Introducing a Two-Level Grammar Concept for Design[1]

(extended abstract)

Klaus P. Jantke

Hochschule für Technik, Wirtschaft und Kultur Leipzig
FB Informatik, Mathematik und Naturwissenschaften
Postfach 30066
04251 Leipzig

jantke@informatik.th-leipzig.de
www.informatik.th-leipzig.de

1 Introduction

The present papers focusses on the introduction of a novel concept of formal grammars. According to the case-based target application sketched below, the ultimate goal is to develop, investigate, and apply the new concept introduced in the area of formal graph languages. Nevertheless, the key ideas will be developed for string languages first. This should enable the reader to easily understand the essentials of the new approach. The paper on hand is intended to be a launching pad for in-depth investigations focussed on learning.

1.1 Case-Based Reasoning

Case-based reasoning is deemed an important technology to alleviate the bottleneck of knowledge acquisition in Artificial Intelligence. Publications are mushrooming. Hence, it is almost impossible to give a reasonable overview. [RS89] and [Kol93] are widely accepted textbooks. There are several introductory papers like [Kol92], e.g. In case-based reasoning, knowledge is represented in the form of particular cases or episodes with an appropriate similarity measure relating them to each other rather than any form of generalized or abstracted rules. One of the underlying key motivations is to avoid unjustified generalization and abstraction during knowledge acquisition. Instead, emphasis is put on adequately representing cases in a so-called case base. This is taken as a basis for further automated reasoning. When new problems are to be solved, a base-based reasoning system may search its case base for similar problems considered in the past. If more or less similar cases are found, the system may try to adapt them to the needs of the recent problem on hand. Thus, similarity based retrieval and adaptation are essential steps of case-based reasoning.

[1]This work has been partially supported by the German Federal Ministry for Education, Science, Research and Technology (BMBF) within the BMBF Joint Project FABEL under grant 413–4001–01 IW 101. The learning-theoretic aspects have been investigated within the project IND–CBL supported by the Deutsche Forschungsgemeinschaft (DFG) under grant Ja 566/2-1.

Cognitive psychology is providing sufficient evidence of the importance of case-based reasoning in automating particular processes of human reasoning adequately. George Lakoff's book [Lak87] is a very nice source putting emphasis on categorization. The early work by Eleanor Rosch (cf. [Ros75], e.g.) contains much details motivating elaborated approaches towards automated case-based reasoning.

Recently, there is an enormous amount of approaches and applications. The majority of publications adopts an attribute-value-pair descrition of cases. In a sense, there is assumed some n-dimensional space for case representation. Every particular case is characterized by some possibly incomplete vector of attribute values. This is appropriate for a huge amount of application domains. But in many interesting domains posing exciting problems to Artificial Intelligence, representing cases in an attribute-value-pair style means already substantial abstraction and generalization (cf. [EOSW94] and [Sch95], e.g.). This essentially misses the original motivation of case-based reasoning approaches. Episodic knowledge should be kept in a form as close as possible to its appearence. Knowledge elicitation is widely underestimated, and investigations like [Gor92], [PAG92], and [RH92] are frequently disregarded in computation-oriented Artificial Intelligence.

The inappropriateness of the attribute-value-pair approach bears abundant evidence of the need of structural case representations together with structural concepts of similarity. The author's paper [Jan94] is intended to be a launching pad for investigations of this type. In response, Katy Börner (cf. [Bör94]) is striving hard for developing and implementing structural similarity concepts in architectural design, e.g.

There is a common motivation for approaches like in [Jan94] and [Bör94]) drawn from the authors' work within the BMFT joint project Fabel (cf. [FC93]).

Another aim of the present paper is to develop certain fundamentals for applying inductive learning techniques in case-based reasoning. A formal framework for case-based learning has recently been developed by [Jan92], [JL93], and [SJL94] in an inductive inference manner.

1.2 Case-Based Learning

The crux of case-based reasoning is that the intended automated reasoning process provided by a case-based reasoning system should be mainly based on prior experience found somehow similar to a certain problem on hand. This similarity has to be determined in an early reasoning phase, whereas its success can only be verified finally. The unavoidable incompleteness of information based on earlier experience makes case-based reasoning a process of inductive hypothesizing.

Therefore, the intelligent behaviour of case-based reasoning systems can be substantially improved by inductive learning techniques. Those techniques are invoked within several application scenarios of case-based reasoning. An in-depth investigation of inductive learning in case-based reasoning needs a number of prerequisites like well-defined case concepts and similarity approaches.

