DYNAMIC CLASS MODEL IN OBJECT-ORIENTED DESIGN

Marcin Skowron

Institute of Computer Science
Jagiellonian University
Nawojki 11, 30-072 Kraków, Poland
Tel. (+48-12) 632-3355 Fax (+48-12) 634-1865
Marcin.Skowron.Student@softlab.ii.uj.edu.pl

Abstract

This paper proposes design with the help of objet-oriented modelling. The class model of designed object can be dynamically changed by a system. Thanks to the specificity of the class model we receive new emergent solutions and a warranty of space sparing. The advantage of the proposed approach is a well-grounded theoretical base and polynomial time of operations. It gives a possibility of practical applications and extensions of proposed model.

Keywords: Object-oriented design. **Abbreviated title**: Dynamic class model.

1. Classes, objects and their relationships

In object-oriented modelling we are using the fact that in perception of world people use three methods of organisation:

- differentiation of experience on individual objects and attributes,
- distinction between whole objects and their parts,
- creation classes of objects and distinction between them [1].

In object-oriented modelling, classes, objects, and their relationships are the primary modelling elements. Classes and objects model the part of the reality describing in the system. The relationships between them describe the structure of this part.

An object is an item we can talk about and manipulate. A class is a description of an object type. All objects are instances of the class, where the class describes the properties and behaviour of one type of objects.

The relationships that can be used are e.g. *associations*, *aggregations* (which is a special case of associations) and *generalisations*:

- an association is a connection between classes, which means that it is also a connection between objects of those classes,
- an aggregation indicates that the relationship between classes is some sort of "whole-part",

- a generalisation is a relationship between a more general and more specific element. The more specific element can contain only additional information. An instance (an object is an instance of a class) of the more specific element may be used wherever the more general element is allowed.

A class diagram is a model type. A class diagram describes the view of system in terms of classes and relationships among the classes. An object diagram shows specific instances of those classes and specific links between those instances.

2. Dynamic class model

Changes of a class model of created software most often demand programmer interference. For example, when cash machines have arisen, bank software has been modified with a class “cash machine” and right relationships.

It seems that in systems aiding design of hierarchical objects (for example floor layouts) the class model should be dynamically changed by a system. In this case the class model can be used as a part of knowledge base of objects already created (see Sec. 8).

This approach gives a warranty of space sparing thanks to the generalisation. It will be presented that thanks to specificity of class model we receive new emergent solutions (see Sec. 9).

The formal definition of the class model is created using one of the most powerful models used in computer science – graphs, exactly hypergraphs. It is defined a hierarchical hypergraph schema – equivalent to the class model. The hierarchical hypergraph schema is a hypergraph in which one subset of nodes and two subsets of edges are distinguished. Nodes from the distinguished subset of nodes describe common features of others nodes and are called abstract nodes. Distinguished subsets of edges correspond to the relation of generalisation („is-a”) and to the relation of aggregation („whole-part”).

3. Example – design of floor layouts

We start with a simple example of representing two floor layouts by means of the proposed formalism [3][4].

Fig.1.

Let us consider two floor layouts shown in Fig. 1. Fig. 2 and Fig. 3 present structures of floor layouts shown in Fig.1a and Fig.1b, respectively. The structures are described by means of hierarchical hypergraphs, which correspond to an object diagram [2]. The hierarchical hypergraph is a hypergraph in which a subset of edges corresponding to the relation of aggregation is distinguished (Definition 5.3). In Fig. 2 and Fig. 3 rectangles with round corners represent nodes. Edges are represented by rectangles with lines spreading out of them. Edges corresponding to the relation of aggregation are marked with a diamond and labelled with ‘A’.

Edges corresponding to the relation of wall between rooms are labelled with ‘W’ and to the relation of wall with a door between rooms are labelled with ‘P’.

The structures described by means of hierarchical hypergraphs in Fig. 2 and Fig. 3 are compatible with the schema of a hierarchical hypergraphs presented in Fig. 4 (Definition 6.7). The schema of hierarchical hypergraphs corresponds to a class diagram [2]. Compatible means that hierarchical hypergraphs in Fig. 2 and Fig. 3 are instances of the schema of a hierarchical hypergraphs in Fig. 4. Hierarchical hypergraphs compatible with the schema of a hierarchical hypergraph constitute a subset of the set of all hierarchical hypergraphs. The schema of a hierarchical hypergraph is a description of this subset. It is possible to observe common parts for all the elements of this subset and differences between them.

Fig. 2.

The hierarchical hypergraph schema is a hypergraph in which one subset of nodes and two subsets of edges are distinguished (Definition 5.1). Nodes from the distinguished subset of nodes do not have their own instances. They describe common features of others nodes and are called abstract nodes. In the schema presented in Fig. 4 there are five abstract nodes labelled “Ground floor {abstract}”, “Hall {abstract}”, “Kitchen {abstract}”, “Dining room {abstract}” and “Living room {abstract}”. Distinguished subsets of edges correspond to a relation of generalisation („is-a”) and to a relation of aggregation („whole-part”). In Fig. 4 edges corresponding to the relation of generalisation are marked with a triangle and labelled with ‘G’.

Fig. 3.

We can see that the house is defined as an object consisting of a ground floor. The ground floor consists of a hall, a bathroom, a dinning-living part and some other elements (for example rooms). The dinning-living part has to consist of rooms functioning as a kitchen, a dining room and a living room. Floor layouts described in Fig. 2 and Fig. 3 differ in the structure of the dinning-living part. Moreover nodes labelled “Hall 2”, “Ground floor 2” in structure of floor layouts in Fig. 3 are specialisations of nodes labelled “Hall 1”, “Ground floor 1” in structure of floor layouts in Fig. 2, respectively.

Fig. 4.

To use a dynamic class model in systems aiding design we should define: a hierarchical hypergraph schema (equivalent to the class model), a hierarchical hypergraph (equivalent to an object model) (Sec. 5), when the hierarchical hypergraph is an instance of the schema, how to receive the hierarchical hypergraph from the schema (Sec. 6) and how to extend the schema (Sec. 7).

4. Preliminaries - hypergraphs

In this section some concepts which are necessary in the sequel are presented. The definition of a hypergraph is recalled [5][6].

Definition 4.1 Let C be a finite set, called the label alphabet. A *hypergraph* (over C) is a system $H = (V, E, s, t, lab_V, lab_E)$ where V is a finite set of *hypernodes*, E is a finite set of *hyperedges*, $s : E \rightarrow V^*$ is a *function of sources of hyperedges*[1], $t : E \rightarrow V^*$ is a *function of targets of hyperedges*, $lab_V : V \rightarrow C$ is a *node labelling function*, $lab_E : E \rightarrow C$ is an *edge labelling function*. The set of all hypergraphs over C is denoted by H_C. □

Definition 4.2 Let $H = (V, E, s, t, lab_V, lab_E)$, $H' = (V', E', s', t', lab'_V, lab'_E) \in H_C$. H is a *subhypergraph* of hypergraph H', what is denoted $H \subseteq H'$, if $V \subseteq V'$, $E \subseteq E'$, $s = s'|_E$, $t = t'|_E$, $lab_V = lab'_V|_V$, $lab_E = lab'_E|_E$. □

Definition 4.3 Let $H = (V, E, s, t, lab_V, lab_E)$, $H' = (V', E', s', t', lab'_V, lab'_E) \in H_C$. H is *isomorphic* with H' if $\exists\ h_V : V \rightarrow V'$, $h_E : E \rightarrow E'$ - bijections $\forall\ e \in E, v \in V : s'(h_E(e)) = h^*_V(s(e)) \ \wedge\ t'(h_E(e)) = h^*_V(t(e)) \ \wedge$ $lab_E'(h_E(e)) = lab_E(e) \ \wedge\ lab_V'(h_V(v)) = lab_V(v)$. □

Definition 4.4 Let $H_1 = (V_1, E_1, s_1, t_1, lab_{V1}, lab_{E1})$, $H_2 = (V_2, E_2, s_2, t_2, lab_{V2}, lab_{E2})$, $H_3 = (V_3, E_3, s_3, t_3, lab_{V3}, lab_{E3}) \in H_C$, $H_1 \subseteq H_3$, $H_2 \subseteq H_3$. *The sum of hypergraph H_1 and hypergraph H_2* we call hypergraph $H = (V, E, s, t, lab_V, lab_E)$ such that: $V = V_1 \cup V_2$, $E = E_1 \cup E_2$, $s = s_3|_E$, $t = t_3|_E$, $lab_V = lab_{V3}|_V$, $lab_E = lab_{E3}|_E$, what we denote $H = H_1 \cup H_2$. □

The following definitions are my propositions of a directed hypergraph, a hypergraph tree and some auxiliary concepts.

Definition 4.5 Let $H_1 = (V_1, E_1, s_1, t_1, lab_{V1}, lab_{E1})$, $H_2 = (V_2, E_2, s_2, t_2, lab_{V2}, lab_{E2})$, $H = (V, E, s, t, lab_V, lab_E) \in H_C$, $H_1 \subseteq H$, $H_2 \subseteq H$. *The set of edges connecting the hypergraph H_1 with the hypergraph H_2* is defined as:

$$
\begin{aligned}
E(H_1,H_2;H) = \{\, e \in E : \exists\ i, j \in N : {} & [\, s_i(e) \in V_1 \setminus V_2 \ \wedge\ t_j(e) \in V_2 \setminus V_1 \,] \vee \\
& [\, t_j(e) \in V_1 \setminus V_2 \ \wedge\ s_i(e) \in V_2 \setminus V_1 \,] \vee \\
& [\, s_i(e) \in V_1 \setminus V_2 \ \wedge\ s_j(e) \in V_2 \setminus V_1 \,] \vee \\
& [\, t_i(e) \in V_1 \setminus V_2 \wedge\ t_j(e) \in V_2 \setminus V_1 \,] \,\}^2.
\end{aligned}
$$
□

[1] For any set A, A* denotes the set of finite sequences over A, including the empty sequence λ.

