visual editors in order to design layouts of spaces, express relations between their functional areas, and design space arrangements. All types of spaces are arranged using an appropriate equipment. The designer's drawings of space layouts are automatically transformed into their internal representations in the form of attributed graphs representing both topological structures and semantic properties of educational facilities (Grabska & Borkowski, 1996). Such graphs can be generated by a graph grammar (Rozenberg, 1999). The main problem of graph generation with graph grammars lies in the complexity and size (understood as the number of rules) of grammars needed in real world problems (that can be encountered in the domains of both engineering and design). Such grammars are also very difficult to define. Moreover, as the design of educational facilities has to encompass different components (a building with different zones, sports facilities, playground, recreational areas etc.) the grammar has to contain many unrelated or loosely related sets of productions.

It thus seems advantageous to use a number of simpler grammars, called a grammar system, cooperating in an effort to generate a design, instead of one complex grammar. A grammar system consists of a finite number of grammars which together influence (change) an environment by applying some operations to it. At a given moment the state of the system is described by the current environment (sub-environments). The system works by changing its state.

In our approach it is assumed that at any given time step there is only one graph being generated. The grammars of the proposed system work in parallel on the common graph independently generating solutions of subproblems. Each system grammar operates on this

CEUR-WS.org/Vol-2394/paper20.pdf

graph according to a communication protocol called the derivation mode. In case of educational facilities design the different elements of such a facility are designed independently with only occasional cooperation when common elements are being processed or common recourses are considered. The grammars of the system jointly generate a set of graphs representing design task solutions with properties specified by graph attributes. Grammar rules of the system are designed in a GraphTool framework (Ryszka & Grabska, 2013).

This approach can be seen as an equivalent of many "design teams" working on different parts of the same design. The only requirement here seems to be ensuring that no two teams work at the same time on the same part of the design, i.e. no two grammars operate on the same nodes of the current graph form. This problem is solved here by introducing a special set of labels which allow for activation of different system grammars at selected locations (Grabska et al., 2008).

The proposed generative method allows for a flexible approach to the design of complex facilities. Depending on the additional requirements some system grammars can be added or deleted from the design system thus representing the presence of some optional elements. The proposed system supports generation of alternative facilities models with various arrangement of spaces, which can be easily adapted to different needs of older and younger children. The presented approach is illustrated through the example of designing kindergarten facilities.

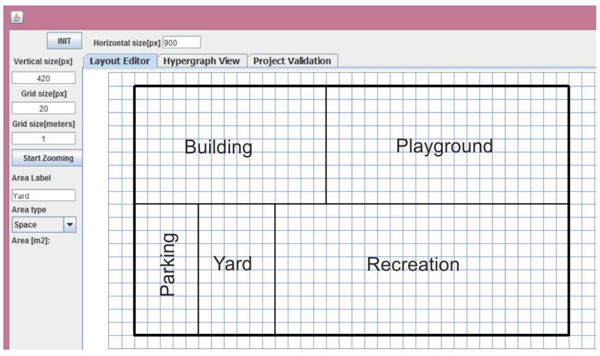
2. CP-graph Representation of Space Layouts

Usually the design process is started by the designer who formulates a set of design requirements on the basis of a design task. These requirements together with legal norms and standards constitute the design criteria for a given project. In our approach to design kindergarten facilities the visual editor (Grabska et al., 2009) supports the conceptual stage of the design by enabling the designer to browse the existing space layouts representing various arrangements of required areas. These layouts are composed of six possible types of areas representing spaces intended for a kindergarten building, yard, playgrounds, recreational areas, sports facilities, and parking lots.

Four possible layouts of kindergarten facilities are presented in Fig.1. The labels *Building, Yard, Playground, Recreation, Sport* and *Parking* denote the areas with specific purpose. The layout presented in Fig.1c contains two areas labeled *Playground*, as the one of them is planned for younger children, while the second one for older children.

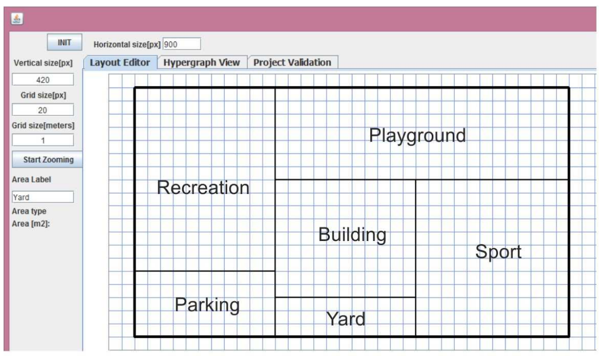
If one of the presented designs satisfies the designer's requirements and expectations he can select it as an initial layout. Using the editor the designer can also modify any of the presented layouts or create a new one. The layout indicated by the designer becomes a starting point for the further development of a task solution.