From the overwhelming scenarios of embedding inductive learning into case-based reasoning, a prototypical example should do for illustration. Assume an interactive design support system possessing its case base together with some similarity measure. Whenever a user is designing a new case, (s)he may ask the system for formerly solved problems similar to a specific problem presented. Usually, the system will respond to the request with one or more cases of maximal similarity. Let us assume that there is always some case among the most similar cases presented which fits the users needs, but which is usually not the most similar one. The user may whish to correct the underlying similarity concept in such a way that his preferred cases would be presented as proposals of highest similarity. This means to change this similarity concept by inductive learning. The input to the assumed learning device is quite clear. First of all, there is a current similarity concept to be modified incrementally. Second, there is a permanently changing case base collecting the system's experience. Third, there is the user's input consisting of pairs containing a recent problem together with a desired output case to be of highest similarity. A sequence of user inputs should result in a sequence of concept changes. In the best case, this may converge to an a priori unknown target similarity concept.

The scenario sketched is a typical instance of the general inductive inference scenario (cf. [AS83], [KW80], [Jan89]) of learning in the limit from usually incomplete information. In the present paper, we are more particularly faced to problems of language learning (cf. [Vid94]). The approach is even more particular, as we are interested in two-level grammar concepts for generating graphs (see below). There are several further motivations of investigating two-level grammars, especially in computational linguistics (cf. [Kos83], e.g.). Inductive learning of two-level grammars is restricted two word languages, so far, like in [Ita94].

The present paper does not deal with the development of inductive learning for advanced case-based reasoning. Its intention is to introduce an appropriate graph-theoretic concept flexible and expressive enough to reflect essential cases of the FABEL application domain. The precision of the introduced concepts and their relation to other formal language concepts should provide a firm background for the intended development and application of inductive learning methods.

2 Structural Case Representation in Design

The present papers addresses case-based reasoning applications where the main objects of reasoning are highly structured. In those domains, exploiting the internal structures of objects is usually crucial.

Many computer science applications are substantially suffering from what the author calls the *paradoxon of levelling.* Although objects in the real world are usually very complex, have a certain hierarchical structure or internal topology, knowledge representation on computers is initially levelling this structure down, in most cases. In later phases of knowledge processing, enormous efforts are necessary to reconstruct the structural information levelled before. This seems to be somehow paradox.

Our underlying FABEL project is an excellent example in this regard. Typical target objects considered within the FABEL application are fresh air supply nets of buildings, for instance. In reality, these nets are highly structured, consist of certain levels both due to stages of architectural design and due to types of constructive parts, have an obvious topology, and are reasonably understood as graphs. However, as an effect of using a particular CAD system during design, these objects are levelled down to lists of tupels representing a bunch of attribute/value pairs. This leads to the somehow crazy necessity to rebuild structured objects from flat representations. In [Bör94] there is sketched the problem of transforming tupel representations back into structured objects for an appropriate case-based knowledge processing.

The present paper focusses on structural concepts for appropriate case-based reasoning, particularly in architectural design. It is an implementation decision whether or not to make a detour via flat case representations when invoking the concepts developed. This may essentially depend on the available hard- and software tools. However, those considerations are beyond the present investigation.

2.1 Case Structures in FABEL

In the FABEL approach (cf. [FK93] and [FC93]), we are dealt with certain architectural design problems. Typical applications are designing complete air supply systems of buildings. The net structures to be created interactively may also represent electricity or water supply, for instance. Characteristically, those nets are huge structures of a high regularity which may be reasonably understood as graphs.

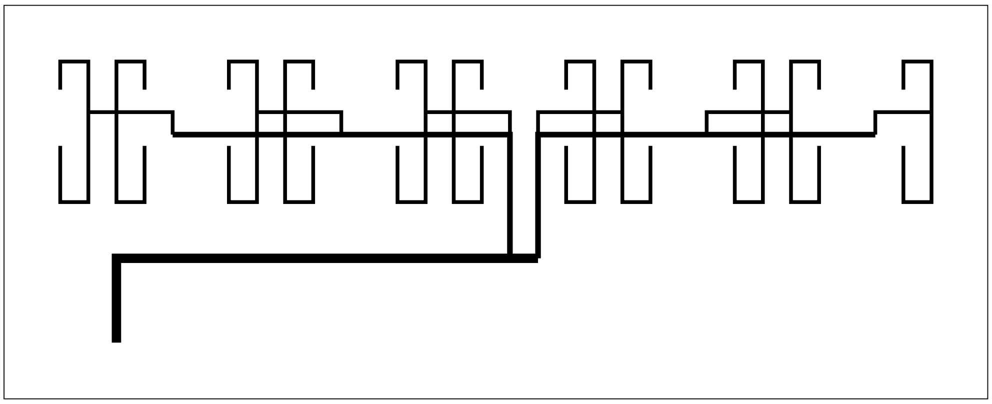

Figure 1: Part of a Typical Fresh-Air Supply Structure

2.2 Graph Objects in Architectural Design

The recent scientific literature yields an extremely inhomogeneous impression of the area. There are a score or more approaches like the one in [Bro94] which sound quite promising, but do not provide any precision. An extreme example is [Lee94] which announces "the formal basis ...", but does not contain any formalism.

