*Artificial Intelligence for Engineering Design, Analysis and Manufacturing* (2008), *22*, 129–145. Printed in the USA.
Copyright © 2008 Cambridge University Press 0890-0604/08 $25.00
DOI: 10.1017/S0890060408000097

# A grammar-based multiagent system in dynamic design

GRAŻYNA ŚLUSARCZYK
Faculty of Physics, Astronomy, and Applied Computer Science, Jagiellonian University, Kraków, Poland
(Received June 21, 2007; Accepted November 30, 2007)

**Abstract**

This paper deals with the system of agents treated as a concurrent modular system, which is able to support the designer in solving complex design tasks. The behavior of design agents is modeled by sets of grammar rules. Each agent manages a graph grammar and a database of facts concerning the subtask for which it is responsible. The course of designing is determined by the interaction between cooperating specialized agents. The design context is expressed by the environment in which agents act and predicates describing design criteria. The organization, design methodology, and a semantic model of a grammar-based multiagent design system are presented. The notions of a valid design solution and a design solution consistent with the design criteria are also introduced. The proposed approach is illustrated by the example of designing a house estate.

**Keywords:** Computer-Aided Design; Graph Grammar; Multiagent System

## 1. INTRODUCTION

As design takes place in an ever-changing environment, recent frameworks for design focus on dynamic character of its context (Campbell et al., 1999; Gero & Kannengiesser, 2003; Sosa & Gero, 2004). This paper proposes a novel computational framework for computer-aided adaptive design. It uses the concept of grammar-based intelligent agents, which control the dynamic multifunctional design process and support the designer in a conceptual phase of this process. The organization, design methodology, and a semantic model of a grammar-based multiagent design system are presented.

Our system of agents is treated as a concurrent modular self-controlling open system that is able to solve complex design tasks. The architecture of the proposed design system is based on a collection of intelligent grammar-based agents that communicate with each other and autonomously control the development of subtask solutions. The design context is expressed by the environment in which agents act and predicates describing design criteria.

The behavior of our design agent is modeled by a set of grammar rules. Each agent has a hypergraph grammar and a database of facts concerning the subtask for which it is responsible. The novelty of our approach lies in the fact that the agents create their own design representations in the form of hypergraphs using rules of hypergraph grammars. Because of the representation that was used, the agents are capable of adapting to changes in the design problem specification and realizing adaptive search for solutions. The design knowledge encoded in hypergraphs enables agents to behave dynamically, as they are able to evaluate not only complete solutions but also partial solutions representing current states of objects being designed, and on this basis make subsequent design decisions. Agents plan their future behavior by choosing grammar productions that are to be applied.

This paper presents our attempt to develop a grammar-based multiagent system that supports the designer in the earliest stages of architectural design. Section 2 describes related work. In Section 3 the architecture of a proposed design system and the information flow in it are presented. The way in which grammar-based agents work is also explained. In the next section formal notions of an agent, a multiagent system, and agent knowledge are invoked. Section 4 describes internal representations of designs in the form of hierarchical hypergraphs and hypergraph grammars used by design agents to derive these hypergraphs. Elements of the language generated by cooperating agents are then interpreted as design solutions. Section 5 presents a semantic model of a grammar-based multiagent design system. The notions of valid designs and consistent design solutions are also discussed. In Section 6, the proposed approach is illustrated by the example of designing a house estate with the assistance of several cooperating grammar-based agents. The manager

Reprint requests to: Grażyna Ślusarczyk, Jagiellonian University, Institute of Physics, Reymonta 4, 30-059 Kraków, Poland. E-mail: gslusarc@uj.edu.pl



https://doi.org/10.1017/S0890060408000097 Published online by Cambridge University Press

agent generates the arrangement of the whole estate compatible with the given specification and invokes agents designing the garden and the house interior. The agent that is responsible for designing furnished floor layouts has agent assistants generating furniture arrangements for different types of spaces in the house. The implementation section gives an outline of our prototype design system. Finally, some conclusions are drawn.

## 2. RELATED WORK

Recently, a great deal of design research has been focussed on developing agent-based design systems (Myers & Pohl, 1994; Lander, 1997). The computational models and implementations of agents and collections of agents based on psychological models of social cognitive abilities are described (Velasquez & Maes, 1997; Kokoszka et al., 2001). Models of motivated, emotional agents (Canamero, 1997) and curious agents (Saunders, 2001; Grabska et al., 2005) are also presented. Sapient agents, which learn to refine the decision-making capability and are able to make long-term strategic decisions, are described in Dzeroski (2002). Adaptive, agent-based methods of conceptual design are presented in Campbell et al. (1998), whereas design systems with learning agents that form some expectations are described in Grecu and Brown (2000). Many of these approaches establish an efficient communication between agents to handle design subtask, but they are not flexible enough to deal with the dynamic nature of design that is particularly important in conceptual stages of design.

In contrast, shape, structural, and graph grammars are widely used in architectural and engineering design (Stiny & Mitchel, 1980; Longenecker & Fitzhorn, 1991; Carlson et al., 1993; Borkowski & Grabska, 1995; Cagdas, 1996; Cagan, 2001; Soman & Campbell, 2002). Most of these approaches use other tools to deal with semantics of design. For example, rules of a shape grammar generating coffeemakers are linked with expressions determining the manufacturing costs of designs in Agarwal et al. (1999).

Lately, different types of grammars are combined with agent-based systems. Some grammatical models of multiagent systems modeling eco systems can be found in Paun and Salomaa (1999). In McCormack and Cagan (2002), each shape grammar rule used in hood panel design is associated with one instantiation agent that controls the application of this rule. However, these approaches do not provide internal representations of designs that would uniformly encode syntactic and semantic design knowledge, and would be appropriate for automatic processing. Therefore, we propose a computational framework, where grammar-based agents operate on hierarchical design representations ground in the hypergraph theory (Habel & Kreowski, 1987; Drewes et al., 2000). The evaluation method based on hierarchical hypergraphs corresponding to partial solutions is more meaningful than the one presented in our previous work (Grabska et al., 2006*b*), where agents evaluate designs using a neural network. Moreover, it allows agents to manipulate design representations in a dynamical way and adapt to changes in design. This approach is modeled in a similar way as a situated function–behavior–structure design framework described in Gero and Kannengiesser (2002).

## 3. A SYSTEM OF INTERFACE DESIGN AGENTS

The proposed multiagent design system consists of the context, which contains the environment and the specified design criteria, the message buffer, and a few kinds of agents acting simultaneously and performing different tasks. In the presented approach interface agents being designer's assistants are used.

The application of intelligent agents in the process of design is illustrated by the example of designing a house estate. The main goal of this task is to provide a well-organized, furnished living space inside a house and a suitably arranged garden.

The distinguished manager agent is responsible for designing the arrangement of the whole estate. The manager agent delegates tasks to the garden designing agent and the house interior designing agent. The latter one is responsible for designing the floor layout. It has agent assistants generating furniture arrangements for different types of spaces in the house. Eventually, the manager agent combines designs generated by itself, the "garden agent" and the "house interior" agent, and sends them to the environment. The environment, which is also treated as an agent, evaluates the obtained solutions and stores them in the global database. It keeps the best designs and a few poorer ones that can lead to long-term benefits. The design solutions put into the database by the environment can be visualized as the graphical models using a given interpretation.

The information flow in the system is shown in Figure 1. A dashed line, which crosses arrows linking agents, denotes the fact that agents and the environment communicate with each other through the message buffer.

The proposed system is suitable for such a design task, as each agent can independently realize adaptive search for solutions of a subtask for which it is responsible. The agents are equipped with programmed hierarchical hypergraph grammars generating attributed hypergraph representations of design solutions and task-oriented databases that together represent their local knowledge. Grammar rules encode the structural aspects of possible solutions, whereas values of hypergraph attributes correspond to design constraints. The flow of information related to the internal processes of a single design agent that communicates with other agents through the message buffer is presented in Figure 2. The names in brackets denote agent's processes that are specified in Definition 4.1.

First, the agent senses the message buffer and reads the obtained message. Then the type of the message, which is in the form of an attributed hypergraph, where values of attributes carry over the design requirements, is recognized in the perception process. When a hypergraph corresponding to a subtask solution generated by the other agent is received in the message, the adequate modification of the generated hypergraph is made.

https://doi.org/10.1017/S0890060408000097 Published online by Cambridge University Press

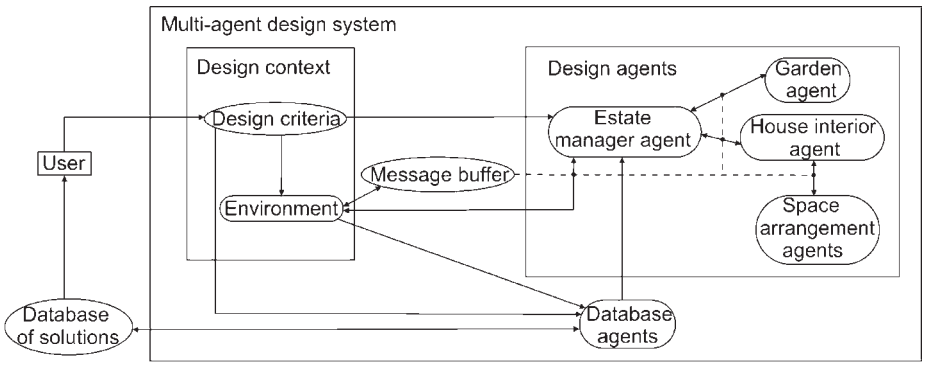


**Fig. 1.** Information flow in the proposed system.

When the message requires solving a subtask, the attributes of a hypergraph representing an internal state of the agent are updated according to the ones included in the obtained message. Then the derivation of a new hypergraph design representation is started. It should be noted that the agent can be invoked by other agents each time with different values of attributes. In this way the design requirements encoded in hypergraph attributes enable agents to adapt to changing design problem specifications.

After each modification of the generated hypergraph the evaluation of the current solution takes place and a decision about the next action is made. Either a message (a generated solution, a request of cooperation or notification of an error) is sent to the buffer or an internal action is taken. As a result of an internal action a successive rewriting step on the derived hypergraph is performed using a selected grammar rule, or a complete solution is stored and the generation of a new one is started. Thus, because of the evaluation of partial solutions representing current states of design, agents make decisions concerning grammar productions that should be applied, and in this way select the direction of search for valid solutions.