Layouts of kindergarten facilities have their internal representations in the form of composition graphs (CP-graphs) (Strug et al., 2017). CP-graphs are labelled and attributed graphs, which represent both topological structures and semantic properties of educational facilities (Grabska & Borkowski, 1996). Nodes of these graphs represent different types of spaces, like a yard, playground, recreational area, spaces for playground facilities, and sports facilities. The type of a component represented by a node is determined by its label. Moreover nodes have attributes specifying properties of components assigned to them. To each node attributes specifying the length, width, area and the type of bedding of the corresponding component are assigned. Bonds assigned to nodes denote possible connections between layout components. Bonds are connected by edges representing spatial relations between components and labelled by the relation names.



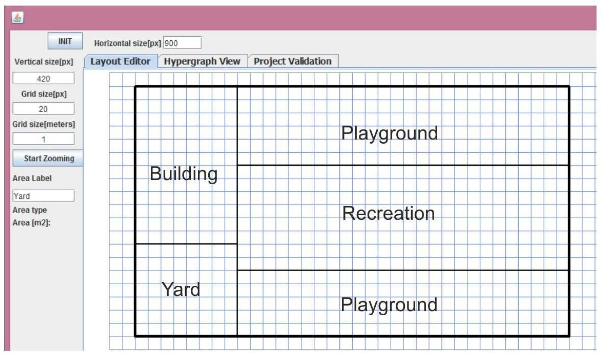
Horizontal size[px] 900
Vertical size[px] 420
Grid size[px] 20
Grid size[meters] 1
Area Label Yard
Area type Space
Area [m2]:

a



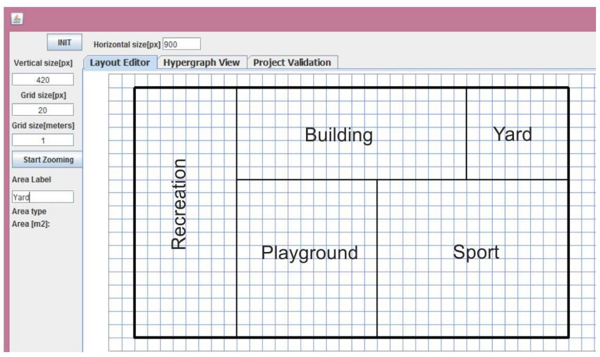
Horizontal size[px] 900
Vertical size[px] 420
Grid size[px] 20
Grid size[meters] 1
Area Label Yard
Area type
Area [m2]:

b



Horizontal size[px] 900
Vertical size[px] 420
Grid size[px] 20
Grid size[meters] 1
Area Label Yard
Area type
Area [m2]:

c



Horizontal size[px] 900
Vertical size[px] 420
Grid size[px] 20
Grid size[meters] 1
Area Label Yard
Area type
Area [m2]:

d

Figure 1: Four space arrangements of kindergarten facilities

The formal definition of a CP-graph is as follows.

Let Σ be a finite alphabet used to label object nodes, bond nodes and edges. Let A be a nonempty, finite set of attributes. For each attribute $a \in A$, let D_a be a fixed, nonempty set of its admissible values, known as the domain of a.

A CP-graph G is a tuple $G = (V, B, E, bd, s, t, lab, atr)$, where:

1. V, B, E are pairwise disjoint finite sets, whose elements are respectively called nodes, bonds and edges,
2. $bd: V \rightarrow 2^B$ is a function assigning sets of bonds to nodes in such a way that $\forall x \in B\ \exists!\ y \in V: x \in bd(y)$, i.e., each bond belongs to exactly one node,
3. $s, t: E \rightarrow B$ are functions assigning to edges source and target bonds, respectively, in such a way that $\forall e \in E\ \exists x, y \in V: s(e) \in bd(x) \wedge t(e) \in bd(y) \wedge x \neq y$,
4. $lab: V \cup B \cup E \rightarrow \Sigma$ is a labelling function,
5. $atr: V \cup B \cup E \rightarrow 2^A$ is an attributing function.

Each of the layouts of kindergarten facilities present or created in the visual editor has its CP-graph representation. Thus the CP-graph corresponding to the space layout selected by the designer becomes the starting graph for the system of grammars, which work in parallel on this graph jointly generating CP-graphs representing design task solutions.

CP-graphs representing space layouts presented in Fig. 1 are shown in Fig. 2. The CP-graph nodes representing different areas intended for a building, yard, playground, recreation, sport and parking are labeled as *BL*, *YR*, *PL*, *RC*, *SP*, and *PR*, respectively. Bonds assigned to nodes represent sides of areas and are drawn as small circles. The edges represent spatial adjacency between components. As the only one relation between components is considered, the edge labels are omitted.