In a minority of papers, one may find well-explained concepts of a certain precision. A valuable introduction leading to grammars for defining topological changes may be found in [HSF+92]. There, the authors have developed a clearly motivated two-level concepts of grammars for architectural design. The languages in use are essentially regular. It is one of the key intentions of the present paper to generalize this approach considerably.

3 Two-Level Design Grammars

The present chapter is devoted to the introduction of two-level graph grammar concepts. On the upper level, classical concepts of the CHOMSKY hierarchy are invoked. On the lower level, the ideas of string patterns is adopted. For the classical formal language concepts, there is a large amount of topical publications. [HU79] is an appropriate reference. The basic concepts of pattern languages have been introduced by DANA ANGLUIN in [Ang80b].

The following figure illustrates the skeleton of some target graph to be described by conventional grammar rules.

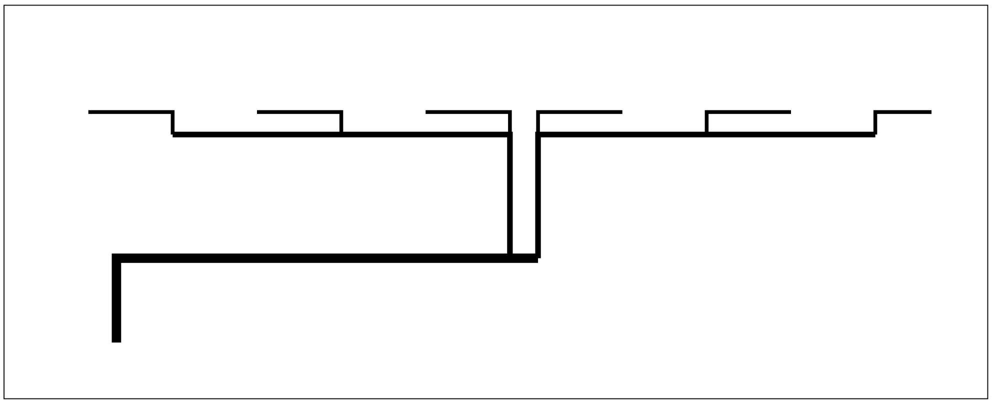

Figure 2: The CHOMSKY Part of a Target Graph

In dependence on the underlying application, those rules may be regular or context free. The resulting graph usually possesses some clearly marked nodes which are to be completed by attaching certain subgraphs taken from a class of more or less well-defined prototypes. This process of completion is somehow similar to the process of variable substitution for pattern languages (cf. [Ang80b]). The next figure is intended to display the "substitution" of "variables" implicitly contained in the figure above.

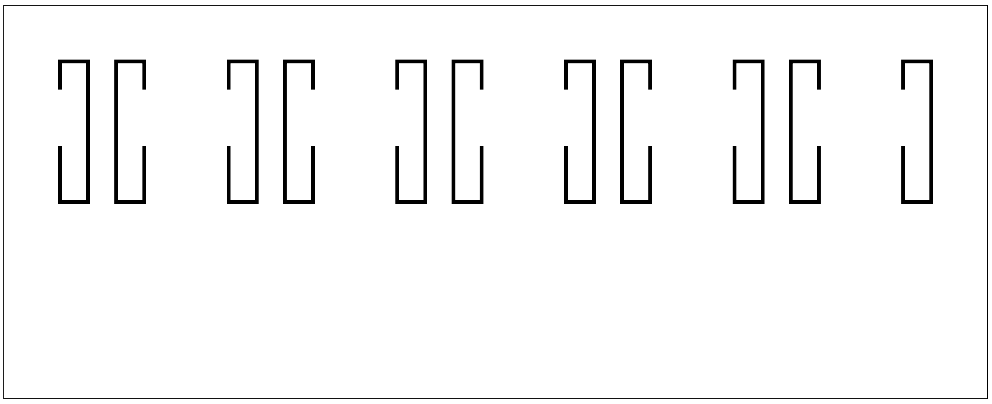

Figure 3: The Pattern Part of a Target Graph

The ultimate goal of the overall approach behind the present paper is to learn appropriate grammatical concepts from actual design processes. This needs clear formalisms to be outlined here.

First, our basic ideas are developed for string languages, for readability. It is assumed that most readers are familiar with the fundamentals of classical formal language theory. Thus, the introduction of our key concepts may look like an easy tour beyond the boundaries of classical theory. Second, they will be straightforwardly generalized to cover graph concepts

3.1 String Languages

The presentation follows classical concepts as tight as possible. When introducing graph grammars below, it seems reasonable to restrict our attention to only context-free substitutions. Therefore, we will apply the same restriction to the string language case discussed first.