[2] For any sets A, B and any function $f : A \rightarrow B^*$, $f_i : A \rightarrow B$ denotes a function given by $f_i(a) = b_i$ for $i \leq k$ and $f_i(a) = \lambda$ for $i > k$, where $f(a) = b_0 b_1 \ldots b_k$ for all $i \in N$ and $a \in A$.

Definition 4.6 Let $H_1 = (V_1, E_1, s_1, t_1, lab_{V1}, lab_{E1})$, $H_2 = (V_2, E_2, s_2, t_2, lab_{V2}, lab_{E2}) \in H_C$, $H_2 \subseteq H_1$. *Difference H_1 and H_2* is called $H \in H_C$ such that:

$H = (V_1 \setminus V_2, E_1 \setminus (E_2 \cup E(H_1,H_2;H_1)), s_1|_{E1\setminus(E2\cup E(H1,H2;H1))}, t_1|_{E1\setminus(E2\cup E(H1,H2;H1))}, lab_{V1}|_{V1\setminus V2}, lab_{E1}|_{E1\setminus(E2\cup E(H1,H2;H1))})$

what is denoted $H = H_1 \setminus H_2$. □

Definition 4.7 *A directed hypergraph* (over C) is a hypergraph $H = (V, E, s, t, lab_V, lab_E) \in H_C$ such that:

$s : E \to V^+,\ t : E \to V^+, \forall\ e \in E\ \ \forall\ i, j \in N\ :\ (s_i(e) \neq \lambda\ \wedge t_j(e) \neq \lambda\ ) \Rightarrow\ (s_i(e) \neq t_j(e))$. □

Definition 4.8 An acyclic hypergraph $H = (V, E, s, t, lab_V, lab_E) \in H_C$ is called *a hypergraph tree* when $s : E \to V^+,\ t : E \to V$. □

Definition 4.9 A *k length path* from a node *u* to a node *v* in a hypergraph $H = (V, E, s, t, lab_V, lab_E) \in H_C$ is a sequence of nodes $(v_0, ... , v_k)$ such that $u = v_0$, $v = v_k$ and $\forall\ i \in \{0, ..., k - 1\}\ \ \exists\ e_i \in E;\ l, h \in N : s_l(e_i) = v_i \ \wedge\ t_h(e_i) = v_{i+1}$, what is denoted by $(v_0, ... , v_k; H)$. □

Definition 4.10 Let $H = (V, E, s, t, lab_V, lab_V) \in H_C$, $V_1 \subseteq V$, $E_1 \subseteq E$. *The set of nodes belonging to V_1 which have an edge belonging to set E_1* is a set $V_1(E_1) = \{\ v \in V_1 :\ \exists\ e \in E_1, i \in N : s_i(e) = v\ \vee\ t_i(e) = v\ \}$. □

Definition 4.11 Let $H = (V, E, s, t, lab_V, lab_E) \in H_C$, $v \in V$.

A hypergraph $SP(H,v) = (V^{SP}, E^{SP}, s^{SP}, t^{SP}, lab_V^{SP}, lab_E^{SP})$ is called *a subhypergraph with a beginning in a node v* if SP(H,v) is the biggest subhypergraph of H including v such that for each node $u \in V^{SP}$ there is a path from the node u to the node v in the hypergraph SP(H,v). □

5. Scheme of hierarchical hypergraph and hierarchical hypergraph

In this section introduced concepts of a hierarchical hypergraph schema and a hierarchical hypergraph are formally defined. There is also defined a schema-isomorphism of hierarchical hypergraph schemes.

Definition 5.1 *A schema of a hierarchical hypergraph* is a hypergraph $G^{SH}=(V^{SH},E^{SH},s^{SH},t^{SH},lab_V^{SH},lab_E^{SH}) \in H_C$ with distinguished sets of *abstract nodes* V_A, *aggregation hyperedges* E^A and *generalisation hyperedges* E^G such that:

$V_A \subseteq V^{SH}(E^G),\ E^A \subseteq E^{SH},\ E^G \subseteq E^{SH},\ E^A \cap E^G = \emptyset$, lab_V , lab_E are injections,

$G^A(G^{SH}) = (V^{SH}(E^A),\ E^A,\ s^{SH}|_{EA},\ t^{SH}|_{EA},\ lab_V^{SH}|_{VSH(EA)},\ lab_E^{SH}|_{EA})$ is a directed hypergraph tree, called *an aggregation hypergraph* of the schema G^{SH},

$G^G(G^{SH}) = (V^{SH}(E^G), E^G, s^{SH}|_{EG}, t^{SH}|_{EG}, lab_V^{SH}|_{VSH(EG)}, lab_E^{SH}|_{EG})$ is a directed acyclic hypergraph, called *a generalisation hypergraph* of the schema G^{SH},

$\forall v \in V^{SH}(E^G)\backslash V_A \ \exists \ v_A \in V_A, (u_0, \ldots, u_k; G^G(G^{SH})) : u_0 = v \wedge u_k = v_A$. □

Definition 5.2. Let $G^{SH1}=(V^{SH1},E^{SH1},s^{SH1},t^{SH1},lab_V^{SH1},lab_E^{SH1})$ and $G^{SH2}=(V^{SH2},E^{SH2},s^{SH2},t^{SH2},lab_V^{SH2},lab_E^{SH2})$ be the hierarchical hypergraph schemes. Let V_A^{SH1}, V_A^{SH2} be sets of abstract nodes of G^{SH1}, G^{SH2}, respectively. The hierarchical hypergraph schema G^{SH1} is *schema-isomorphic* with the hierarchical hypergraph schema G^{SH2} when the hypergraph G^{SH1} is isomorphic with the hypergraph G^{SH2}, the aggregation hypergraph $G^A(G^{SH1})$ is isomorphic with the aggregation hypergraph $G^A(G^{SH2})$, the generalisation hypergraph $G^G(G^{SH1})$ is isomorphic with the generalisation hypergraph $G^G(G^{SH2})$ and $\exists \ h : V_A^{SH1} \rightarrow V_A^{SH2}$ - bijection $\forall v \in V_A^{SH1} : lab_V^{SH2}(h(v)) = lab_V^{SH1}(v)$. □

Let us notice very important fact for practical applications of a hierarchical hypergraph schema. Checking schema-isomorphism of hierarchical hypergraph schemes can be done in time $O(n^2 + m^2n^2)$, where n is a number of nodes and m is a number of edges. This is because each node and edge has unique label.

Definition 5.3 A schema of a hierarchical hypergraph $G^H = (V^H, E^H, s^H, t^H, lab_V^H, lab_E^H)$ such that the set of abstract nodes $V_A = \varnothing$ and the set of generalisation hyperedges $E^G = \varnothing$ is called *a hierarchical hypergraph.* □

6. Receiving a hierarchical hypergraph from a hierarchical hypergraph schema

A hierarchical hypergraph can be received from a hierarchical hypergraph schema by a sequence of abstract nodes replacing operations. We replace abstract nodes by one of his specialisation's until there are no abstract nodes left (Definition 6.5). This operation inherits of features, in this case edges, from node ancestors in the generalisation hypergraph. For example, in a hypergraph schema in Fig. 4 a node labelled "Ground floor 1" inherits consisting of nodes labelled "Hall", "Bathroom" and "Dinning-living part" from a node labelled "Ground floor {abstract}". In this hypergraph schema we can replace "Ground floor {abstract}" with "Ground floor 1" (Fig. 2) or with "Ground floor 2" (Fig. 3). Similarly, we can replace nodes labelled "Kitchen {abstract}", "Dining room {abstract}" and „Living room {abstract}" with a node labelled "Kitchen, dining room and living room" (Fig. 2) or with nodes labelled "Kitchen and dining room" and „Living room" (Fig. 3). Let us see at the following auxiliary definitions.