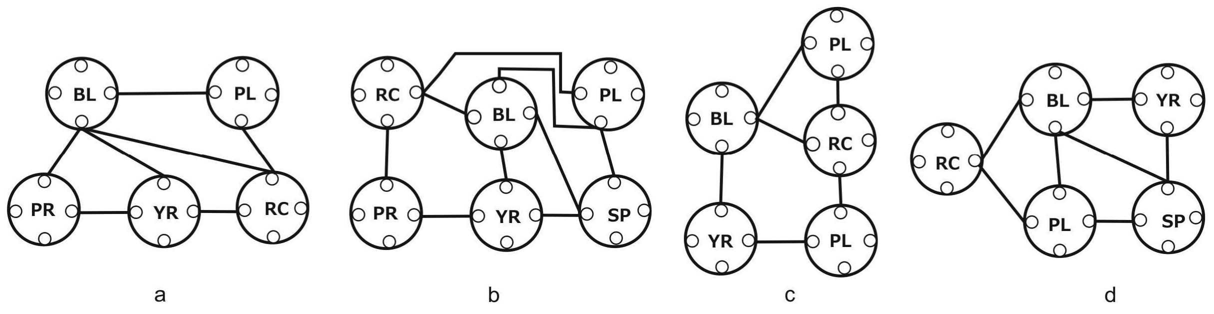


Figure 2: CP-graphs representing space layouts from Fig. 1

3. CP-graph Grammars

CP-graph representing structures of educational facilities designs are generated by CP-graph grammars. A CP-graph grammar is composed of a set of productions and an axiom being an initial CP-graph. In order to apply grammar rules some bonds of CP-graphs occurring in these rules are specified as external ones. Let $G = (V, B, E, bd, s, t, lab, atr)$ be a CP-graph. A partial function $ext: B \rightarrow \aleph$, where $\aleph$ is a set of integers, determines external bonds of G.

A CP-graph transformation rule is of the form $p = (l, r, sr)$, where l and r are attributed CP-graphs with the same number of different values of external bonds and sr is a set of functions specifying the way in which attributes assigned to nodes of l are transferred to the attributes assigned to nodes of r. Each attribute has a set of possible values, which are established during the rule application. The application of the rule p to a CP-graph G consists in substituting r for a CP-graph being an isomorphic image of l in G, replacing external bonds of the CP-graph being removed with the external bonds of r with the same numbers, and specifying values of attributes assigned to elements of r according to functions of sr. After inserting r into a host CP-graph all edges which were coming into (or out of) a bond with a given number in the CP-graph l, are coming into (or out of) bonds of r with the same number.

The formal definition of a CP-graph grammar is as follows.

Let N and T denote sets of non-terminal and terminal node labels, respectively.

A CP-graph grammar GG over N and T is a pair $GG = (P, S)$, where:

1. P is a finite set of productions of the form $p = (l, r, sr)$ satisfying the following conditions:

   – l and r are attributed CP-graphs such that the set of numbers defined for bonds of l by ext is the same as the set of numbers defined by ext for r, i.e., $set(ext(B_l)) = set(ext(B_r))$,

   – l contains at least one node with a label of N (i.e., $\exists v \in V_l: lab_l(v) \in N$),

2. S is an initial attributed CP-graph containing at least one node with a label of N, and called an axiom of GG.

The chosen rules of a CP-graph grammar, which generates CP-graph representations of building floor layouts, are shown in Fig. 3. Non-terminal labels of nodes start with a capital letter, while terminal labels are written in lower case letters. The rule $p1$ divides the floor area into a communication area represented by a node labelled Cmc, two play areas represented by nodes labelled as *Play*, a dining area (*Dining*), an area for bathrooms (*Bthrs*), an entrance, an utility room and a cloakroom. The productions *p2* and *p3* place one or two playrooms in each play area. The embedding transformation for all these productions is such that all edges which are connected in the host CP-graph with bond number *i* of the left-hand side of a production are replaced by edges connected with all bonds with number *i* on the right-hand side of this production. The way of determining the values of the attribute *area* for the nodes of the production right-hand side CP-graph in respect to the value of this attribute specified for the left-hand side CP-graph is shown for productions *p2* and *p3*. One of the possible floor layouts corresponding to a CP-graph generated by the presented CP-graph grammar is shown in Fig.4.