Definition 1

A two-level context-free design grammar $\mathcal{DG} = [N, T_1, V, T_2, R, s]$ is given by the following parameters:

- N is a non-empty set (the so-called nonterminals).
- T_1 is a (possibly empty) set (the so-called first-level terminals).
- V is a non-empty set (the so-called pattern variables).
- T_2 is a non-empty set (the so-called second-level terminals).
- For R it holds $R \subseteq N \times (N \cup V \cup T_1)^+$ (the rule system of the grammar).
- For s it holds $s \in N$ (the start symbol of the grammar).

A design grammar $\mathcal{DG} = [N, T_1, V, T_2, R, s]$ is said to be finite, if all sets N, T_1, V, T_2, and R are finite. $\square$

Usually, grammars are used to define languages. We will introduce a formal semantics specifing for every grammar $\mathcal{DG}$ its formal language $L(\mathcal{DG})$. The standard concepts of the CHOMSKY hierarchy are assumed. Additionally, we invoke the basic concepts of pattern languages (cf. [Ang80b]).

In particular, the following notions and notations are in use:

- For every CHOMSKY grammar $\mathcal{G}$ on any alphabet A, $L(\mathcal{G})$ denotes the formal language generated by $\mathcal{G}$ over A.
- For every pattern p on any alphabet A and for any set of variables X disjoint with the alphabet A, $L(p)$ denotes the pattern language generated by p via non-erasing substitutions over A.

Definition 2

The formal semantics of any context-free design grammar $\mathcal{DG} = [N, T_1, V, T_2, R, s]$ is some formal language $L(\mathcal{DG})$ on the alphabet $T_1 \cup T_2$ defined by

$$L(\mathcal{DG}) \;=\; \bigcup_{p \in L(\mathcal{G})} L(p)$$

where

- $\mathcal{G} = [N, V \cup T_1, R, s]$ is a CHOMSKY grammar and
- every string $p \in L(\mathcal{G})$ is a pattern with variables in V varying over T_2 .

For any given alphabet A, the notation $\mathcal{L}_{DG}$ means the class of all formal languages defined by any context-free design grammar over $A = T_1 \cup T_2$. □

One could easily introduce a couple of requirements of naturalness. For instance, if $V \neq \emptyset$ and $L(\mathcal{G}) \setminus T_1^* \neq \emptyset$, one should reasonably assume $T_2 \neq \emptyset$. Vice versa, if V is empty, one does not need any letter in T_2. The investigation of dependencies of this type is left to the reader as an introductory excercise.

For extremely simple special cases, one gets well-known concepts. For instance, $V = \emptyset$ implies that $L(\mathcal{DG})$ is a usual CHOMSKY language. If $L(\mathcal{G})$ is a singleton set, $L(\mathcal{DG})$ is a standard pattern language. Already in the simple case where $L(\mathcal{G})$ is finite, we are faced to a type of formal languages not investigated so far.

There is a couple of reasonable generalizations and refinements of the basic approach introduced above. For example, one may generalize the approach to multi-layer grammars. Furthermore, one may refine the approach by assigning particular alphabets to different variables in V, or by guarding substitutions. In the author's opinion, one should stick with the basic approach for some time. First, this approach seems to meet the needs of the driving application described above quite well. There is no obvious need to develop more than two levels. Second, one should try to solve some of the open problems in the simplest setting before attacking more general cases. Hence, the approach will not be generalized throughout this paper.

A few investigation may first illustrate the differences of our new concepts from known hierarchies.

Even if $L(\mathcal{G})$ is regular, $L(\mathcal{DG})$ may be not context-free.

There is no need for a proof as it is folklore that pattern languages are usually not context-free.

We present a few further trivialities to warm up:

In case $|V| = 1$, $L(\mathcal{DG})$ is regular if and only if $L(\mathcal{G})$ is regular.

If T_1 and T_2 are disjoint, it holds:

1. Membership for $L(\mathcal{DG})$ is uniformly decidable.
2. Emptyness for $L(\mathcal{DG})$ is uniformly decidable.
3. Finiteness for $L(\mathcal{DG})$ is uniformly decidable.

There is no need for an explicit proof. The results above are immediately inherited from classical results in formal language theory (cf. [HU79]).

Further properties seem to require particular assumptions. For the intended application domain, languages which satisfy $T_1 \setminus T_2 \neq \emptyset$ are of a particular importance. It may also be of a special interest to consider languages with a certain rate of letters from $T_1 \setminus T_2$ in terminal words.

3.2 Graph Grammars

The area of graph grammar is quite rich. First, one needs to specify whether the generation process should be based on node replacements or edge replacement (or even hyperedge replacements). The FABEL application does no govern this decision in any way. Throughout this paper, we deal with node substitutions, exclusively.

A key question is how to embed graphs when inserted for a node of a given host graph. The embedding rule essentially determines the peculiarities of the whole approach. In the case of undirected graphs, there is an enormous number of concurring ideas.