Definition 6.1 Let $G^{SH} =(V^{SH}, E^{SH}, s^{SH}, t^{SH}, lab_V^{SH}, lab_E^{SH})$ be a hierarchical hypergraph schema, $v \in V^{SH}$. A set $Spec(v) = \{ u \in V^{SH} : \exists \ (u_0, \ldots, u_k; G^G(G^{SH})) : u_0 = u \ \wedge \ u_k = v \}$ is called *node v specialisation's*. □

Definition 6.2 Let $G^{SH} = (V^{SH}, E^{SH}, s^{SH}, t^{SH}, lab_V{}^{SH}, lab_E{}^{SH})$ be a hierarchical hypergraph schema, $v_A \in V_A{}^{SH}$. A set $Spec_w(v_A) = \{ v \in V^{SH} \setminus V_A{}^{SH} : \exists\ (u_0, \ldots ,u_k; G^G(G^{SH})) : u_0 = v \wedge u_k = v_A \wedge u_1, \ldots , u_{k-1} \not\in V_A{}^{SH} \}$ is called *node v_A proper specialisation's*. □

Definition 6.3 Let $G^{SH} = (V^{SH}, E^{SH}, s^{SH}, t^{SH}, lab_V{}^{SH}, lab_E{}^{SH})$ be a hierarchical hypergraph schema, $v \in V^{SH}$. A set $Gen(v) = \{ u \in V^{SH} : v \in Spec(u) \}$ is called *node v generalisations*. □

Definition 6.4 Let $G^{SH} = (V^{SH}, E^{SH}, s^{SH}, t^{SH}, lab_V{}^{SH}, lab_E{}^{SH})$ be a hierarchical hypergraph schema, $v \in V^{SH}$. A set $Gen_A(v) = \{ v_A \in V_A{}^{SH} : v \in Spec_w(v_A) \}$ is called *node v abstract generalisations*. □

Let us denote schema shown in Fig. 4 by $G^{SH} = (V^{SH}, E^{SH}, s^{SH}, t^{SH}, lab_V{}^{SH}, lab_E{}^{SH})$. In the schema shown in Fig.4 we have: $Spec((lab_V{}^{SH})^{-1}$(“Ground floor {abstract}”)) = $Spec_w((lab_V{}^{SH})^{-1}$(“Ground floor {abstract}”)) = { $(lab_V{}^{SH})^{-1}$(“Ground floor 1”), $(lab_V{}^{SH})^{-1}$(“Ground floor 2”) }, $Gen((lab_V{}^{SH})^{-1}$(“Ground floor 2”)) = { $(lab_V{}^{SH})^{-1}$(“Ground floor 1”), $(lab_V{}^{SH})^{-1}$(“Ground floor {abstract}”) }, $Gen_A((lab_V{}^{SH})^{-1}$(“Ground floor 2”)) = { $(lab_V{}^{SH})^{-1}$(“Ground floor {abstract}”) }.

Definition 6.5 Let $G^{SH} = (V^{SH}, E^{SH}, s^{SH}, t^{SH}, lab_V{}^{SH}, lab_E{}^{SH})$ be a hierarchical hypergraph schema, E^G be a set of generalisation hyperedges of schema G^{SH}, $v \in V^{SH}$, $Gen_A(v) \neq \varnothing$, $Spec(v) \cap V_A{}^{SH} = \varnothing$. A hierarchical hypergraph schema created by *replacing the node v abstract generalisations by the node v* in the schema G^{SH} is a hierarchical hypergraph schema $G^{GEN}(G^{SH}; v) = (V^{GEN} \setminus V^{REM}, E^{GEN} \setminus E^{REM}, s^{GEN}, t^{GEN}, lab_V{}^{GEN}, lab_E{}^{GEN})$ such that:

a) $G = (V^{GEN}, E^{GEN}, s, t, lab_V, lab_E) := G^{SH} \setminus G^R$ where

$$G^R = \bigcup_{v_A \in Gen_A(v)} \ \bigcup_{v_S \in Spec(v_A) \setminus (Gen(v) \cup \{v\})} (SP(G^A(G^{SH}) \cup G^G(G^{SH}), v_S) ,$$

b) $$V^{REM} = \bigcup_{v_A \in Gen_A(v)} \{v_A\} \ \cup \bigcup_{v_A \in Gen_A(v)} \ \bigcup_{v_S \in Spec(v_A) \cap Gen(v)} \{v_S\},$$

$$E^{REM} = \{ e \in E^G : \exists\ i, j \in N : s_i(e) \in V^{REM} \cup \{v\} \ \wedge\ t_j(e) \in V^{REM} \cup \{v\}\},$$

c) $$s^{GEN}{}_i(e) = \begin{cases} s_i(e) & \text{for } s_i(e) \in V^{GEN} \setminus V^{REM} \\ v & \text{for } s_i(e) \in V^{REM} \end{cases} \quad \text{for } \forall\ e \in E^{GEN} \setminus E^{REM}, \forall\ i \in N$$

$$t^{GEN}{}_i(e) = \begin{cases} t_i(e) & \text{for } t_i(e) \in V^{GEN} \setminus V^{REM} \\ v & \text{for } t_i(e) \in V^{REM} \end{cases} \quad \text{for } \forall\ e \in E^{GEN} \setminus E^{REM}, \forall\ i \in N,$$

d) $lab_V{}^{GEN} = lab_V|_{V^{GEN} \setminus V^{REM}}$, $lab_E{}^{GEN} = lab_E|_{E^{GEN} \setminus E^{REM}}$ □

Let us consider example of this operation working. We will replace a set $\mathrm{Gen}_A((\mathrm{lab}_V^{SH})^{-1}$(“Ground floor 1”)) = { $(\mathrm{lab}_V^{SH})^{-1}$(“Ground floor {abstract}”) } by a node labelled “Ground floor 1” in a schema in Fig. 4. In Fig. 5 the schema G^R of this operation is marked with red. Nodes from set V^{REM} = {lab_V^{-1}(“Ground floor abstract}”)} and edges from set E^{REM} = {lab_E^{-1}(“GG1”), lab_E^{-1}(“G1G2”)}are greyed.

Fig. 5.

A result of a replacing a set $\mathrm{Gen}_A((\mathrm{lab}_V^{SH-})^1$(“Ground floor 1”)) by a node labelled “Ground floor 1” is shown in Fig. 6. Edges inherited from ancestors of the node labelled “Ground floor 1” are greyed.

A hierarchical hypergraph received from a hierarchical hypergraph schema is called a compressed schema of the hierarchical hypergraph.

Definition 6.6 Let $G_0^{SH} = (V_0^{SH}, E_0^{SH}, s_0^{SH}, t_0^{SH}, \mathrm{lab}_{V0}^{SH}, \mathrm{lab}_{E0}^{SH})$ be a hierarchical hypergraph schema. *A compressed schema of the hierarchical hypergraph G_0^{SH}* is a hierarchical hypergraph schema $G_k^{SH} = (V_k^{SH}, E_k^{SH}, s_k^{SH}, t_k^{SH}, \mathrm{lab}_{Vk}^{SH}, \mathrm{lab}_{Ek}^{SH})$, such that:

a) $k \in N$,

b) $G_i^{SH} = G^{GEN}(G_{i-1}^{SH}; v_i)$ where $v_i \in V_{i-1}^{SH}(E^G)$, $\mathrm{Gen}_A(v_i) \neq \emptyset$, $\mathrm{Spec}(v_i) \cap V_{iA}^{SH} = \emptyset$ for $i \in \{1, \dots, k\}$,

c) $V_{kA}^{SH} = \emptyset$. □

Fig. 6.

Now it is easy to define when a hierarchical hypergraph is an instance of a hierarchical hypergraph schema.

Definition 6.7 A hierarchical hypergraph G^H *is compatible with a schema of a hierarchical hypergraph G^{SH}* if there is a compressed schema of the hierarchical hypergraph G^{SH} schema-isomorphic with G^H. □

Hierarchical hypergraphs shown in Fig.2 and Fig.3 are compatible with the schema of the hierarchical hypergraph shown in Fig.4.

7. Extending a hierarchical hypergraph schema

The hierarchical hypergraph schema is created by separated adding to it hierarchical hypergraphs. It means, we add only nodes and edges, which are not in the schema yet. New nodes and edges are adequately connected to generalisation hypergraph of the schema by means of *new generalisations function.* Than we verify if the added hierarchical hypergraph is compatible with the modified hierarchical hypergraph schema.

The following example will help to clarify introduced concepts. Let us suppose that a hierarchical hypergraph schema shown in Fig. 7 is an input schema. We will design ground floor houses, which have to consist of a hall, a bathroom and a dinning-living part. The dinning-living part has to consist of rooms functioning as a kitchen, a dining room and a living room.

In Fig. 8 there is the input schema from Fig. 7 after separated adding to it hypergraph schema shown in Fig. 2 by means of a new generalisations function N_1. The function N_1 maps a node labelled “Ground floor 1” in a set consists of node labelled “Ground floor {abstract}”, a node labelled “Hall 1” in a set consists of node labelled “Hall {abstract}”, a node labelled “Kitchen, dining room and living room” in a set consists of nodes labelled “Kitchen {abstract}”, “Dining room {abstract}”, ”Living room {abstract}” and the rest of nodes into ∅. New nodes and edges are greyed in Fig. 8. Let us see that nodes labelled “House”, “Bathroom”, “Dining-living part” and edges labelled “A_{HG}”, “A_{GHBDL}”, “A_{DLKDL}”, “P_{HK}”, “P_{HB}” from hypergraph in Fig. 2 are not added to the schema.

In Fig. 9 there is the schema from Fig. 8 after separated adding to it hypergraph schema shown in Fig. 3 by means of a new generalisations function N_2. The function N_2 maps a node labelled “Ground floor 2” in a set consists of node labelled “Ground floor {abstract}”, a node labelled “Hall 2” in a set consists of node labelled “Hall {abstract}”, a node labelled “Kitchen and dining room” in a set consists of nodes labelled “Kitchen {abstract}”, “Dining room {abstract}”, a node labelled “Living room” in a set consists of node labelled ”Living room {abstract}” and the rest of nodes into ∅. New nodes and edges are greyed in Fig. 9. Let us notice that a node labelled “Ground floor 2” is not directly connected to a node labelled “Ground floor {abstract}” and a node labelled “Hall 2” is not directly connected to a node labelled “Hall {abstract}” too.

Fig. 7.

This is because new nodes are connected to generalisation hypergraph of the schema in such a way that all edges, which are already in the schema, must be connected to ancestors of connected node. Therefore the node labelled “Ground floor 2” is connected to a node labelled “Ground floor 1” and the node labelled “Hall 2” is connected to a node labelled “Hall 1”.

Fig. 8.

The following definitions are needed to formalisation of extending a hierarchical hypergraph schema concept.