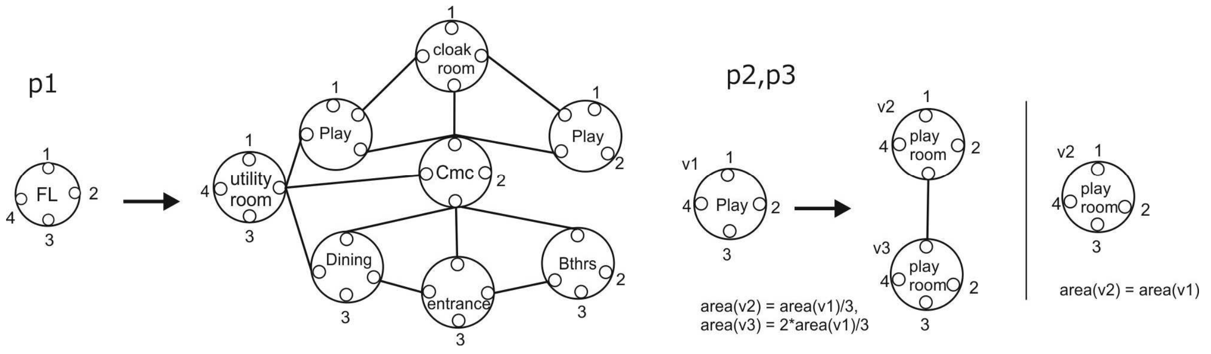
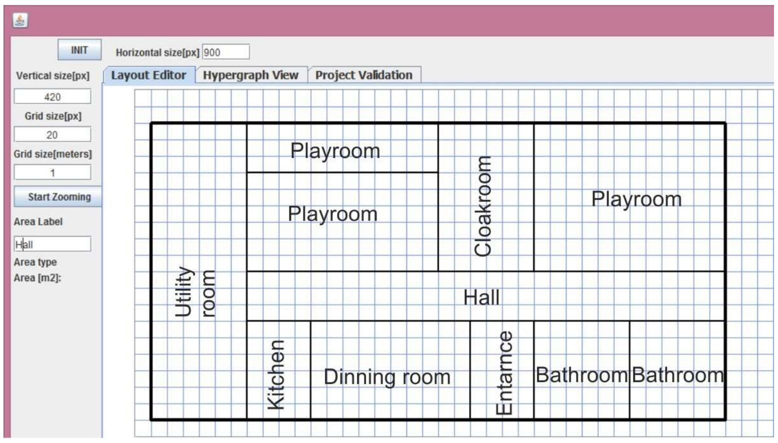


Figure 3: The chosen rules of a CP-graph grammar generating representations of floor layouts



Horizontal size[px] 900

Vertical size[px] 420

Grid size[px] 20

Grid size[meters] 1

Area Label Hall

Area type

Area [m2]:

Figure 4: A floor layout corresponding to a CP-graph generated by the CP-graph grammar from Fig.3

In Fig. 5 twelve selected rules of a CP-graph grammar, which supports the user in designing various arrangements of playground facilities, are presented. The productions *p1* and *p2* generate arrangements of areas with bedding filled with grass (nodes labelled by *Lawn*), rubber mat (*Rubber*) and synthetic grass (*Synthetic*). The fourth type of area in the playground can be planted with trees (*Trees*). Productions *p3* and *p4* allow for generating an arbitrary number of trees, but at least two trees should be present. To production *p3* a variable, which specifies the maximum number of times this production can be applied, is attached. Productions *p10*, *p11* and *p12* enable us to obtain lawns on which there are paths, benches and/or shrubs. The number of benches and shrubs is generated in the same way as trees by means of productions *p3* and *p4*. Productions *p5* and *p6* generate arrangements of activity towers, swings, and spring rockers, which are located on rubber mats, while productions *p7* and *p8* generate arrangements of carousels, outdoor boards and play houses with sandpits, which are located on synthetic grass. Production *p9* is a context sensitive one, as its left-hand side CP-graph contains more than one node. This production allows us to place a bench between a sandpit and boards if this sandpit is to be located near outdoor boards. The area of spaces and the material used for different types of beddings are specified by graph attributes. Attributes allow also for selecting the right playground equipment for an appropriate age group (for children of age (3-4) and (5-6)). This selection should be taken into account for example when the equipment for the youngest children (age 3-4) is placed. Their equipment should be located close to the place where the kindergarten teachers are.