In the present paper, we assume directed acyclic graphs. This approach has two basic roots. First, it is a direct generalization of approaches for two-level word grammars as sketched in [HSF$^+$92]. Second, it is based on graph grammar concepts mainly proposed by OKSANA ARNOLD in [AJ94b], [AJ94c] and [JA95]. Although these grammatical concepts are originally designed to meet the necessities of therapy plan generation for complex dynamic processes, they fit our present needs quite well.

The non-terminal symbols of the above definitions collected in N correspond to compound vertices in C_i below. Analoguously to the above set of variables V, there will be sets of variables X_i in every graph.

For the following definitions, there are two basic alternatives for representing what corresponds to the start symbol of CHOMSKY grammars: One may either choose a node or a graph. The choice of a graph would directly correspond to the underlying approach in [AJ94b], [AJ94a] and [JA95]. However, some reader may find it more intuitive to have a start object which is properly atomic. As this decision is not essential, we have chosen the syntactically simpler second alternative.

Definition 3

A two-level context-free graph design grammar $\mathcal{GDG} = [\mathcal{F}, \mathcal{H}, c_0]$ is given as follows:

1. $\mathcal{F} = \{\mathcal{G}_1, \ldots, \mathcal{G}_m\}$ is a set of so-called pin graphs.
2. $\mathcal{H} = \{\mathcal{G}_{m+1}, \ldots, \mathcal{G}_n\}$ is another set of pin graphs.
3. Every pin graph in $\mathcal{F}$ has the form $\mathcal{G}_i = [V_i, E_i, P_i^{in}, P_i^{out}, C_i, sub_i, X_i]$.
4. Every pin graph in $\mathcal{H}$ has the form $\mathcal{G}_i = [V_i, E_i, P_i^{in}, P_i^{out}]$.
5. Every pin graph $\mathcal{G}_i$ in $\mathcal{F} \cup \mathcal{H}$ with $i \in \{1, \ldots, n\}$ meets the following conditions:

   5.1 $\mathcal{G}_i' = [V_i, E_i]$ is a finite, directed, acyclic graph with the set V_i of vertices and the set E_i of edges.

   5.2 $P_i^{in} \cup P_i^{out}$ contains the so-called input pins and output pins of $\mathcal{G}_i$ defined as follows:

   5.3 $P_i^{in} = \{\, v \mid v \in V_i \wedge \neg\exists u \in V_i\,(\,(u, v) \in E_i\,)\,\}$

   5.4 $P_i^{out} = \{\, v \mid v \in V_i \wedge \neg\exists u \in V_i\,(\,(v, u) \in E_i\,)\,\}$

6. Every pin graph $\mathcal{G}_i$ in $\mathcal{F}$ with $i \in \{1, \ldots, m\}$ meets the following conditions:

   6.1 The vertices in $C_i \subseteq V_i$ are called compound.

   6.2 $sub_i : C_i \to 2^{\{1,\ldots,m\}} \setminus \emptyset$ defined the admissible substitutions of graphs $\mathcal{G}_j$ for compound vertices in C_i.

   6.3 The vertices in $X_i \subseteq V_i$ are called variables. For all $i \in \{1, \ldots, m\}$ it holds $C_i \cap X_i = \emptyset$.

7. $c_0 \in \bigcup_{i=1\ldots m} C_i$ is some start vertex. □

Obviously, the specification of P_i^{in} and P_i^{out} is redundant. It is given exclusively for clarity.

The pin graphs in $\mathcal{H}$ can be understood as special cases of graphs in $\mathcal{F}$ with $C_i = \emptyset$ and $X_i = \emptyset$. The property $C_i = \emptyset$ implies that the mapping sub_i (understood as a relation) must be empty. Intuitively, the graphs in $\mathcal{F}$ are to be substituted one into another according to $\{sub_i\}_{i=1\ldots m}$, whereas the graphs in $\mathcal{H}$ are used for replacing the varaibles in $\{X_i\}_{i=1\ldots m}$.

For illustration, recall figure 3 above. It depicts 2 pin graphs (one occurs in 5 copies) of $\mathcal{H}$. The graph of figure 2 belonging to $\mathcal{F}$ contains variables in 6 places. A possible substitution yields the graph originally displayed in figure 1.

Decisions about the substitution mechanism, i.e. embedding, define the underlying formal semantics. As introduced before, there isa two-level mechanism. For understandability, it is split into the following concepts:

- a rewrite relation,
- a language of pattern graphs,
- graph substitution, and
- a language of ground graphs.

There are some useful notations. For example, there is some desire to avoid name conflicts during substitution. Therefore, the following concentions will be adopted:

- If c and d are names of vertices (perhaps, in different graphs), then $c.d$ is an acceptable name of a vertex as well.
- Similarly, this notation is applied to pairs of vertices: $c.(d_1, d_2) = (c.d_1, c.d_2)$
- If c is a vertex and C a set of vertices, then holds $c.C = \{c.d \mid d \in C\}$.