Definition 7.1 Let $H^1 = (V^1, E^1, s^1, t^1, lab_V^1, lab_E^1)$, $H^2 = (V^2, E^2, s^2, t^2, lab_V^2, lab_E^2) \in H_C$. Let us denote $V' := \{ v_2 \in V^2 : \exists\, v_1 \in V^1\ lab_V^1(v_1) = lab_V^2(v_2) \}$, $E' := \{ e_2 \in E^2 : \exists\, e_1 \in E^1\ lab_E^1(e_1) = lab_E^2(e_2) \}$.

A separated sum of hypergraphs H^1 and H^2 is called a hypergraph $H = (V, E, s, t, lab_V, lab_E)$ such that:

$V = V^1 \cup (V^2 \setminus V')$,

$E = E^1 \cup (E^2 \setminus E')$,

$\forall$ v $\in$ V, $\forall$ e $\in$ E, $\forall$ i $\in$ N

$$s_i(e) = \begin{cases} s_i^1(e) & \text{for } e \in E^1 \\ s_i^2(e) & \text{for } e \in E^2,\ s_i^2(e) \notin V' \\ v & \text{for } e \in E^2,\ s_i^2(e) \in V' \text{ where } v \in V^1,\ lab_V^1(v) = lab_V^2(s_i^2(e)) \end{cases}$$

$$t_i(e) = \begin{cases} t_i^1(e) & \text{for } e \in E^1 \\ t_i^2(e) & \text{for } e \in E^2,\ t_i^2(e) \notin V' \\ v & \text{for } e \in E^2,\ t_i^2(e) \in V' \text{ where } v \in V^1,\ lab_V^1(v) = lab_V^2(t_i^2(e)), \end{cases}$$

$$lab_V(v) = \begin{cases} lab_V^1(v) & \text{for } v \in V^1 \\ lab_V^2(v) & \text{for } v \in V^2 \setminus V', \end{cases}$$

$$lab_E(e) = \begin{cases} lab_E^1(e) & \text{for } v \in E^1 \\ lab_E^2(e) & \text{for } v \in E^2 \setminus E' \end{cases}$$

what is denoted by $H = H^1 \oplus H^2$. □

Definition 7.2 Let $G^{SH1} = (V^{SH1}, E^{SH1}, s^{SH1}, t^{SH1}, lab_V^{SH1}, lab_E^{SH1})$ and $G^{SH2}=(V^{SH2}, E^{SH2}, s^{SH2}, t^{SH2}, lab_V^{SH2}, lab_E^{SH2})$ be hierarchical hypergraph schemes. A *separated sum of hierarchical hypergraph schemes* G^{SH1} and G^{SH2} we call a hypergraph $G^{SH1\oplus SH2}$ such that:

$$G^{SH1\oplus SH2} = G^{SH1} \oplus G^{SH2},$$

$$G^A(G^{SH1\oplus SH2}) = G^A(G^{SH1}) \oplus G^A(G^{SH2}),$$

$$G^G(G^{SH1\oplus SH2}) = G^G(G^{SH1}) \oplus G^G(G^{SH2}),$$

$$V_A^{SH1\oplus SH2} = V_A^{SH1} \cup (V_A^{SH2} \setminus \{ v_2 \in V_A^{SH2} : \exists\ v_1 \in V_A^{SH1}\ lab_V^{SH1}(v_1) = lab_V^{SH2}(v_2) \}).$$ □

Fig. 9.

Concepts of a new generalisations function and extending a hierarchical hypergraph schema are defined us follows.

Definition 7.3 Let $G^{SH} = (V^{SH}, E^{SH}, s^{SH}, t^{SH}, lab_V^{SH}, lab_E^{SH})$ be a hierarchical hypergraph schema and $G^H = (V^H, E^H, s^H, t^H, lab_V^H, lab_E^H)$ be a hierarchical hypergraph.

A function $N : V^H \rightarrow P(V_A^{SH}(E^G))$ is called *new generalisations function* of schema G^{SH}. □

Definition 7.4 Let $G^{SH} = (V^{SH}, E^{SH}, s^{SH}, t^{SH}, lab_V^{SH}, lab_E^{SH})$ be a hierarchical hypergraph schema, $v_A \in V_A^{SH}$, $G^H = (V^H, E^H, s^H, t^H, lab_V^H, lab_E^H)$ be a hierarchical hypergraph and $v \in V^H$.

A set $Gen_{PROP}(v_A, v) = \{ u \in Spec(v_A) \cup \{v_A\}: \forall\ w \in Spec(u) :$

$$\neg[\,\exists\, e_{SH} \in E^{SH}, e_H \in E^H, i, j \in N : (lab_E^{SH}(e_{SH}) \; = lab_E^H(e_H)) \wedge$$

$$(s_i^{SH}(e_{SH}) = w \;\vee t_i^{SH}(e_{SH}) = w) \wedge$$

$$(s_j^{H}(e_H) = v \;\vee t_j^{H}(e_H) = v)]\}$$

is called *node v proper generalisations relatively to v_A* . □

For example proper generalisations of a node labelled “Ground floor 2” in hierarchical hypergraph shown in Fig. 3 relatively to a node labelled “Ground floor {abstract}” in schema shown in Fig. 8 is a set consists of a node labelled “Ground floor 1”. The node labelled “Ground floor {abstract}” is not a proper generalisation of a node labelled “Ground floor 2”. This is because a node labelled “Ground floor 1” is connected with a node labelled “Room 1” by an edge labelled “A_{G1R1}”.

Definition 7.5 Let $G^{SH} = (V^{SH}, E^{SH}, s^{SH}, t^{SH}, lab_V^{SH}, lab_E^{SH})$ be a hierarchical hypergraph schema, $G^H = (V^H, E^H, s^H, t^H, lab_V^H, lab_E^H)$ be a hierarchical hypergraph, a function $N : V^H \rightarrow P(V_A^{SH}(E^G))$ be a new generalisations function of schema G^{SH}. Let us denote:

$G^{EXP}(G^{SH},G^H,N) := (V, E \cup E', s, t, lab_V, lab_E^{EXP})$ where:

a) $G := (V, E, s, t, lab_V, lab_E) = \; G^{SH\oplus H}$,

b) E’ is a subset of a set of generalisation edges of the schema $G^{EXP}(G^{SH},G^H,N)$,

c) $\forall\; v \in V^H : N(v) \neq \varnothing \;\; \exists!\; e \in E' : \; s(e) = v \wedge \; t(e) = v_1 ... v_n$ where

$$n = \#\,(\bigcup_{v_A \in N(v)} Gen_{PROP}(v_A, v)) \;\text{ and }\; v_i \in \bigcup_{v_A \in N(v)} Gen_{PROP}(v_A, v) \;\text{ and }\; v_i \neq v_j \;\text{ for } i{\neq}j \,,\; i, j \in \{1, ... ,n\},$$

d) $lab_E^{EXP}|_E = lab_E$.

The hierarchical hypergraph G^H is *possible extension* of the hierarchical hypergraph schema G^{SH} when G^H is compatible with the hierarchical hypergraph schema $G^{EXP}(G^{SH},G^H,N)$. □

8. The class model as a part of design solution space

One of design problems is a choice a solution from a design solution space compatible with design requirements [7]. Proper construction of the design solution space and proper connections between elements of the design solution space and elements of the design requirements space let us to solve this problem.

Let the design solution space consists of hierarchical hypergraph schema and designing objects. Let connect hierarchical hypergraph schema nodes and/or edges with parts of designing objects. In our example, let describe floor layouts by means of polygons nets represented as pointers to edges list. In this representation it is better to connect edges of hierarchical hypergraph schema with edges of polygons net. In Fig. 10 this connections are marked with blue.

Let suppose that the design requirements space is a sum of design features sets. The choice of solution is possible with *the choice function* that maps nodes of hierarchical hypergraph schema and parts of solutions into set of pairs: (“design feature”, “number from range [0, 1]”). The second element of that pair tells how much a choice of node/part guarantees fulfilment design requirements given by the first element of the pair. Finding a solution in the design solution space compatible with design requirements depends on replacing each abstract node by the node, which is the most compatible with design requirements (see Definition 6.6). In Fig. 10 the choice function mappings are marked with red.

Fig. 10.

Such structure of solution space gives us a possibility of some additional mechanisms. For example we can create rules [8]. Rules permit modifying found solution, so that it becomes more compatible with design requirements. A rule consists of feature sets, hypergraph schema and solution parts. Example of a rule is given in Fig. 11.

Fig. 11.

9. An emergent solution

A hierarchical hypergraph schema shown in Fig. 4 was constructed out of two hierarchical hypergraphs shown in Fig. 2 and Fig. 3. Let us notice that it can be received new emergent hierarchical hypergraphs from this hierarchical hypergraph schema. It is possible thanks to specificity of class model, which has been used in a definition of replacing node abstract generalisations by this node. One of emergent hierarchical hypergraphs is shown in Fig. 12. It was received from the schema shown in Fig. 4 by replacing nodes labelled “Ground floor {abstract}”, “Hall {abstract}”, “Kitchen {abstract}”, “Dining room {abstract}” and „Living room {abstract}” with a nodes labelled “Ground floor 1”, “Hall 2”,“Kitchen and dining room” and „Living room”, respectively.

Fig. 12.

One of possibly floor layouts of structure from Fig. 12 is shown in Fig. 13.