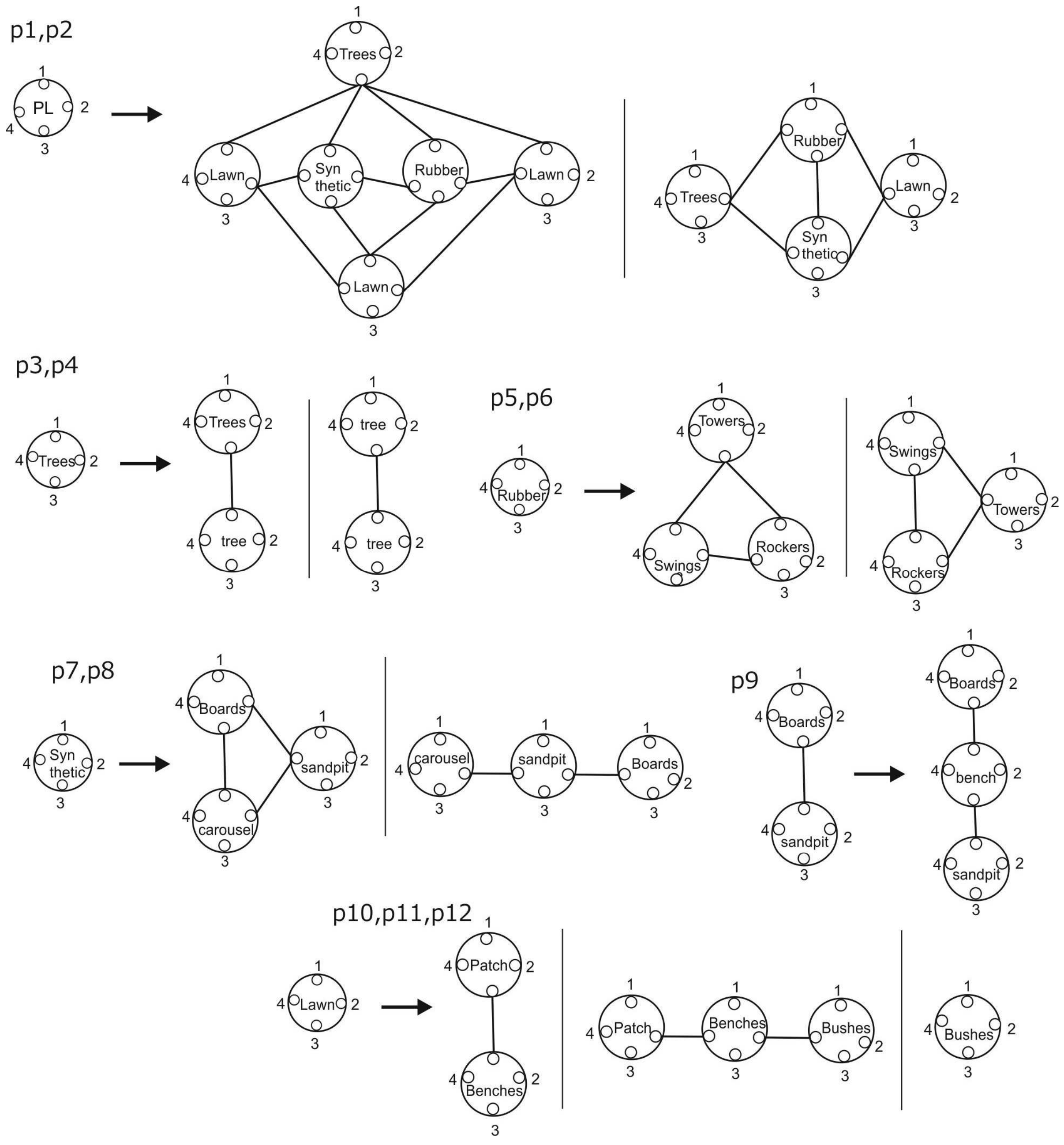


Figure 5: Twelve selected rules of a CP-graph grammar generating representations of playgrounds

Fig. 6 shows three examples of playground designs with equipment. The following categories of equipment are presented: activity towers (1, 3), swings (2, 7), spring rockers (6,8), carousel (9), outdoor tic-tac-toe board (5), and kids play house with sandpit (4).