Names of vertices of the form "c.d" are said to be structured. Originally, there are no structured names.

Definition 4

Assume any two-level context-free graph design grammar $\mathcal{GDG} = [\mathcal{F}, \mathcal{H}, c_0]$. For any graph $\mathcal{G}_i \in \mathcal{F}$, any vertex $c \in C_i$ and any index $j \in sub_i(c)$, the *Substitution of* $\mathcal{G}_j$ *in* $\mathcal{G}_i$ *at position* $c \in C_i$ yields a graph $\mathcal{G}_i[c \hookleftarrow \mathcal{G}_j]$ where $\mathcal{G}_i[c \hookleftarrow \mathcal{G}_j] = [V, E, P^{in}, P^{out}, C, sub, X]$ is defined by:

1. $V \;=\; (\,V_i \setminus \{c\}\,) \cup c.V_j$
2. $E \;=\; (\,(E_i \cup c.E_j) \setminus (V_i \times \{c\} \cup \{c\} \times V_i)\,)$
   $\cup\,((\{v \mid (v,c) \in E_i\} \times c.P_j^{in}) \cup (c.P_j^{out} \times \{v \mid (c,v) \in E_i\}))$

   3.1 $P^{in} \;=\; P_i^{in}$ if $c \notin P_i^{in}$
   3.2 $P^{in} \;=\; (\,P_i \setminus \{c\}\,) \;\cup\; c.P_j^{in}$ if $c \in P_i^{in}$

   4.1 $P^{out} \;=\; P_i^{out}$ if $c \notin P_i^{out}$
   4.2 $P^{out} \;=\; (\,P_i \setminus \{c\}\,) \;\cup\; c.P_j^{out}$ if $c \in P_i^{out}$

5. $C \;=\; (\,C_i \setminus \{c\}\,) \cup c.C_j$
6. $sub \;=\; sub_{i/C_i \setminus \{c\}} \;\cup\; c.sub_j$
7. $X \;=\; X_i \;\cup\; c.X_j$ $\square$

The resulting pin graph $\mathcal{G} = \mathcal{G}_i[c \hookleftarrow \mathcal{G}_j]$ is of the same structure as the graphs in $\mathcal{F}$. Therefore, the definition above applies to those generated host graphs $\mathcal{G}$, for arbitrary of their vertices c, and for all graphs $\mathcal{G}_j$ in $\mathcal{F}$ as well. This allows to write $\mathcal{G}[c \hookleftarrow \mathcal{G}_j]$ even if $\mathcal{G}$ is the result of any preceeding substitution process.

Definition 5

Among pin graphs of the structure $[V, E, P^{in}, P^{out}, C, sub]$, one defines a rewrite relation $\Rightarrow_{\mathcal{F}}$ by

1. $\mathcal{G} \;\Rightarrow_{\mathcal{F}}\; \mathcal{G}'$ if and only is $\exists c\, \exists j\; (\, \mathcal{G}' = \mathcal{G}[c \hookleftarrow \mathcal{G}_j]\,)$

One may interpret the vertex c_0 as a graph consisting of only the one (compound) vertex c_0 for justifying the notation $c_0 \;\Rightarrow_{\mathcal{F}}\; \mathcal{G}$.

2. $c_0 \;\Rightarrow_{\mathcal{F}}\; \mathcal{G}$ gilt if and only if
   $\exists i \in \{1, \ldots, m\}\; \exists j \in sub_i(c_0)\; (\, c_0 \in C_i \;\wedge\; \mathcal{G} = \mathcal{G}_j\,)$

$\Rightarrow_{\mathcal{F}}^{+}$ denotes the transitive closure of $\Rightarrow_{\mathcal{F}}$ and the irreducibility of any graph $\mathcal{G}$ with respect to $\Rightarrow_{\mathcal{F}}$ is indicated by $\mathcal{G}{\downarrow_{\mathcal{F}}}$. □

This is sufficient for defining the language of patterns generated by any given two-level context-free graph design grammar. The graph depicted in figure 2 is a typical element of such a language, for instance.

Definition 6

Assume any two-level context-free graph design grammar $\mathcal{GDG} = [\mathcal{F}, \mathcal{H}, c_0]$. This defines a language $\mathcal{P}(\mathcal{GDG})$ as follows:

- $\mathcal{P}(\mathcal{GDG}) \;=\; \{\, \mathcal{G} \mid c_0 \;\Rightarrow_{\mathcal{F}}^{+}\; \mathcal{G} \;\wedge\; \mathcal{G}{\downarrow_{\mathcal{F}}}\, \}$ □

The following technical concepts seem quite appropriate.