Fig. 13

10. A hierarchical hypergraph schema and evolutionary algorithms

Formalism presented above can be used in evolutionary algorithms working on hypergraphs [9][10][11]. A hierarchical hypergraph schema can be used to determine correct crossing points. Let hierarchical hypergraphs G_1^H, G_2^H are compatible with one schema G^{SH}. We say that the set of nodes $\{u_0, ... , u_n\}$ belonging to the hierarchical hypergraph G_1^H is the same type as the set of nodes $\{w_0, ... , w_m\}$ belonging to the hierarchical hypergraph G_2^H when each node belonging to G^{SH} which is an a generalisation of one of nodes $\{u_0, ... ,u_n\}$ is also a generalisation of one of nodes $\{w_0, ... , w_m\}$. For example the set of nodes consists of a node labelled “Kitchen, dining room and living room” in Fig. 2 is of the same type as the set of nodes consists of nodes labelled “Kitchen and dining room” and „Living room” in Fig.3. Nodes sets of the same type determine crossing points in evolutionary algorithms.

It can be defined the operation of exchanging nodes sets of the same type such that the hierarchical hypergraph generated as a result is compatible with the same schema as input hierarchical hypergraphs. But this is not necessary. Let us see that the evolutionary algorithm give us the same result as an algorithm based on receiving a hierarchical hypergraph from a hierarchical hypergraph schema on the basis of the choice function. The hierarchical hypergraph shown in Fig. 12 is a result of crossing hierarchical hypergraph shown in Fig.2 and Fig. 3. The first crossing point is given by a node labelled “Hall 1” in Fig. 2 and a node labelled “Hall 2” in Fig. 3. The second crossing point is given by a node labelled “Kitchen, dining room and living room” in Fig. 2 and nodes labelled “Kitchen and dining room” and “Living room” in Fig. 3. In other point of view this hypergraph is the result of a receiving a structure of floor layout consists of one room, a living room, a dining room and kitchen in one room from a hierarchical hypergraph schema shown in Fig. 4.

10. Conclusions

The approach proposed in this paper makes possible smooth transition from object-oriented design to object-oriented programming. The class model of designed object can be dynamically changed by a system. Changes of the class model keep design criteria. Thanks to the specificity of the class model we receive new emergent solutions and a warranty of space sparing. The advantage of the proposed approach is a well-grounded theoretical base and a possibility of practical applications thanks to polynomial time of operations.

It seems to be valuable to make use of another ideas of object-oriented design. For example multiplicity of edges let us spare much more space.

REFERENCES

[1] Coad P., and Yourdoun E.: Object-Oriented Design. Prentice Hall, Inc., A Simon & Schuster Company, 1991.

[2] Eriksson H.E, and Penker M.: UML Toolkit. John Wiley & Sons, Inc. 1998.

[3] Bailey S.F. and Smith I.: Structural and architectural integration in building design, Proc. EG-SEA-AI Workshop on AI in CE, Lausanne, 1994, p. 327-341.

[4] Grabska E. And Borkowski A.: Generating floor layouts by means of composite representation, ECCE

Symposium 1997, RIL, Finland, p. 154-158.

[5] Habel A.: Hyperedge Replacement: Grammars and Languages. Lecture Notes in Computer Science, Springer Verlag 643, 1992.

[6] Habel A.: Hyperedge Grammars: Transformational and Algorithmic Aspects. J. Inform. Process. Cybernet. EIK 28 (1992) 5, 241-277.

[7] Aamodt A. and Plaza E.: Case-Based Reasoning: Foundational Issues, Methodological Variations, and System Approaches, AI Communications. IOS Press, Vol. 7: 1, pp 39-59.

[8] Ron Sun: Robust Reasoning: Integrating Rule-Based and Similarity-Based Reasoning. Artificial Intelligence, Volume 75, Number 2, June 1995, 241-295.

[9] Mitchel, M.: An Introduction to Genetic Algorithms, MIT Press, 1996.

[10] Hliniak G., and Strug B.: Graph grammars and evolutionary methods in graphic design. Machine GRAPHICS & VISION, Vol.9 No 1/2,2000, 5-13.

[11] Gero, J. and Kozakov, V.: An Exploration-based Evolutionary Model of Generative Design Process. Microcomputers in Civil Engineering No. 10, 1996, 209-216.

Captions to illustrations

Fig.1. Examples of floor layouts.

Fig. 2. The structure of floor layout from Fig.1a.

Fig. 3. The structure of floor layout from Fig.1b.

Fig. 4. Structures of floor layouts from Fig.1 described by means of hypergraph schema.

Fig. 5. The schema G^R (marked with red), nodes from set V^{REM} (greyed) and edges from set E^{REM} (greyed) of replacing the set $Gen_A((lab_V^{SH-})^1($ “Ground floor 1”$))$ = { lab_V^{-1}(“Ground floor {abstract}”) } by a node labelled “Ground floor 1”.

Fig. 6. The schema shown in Fig. 4 after replacing the set $Gen_A((lab_V^{SH-})^1($ “Ground floor 1”$))$ = { lab_V^1(“Ground floor {abstract}”) } by a node labelled “Ground floor 1”. Edges inherited from ancestors of the node labelled “Ground floor 1” are greyed.

Fig. 7. An input hierarchical hypergraph schema.

Fig. 8. An input schema from Fig. 7 after separated adding to it hypergraph schema shown in Fig. 2 by means of a new generalisations function N_1. The function N_1 maps a node labelled “Ground floor 1” in a set consists of node labelled “Ground floor {abstract}”, a node labelled “Hall 1” in a set consists of node labelled “Hall {abstract}”, a node labelled “Kitchen, dining room and living room” in a set consists of nodes labelled “Kitchen {abstract}”, “Dining room {abstract}”, ”Living room {abstract}” and the rest of nodes into ∅. New nodes and edges are greyed.

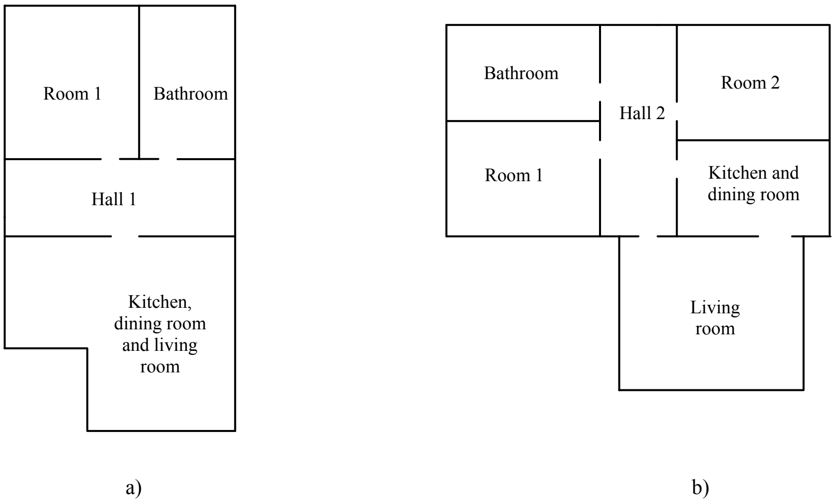
Fig. 9. A schema from Fig. 8 after separated adding to it hypergraph schema shown in Fig. 3 by means of a new generalisations function N_2. The function N_2 maps a node labelled “Ground floor 2” in a set consists of node labelled “Ground floor {abstract}”, a node labelled “Hall 2” in a set consists of node labelled “Hall {abstract}”, a node labelled “Kitchen and dining room” in a set consists of nodes labelled “Kitchen {abstract}”, “Dining room {abstract}”, a node labelled “Living room” in a set consists of node labelled ”Living room {abstract}” and the rest of nodes into ∅. New nodes and edges are greyed.

Fig. 10. The design requirements space and the design solution space consists of hierarchical hypergraph schema and designing objects.

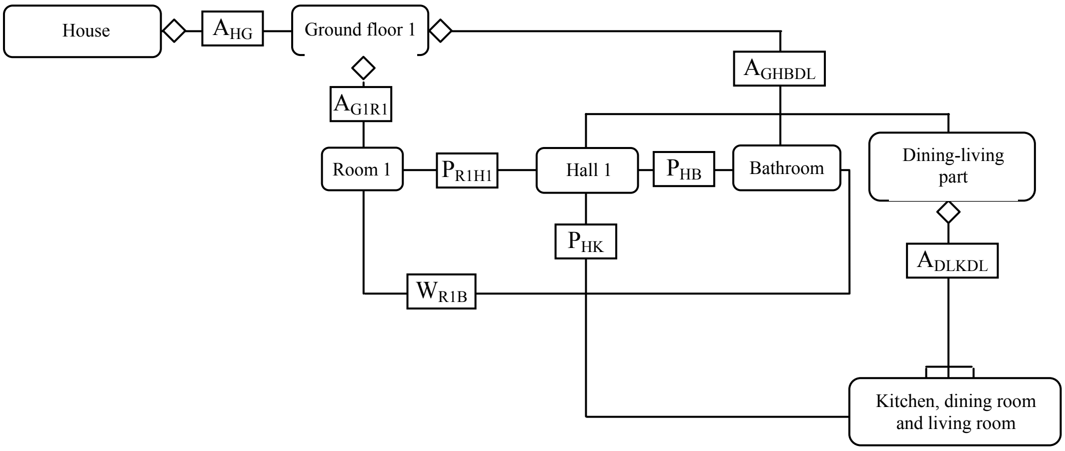
Fig. 11. Example of a rule.

Fig. 12. One of emergent hierarchical hypergraphs that can be received from the schema show in Fig. 4.