## 4. KNOWLEDGE REPRESENTATION IN A MULTIAGENT DESIGN SYSTEM

A design agent is a computational entity situated in some environment and capable of acting autonomously in this environment to satisfy requirements of the assigned design task. Its behavior is determined by processes of sensing,

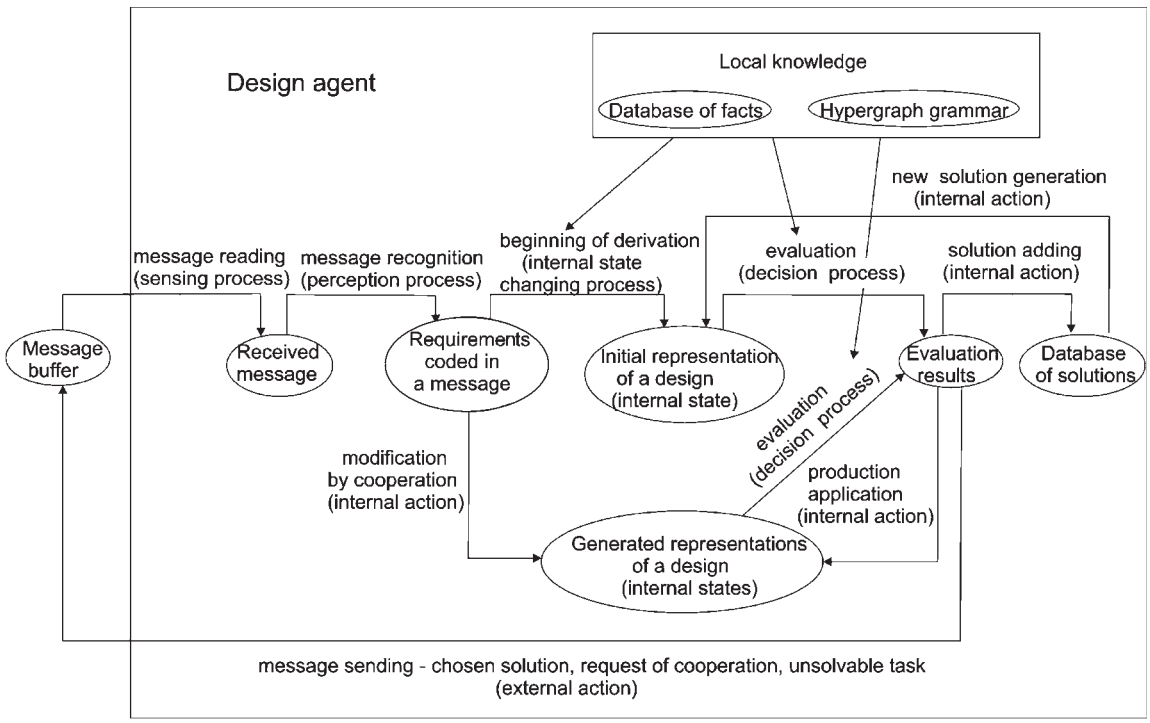


**Fig. 2.** The way in which a grammar-based design agent works.

https://doi.org/10.1017/S0890060408000097 Published online by Cambridge University Press

perception, evaluation of solutions, decision making, performing internal actions, and affecting the environment by external actions. The effectoric capability of an agent is assumed to be represented by the set of its actions (Wooldridge, 1999). Each intelligent agent should possess two types of memory: a short-term memory and a long-term one. The short-term memory allows the agent to remember a few recent perceptual and conceptual states. The long-term memory stores agent's generalized experiences from the past.

Let us define an intelligent agent in a formal way. Let $W$ denote a set of possible states of the world. Let $S = \{s^0, s^1, \ldots\}$, $P = \{p^0, p^1, \ldots\}$ and $C = \{c^0, c^1 \ldots\}$ denote sets of sensor, perceptual, and conceptual states, respectively.

DEFINITION 4.1. An intelligent agent with a short-term ($M_S$) and long-term memory ($M_L$) is defined as a tuple $\mathrm{Ag} = (\mathrm{Int}, A, \sigma, \rho, \eta, \chi, \alpha)$, where $\mathrm{Int} = \{\mathrm{int}^0, \mathrm{int}^1, \ldots\}$ is a set of agent's internal states, $A = \{a^0, a^1, \ldots\}$ is a set of its actions, $\sigma$: $W \rightarrow S$ is the world sensing process, $\rho$: $S \rightarrow P$ is the perception process being the interpretation of data obtained from sensors, $\eta$: $\mathrm{Int} \times P \rightarrow \mathrm{Int}$ is the internal state changing process, $\chi$: $\mathrm{Int} \rightarrow C$ is the evaluation and decision-making process during which an internal state is evaluated and new goals are specified, and $\alpha$: $C \rightarrow A$ is the action process that translates goals specified by a conceptual state to an action that should be taken. ■

In our approach, the world $W = \{w_0, w_1, \ldots\}$ of design agents is a set of hypergraphs representing solutions and generated by all agents in a parallel way. A set of agent's internal states *Int* includes hypergraphs being derived using productions of its grammar. Perceptual states correspond to requirements obtained in messages, whereas conceptual states represent hypergraph evaluation results, which are then stored in variables of the short-term memory. The long-term memory is in the form of a database of the created admissible solutions. The agents' actions are depicted in Figure 2.

A multiagent design system is composed of a collection of agents $\mathrm{Ag}_1, \ldots, \mathrm{Ag}_n$ and the environment that they occupy. The definition of a multiagent system is preceded by the one of an environment that follows (Fagin et al., 1995; Wooldridge & Lomuscio, 2000).

DEFINITION 4.2. An environment is a tuple: $\mathrm{Env} = (W, A_E, K_1, \ldots, K_n, \tau)$, where $W = \{w_0, w_1, \ldots\}$ is a set of possible states of the world; $A_E = \{a_E^0, a_E^1, \ldots\}$ is a set of environment actions; $K_i$: $W \rightarrow W_{\mathrm{Ag}_i} \in 2^W$ is a partition of $W$ for every agent $\mathrm{Ag}_i$, which characterizes information available to an agent in every environment state; and $\tau$: $A_E \times A_1 \times \cdots \times A_n \rightarrow 2^W$ is a state transformer function, which maps an action of the environment being in a given state and one action of each agent to the set of environment states that can result from the performance of these actions in this state. ■

DEFINITION 4.3. A multiagent system (Wooldridge & Lomuscio, 2000) is a structure $\mathrm{AS} = (\mathrm{Env}, \mathrm{Ag}_1, \ldots, \mathrm{Ag}_n)$, where Env is an environment specified as in Definition 4.2 and $\mathrm{Ag}_1, \ldots, \mathrm{Ag}_n$ are intelligent agents determined as in Definition 4.1. ■

At any moment of time the system is in a global state. The set of global states *GS* of the system *AS* contains subsets of $W \times \mathrm{Int}_1 \times \cdots \times \mathrm{Int}_n$. The functioning of the system is realized by changes of the global states described by the environment state transformer function $\tau$ and the agents' internal state changing functions $\eta_i$.

The initial global state is of the form $g_0 = (w_0, \mathrm{int}_1^0, \ldots, \mathrm{int}_n^0)$. At this point every agent $\mathrm{Ag}_i$ senses the environment computing $s_i^0 = \sigma_i(K_i(w_0))$, on the basis of $s_i^0$ establishes its initial perception state $p_i^0$, updates its internal state, which becomes $\mathrm{int}_i^1 = \eta_i(\mathrm{int}_i^0, p_i^0)$ and selects the first action, which should be performed as $a_i^0 = \alpha_i(\chi_i(\mathrm{int}_i^1))$. The state of the environment is updated as $w_1 = \tau(a_E^0, a_1^0, \ldots, a_n^0)$, and the system reaches the next global state $g_1 = (w_1, \mathrm{int}_1^1, \ldots, \mathrm{int}_n^1)$.

In our system the initial state $w_0$ has a form of one hyperedge from which the derivation starts. Its attributes specify the design requirements like the number of rooms in a house. Initial internal states of the agents $\mathrm{int}_1^0, \ldots, \mathrm{int}_n^0$ are empty hypergraphs. When an agent $\mathrm{Ag}_i$ senses that it is invoked by the environment or the other agent, it identifies the obtained message with the initial hyperedge of its grammar, which becomes its next internal state $\mathrm{int}_i^1$.

A prepositional multimodal logic is often used to represent and reason about various aspects of multiagent systems (Gaborit et al., 1990; Traverso & Spalazzi, 1995; Wooldridge, 1995). To represent knowledge possessed by design agents we define an interpreted system $I_{\mathrm{AS}}$ and then associate with it a Kripke structure (Fagin et al., 1995).

We assume that we have a set $\Phi$ of primitive propositions that describe basic facts about the system. A run of AS over GS is a function $r$: $N \rightarrow$ GS so it can be identified with any sequence of global states over GS. Let $R$ denote a set of runs over GS.

DEFINITION 4.4. An interpreted multiagent system $I_{\mathrm{AS}}$ generated by the structure AS is a pair $(\mathrm{GS}_{\mathrm{AS}}, \pi)$, where $\mathrm{GS}_{\mathrm{AS}} = \cup_{r \in R} r(N)$ is a set of all global states reachable by AS and $\pi$ is an interpretation for the propositions in $\Phi$ over GS, which assigns truth values to the primitive propositions at the global states. ■

There is a one-to-one correspondence between each partition on $W$, which defines the possible worlds indistinguishable to the agent $\mathrm{Ag}_i$ in the given state of $W$ (see Definition 4.2), and an equivalence relation on $W$. Given a partition $K_i$ on $W$ the corresponding equivalence relation $\mathbb{K}_i$ is defined by $(w', w'') \in \mathbb{K}_i$ iff $K_i(w') = K_i(w'')$.

A system $I_{\mathrm{AS}}$ corresponds to the Kripke structure $M_{I_{\mathrm{AS}}} = (\mathrm{GS}_{\mathrm{AS}}, \pi, \mathbb{K}_1, \ldots, \mathbb{K}_N)$, where $\mathbb{K}_i \subseteq \mathrm{GS}_{\mathrm{AS}} \times \mathrm{GS}_{\mathrm{AS}}$ is the equivalence relation corresponding to the partition $K_i$ on $W$ and extended on $\mathrm{GS}_{\mathrm{AS}}$ in such a way that for each two global states $g' = (w', \mathrm{int}_1', \ldots, \mathrm{int}_n')$ and $g'' = (w'', \mathrm{int}_1'', \ldots, \mathrm{int}_n'')$, $(g', g'') \in \mathbb{K}_i$ if $(w', w'') \in \mathbb{K}_i$ and $\mathrm{int}_j' = \mathrm{int}_j''$.

The knowledge of a design agent is determined by its database, grammar rules, and generated hypergraphs. Having

https://doi.org/10.1017/S0890060408000097 Published online by Cambridge University Press

a set $\Phi$ of propositions describing design criteria the agent can test the validity of a generated solution by comparing values of hypergraph attributes with a given requirement.

Let $\mathbb{K}_1, \ldots, \mathbb{K}_n$ denote modal operators and let $L$ denote a language obtained by closing off $\Phi$ under negation, conjunction, and the modal operators $\mathbb{K}_1, \ldots, \mathbb{K}_n$.

DEFINITION 4.5. Agent $\mathrm{Ag}_i$ knows the formula $\varphi \in L$ in a global state $g = (w, \mathrm{int}_1, \ldots, \mathrm{int}_n)$ of AS exactly if $(M_{I_{\mathrm{AS}}}, g)| = \varphi$ (i.e., $\varphi$ is satisfied in state $g$ of a structure $M_{I_{\mathrm{AS}}}$). In other words, $(I_{\mathrm{AS}}, g)| = \mathbb{K}_i\varphi \Leftrightarrow \pi(h)(\varphi) = \mathit{true}$ for all $h$ such that $(g, h) \in \mathbb{K}_i$. ■

## 5. HYPERGRAPHS AND HYPERGRAPH GRAMMARS USED BY DESIGN AGENTS

As many designed objects have hierarchical structures there is a preference for a hierarchical approach to modeling design (Suh, 1990; Rosenman & Gero, 1999). Therefore, in this paper the representation of design object structures used by design agents is in the form of hierarchical hypergraphs (Ślusarczyk, 2003; Grabska et al., 2006*a*), which allows representing objects on different levels of detail. Hyperedges of these hypergraphs enable us to express multiargument relations between different parts and subparts of objects.

Hypergraphs used in a multiagent design system contain two types of labeled hyperedges. Hyperedges of the first type are nondirected and represent parts of objects, whereas hyperedges of the second type represent relations among them. They are directed unless they represent symmetrical relations. Hyperedges of the hypergraph are labeled by names of the corresponding components or relations.