Definition 7

Assume any grammar $\mathcal{GDG} = [\mathcal{F}, \mathcal{H}, c_0]$. The set of its variables is

1. $X_{\mathcal{GDG}} \;=\; \bigcup_{i=1\ldots m} X_i$

The final segment relation $\sqsupseteq$ of vertices is introduced as follows.

2. $\forall d, e\; (\, e \sqsupset_1 d \iff \exists c\; (\, e = c.d\,)\,)$

3. $\sqsupseteq_1$ denotes the reflexive closure of $\sqsupset_1$.

4. $\forall d, e\; (\, e \sqsupseteq d \iff e = d \;\vee\; \exists c\; (\, e \sqsupset_1 c \;\wedge\; c \sqsupseteq d\,)\,)$

ν computes the last letter of any vertex name via $\sqsupseteq$.

5. $\forall c, d\; (\, d = \nu(c) \iff c \sqsupseteq d \wedge\; \nexists e\, (\, d \sqsupset e\,)\,)$. □

Structured vertices c which have been constructed during stepwise substitution are finally identified with its last letter $\nu(c)$. First, the substitution concept is introduced.

Definition 8

Assume any two-level context-free graph design grammar $\mathcal{GDG} = [\mathcal{F}, \mathcal{H}, c_0]$ with the language of patterns $\mathcal{P}(\mathcal{GDG})$. A substitution σ is some mapping

1. $\sigma \ : \ X_{\mathcal{GDG}} \ \longrightarrow \ \{m+1, \ldots, n\}$

Every substitution σ specifies a non-deterministic rewrite operator $\Rightarrow_\sigma$ on $\mathcal{P}(\mathcal{GDG})$.

2. $\mathcal{G} \ \Rightarrow_\sigma \ \mathcal{G}'$, if $\exists c \ (\ \mathcal{G}' = \mathcal{G}[c \hookleftarrow \mathcal{G}_{\sigma(\nu(c))}]\ )$
3. $\Rightarrow_\sigma^+$ denotes the transitive closure and $\Rightarrow_\sigma^*$ denotes the reflexive and transitive closure of $\Rightarrow_\sigma$.
4. $\mathcal{G}\!\downarrow_\sigma$ means that $\mathcal{G}$ is irreducible w.r.t. $\Rightarrow_\sigma$.

The target concept to be developed in the present paper is the formal graph language generated by any given two-level context-free graph design grammar.

Definition 9

Assume any two-level context-free graph design grammar $\mathcal{GDG} = [\mathcal{F}, \mathcal{H}, c_0]$ with the language of patterns $\mathcal{P}(\mathcal{GDG})$. $\mathcal{GDG} = [\mathcal{F}, \mathcal{H}, c_0]$ defines a formal language $\mathcal{L}(\mathcal{GDG})$ by

$$\mathcal{L}(\mathcal{GDG}) \ = \ \bigcup_{\substack{\mathcal{G} \in \mathcal{P}(\mathcal{GDG}) \\ \sigma \in X_{\mathcal{GDG}}}} \sigma(\mathcal{G})$$

The two levels of the derivation process are easy to see. For generating any graph of $\mathcal{L}(\mathcal{GDG})$, one first generates a pattern graph describing the skeleton of the target graph which belongs to $\mathcal{P}(\mathcal{GDG})$. Second, the final graph is constructed by an appropriate variable substitution.

This process suggests several similarity concepts. The simplest one is to consider two (or even more) graphs as somehow similar, exactly if they have the same skeleton. Another interesting approach is to choose the largest common subgraph of two skeletons as the similarity of two given graphs. A quite opposite approach compares the similarities of final substitutions.

Furthermore, the novel formalism developed may be taken as a firm basis for learning-theoretic investigations. One may try to learn one level of the grammar under the assumption of the other one. And one may try to learn similarity concepts for given two-level context-free graph design grammars.

Despite the attractiveness of these questions, the reader should recall that the aim of the present extended abstract was exclusively to set the stage both for the case-based reasoning investigations and for the learning-theoretic investigations by providing a well-motivated innovative concept of graph grammars and languages.

References

[AJ94a] Oksana Arnold and Klaus P. Jantke. Therapy plan generation as program synthesis. In Setsuo Arikawa and K.P. Jantke, editors, *Algorithmic Learning Theory, AII'94 & ALT'94*, volume 872 of *LNAI*, pages 40–55. Springer–Verlag, 1994.

[AJ94b] Oksana Arnold and Klaus P. Jantke. Therapy plans as hierarchically structured graphs. WISCON Report 02/94, HTWK Leipzig (FH), Fachbereich IMN, April 1994.

[AJ94c] Oksana Arnold and Klaus P. Jantke. Therapy plans as hierarchically structured graphs. In *Fifth International Workshop on Graph Grammars and their Application to Computer Science, Williamsburg, Virginia, USA*, November 1994.