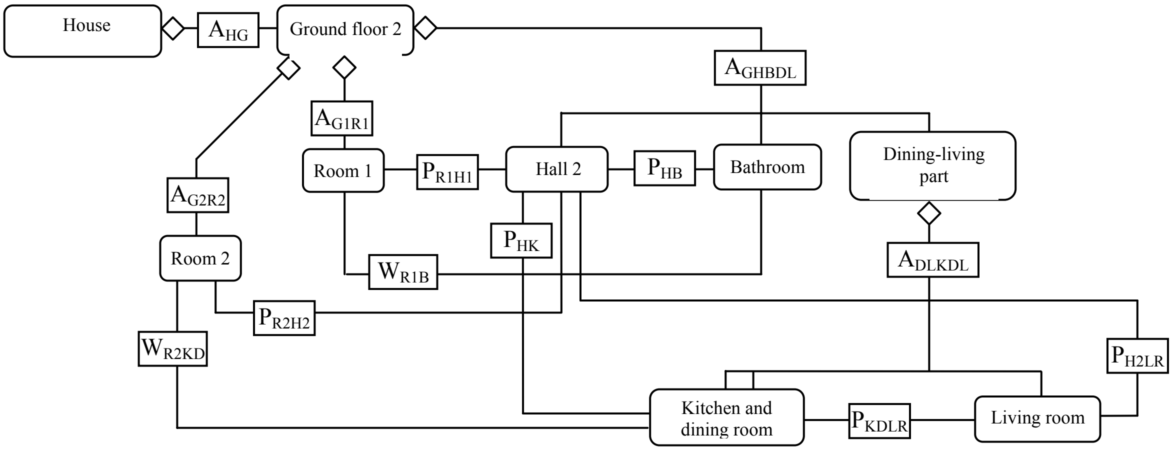
Fig. 13. One of possibly floor layouts of structure shown in Fig. 12.



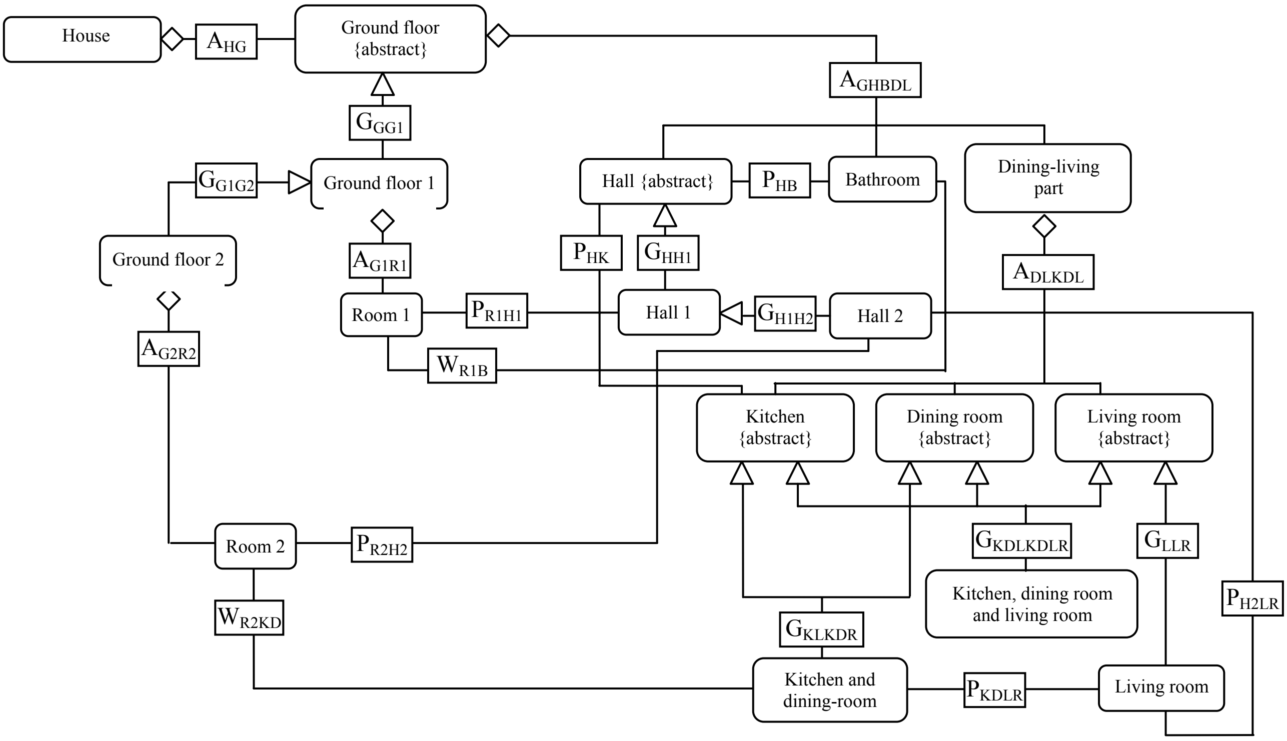
Dynamic class model in object-oriented design, Marcin Skowron, Fig.1.



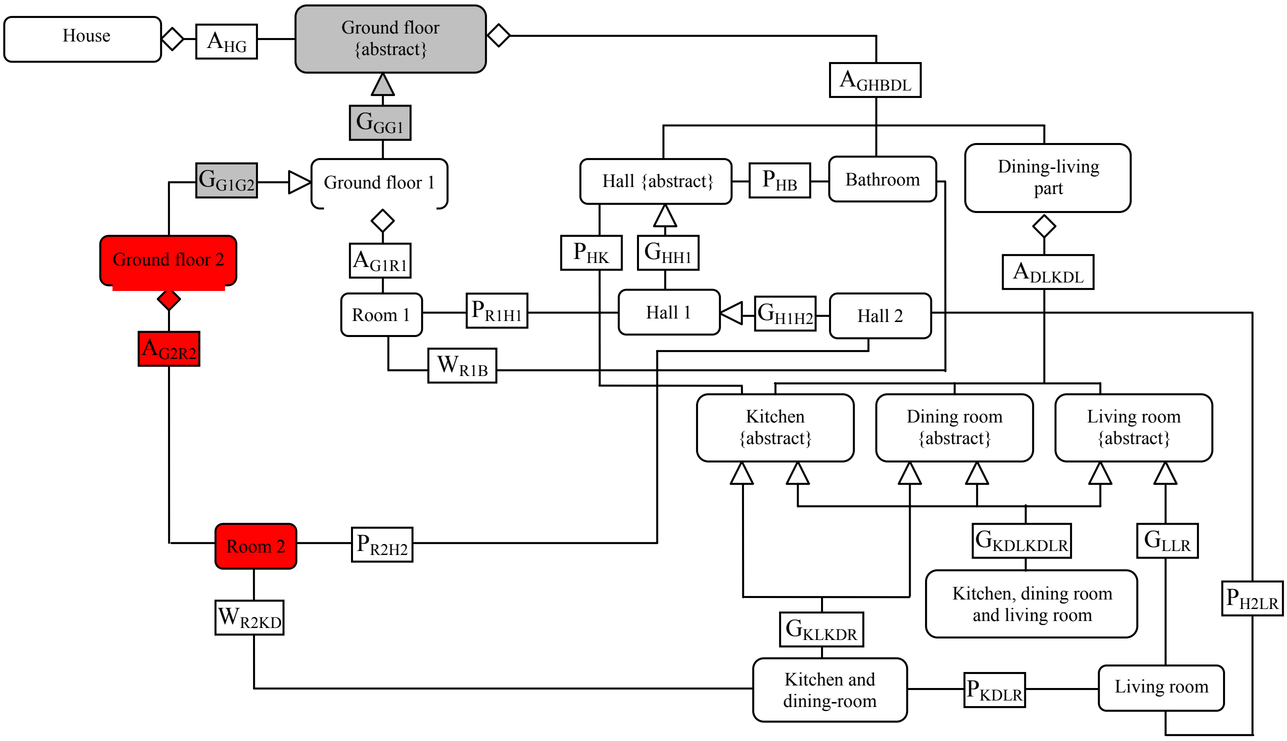
Dynamic class model in object-oriented design, Marcin Skowron, Fig. 2.



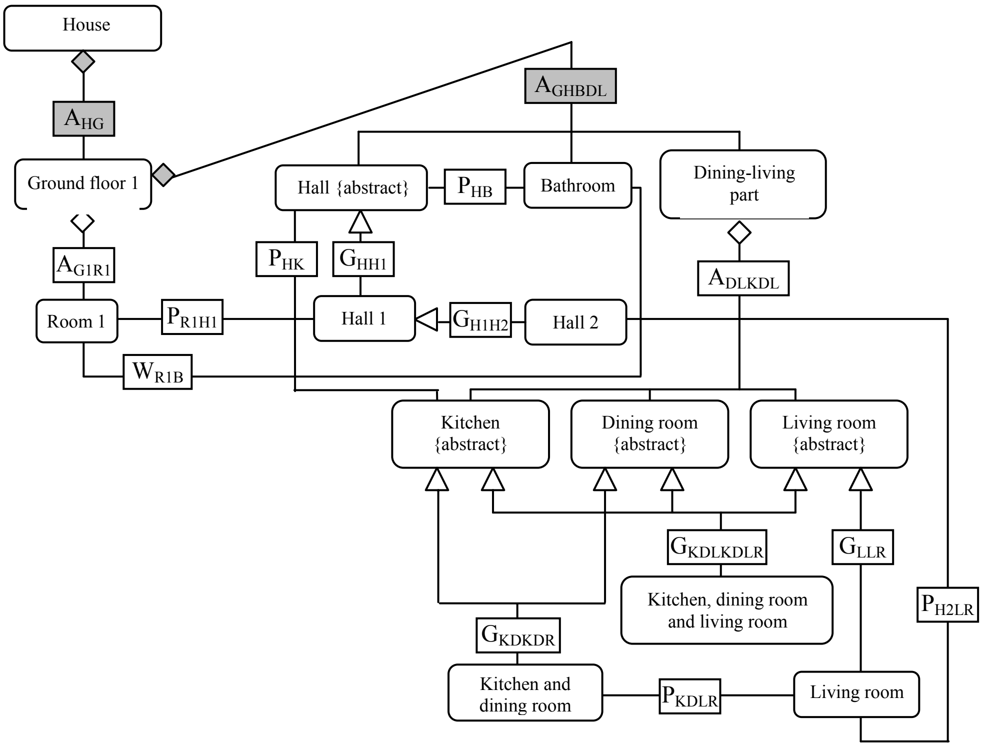
Dynamic class model in object-oriented design, Marcin Skowron, Fig. 3.