To represent features of objects and relations between them, attributing of hyperedges is used. Attributes represent properties (like shape, size, position, color, material) of elements corresponding to hyperedges. Values of attributes obtained by the agent when it is invoked specify the design requirements that should be met by the solutions.

Hyperedges representing object components can contain nested hypergraphs. Hierarchical hyperedges (with nonempty contents) represent object components with different functions or some groups of components. Hierarchical hypergraphs can be then mapped into various graphical models of the objects according to attributes assigned to these hypergraphs.

Let $[i]$ denote the interval $\{1, \ldots, i\}$ for $i \geq 0$ (with $[0] = \emptyset$). Let $\Sigma = \Sigma_{\mathrm{C}} \cup \Sigma_{\mathrm{R}}$, where $\Sigma_{\mathrm{C}} \cap \Sigma_{\mathrm{R}} = \emptyset$, be a fixed alphabet of hyperedge labels. Let $At$ be a set of hyperedge attributes.

DEFINITION 5.1. An attributed hierarchical hypergraph over $\Sigma$ is a system $H = (E_H, V_H, s_H, t_H, \mathrm{lb}_H, \mathrm{att}_H, \mathrm{ext}_H, \mathrm{ch}_H)$, where

1. $E_H = E_{\mathrm{C}} \cup E_{\mathrm{R}}$, where $E_{\mathrm{C}} \cap E_{\mathrm{R}} = \emptyset$, is a finite set of hyperedges, where elements of $E_{\mathrm{C}}$ represent object components and elements of $E_{\mathrm{R}}$ represent relations,
2. $V_H$ is a finite set of nodes,
3. $s_H: E_H \to V_H{}^*$ and $t_H: E_H \to V_H{}^*$ are two mappings assigning to hyperedges sequences of source and target nodes, respectively, in such a way that $\forall e \in E_{\mathrm{C}} s_H(e) = t_H(e)$,
4. $\mathrm{lb}_H: E_H \to \Sigma$ is a hyperedge labeling function, such that $\forall e \in E_{\mathrm{C}}\ \mathrm{lb}_H(e) \in \Sigma_{\mathrm{C}}$ and $\forall e \in E_{\mathrm{R}}\ \mathrm{lb}_H(e) \in \Sigma_{\mathrm{R}}$,
5. $\mathrm{att}_H: E_H \to P(At)$ is a function assigning sets of attributes to hyperedges,
6. $\mathrm{ext}_H: [n] \to V_H$ is a mapping specifying a sequence of hypergraph external nodes,
7. $\mathrm{ch}_H: E_{\mathrm{C}} \to P(A)$ is a child nesting function, where $A = V_H \cup E_H$ is called a set of hypergraph atoms, and the following conditions are satisfied:

- one atom cannot be nested in two different hyperedges,
- a hyperedge cannot be its own child,
- source and target nodes of a nested hyperedge $e$ are nested in the same hyperedge as $e$. ■

Hypergraph nodes express potential connections between hyperedges. To each hyperedge a sequence of source and target nodes is assigned. A hyperedge is called nondirected if sequences of its source and target nodes are equal. Moreover, for each hypergraph a sequence of its external nodes is determined. The length of this sequence specifies the type of a hypergraph.

EXAMPLE 5.1. A hierarchical hypergraph representing an arrangement of kitchen equipment and furniture shown in Figure 3b is presented in Figure 3a. It contains three hierarchical component hyperedges. The hierarchical hyperedge labeled *Kitchen* represents a given space where the pieces are to be located. The four external nodes of this hyperedge represent four walls of the kitchen. The hyperedge labeled *Eating place* contains a hierarchical hypergraph composed of a hyperedge representing a table that is connected by a relational hyperedge with a hierarchical hyperedge labeled *Chairs* representing a group of four chairs. All other component hyperedges represent single pieces of furniture or kitchen equipment. A hyperedge labeling function assigns one name of the set $\Sigma_{\mathrm{C}} = \{$*Chairs, Eating place, Kitchen, chair, cooker, counter, fridge, sideboard, sink, stool, dishwasher*$\}$ to each component hyperedge.

There are five hyperedges representing distance relations between pieces. These hyperedges are labeled *near* or *far*, which means that some pieces should be located close to each other or possibly far away from each other, respectively. For example, sideboards should be placed close together, whereas a fridge and a cooker should be far away from each other. It should be noted that if a relational hyperedge is incident to a node connected with a component hyperedge with no empty contents, then the relation represented by this hyperedge is inherited by all hyperedges nested in the hierarchical one (e.g., all furniture located in the eating place should be placed far away from the cooker).

https://doi.org/10.1017/S0890060408000097 Published online by Cambridge University Press

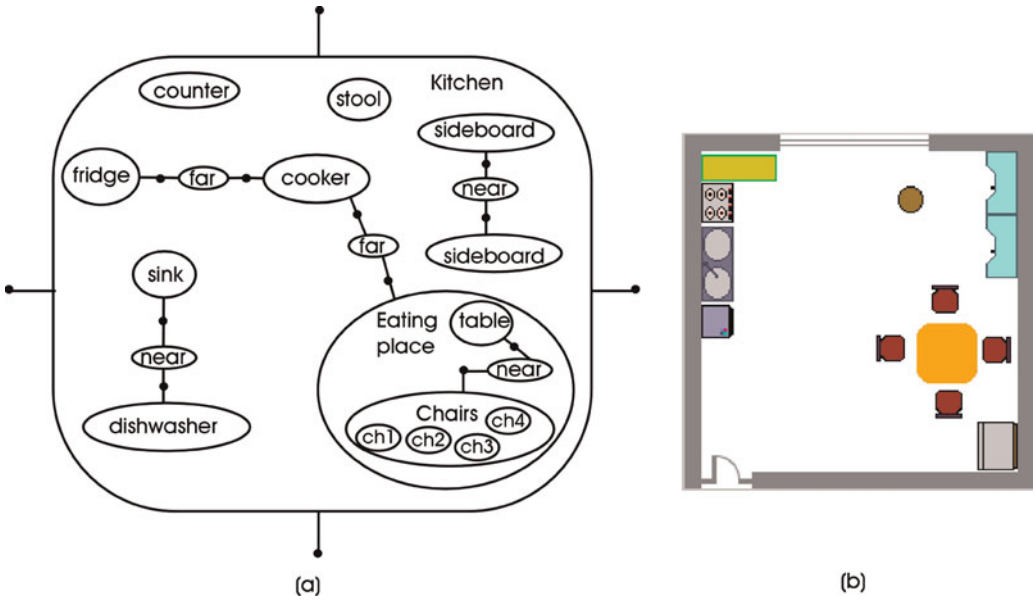


**Fig. 3.** (a) A hierarchical hypergraph representing a kitchen layout and (b) its visualization. [A color version of this figure can be viewed online at www.journals.cambridge.org]

Shapes, colors, and location of the furniture or equipment are specified by values of three attributes, *shape, color*, and *location*, assigned to all nonhierarchical component hyperedges. The *location* can take value 0, which denotes that there are no constrains as to the location of the piece (e.g., *stools, chairs*), 1—the element should be placed by a wall (e.g., *cooker, fridge, sink*), 2—it can be located by the window (e.g., *counter, table*), or 3—it has to be placed by a wall or by the window (e.g., *sideboard*). The attribute *area* is assigned to hierarchical hyperedges. ■

When structures of designs are described in terms of hierarchical hypergraphs, hierarchical hypergraph grammars can serve as efficient tools for generating these structures. Application of grammar rules chosen by design agents corresponds to successive design actions. In this way design agents dynamically control the design process driving it toward solutions compatible with the design criteria.

A hierarchical hypergraph grammar is composed of a set of hypergraph edges with terminal and nonterminal labels, a set of hypergraph nodes, a set of productions, and an axiom being an initial hypergraph. Each grammar production is of the form $p = (l, r, \xi, \mathrm{sr})$, where $l$ and $r$ are attributed hierarchical hypergraphs with the same number of external nodes equipped with ordering relations, $\xi$ is a predicate of applicability, and sr is a set of semantic rules that specify the way in which values of attributes assigned to hyperedges of $l$ are transferred to the attributes assigned to hyperedges of $r$.

Let $N$ and $T$ denote sets of nonterminal and terminal labels, respectively. Let $A = V \cup E$ be a set of atoms of $H$, where $H$ denotes a family of attributed hierarchical hypergraphs over $N \cup T$. Let $\mathrm{lb}_H$ be a hyperedge labeling function.

DEFINITION 5.2. A hierarchical hypergraph grammar over $N$ and $T$ is a system $G = (V, E, P, X)$, where

1. $V$ is a finite set of nodes,
2. $E = E_N \cup E_T$ is a finite set of hyperedges, with a child nesting partial function ch: $E \rightarrow P(A)$, where $\mathrm{lb}_H$: $E_N \rightarrow N$ assigns nonterminal labels to hyperedges of $E_N$, whereas $\mathrm{lb}_H$: $E_T \rightarrow T$ assigns terminal labels to hyperedges of $E_T$,
3. $P$ is a finite set of productions of the form $p = (l, r, \xi, \mathrm{sr})$ satisfying the following conditions:
   - $l$ and $r$ are attributed hierarchical hypergraphs of the same type composed of nodes of $V$ and hyperedges of $E_N \cup E_T$,
   - $l$ contains at least one hyperedge of $E_N$,
   - $\xi$: $H \rightarrow \{\mathrm{TRUE}, \mathrm{FALSE}\}$ is a predicate of applicability,
   - sr is a set of semantic rules defining values of attributes assigned to hyperedges of $r$.
4. $X$ is a hyperedge of $E_N$ with its source and target nodes and such that $\mathrm{ch}(X) = \emptyset$ and called an axiom of $G$. ■

Our design agents use context-free hierarchical hypergraph grammars.