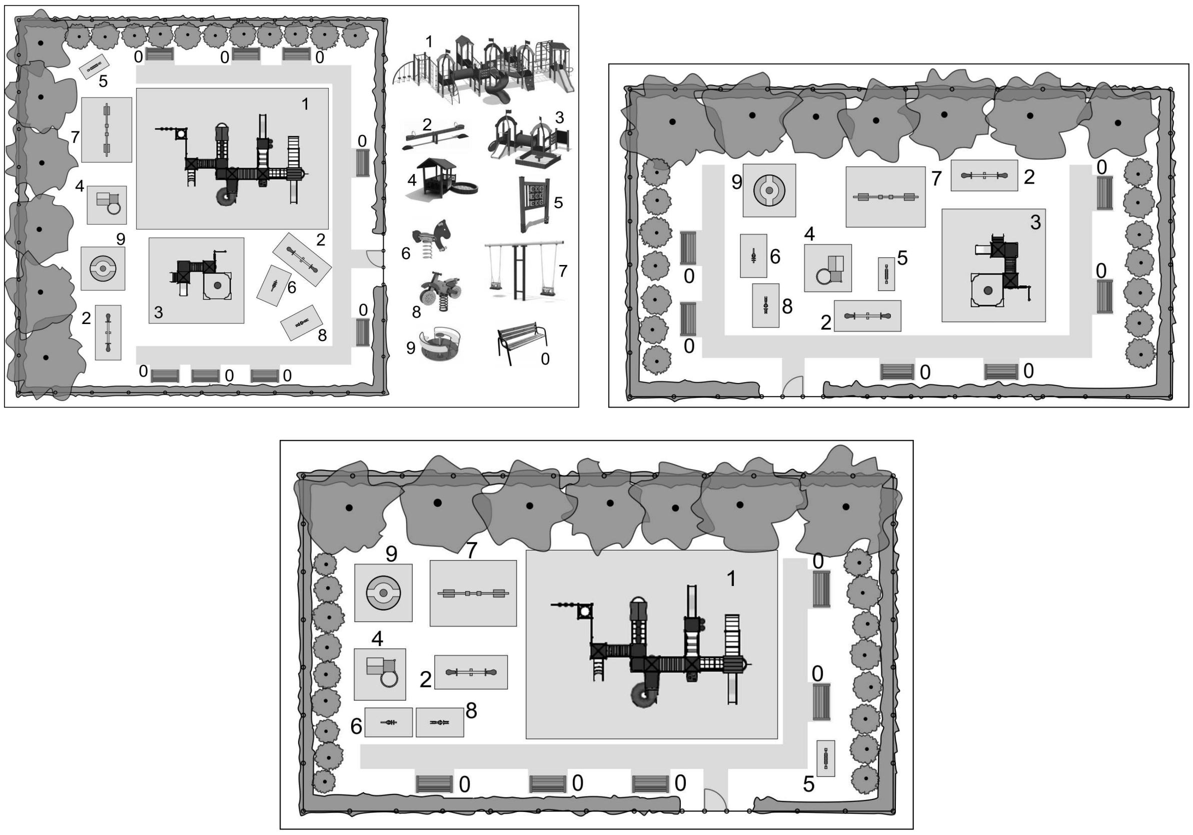


Figure 6: Three examples of playground designs with equipment

In Fig. 7a a recreation area arrangement, which corresponds to a CP-graph generated by CP-graph grammar supporting the design of various arrangements of trees, shrubs, flower beds, paths, bridges, benches and stones, is presented. A fragment of this arrangement and a corresponding subCP-graph are shown in Figs. 7b and 7c, respectively. The CP-graph nodes represent seven path segments, four flower beds, a tree, a bench, and a bridge.

The described CP-graph grammars are created with the use of *GraphTool*, which is an Integrated Development Environment providing tools necessary to define graphs, and graph transformation rules. It is implemented in the Java language as a plugin for the Eclipse framework, and is based on the EMF library (Eclipse Modeling Framework). By replacing CP-graph subgraphs in the way specified in CP-graph rules the *GraphTool* generates different CP-graphs representing possible arrangements of kindergarten facilities. The possibility of relating attributes of right-hand sides of CP-graph rules to attributes of their left-hand sides enables us to capture parametric modeling knowledge.

4. A Grammar System

As the kindergarten is actually a composition of different design components the design of each of them can be described by an independent grammar. Each of such grammars can be seen as an independent "design team" working on its own part of a design with minimal or no contact/interference/communication with other teams. Only when a given team does not know how to deal with a certain part of a design it can call another team – such an action is represented by triggering a specialized grammar able to solve a particular design subproblem (for example to design a part of a playground or to design a recreation area).

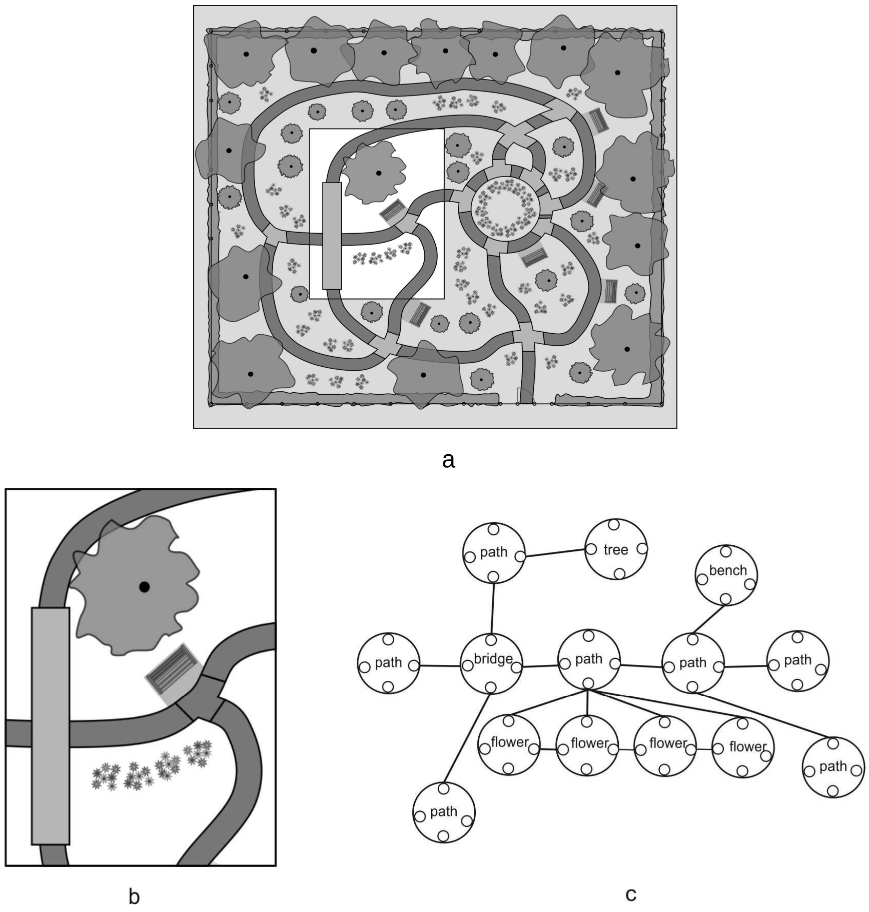


Figure 7: a) A recreation area arrangement, b) a fragment of this area, c) a CP-graph representation of this fragment arrangement

A family of grammars is called a grammar system. A grammar system consists of a finite number of grammars which can work together to change an environment by applying some operations to it. At a given moment the state of the system is described by the current environment (sub-environments). The system works by changing its state.