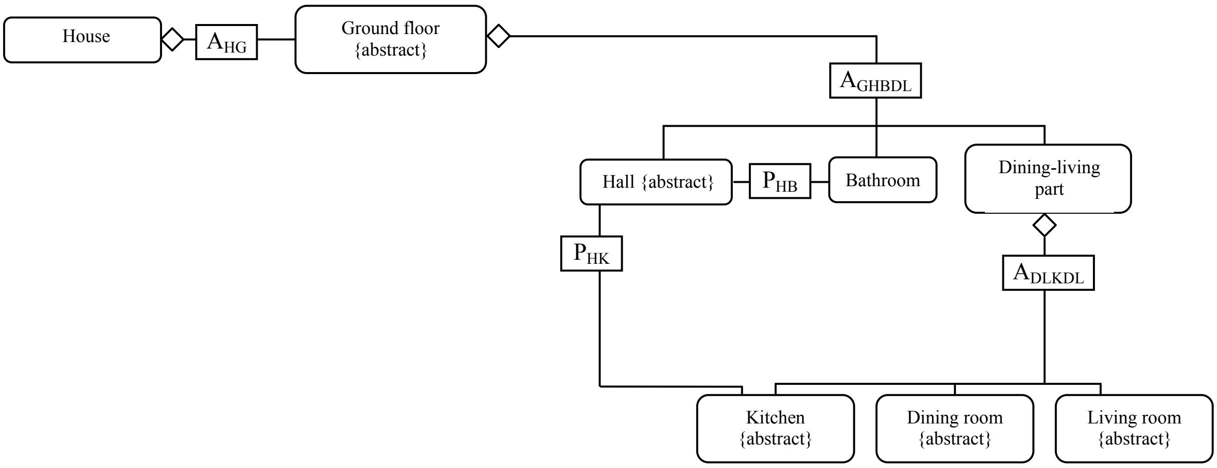
Dynamic class model in object-oriented design, Marcin Skowron, Fig. 4.



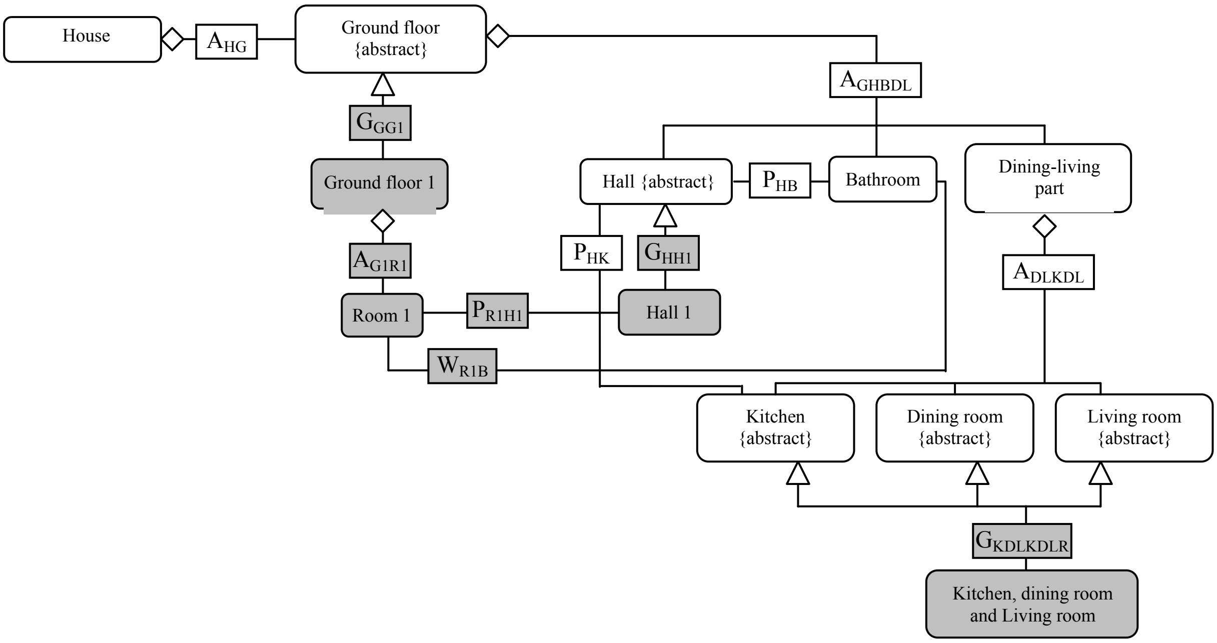
Dynamic class model in object-oriented design, Marcin Skowron, Fig. 5.



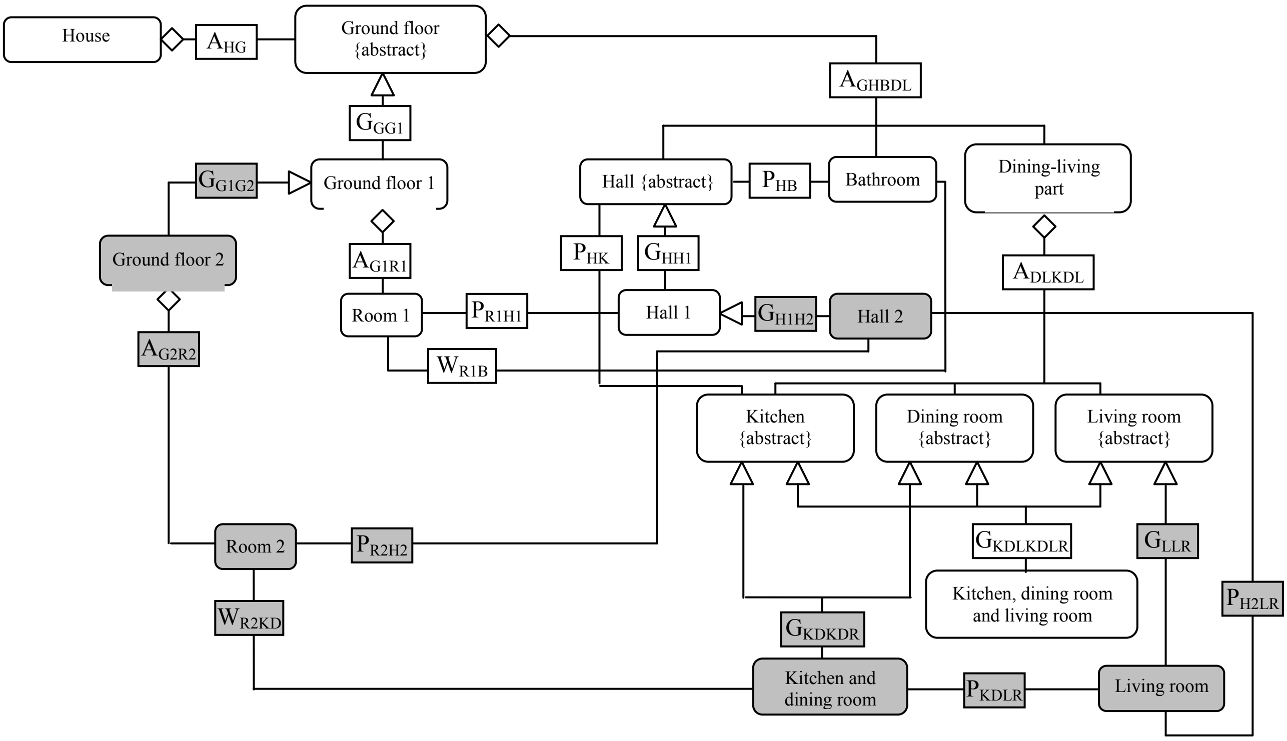
Dynamic class model in object-oriented design, Marcin Skowron, Fig. 6.