DEFINITION 5.3. A context-free hierarchical hypergraph grammar is a system $G = (V, E, P, X)$, where $P$ is a finite set of productions of the form $p = (l, r, \xi, \mathrm{sr})$ and such that $l$ contains only one hyperedge of $E_N$. ■

EXAMPLE 5.2. Some productions of a context-free hierarchical hypergraph grammar generating structures describing

https://doi.org/10.1017/S0890060408000097 Published online by Cambridge University Press

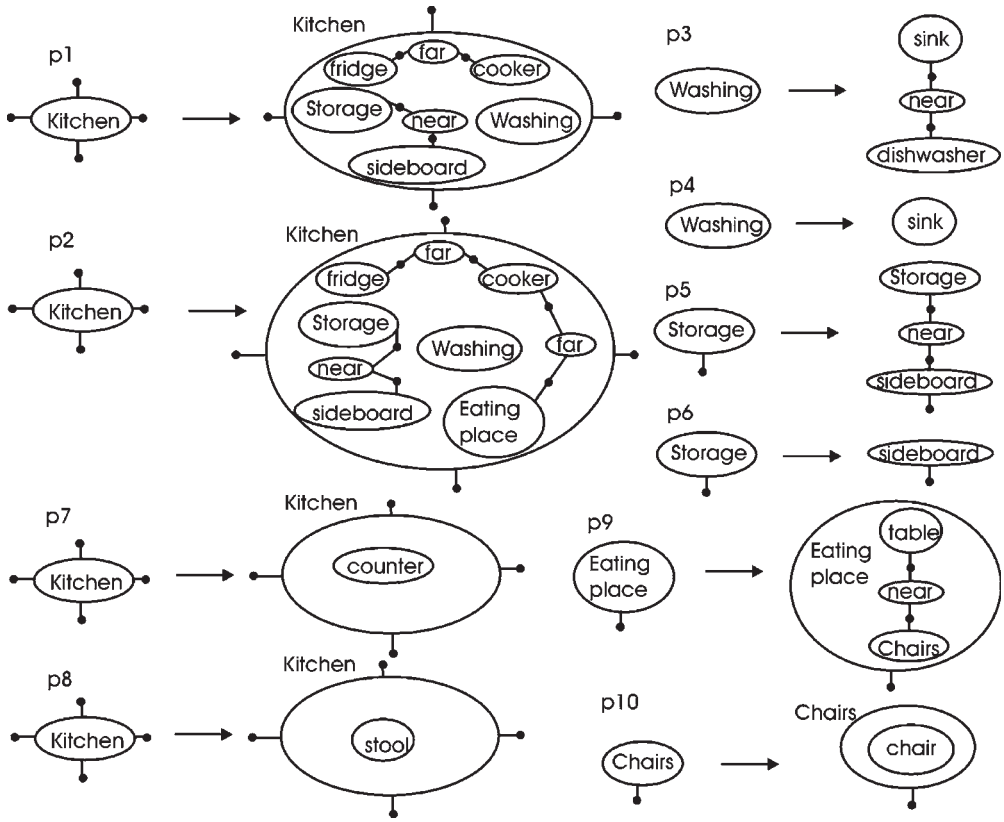


**Fig. 4.** A context-free hierarchical hypergraph grammar generating structures of kitchen layouts.

kitchen layouts are presented in Figure 4 (the predicates of applicability and semantic rules are omitted for the sake of simplicity; they can be found in grammar rules shown later). The first two productions enable the agent to create a kitchen layout with or without the eating place. The third and fourth productions enable the agent to locate a sink and a dishwasher near it or only a sink, respectively. The next two productions allow to fix a required number of sideboards, which should be placed close to each other. The productions $p7$ and $p8$ allow to add a counter and a stool, respectively. The last two productions enable the agent to locate a table and some chairs in the eating place.■

The production $p$ can be applied to a hierarchical hypergraph $H$ if its predicate of applicability is satisfied. The application of $p$ consists in substituting $r$ for a hypergraph isomorphic with $l$, replacing external nodes of the hypergraph being removed isomorphic with nodes of $\mathrm{ext}_l$ by the corresponding external nodes of $\mathrm{ext}_r$ and determining values of attributes of $r$ according to semantic rules of sr.

Definition 5.4. Let $G = (V, E, P, X)$ be a context-free hierarchical hypergraph grammar and $h', h''$ be two attributed hypergraphs, where $h''$ is said to be directly derivable from $h'$ ($h' \Rightarrow h''$) if there exists a production $p = (l, r, \xi, \mathrm{sr})$ of $G$ such that $\xi$ is satisfied for $h'$, $h$ is a subgraph of $h'$ isomorphic with $l$, and $h''$ is isomorphic with the result of replacing $h$ in $h'$ by $r$, substituting external nodes of $h$ by the corresponding external nodes of $r$ and assigning values to attributes of $r$ according to sr. ■

The language generated by a given context-free hierarchical hypergraph grammar is a set of hypergraphs generated starting from the grammar axiom and such that all their hyperedges have terminal labels.

Definition 5.5. Let $G = (V, E, P, X)$ be a context-free hierarchical hypergraph grammar. The language generated by $G$ is a set $L(G) = \{h \in H | X \Rightarrow^* h \text{ and } \forall e \in E_h \mathrm{lb}_h(e) \in T\}$. ■

A hierarchical hypergraph grammar of each agent is equipped with a control diagram. A control diagram, which determines the order in which grammar productions can be applied, is a directed graph whose nodes are labeled by names of productions. Moreover, there are two distinguished nodes, the initial one with no edges coming into it and the final one with no edges coming out of it, labeled by $I$ and $F$, respectively. If a subgraph isomorphic with the left-hand side of a production that should be used is not found in a generated hypergraph, the production will not be applied.

Definition 5.6. A programmed hierarchical hypergraph grammar is a pair $G_P = (G, \mathrm{CD})$, where $G$ is a hierarchical hypergraph grammar and CD is a control diagram for $G$. ■

Example 5.3. A control diagram for a hierarchical hypergraph grammar generating structures of kitchen layouts is presented in Figure 5. Each path from the node labeled $I$ to the node labeled $F$ represents one possible way of obtaining a hypergraph representing a potential arrangement. Application

https://doi.org/10.1017/S0890060408000097 Published online by Cambridge University Press

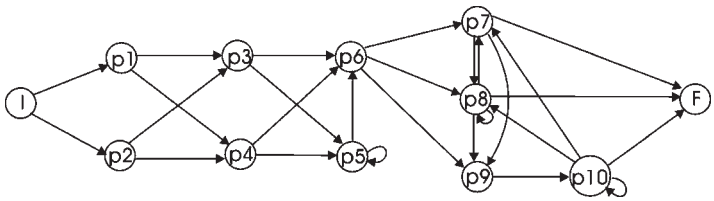


**Fig. 5.** A control diagram for a hypergraph grammar generating structures of kitchen layouts.

of a production sequence $p2$, $p3$, $p6$, $p7$, $p8$, $p9$, $4 \times p10$ results in the hypergraph shown in Figure 3a. ■

It should be noted that although a control diagram for a hypergraph grammar determines the order in which the productions should be applied, it still leaves the possibility of producing a variety of design alternatives as a great or even infinite number of paths leading to different solutions can be specified.

We are interested in a language that is generated by the cooperating agents that not only simultaneously develop different parts of a derived hypergraph but also delegate the derivation of some parts to other agents. Let $G_{P1}, \ldots, G_{Pn}$ denote context-free programmed hierarchical hypergraph grammars of design agents. The generation of a hypergraph representing a solution starts from the axiom $X_1$ of the manager agent's grammar $G_{P1}$ and in the first step one of its productions is used. Then it can invoke other agents and the derivation process can take place in *parallel*. When an agent $\mathrm{Ag}_i$ obtains a message in the form of a hyperedge $h$ it identifies $h$ with the axiom $X_i$ of its grammar and starts the local derivation process. In each step an agent $\mathrm{Ag}_i$ either performs a rewriting step on a locally derived hypergraph using one of the productions of a grammar $G_{Pi}$ or replace a hyperedge in this hypergraph by a hierarchical hypergraph generated by the earlier invoked agent.

Let $H$ denote a family of attributed hierarchical hypergraphs over $N \cup T$.

DEFINITION 5.7. Let $h, h', h'', g \in H$. A derivation process in a multiagent design system with agents equipped with programmed context-free hierarchical hypergraph grammars $G_{P1}, \ldots, G_{Pn}$ is composed of direct derivations of the two following forms:

1. $h''$ is directly derivable from $h'$ ($h' \Rightarrow {}_1 h''$) using one of the productions of $G_{Pi}$, $1 \leq i \leq n$, as specified in Definition 5.4.
2. $h''$ is directly derivable from $h'$ ($h' \Rightarrow_2 h''$) if there exists a hyperedge $h$ in $h'$(isomorphic with axiom $X_i$ of $G_{Pi}$, $1 \leq I \leq n$, $g$ is an attributed hierarchical hypergraph generated starting from $X_i$, and $h''$ is isomorphic with the result of replacing $h$ in $h'$ by $g$ and substituting external nodes of $h$ by the corresponding external nodes of $g$. ■

The language generated by a multiagent design system is a set of hypergraphs, where all hyperedges have terminal labels, and generated by different agents starting from the axiom $X_1$ of the grammar $G_{P1}$ of the manager design agent.

DEFINITION 5.8. Let DAS be a multiagent design system with context-free programmed hierarchical hypergraph grammars $G_{P1}, \ldots, G_{Pn}$.

The language generated by DAS is a set $\mathrm{L(DAS)} = \{h \in H | X_1 \Rightarrow_1 h_1 \{\Rightarrow_1, \Rightarrow_2\}^* h$ and $\forall e \in E_h \ \mathrm{lb}_h(e) \in T\}$. ■

## 6. A SEMANTIC MODEL OF A GRAMMAR-BASED MULTIAGENT DESIGN SYSTEM

Each agent Ag of the design system has a context-free programmed hierarchical hypergraph grammar $G_P$ generating hypergraph representations of design solutions and a database $B$ of facts concerning the subtask for which it is responsible. For example, an agent that designs a house interior has a set of architectural norms, like the minimal surface of rooms and their best geographical location, in its database.

Let a pair $\mathrm{LK} = (G_P, B)$ represent a local knowledge of an agent.

DEFINITION 6.1. A grammar-based design agent $\mathrm{DA} = (\mathrm{Ag}, \mathrm{LK})$ is defined as an intelligent agent $\mathrm{Ag} = (\mathrm{Int}, A, \sigma, \rho, \eta, \chi, \alpha)$ equipped with a domain-dependent local knowledge $\mathrm{LK} = (G_P, B)$. ■

A design process takes place in a specified context that changes with time. This context is expressed by the environment in which agents act and predicates describing design criteria. In addition, agents inform and provide a context for one another. When the agent requires a cooperation it invokes other agents with the design requirements encoded in message attributes. Their values each time can be different according to new predicates.

Let GS denote a set of multiagent design system global states.

DEFINITION 6.2. A context of a multiagent design system is defined as a tuple $\gamma = (\mathrm{Env}, g_0, \psi)$, where Env is the system environment, $g_0 \in \mathrm{GS}$ is the initial global state, and $\psi$ is a set of predicates describing design criteria (functional requirements and constraints). ■

A collection of grammar-based design agents, a specified context and a communication buffer together constitute a grammar-based multiagent design system.

DEFINITION 6.3. A grammar-based multiagent design system is a tuple $\mathrm{DAS} = (\gamma, M_\mathrm{B}, \mathrm{DA}_1, \ldots, \mathrm{DA}_n)$, where

https://doi.org/10.1017/S0890060408000097 Published online by Cambridge University Press

$\gamma$ is a context of a multiagent design system, $M_B$ is a message buffer used by agents and the environment, and $DA_1, \ldots, DA_n$ are grammar-based design agents. ■

As mentioned, the world $W$ of design agents is as a set of hypergraphs that are generated by all agents in a parallel way. The multiagent design system works in the following way. Being in the initial global state the environment sends the message $send(h_0, 1, 0)$, where 0 and 1 denote the environment and the manager agent, respectively, and $h_0$ is the hyperedge with well-specified values of all its attributes. At the next step the manager agent starts the hypergraph derivation process by identifying $h_0$ with the axiom of its grammar. In the successive steps it applies appropriate grammar rules or delegates further derivation to other agents by sending messages.

A set of each design agent's internal states Int is a set of hypergraphs derived using productions of its hierarchical hypergraph grammar. Each internal state $int^k$ corresponds to a hypergraph generated after $k$ steps of derivation. The set of an agent actions $A = A_I \cup A_E$ contains internal actions $A_I$ and actions in the form $a^i = send(\mu, j, n) \in A_E$, where $\mu$ denotes a message of a set $M$ of messages, $j$ is the identifier of the agent to which the message is sent, and $n$ is the identifier of an agent that sends the message. The set $M$ contains three types of messages: an error message, a generated hypergraph, which is returned to the agent that earlier requested cooperation, and a hyperedge with specified values of all attributes, which corresponds to delegating a design subtask to the other agent. Each internal action $a^i \in A_I$ changes the current internal state of an agent by applying a hypergraph grammar rule to the hypergraph being derived, by replacing a hyperedge of the hypergraph with an obtained hypergraph, or by storing a complete solution and starting generation of a new one. In addition, $A$ contains a special *null* action that corresponds to the agent performing no action.