In general usually two types of grammar systems are distinguished: parallel communicating grammar systems (PC grammar systems) (Csuhaj-Varjú & Vaszil, 2001) and cooperating distributed grammar systems (CD grammar systems) (Csuhaj-Varjú, 2004). The main difference between these types of grammar systems consists in the way they work and communicate. In PC grammar systems each grammar operates on its own copy of the graph under derivation and they only exchange information when there is a specific request. In case of CD grammar systems the grammars operate on one common graph, one grammar at a time.

As in computer aided design domain a single object is usually developed, the CD grammar systems seem more appropriate here. They allow for a number of grammars to work together on one object. At any given time step there is only one graph being generated. Each system grammar operates on this graph according to a communication protocol called the *derivation mode*. There is a number of ways such a mode can be defined, for example it may allow a single grammar to perform a predefined number of steps or to work until no production of this grammar can be applied. The method of selecting which grammar should be used as the next "active" grammar is also important.

It has to be noticed that in case of CP-graph grammars the requirement that only one grammar can operate on a given temporary form seems too strong. Moreover in design systems it would be useful to allow more than one grammar to operate on the same common graph. The only requirement here seems to be ensuring that no two teams work at the same time on the same part of the design, i.e., no two grammars operate on the same nodes of the graph being derived.

Another aspect that has to be properly defined is a way of activating particular grammars. Such a method, called a *cooperation strategy*, may either be based on some predefined order - thus leading the system to operate under the control of an "external supervisor". Or, it can be based on dynamic activation of grammars related to the current state of a graph being derived. The way a given grammar works i.e. how long it performs its operations must also be defined. In this paper a so called *terminal derivation mode* is used, i.e. each grammar works as long as it contains a production that can be applied to a current graph.

An advantage of such an approach is the possibility to use grammars defined for different aspects of the design process. In case of adding some elements to the design it is enough to add a grammar to the grammar system. In case of a single grammar approach a grammar has to be redesigned to incorporate new production rules and new symbols. Moreover, it has to be noted that the use of a grammar system can lead to the reduction of computational costs, as different types of grammars can be used in one system. There exist different types of grammars. Context free grammars have lower computational cost than context sensitive grammars, but at the same time context sensitive grammars have higher expressive power. Yet in some cases context sensitive productions cannot be avoided. An example of such a production, which adds a bench only when a sandpit and a board are present, is depicted in Fig. 5 (production *p9*). In a single grammar adding just one context sensitive rule makes the whole grammar context sensitive. In case of a grammar system adding such a rule makes only one grammar to be context sensitive without affecting other grammar types and associated computational cost.

In the presented approach a CP-graph grammar system consists of a set of initial CP-graphs and a family of n (but at least two) CP-graph grammars, which communicate using a set of so-called *communication symbols*. This set is composed of all non-terminal labels of nodes occurring in axioms of the system CP-graph grammars. Each grammar has its own specific communication symbol, and therefore the number of communication symbols equals to the number of the system grammars.

Formally, a CP-graph grammar system defined over N and T is as a tuple

$GGS = (G_1, G_2, \ldots, G_n, C, S_0)$, where

- $G_i = (P_i, S_i)$, $(i=1 \ldots n)$ are CP-graph grammars such that each S_i is composed of only one node with a label of N,
- C is a set of communication symbols, such that $C \subset N$, is composed of all non-terminal labels occurring in grammar axioms and there are no identical communication symbols in axioms of different grammars.
- S_0 is a set of initial CP-graphs.

Nodes labelled with terminal, non-terminal and communication symbols will be called, terminal, non-terminal and communication nodes, respectively.

Each system grammar contains productions that have at least one node on the left side labelled by a non-terminal symbol. The nodes of the graph on the right side of the production can labelled by any symbol, terminal, non-terminal or communication one.

The non-terminal nodes can be intuitively understood as representing a part of a design that is not finished but a given grammar knows how to deal with it further. Taking this intuition further a communication symbol means that a particular grammar knows that some part of design is not finished but it is not "specialist" in this part and thus it "communicates" with other grammar that knows how to deal with this part. Terminal symbols represent part of a design that is considered to be finished.

The definition of a grammar system guarantees that if any grammar introduces a communication symbol into the graph, then at least one other grammar exists such that it contains an axiom node labelled by the same symbol. Thus communication symbols occurring in the derived graph allow for triggering appropriate system grammars.

For the kindergarten design, let $C = \{BL, PL, PR, YR, RC, SP\}$ and S_0 contain among other all of the CP-graphs representing initial space layouts depicted in Fig. 2. In this case after selecting an initial graph from S_0 a graph system can be run. Each of the labels of the initial graph triggers one grammar from the proposed CP-grammar system.