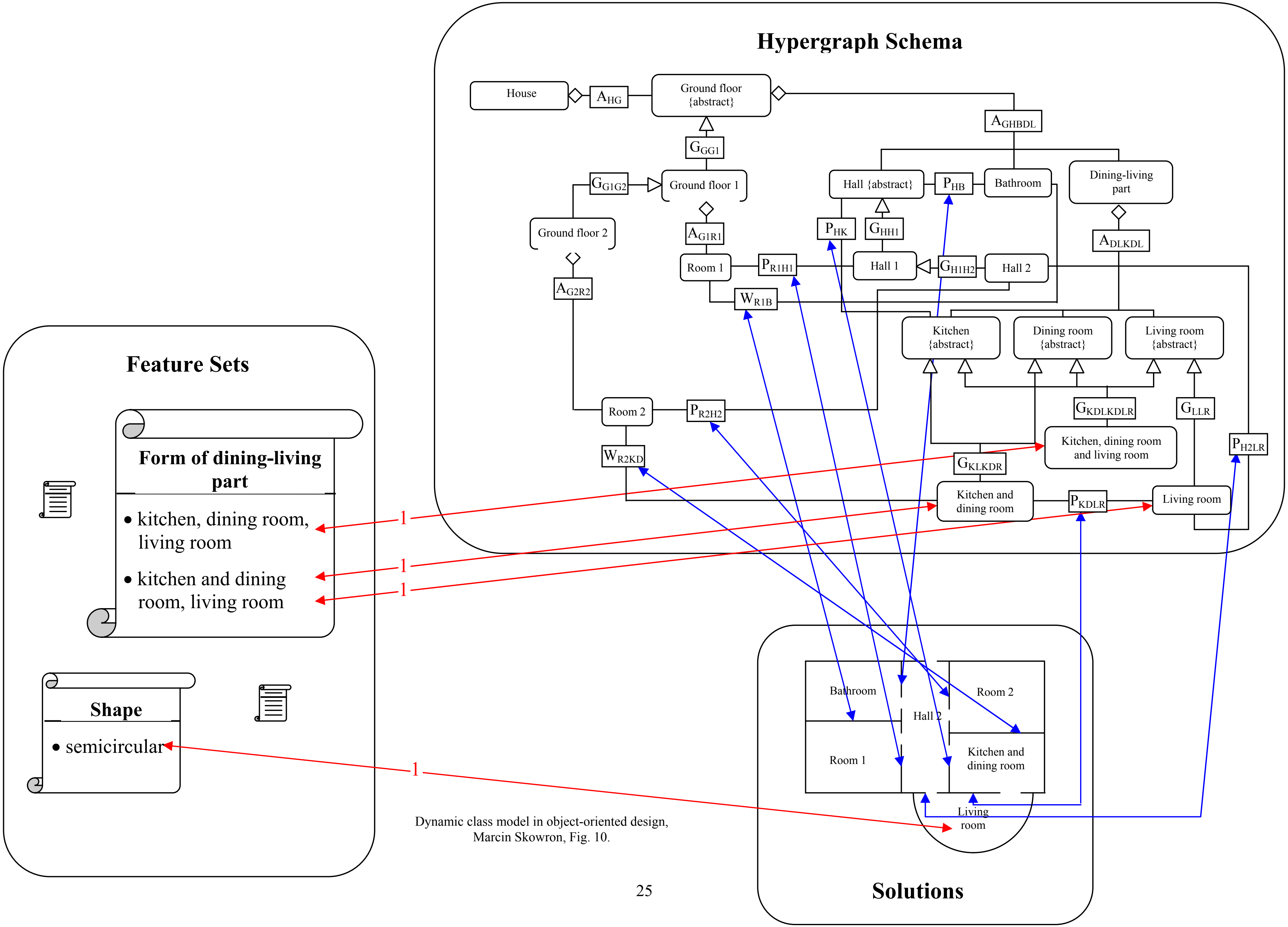
Dynamic class model in object-oriented design, Marcin Skowron, Fig. 7.



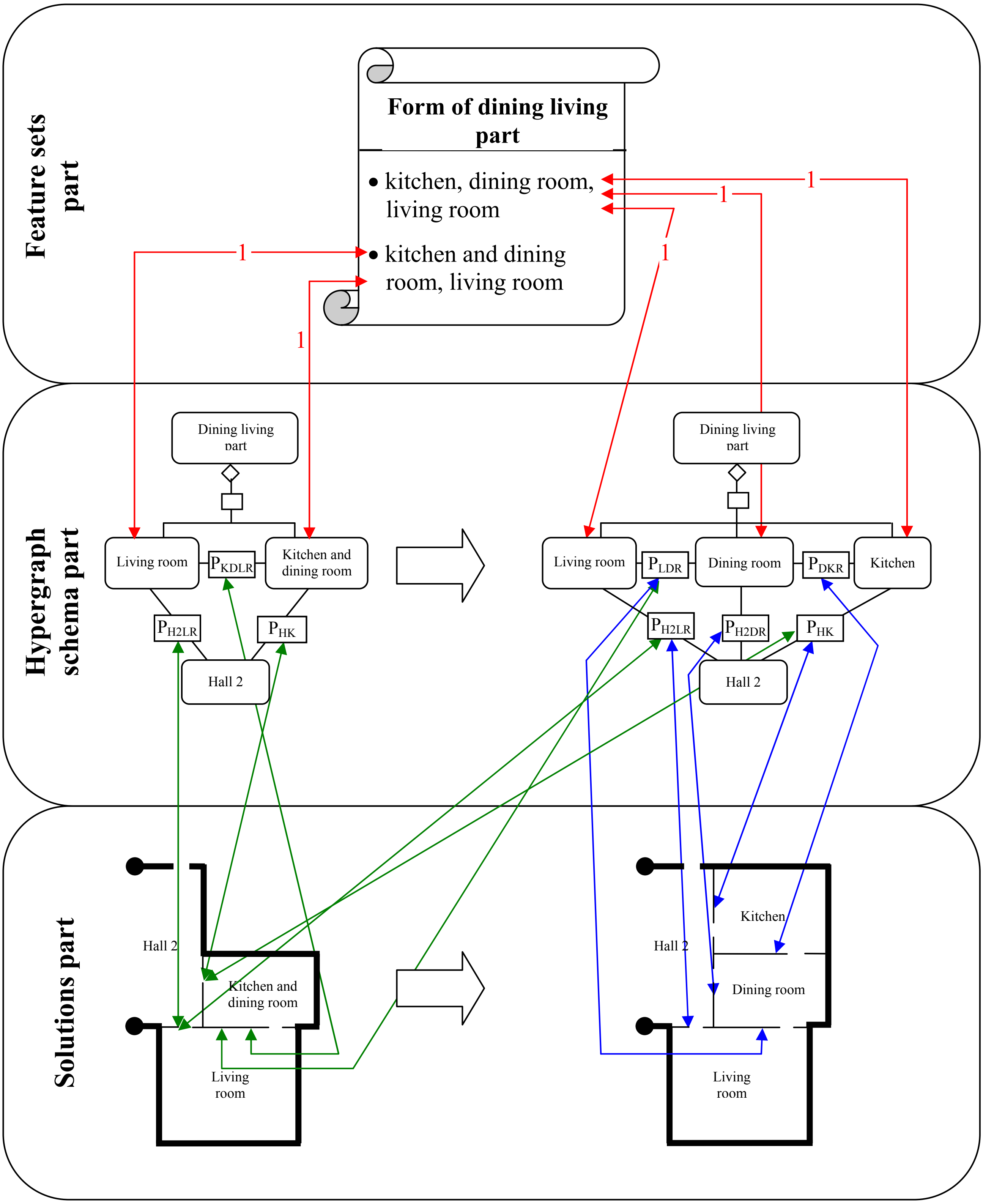
Dynamic class model in object-oriented design, Marcin Skowron, Fig. 8.



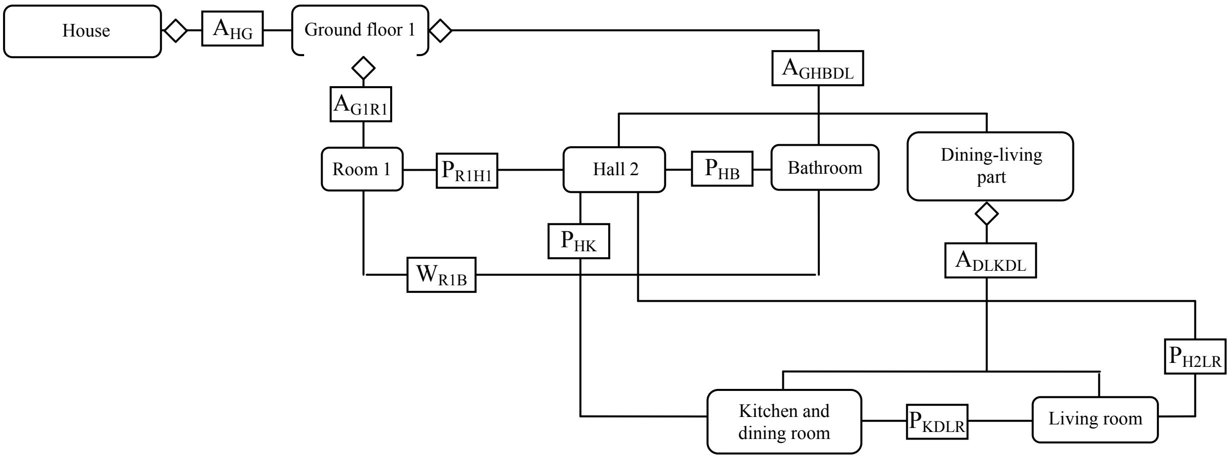
Dynamic class model in object-oriented design, Marcin Skowron, Fig. 9.



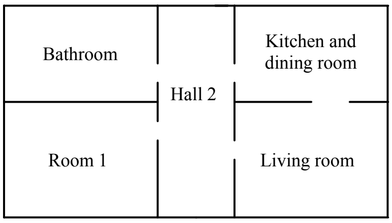
Dynamic class model in object-oriented design,
Marcin Skowron, Fig. 10.



Dynamic class model in object-oriented design, Marcin Skowron, Fig. 11.



Dynamic class model in object-oriented design, Marcin Skowron, Fig. 12.



Dynamic class model in object-oriented design, Marcin Skowron, Fig. 13.

Biographical note

I graduated Computer Science from Jagiellonian University, Kraków, Poland. Now I am PhD student in Jagiellonian University, in Institute of Mathematics.
I am interested in artificial intelligence and computer graphics.