In the sensing process a design agent searches the buffer for messages addressed to it. Thus, if the agent $DA_j$ performed an action $send(\mu, i, j)$, then the agent $DA_i$ in the $k$ step of its run obtains the message as the result of sensing process $\sigma$. Then the kind of the obtained message $\mu$ is determined in the perception process $\rho$. In the next step, in the internal state changing process the values of design variables are updated according to the kind of the obtained message. $\mu$ is always in the form of an attributed hypergraph. If this hypergraph is empty, the message means that the agent $DA_j$ was unable to perform a delegated subtask. Therefore, the hyperedge in the hypergraph generated by the agent $DA_i$ that was to be further developed by the agent $DA_j$ remains unchanged. If $\mu$ has the form of a hypergraph containing more elements than one nonhierarchical hyperedge, it means that the agent $DA_j$ has returned a hypergraph corresponding to a subtask solution. In this case the appropriate hyperedge in the hypergraph generated by the agent $DA_i$ is replaced by $\mu$. If $\mu$ in the form of such a hypergraph is obtained by the environment, then it represents a complete solution of a given design task, and is added to the global database of solutions. If $\mu$ has the form of one hyperedge, it means that the agent $DA_j$ delegated a subtask to the agent $DA_i$. Then the values of design variables in the short-term memory are updated according to the values of attributes of $\mu$, $\mu$ is identified with the axiom of the agent's grammar $G_{Pi}$ and the generation of a new hypergraph is started.

Before each step of a hypergraph derivation an agent evaluates its current internal state representing a partial solution and selects an action in a decision making process $\chi$. As a result of the next grammar rule application, the internal state of the agent is changed. When no further rule can be applied and the derived hypergraph $h$ does not belong to the language generated by the agent ($h \notin L(G_{Pi})$), a message in the form of an empty hypergraph is sent to the agent $DA_j$ by which the considered one was invoked. If $h \in L(G_{Pi})$, the message $\mu = h$ is sent to the agent $DA_j$. If the cooperation of the other agent is needed, a message in the form a hyperedge, which should be further developed by the agent $DA_j$, is sent to the message buffer.

The agent's evaluation capability depends on the possessed knowledge. This evaluation is based on the design knowledge included in the database of facts concerning the subtask it is responsible for and values of hypergraph attributes describing requirements and constraints specified in the design context. The agent evaluates both partial and complete solutions. The results of partial solutions evaluation influence the agent decisions about required modifications and the direction of further search for solutions.

Each design agent $DA_i$ has in its database $B_i$ an interpretation $I_i$ for hierarchical hypergraphs of $L(G_{Pi})$. The interpretation determines the possible ways of mapping an attributed hierarchical hypergraph into graphical models by specifying a semantic meaning of hypergraph elements. It assigns geometrical objects to hyperedges representing artifact components, and establishes a correspondence between relational hyperedges and relations between these objects. The interpretation enables the agents to appropriately arrange object components and create a visualization of the whole designed object.

DEFINITION 6.4. Let $H = (E_H, V_H, s_H, t_H, lb_H, att_H, ext_H, ch_H)$ be an attributed hierarchical hypergraph over $\Sigma = \Sigma_C \cup \Sigma_R$.

An interpretation for $H$ is a function $I = (I_C, I_R)$, where

1. $I_C$: $E_C \times \Sigma_C \rightarrow O$, where $O$ is a finite set of geometric objects, assigns objects to labeled component hyperedges,
2. $I_R$: $E_R \times \Sigma_R \rightarrow R$, where $R$ is a finite set of relations between geometric objects, assigns relations to labeled relational hyperedges. This function enables to correctly assign transformations that should be applied to geometric objects represented by component hyperedges when a hierarchical hypergraph visualization is created. ■

https://doi.org/10.1017/S0890060408000097 Published online by Cambridge University Press

EXAMPLE 6.1. The interpretation for the hierarchical hypergraph shown in Figure 3a assigns icons representing appropriate pieces to all component hyperedges according to their labels. A set of icons representing a fridge, cooker, sink, stool, table, counter, sideboard, and a dishwasher (from left to right) is shown in Figure 6. Then the transformations being compositions of translations and rotations are assigned to component hyperedges in a way that enable location of the icons according to the relations specified by relational hyperedges and values of the component hyperedge attribute *location* described in Example 5.1. They ensure that the *cooker, fridge*, and *sink* are located by the wall, the *cooker* is placed far from the *fridge, table, chairs*, and so forth. Three graphical models corresponding to the considered hypergraph are shown later. ■

Let $G_{P1}, \ldots, G_{Pn}$ denote hierarchical hypergraph grammars of design agents $DA_1, \ldots, DA_n$ and $I_1, \ldots, I_n$ denote agents' interpretations for elements of $L(G_{P1}), \ldots, L(G_{Pn})$, respectively. For each hypergraph $h$ of $L(G_{Pi})$, $I_i(h)$ is called a graphical model of $h$.

The interpretation $I$ for the DAS is defined as $I = I_1 \cup \cdots \cup I_n$. The global database DB of the system contains hypergraphs belonging to the language $L$(DAS) generated by a DAS with grammars $G_{P1}, \ldots, G_{Pn}$. Thus, $I(L(\text{DAS}))$ denotes a set of all graphical models reachable by the grammar-based multiagent design system.

DEFINITION 6.5. Let $\pi$ be an interpretation for the predicates of $\psi$ over $I(L(\text{DAS}))$. An interpreted design system $I_{\text{DAS}}$ generated by the structure DAS is a triple (DB, $I$, $\pi$). ■

STATEMENT 6.1. Given an interpreted design system $I_{\text{DAS}} = (\text{DB}, I, \pi)$.

A hypergraph $h \in DB$ is a valid design solution in respect to $\varphi \in \psi$ if the manager design agent $DA_M$, which starts the generation of solutions and sends the generated ones to the environment, knows that the formula $\varphi$ is satisfied for $h$. In other words, $(I_{\text{DAS}}, h) \models \mathbb{K}_M \varphi \Leftrightarrow \pi(\varphi(I(g))) = true$ for all $g$ isomorphic with $h$. ■

STATEMENT 6.2. An interpreted design system $I_{\text{DAS}}$ generates design solutions consistent with the design criteria if $\forall h \in \text{DB} \forall \varphi \in \psi (I_{\text{DAS}}, h) \models \mathbb{K}_M \varphi$. ■

The agents of a grammar-based multiagent design system jointly solve a problem by modifying the contents of a global database. Thus, computations are situated and connected to the context through interaction.

**Fig. 6.** Icons representing kitchen pieces. [A color version of this figure can be viewed online at www.journals.cambridge.org]

## 7. CREATIVE DESIGN WITH INTELLIGENT GRAMMAR-BASED AGENTS

At the beginning of the design process all the information needed to perform a design task is supplied by the designer who has to take into account all needs of the client. Input information includes a description of the estate, garden, garage, and rooms required in the house, as well as the style and types of furniture. The global database of generated solutions can be empty or include some prototypical solutions of similar tasks, whereas the communication buffer where messages from agents are stored is empty.

### 7.1. The estate manager design agent

The requirements and constraints of the design project are specified in the initial state of the environment as values of attributes of the initial hyperedge. The manager agent, which is responsible for the whole design task, is expected to generate the arrangements of the whole estate using rules of the given hypergraph grammar, combine it with furnished floor layouts and garden arrangements generated by other agents, and return task solutions to the environment.

At the outset of the design the environment sends a message to the manager agent in the form of an attributed hyperedge. The perception process of the manager agent consist of reading and storing attributes of this hyperedge describing the design requirements, like the number of rooms in the layout, their area and shape, the area and the features of the garden, the necessity of creating a garage. The internal state of the agent is changed by updating the values of design variables (stored in the initial state of agent's short-term memory) and identifying the obtained hyperedge with the axiom of the agent's grammar. Thus, the new internal state is obtained and the agent starts the hypergraph derivation process using grammar rules.

EXAMPLE 7.1. Two selected productions of a hierarchical hypergraph grammar of the estate manager agent with predicates $\xi$ determining the conditions of their applicability are presented in Figure 7b. Some semantic rules defining the way in which values of attributes of the right-hand sides are inherited from the ones of the left-hand sides are shown. The first production is applied if a garage should be present and the garden should be accessible from the north side of the house. The second production allows the agent to generate an estate without a garage and with the garden accessible from the east side of the house. The hyperedge labels *House Interior* and *Garden* indicate that the design of these parts will be delegated to other agents. ■

On the basis of the current internal state the agent either chooses a local action or sends a message. A local action consists in applying a grammar rule that corresponds to the specified values of design variables.

EXAMPLE 7.2. Let us assume that the manager agent obtains an initial hyperedge shown in Figure 7a with fixed

https://doi.org/10.1017/S0890060408000097 Published online by Cambridge University Press

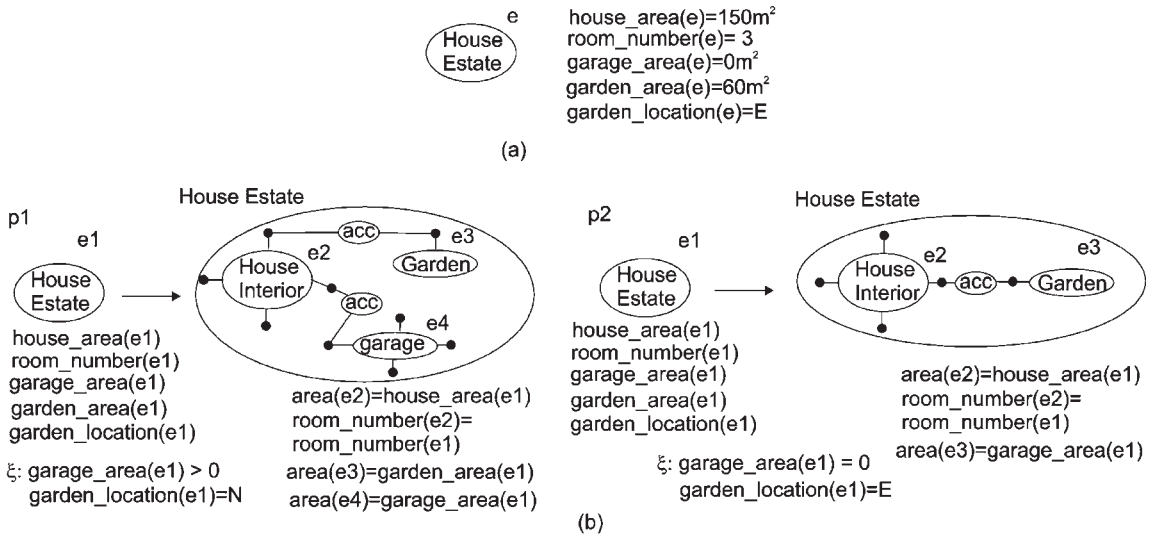


**Fig. 7.** (a) A hyperedge with design requirements and (b) two rules of a hierarchical hypergraph grammar of the manager design agent.

values of hyperedge attributes, which reflect the requirements concerning the area of a house, garden, and garage, the number of bedrooms, and the garden location. Given this set of example constrains the agent searches for a feasible layout of the estate. The values of the attributes obtained from the environment make the agent apply the second production. ■

When the solution being generated does not satisfy the required conditions the agent sends an "error" message in the form of an empty hypergraph to the environment. If the agent finds in the generated hypergraph a hyperedge with a label *House Interior* or *Garden*, which indicates that the cooperation of another agent is needed, it sends a message in the form of this hyperedge with the current values of its attributes to the agent, which should generate an arrangement for the space corresponding to the found label. As the agent sensors scan the message buffer it can easily spot the solution returned by the earlier invoked agent. Then it takes the other local action and combines the returned arrangement with the solution generated by itself by replacing the earlier sent hyperedge with the obtained hypergraph.