In Fig. 8 an example of the application of a grammar system is presented. In the first step the user selects an initial layout. In this case, the layout depicted in Fig. 1c and represented by the initial CP-graph depicted in Fig. 2c is selected. This CP-graph contains five nodes labelled with communication symbols: two of them are labelled by *PL*, and three nodes by *YR*, *BL* and *RC*, respectively. In the next step, four grammars from the grammar system are triggered by their respective symbols. For example the symbol *BL* triggers the grammar responsible for the design of the layout of the building. As there are two nodes labelled with the communication symbol *PL*, the grammar responsible for the design of the playground area is triggered twice. As the grammar is non-deterministic it allows us to obtain different designs in both areas. The productions can be selected either randomly or may be determined by attributes assigned to communication symbols (for example they may include a number or type of objects expected in a given area). It can be observed that by running the same process for a second time we may obtain a slightly different design but still following some basic structure.

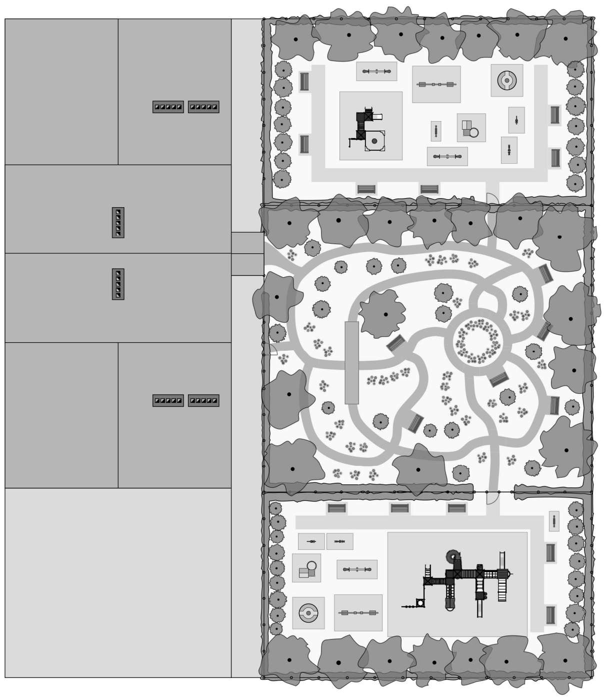


Figure 8: A design of kindergarten facilities corresponding to the CP-graph obtained using the proposed CP-graph grammar system

5. Conclusions

In this paper a graph grammar system, which supports the conceptual stage of designing layouts and arrangement of spaces in educational facilities is considered. The main advantage of such approach is the possibility of adding specialized grammars when a different type of space is required.

This system is equipped with a visual editor, where space layouts representing various arrangements of required areas are presented, and *GraphTool* supporting defining CP-graphs, CP-graph grammar rules, as well as applying them.

Up to now we have considered only one-storey kindergarten buildings. In case of designing higher buildings, the hierarchical CP-graphs will be used to represent their topological structure. In such a CP-graph each hierarchical node represents one floor and contains the CP-graph representing the space arrangement of this floor. At first the designer specifies for each floor the general arrangement of functional areas with places intended for stairs and lifts. These arrangements are represented by CP-graphs, which are further developed by a CP-graph grammar generating CP-graph representations of building floor layouts composed of specific rooms and relations among them.

In the next step of our research, we intend to define a control diagram for the whole graph grammar system. This will require adding "control" productions to the component graph grammars supporting verification of the selection of individual graph grammars in the process of generating complex projects.

The kindergartens presented in this paper are just one of the types of buildings illustrating the advantages of a graph grammar system in their design process. Designing hotel buildings with more demanding requirements is another example that would be worth considering in the framework of a graph grammar system.

References

Grabska, E., Borkowski, A. (1996). Assisting Creativity by Composite Representation. Artificial Intelligence in Design '96, Kluwer Academic Publishers, pp. 743-759.

Rozenberg, G. (1999). Handbook of Graph Grammars and Computing by Graph Transformations, vol.2 Applications, Languages and Tools, World Scientific, London.

Ryszka, I., Grabska, E. (2013). GraphTool - a new System of Graph Generation, ADVCOMP 2013, The Seventh International Conference on Advanced Engineering Computing and Applications in Sciences, ISBN: 978-1-61208-290-5, pp. 79-83.

Grabska, E., Strug. B., Ślusarczyk, G., Grabski W. (2008). Grammar-Based Distributed Design, in L. Rutkowski et al. (eds.), Computational Intelligence: Methods and Applications, EXIT, Warszawa, pp. 493-502.

Csuhaj-Varjú, E., Vaszil. Gy. (2001). On context-free parallel communicating grammar systems: Synchronization, communication, and normal forms. Theoretical Computer Science, 255, pp. 511-538.

Csuhaj-Varjú., E. (2004). Grammar systems: A short survey, Proceedings of Grammar Systems Week 2004, pp. 141-157, Budapest, Hungary.

Strug, B., Ślusarczyk, G., Grabska, E. (2017). A graph-based generative method for supporting bridge design, 24th EG-ICE International Workshop on Intelligent Computing in Engineering (EG-ICE 2017), Nottingham, pp. 294-302.

Grabska, E., Borkowski, A., Palacz, W., Gajek, Sz. (2009). Hypergraph System Supporting Design and Reasoning. In: Wolfgang Huhnt, ed., Computing in Engineering, EG-ICE Conference 2009, pp. 134-141.