### 7.2. The house interior design agent

The house interior design agent generates structures of floor layouts. It starts the generation of a hypergraph after obtaining a message in the form of a hyperedge labeled *House Interior*.

EXAMPLE 7.3. Some rules of a hierarchical hypergraph grammar of an agent generating house interiors are shown in Figure 8. The attributes of component hyperedges of both sides of productions, some semantic rules, and predicates of applicability are depicted for the first three productions. All relational hyperedges represent accessibility between the rooms. The first production divides the floor area into three functional units, a hall, a kitchen, and an antechamber under the condition that there is more than 60 $m^2$ to be used. The second and third production allow us to choose a layout with one or two bedrooms in the first sleeping area. The fourth production adds a living room connected with a terrace. The productions $p5$ and $p6$ enable the agent to obtain different layouts composed of two rooms, a shower, and a small hall. The former rule gives a possibility to locate the shower between the rooms, whereas the latter one allows the shower to be adjacent to the one room only, but to have two entrances, one from the *hall1* and the second from the *room*. A control diagram for this hypergraph grammar is shown later. ■

After each step of derivation the house interior design agent evaluates the partial solutions, which are possible to obtain by using admissible grammar rules in respect to the information in the database and requirements specified in the design context.

EXAMPLE 7.4. Let us assume that the agent generating floor layouts obtained a hyperedge $e$ labeled *House Interior* with $house\_area(e) = 150\ m^2$, $room\_number(e) = 3$. Then these values are given to the attributes of the grammar axiom, which is isomorphic with the left-hand side of the first production. After application of this production the value of the attribute *room_number* of the hyperedge labeled *Sleeping1* is equal to 1 and therefore in the next step the second production is applied. A hypergraph generated by the grammar according to the mentioned criteria is shown in Figure 9, whereas one of the possible floor layouts corresponding to this hypergraph is shown in Figure 10a.

The considered grammar contains two productions: one generating two rooms and another generating three rooms in the *Sleeping1* area, both applicable when the value of *room_number* attribute is greater than 1. In such a case, the agent selects a production to be applied on the basis of the

https://doi.org/10.1017/S0890060408000097 Published online by Cambridge University Press

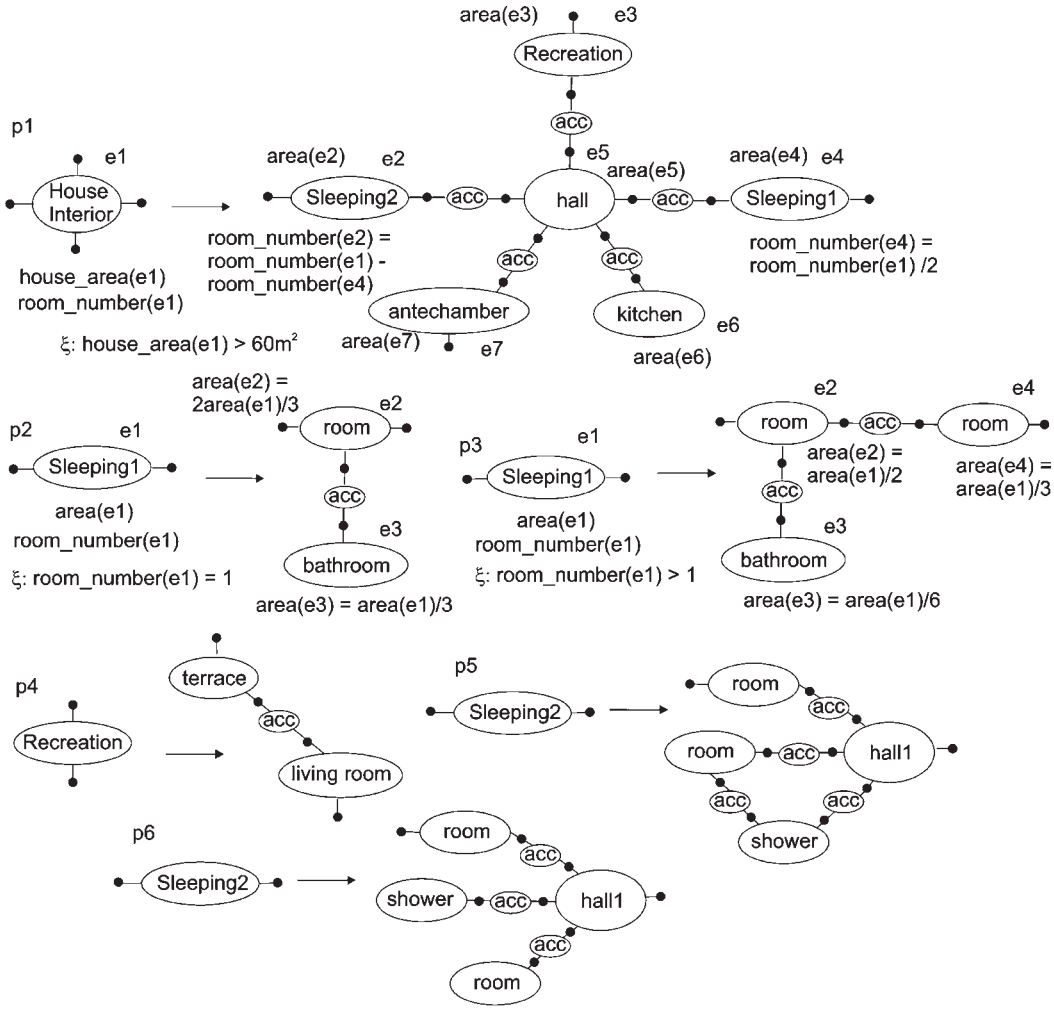


**Fig. 8.** Some of the rules of a hypergraph grammar generating structures of floor layouts.

current value of the attribute *area* of the hyperedge *Sleeping1*. If this value is smaller than 30 $m^2$, the agent creates only two rooms using production *p*3. After application of this production the agent evaluates the obtained hypergraph using its database. It checks if the values of the attribute *area* of both hyperedges labeled *room* are in the range specified for a bedroom area in the database. If they are smaller than the stored norms an error message is sent to the manager agent. The hypergraph generated using productions *p*3 and *p*5, according to the design requirements including four bedrooms and a shower accessible from two sides, together with the corresponding floor layout are presented later. ■

At each step the agent applies the most promising grammar production selected from the set of rules that can be used at this point of derivation or sends a message to the other agent.

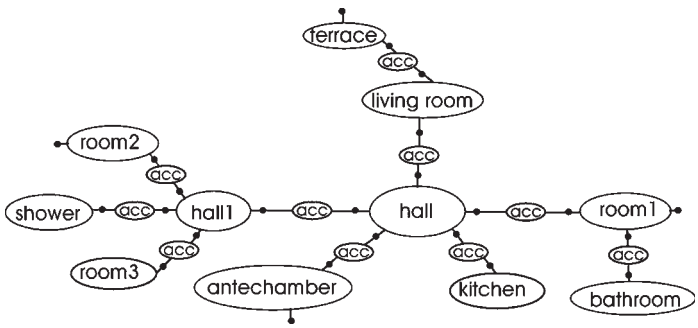


**Fig. 9.** A hypergraph generated by the hypergraph grammar from Figure 8.

https://doi.org/10.1017/S0890060408000097 Published online by Cambridge University Press

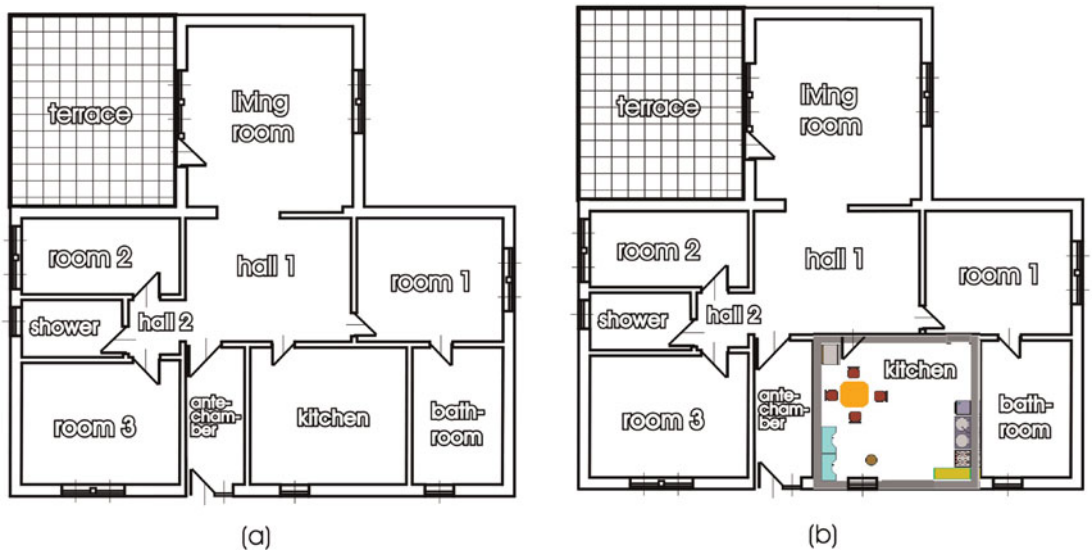


**Fig. 10.** (a) A floor layout corresponding to the hypergraph from Figure 9 and (b) a floor layout with a kitchen arrangement. [A color version of this figure can be viewed online at www.journals.cambridge.org]

If the generated hypergraph contains hyperedges labeled by elements of the set {*living-room, room, bath-room, kitchen*} the house interior design agent delegates tasks to the agents generating furniture arrangements for appropriate types of spaces in the house. Then it combines the returned arrangements with the solution generated by itself. The floor layout combined with a kitchen arrangement (a fridge, cooker, sink, dishwasher, counter, stool, two sideboards, and a table with four chairs) is shown in Figure 10b.

### 7.3. The furniture arrangement design agents

There are four agents generating furniture arrangements in the system. They produce layouts for livingrooms, bedrooms, bathrooms, and kitchens. Each of them is invoked by the house interior design agent when the appropriate hyperedge label occurs in the hypergraph generated by this agent.

The knowledge bases of these agents contain sets of icons (graphic representations) corresponding to the predefined furniture and equipment. The values of the attributes assigned to these icons specify the shape, color, and location of the furniture or equipment.

EXAMPLE 7.5. Two-dimensional graphic representations of a furniture set in the knowledge base of an agent generating arrangements for living rooms are shown later. They correspond to a couch, armchair, table, bookcase, coffee table, television set, chair, wardrobe, and a cupboard. ■

When an arrangement agent senses the message in the communicating buffer it reads the attributes describing the design criteria (like the area of the space, the location of doors and windows) in the perception process. Then it starts generation of the arrangement using a given hypergraph grammar and a knowledge base where the information about furniture and equipment for the given space is stored.

As the behavior of all furniture arrangement agents is similar, it will be explained on the example of the kitchen arrangement agent. Its database of equipment contains information about a sink, dishwasher, cooker, fridge, sideboard, counter, stool, table, and chair. Analogously as the manager and the house interior design agents, the furniture arrangement agent after each step of derivation also evaluates the partial solutions possible to obtain in respect to the given requirements and makes the decision about the action that is to be taken next.

EXAMPLE 7.6. The agent's hypergraph grammar generating structures describing kitchen layouts is presented in Figure 4. A production that generates a hypergraph with a hyperedge representing an eating place in the kitchen is used only when the value of the attribute *area* assigned to the hyperedge *Kitchen* (rule $p2$) is greater than 10 $m^2$. In the opposite case, there would not be enough space for all needed equipment or the arrangement of furniture would not be comfortable. ■

Hypergraphs obtained in the derivation process and representing partially specified solutions can be mapped into graphical models using a given hypergraph interpretation (Grabska et al., 2003). Thus, each furniture arrangement design agent has the following three types of rules: hypergraph grammar rules, which are used to generate hypergraphs representing pieces arrangements; interpretation rules, which enable the agent to map a generated hypergraph into a graphical model by assigning semantic information to hypergraph elements; and constrain rules, which describe the possible sizes and orientations of geometric primitives of a database by specifying admissible scaling and rotations.

Evaluation of partial solutions enables the agent to select the grammar rule, which should be used in the next derivation step, take decisions about required modifications or find some new design concepts and requirements. If some modifications

https://doi.org/10.1017/S0890060408000097 Published online by Cambridge University Press

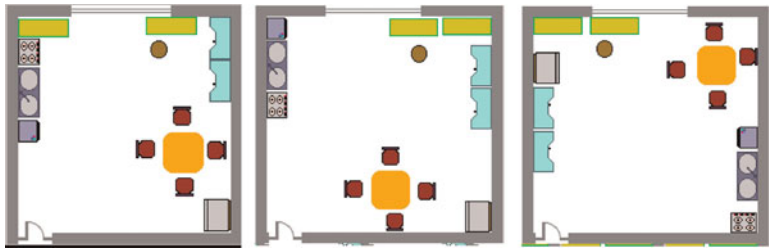

**Fig. 11.** Three different kitchen arrangements compatible with the same criteria. [A color version of this figure can be viewed online at www.journals.cambridge.org]

of the generated solution are required, modifying rules, which can be added to the hypergraph grammar, may be used (Ślusarczyk, 2004). When new design ideas or requirements emerge the agent can create new grammar rules and add them to the control diagram (Grabska et al., 2006*c*). The inspection of partial solutions at successive stages of the design process allows the agent to evaluate many possible variants of component arrangements and to choose the proper direction of search.

Each arrangement design agent generates a set of solutions that are compatible with the initial requirements. This set can be treated as the contents of the agent's long-term memory. Therefore, when the house interior design agent invokes an arrangement agent for the second time with the same initial requirements it can behave in a reactive manner returning one of the earlier generated solutions. However, in case of new initial requirements it has to start the generation again.

EXAMPLE 7.7. Three different kitchen arrangements generated for the same the criteria specification (an eating place with a table and chairs, a dishwasher near a sink, a fridge far from the cooker, and at last two sideboards and one counter) are presented in Figure 11. ■

Grammar-based design agents can effectively assist the in the design process. They not only cooperate with each other but also dynamically react to changes in the design context.

EXAMPLE 7.8. Three examples of the results obtained using five cooperating grammar-based agents to the problem of designing house interiors with furniture arrangements are shown in Figure 12. The rooms of the floor layout from Figure 10a are filled with different furniture arrangements. ■

## 8. IMPLEMENTATION OF THE SYSTEM

The prototype multiagent design system is written in Java, and enables us to verify the proposed approach. It contains one module for each agent type. In each module the designer can define the agent's hypergraph grammar and add to a database constrains concerning geometric primitives corresponding to component hyperedges. The interface window of the agent that generates floor layouts of a house is shown


**Fig. 12.** Three different furniture arrangements for the same floor layout. [A color version of this figure can be viewed online at www.journals.cambridge.org]

https://doi.org/10.1017/S0890060408000097 Published online by Cambridge University Press

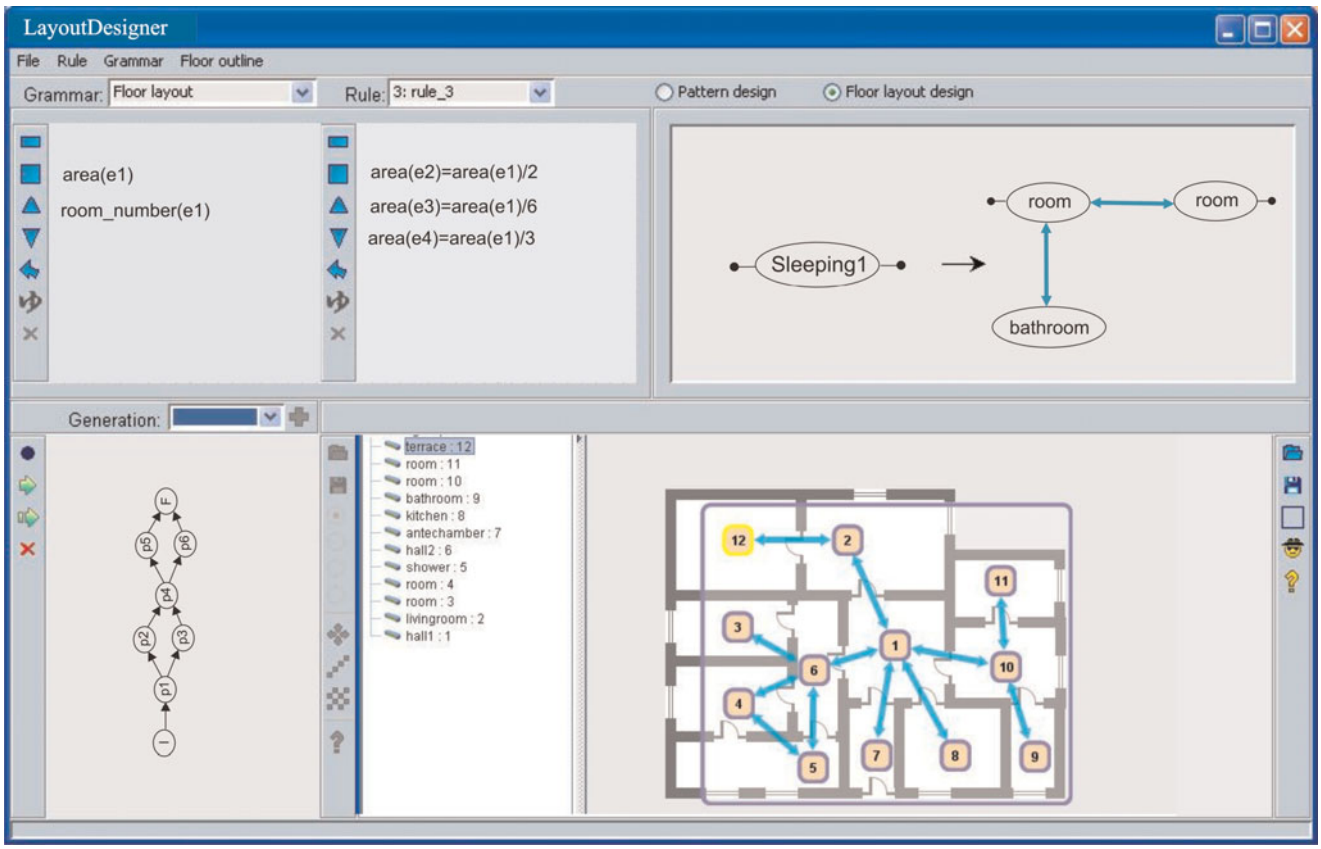


**Fig. 13.** An interface of a floor layout design agent. [A color version of this figure can be viewed online at www.journals.cambridge.org]

in Figure 13. In its upper part the designer defines a hypergraph grammar of the agent. One of the productions of this grammar is presented in the right-hand side panel. On the left-hand side of the upper part of the window the attributes and semantic rules of productions are specified. A control diagram for the hypergraph grammar can be defined and edited in the bottom left-hand side panel.

When the manager agent is activated and the system starts to work, the designer can switch between the windows of different agents. In the bottom right-hand side panel the hypergraph generated by a given agent, its graphical model, or both views together with the transparent hypergraph on the top of the layout can be seen. A hypergraph generated by a house interior design agent together with a layout being its graphical model is presented on the right-hand side of Figure 13. The labels of hyperedges are listed on the left of the hypergraph. The same panel with a furniture layout for a living room generated by one of the arrangement agents is shown in Figure 14a. Some icons representing furniture of this agent database are shown on the right-hand side of this panel.

The bottom right-hand side panel of the agent's window is also used to present hypergraphs obtained after replacing some hyperedges by hypergraphs generated by cooperating agents and their graphical interpretations. A panel with a floor layout of a house with a living room filled with furniture, which corresponds to the hypergraph generated by the house interior design agent, where the hyperedge labeled *living_room* was replaced by a hypergraph returned by the living room arrangement agent, is shown in Figure 14b. In the present version of the system a graphical layout is created by locating first one primitive representing a room and then successively fitting others one by one to the previously located ones. There is no constraints solver used but all eventual inconsistencies can be modified in a free-hand mode.

## 9. CONCLUSIONS

Cooperating grammar-based design agents seem to be promising assistants in the process of creative design. In this paper a grammar-based multiagent design system was presented. Solutions created by such a design system can be easily adapted to the clients' individual needs without the designer intervention. The proposed semantic model of a design system enables agents to check whether generated hypergraphs representing design solutions are consistent with the design criteria. The interpretation of hypergraphs generated by the system allows the designer to view and evaluate graphical models of solutions.

A multiagent system with a collection of communicating interface design agents can model diverse requirements and constrains characteristic for dynamic design context. Context

https://doi.org/10.1017/S0890060408000097 Published online by Cambridge University Press

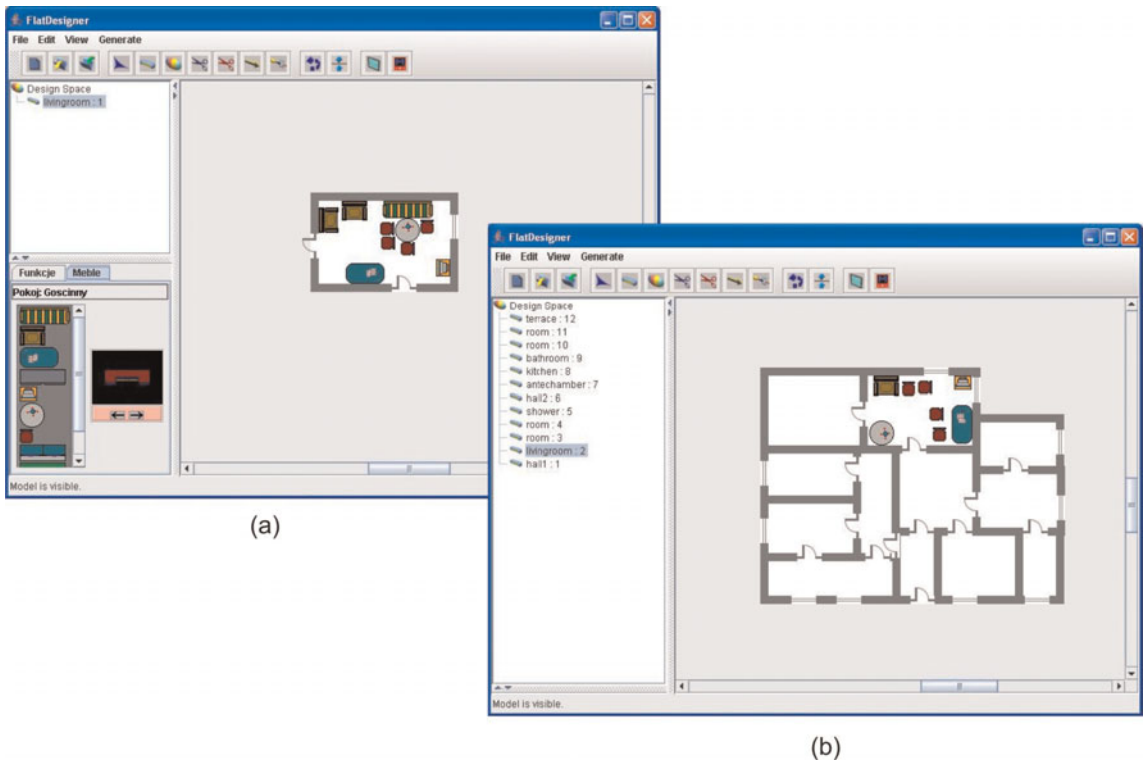


**Fig. 14.** (a) A furniture layout of a living room and (b) a floor layout of a house with a furnished living room. [A color version of this figure can be viewed online at www.journals.cambridge.org]

drives the design process by the influence on the way in which the agent makes the decisions: which rule to apply, when and where to apply it, when to terminate the generation, and which design solution to select. Thus, design requirements drive the solution development at each step of the generation process. Interaction with other agents and the feedback from the environment make the successful design possible.

In this paper we focused on systems composed of agents with one type of grammars. In our future work we intend to extend this approach to incorporate agents equipped with nonvisual descriptive grammars and working on the global database of already created designs. Such agents could, for example, compute quantitative or qualitative information about designs generated by other agents. They could query the database to find the most plausible designs, estimate the costs of building and furnishing the designed houses, and arranging the planned gardens, evaluate the accessibility of needed materials and garden plants. They should also be able to search the database for already existing solutions to a problem similar to a given new one.

As an agent's decision method for choosing goals is based on current information about the design task, the agent should be capable of learning and abstracting knowledge. Therefore, we intend to equip design agents with a learning function. On the basis of the evaluation notes given by other agents, environment, or the designer to hypergraphs representing generated solutions the agent will learn which sequences of grammar productions are preferred and modify the control diagram by updating weights determining the probability of production application.

In the present version of the system the agents cooperate and inform each other about the impossibility of finding valid solutions, but they do not negotiate. In the future, we would like the agents to have more influence on the other agents' decisions. For example, the furniture arrangement agent should be able to make the floor layout design agent move some doors or windows.

## REFERENCES

Agarwal, M., Cagan, J., & Constantine, G.C. (1999). Influencing generative design through continuous evaluation: associating costs with the coffeemaker shape grammar. *Artificial Intelligence for Engineering Design, Analysis and Manufacturing 13*, 253–275.

Borkowski, A., & Grabska, E. (1995). Representing design by composition graphs. In *IABSE Colloquium, Knowledge Support Systems in Civil Engineering, IABSE Reports*, pp. 27–36, Zurich.

Cagan, J. (2001). Engineering shape grammars. In *Formal Engineering Design Synthesis* (Antonsson, E.K., & Cagan, J., Eds.). Cambridge: Cambridge University Press.

Cagdas, G. (1996). A shape grammar: the language of traditional Turkish houses. *Environment and Planning B 23*, 443–464.

Campbell, M., Cagan, J., & Kotovsky, K. (1998). A-design: theory and implementation of an adaptive agent-based method of conceptual design. In *Artificial Intelligence in Design'98* (Gero, J.S., & Sudweeks, F., Eds.), pp. 579–598. Dordrecht: Kluwer.

https://doi.org/10.1017/S0890060408000097 Published online by Cambridge University Press

Campbell, M., Cagan, J., & Kotovsky, K. (1999). A-design: an agent-based approach to conceptual design in a dynamic environment. *Research in Engineering Design 11*, 172–192.

Canamero, D. (1997). Modeling motivations and emotions as a basis for intelligent behavior. *Proc. 1st Int. Conf. Autonomous Agents*, pp. 148–155, Marina del Rey, CA.

Carlson, C., Woodbury, R., & McKelvey, R. (1993). An introduction to structure and structure grammars. *Environment and Planning B 18*, 417–426.

Drewes, F., Hoffmann, D., & Plump, D. (2000). Hierarchical graph transformation. *Proc. FOSSACS 2000, Lecture Notes in Computer Science*, Vol. 1784, pp. 98–113.

Dzeroski, S. (2002). Relational reinforcement learning for agents in worlds with objects. *Proc. Symp. Adaptive Agents and Multi-Agent Systems (AISB'02)*, pp. 1–8.

Fagin, R., Halpern, J.Y., Moses, Y., & Vardi, M.Y. (1995). *Reasoning About Knowledge*. Cambridge, MA: MIT Press.

Gaborit, P., Potet, A., & Sayettat, C. (1990). Semantics and validation procedures of a multi-modal logic for formalization of multi-agent universes. *Proc. 9th European Conf. Artificial Intelligence*, pp. 289–291. Stockholm: Pitman.

Gero, J.S., & Kannengiesser, U. (2002). The situated function–behaviour–structure framework. In *Artificial Intelligence in Design'02* (Gero, J.S., Ed.), pp. 89–104. Dordrecht: Kluwer.

Gero, J.S., & Kannengiesser, U. (2003). Towards a framework for agent-based product modelling. *Int. Conf. Engineering Design, ICED'03*, pp. 1621–1622, Stockholm.

Grabska, E., Ślusarczyk, G., & Papiernik, K. (2003). Interpretation of objects represented by hierarchical graphs. *Proc. Conf. Computer Recognition Systems, KOSYR'03*, pp. 287–293, Wrocław, Poland.

Grabska, E., Ślusarczyk, G., & Grześ, P. (2005). Dynamic design with the use of intelligent agents. *Proc. 4th Int. Conf. Computer Recognition Systems, CORES'05*, pp. 827–834. Berlin: Springer.

Grabska, E., Grzesiak-Kopeć, K., Lembas, J., Łachwa, A., & Ślusarczyk, G. (2006*a*). Hypergraphs in diagrammatic design. In *Computer Vision and Graphics* (Wojciechowski, K., et al., Eds.), pp.111–117. Berlin: Springer.

Grabska, E., Grzesiak-Kopeć, K., & Ślusarczyk, G. (2006*b*). Designing floor-layouts with the assistance of curious agents. In *ICCS 2006 Part III, Lecture Notes in Computer Science* (Alexandrov, V.A., et al., Eds.), Vol. 3993, pp. 883–886. Berlin: Springer.

Grabska, E., Grzesiak-Kopeć, K., & Ślusarczyk, G. (2006*c*). Visual creative design with the assistance of curious agents. In *Proc. 4th Int. Conf. Diagrammatic Representation on Inference, Diagrams 2006*, *Lecture Notes in Computer Science* (Baker-Plummer, D., et al., Eds.), Vol. 4045, pp. 218–220. Stanford, CA: Springer.

Grecu, D.L., & Brown, D.C (2000). Expectation formation in multi-agent design systems. In *Artificial Intelligence in Design'00* (Gero, J.S., Ed.), pp.651–671. Dordrecht: Kluwer.

Habel, A., & Kreowski, H.J. (1987). Some structural aspects of hypergraph languages generated by hyperedge replacement. In *Lecture Notes in Computer Science*, Vol. 247, pp. 207–219. Berlin: Springer–Verlag.

Kokoszka, A., Bielecki, A., & Holas, P. (2001). Mental organization according to metabolism of information and its mathematical description. *International Journal of Neuroscience 107*, 173–184.

Lander, S.E. (1997). Issues in multiagent design systems. *IEEE Expert 12*, 18–26.

Longenecker, S.N., & Fitzhorn, P.A. (1991). A shape grammar for non-manifold modeling. *Research in Engineering Design 3*, 159–170.

McCormack, J.P., & Cagan, J. (2002). Designing inner hood panels through a shape grammar based framework. *Artificial Intelligence for Engineering Design, Analysis and Manufacturing 16*, 273–290.

Myers, L., & Pohl, J. (1994). ICDM: integrated cooperative decision making-in practice. *Tools With Artificial Intelligence. Proc. 6th Int. Conf.*, pp. 608–614.

Paun, Gh., & Salomaa, A. (Eds.). (1999). *Grammatical Models of Multi-Agent Systems*. Amsterdam: Gordon & Breach.

Rosenman, M., & Gero, J.S (1999). Evolving designs by generating useful complex gene structures. In *Evolutionary Design by Computers* (Bentley, P.J., Ed.), pp. 345–364. San Francisco, CA: Morgan Kaufmann.

Saunders, R. (2001). Curious design agents and artificial creativity. PhD Thesis. University of Sydney.

Ślusarczyk, G. (2003). Hierarchical hypergraph transformations in engineering design. *Journal of Applied Computer Science 11*, 67–82.

Ślusarczyk, G. (2004). Heuristic methods and hierarchical graph grammars in design. *Visual and Spatial Reasoning in Design III*, pp. 45–66, University of Sydney, Key Centre of Design Computing and Cognition.

Soman, A., & Campbell, M. (2002). A grammar-based approach to sheet metal design. *Proc. DETC'02 ASME Design Engineering Technical Conf.*, pp. 1–9, Montreal.

Sosa, R., & Gero, J.S. (2004). A computational framework for the study of creativity and innovation in design: effects of social ties. In *Design Computing and Cognition'04* (Gero, J.S., Ed.), pp. 499–517. Dordrecht: Kluwer.

Stiny, G., & Mitchell, W.J. (1980). The grammar of paradise: on the generation of Mughul gardens. *Environment and Planning B 7*, 209–226.

Suh, N.P. (1990). *The Principles of Design*. New York: Oxford University Press.

Traverso, P., & Spalazzi, L. (1995). A logic for acting, sensing and planning. *Proc. 14th Int. Joint Conf. Artificial Intelligence*, pp. 1941–1949, Montreal.

Wooldridge, M.J (1995). This is MYWORLD: the logic of an agent oriented DAI testbed. *Intelligent Agents: Proc. ECAI'94 Workshop on Agent Theories, Architectures and Languages, Lecture Notes in Artificial Intelligence* (Wooldridge, M., & Jennings, N.R., Eds.), Vol. 890, pp. 160–178. Amsterdam: Springer–Verlag.

Wooldridge, M.J. (1999). Intelligent agents. In *Multiagent Systems: A Modern Approach to Distributed Artificial Intelligence* (Weiss, G., Ed.), pp. 27–77. Cambridge, MA: MIT Press.

Wooldridge, M.J., & Lomuscio, A. (2000). Multi-agent VSK logic. *Proc. 7th European Workshop on Logics in Artificial Intelligence (JELIAI-2000)*. Berlin: Springer–Verlag.

Velasquez, J.D., & Maes, P. (1997). Cathexis: a computation model of emotions. *Proc. 1st Int. Conf. Autonomous Agents*, pp. 518–519, Marina del Rey, CA.

---

**Grażyna Ślusarczyk** received her PhD degree from the Institute of Fundamental Technological Research PAS in 1999. She is a Lecturer in the Department of Design and Computer Graphics, Faculty of Physics, Astronomy, and Applied Computer Science, Jagiellonian University. Her research interests include graph transformations, computer-aided graphic design, and multiagent systems in design.

https://doi.org/10.1017/S0890060408000097 Published online by Cambridge University Press