[Ang80a] Dana Angluin. Finding patterns common to a set of strings. *Journal of Computer and System Sciences*, 21:46–62, 1980.

[Ang80b] Dana Angluin. Inductive inference of formal languages from positive data. *Information and Control*, 45(2):117–135, 1980.

[AS83] Dana Angluin and Carl H. Smith. A survey of inductive inference: Theory and methods. *Computing Surveys*, 15:237–269, 1983.

[Bör94] Katy Börner. Structural similarity as guidance in case-based reasoning. In Stefan Wess, Klaus-Dieter Althoff, and Michael M. Richter, editors, *1st European Workshop on Case–Based Reasoning (EWCBR'93), November 1-5, 1993, Selected Papers*, volume 837 of *Lecture Notes in Artificial Intelligence*, pages 197–208. Springer-Verlag, 1994.

[Bro94] Ken N. Brown. Shape grammars and semantics: Formalising the link between geometry, features and the design process. In *Third International Conference on Artificial Intelligence in Design, Lausanne, Switzerland, August 15–18, 1994, Workshop on Reasoning with Shapes in Design, Workshop Notes*, pages 1–4, 1994.

[EOSW94] Dieter Ehrenberg, Thomas J. Olbrich, Ralf Schulz, and Herbert Wiesner. Distributed reasoning support systems – a case-based approach to solving ill-structured decision problems. Arbeitsbericht 5, Universität Leipzig, Wirtschaftswissenschaftliche Fakultät, Lehrstuhl Wirtschaftsinformatik, May 1994.

[FC93] FABEL-Consortium. Survey of FABEL. FABEL–Report 2, Gesellschaft für Mathematik und Datenverarbeitung mbH, Forschungsbereich Künstliche Intelligenz, February 1993.

[FK93] Fabel-Konsortium. FABEL im Überblick. FABEL–Report 1, Gesellschaft für Mathematik und Datenverarbeitung mbH, Forschungsbereich Künstliche Intelligenz, February 1993.

[Gor92] Dallie E. Gordon. Implication of cognitive theory for knowledge acquisition. In Robert R. Hoffman, editor, *The Psychology of Expertise. Cognitive Research of Empirical AI*, pages 99–120. Springer-Verlag, 1992.

[HSF+92] K. Hua, I. Smith, B. Faltings, S. Shih, and G. Schmitt. Adaptation of spatial design cases. In John S. Gero, editor, *Artificial Intelligence in Design'92*, pages 559–575. Kluwer Academic Publ., 1992.

[HU79] John E. Hopcroft and Jeffrey. D. Ullman. *Introduction to Automata Theory, Languages, and Computation.* Addison-Wesley, 1979.

[Ita94] Alon Itai. Learning morphology – practice makes good. In Rafael C. Carrasco and Jose Oncina, editors, *Grammatical Inference and Applications, Second International Colloquium, ICGI-94, Alicante, Spain, September 21–23, 1994*, volume 862 of *Lecture Notes in Artificial Intelligence*, pages 5–15. Springer–Verlag, 1994.

[JA95] Klaus P. Jantke and Oksana Arnold. Graphgrammatik-Konzepte in der Therapieplanung für komplexe dynamische Prozesse. In Markus Holzer, editor, *4. GI Theorietag "Automaten und Formale Sprachen", 1994, Proc., Herrsching*, pages 42–47. Universität Tübingen, Wilhelm-Schickard-Institut, 1995.

[Jan89] Klaus P. Jantke. Algorithmic learning from incomplete information: Principles and problems. In J. Dassow and J. Kelemen, editors, *Machines, Languages, and Complexity*, Lecture Notes in Computer Science, pages 188–207. Springer-Verlag, 1989.

[Jan92] Klaus P. Jantke. Case based learning in inductive inference. In *Proc. 5th Annual ACM Workshop on Computational Learning Theory, (COLT'92), July 27-29, 1992, Pittsburgh, PA, USA*, pages 218–223. ACM Press, 1992.

[Jan94] Klaus P. Jantke. Nonstandard concepts of similarity in case-based reasoning. In Hans-Hermann Bock, Wolfgang Lenski, and Michael M. Richter, editors, *Information Systems and Data Analysis: Prospects – Foundations – Applications, Proceedings of the 17th Annual Conference of the GfKl, Univ. of Kaiserslautern, 1993*, Studies in Classification, Data Analysis, and Knowledge Organization, pages 28–43. Springer-Verlag, 1994.

[JL93] Klaus P. Jantke and Steffen Lange. Case–based representation and learning of pattern languages. In K.P. Jantke, S. Kobayashi, E. Tomita, and T. Yokomori, editors, *4th Intern. Workshop on Algorithmic Learning Theory (ALT'93), Proc., November 1993, Tokyo, Japan*, volume 744 of *Lecture Notes in Artificial Intelligence*, pages 87–100. Springer–Verlag, 1993.

[Kol92] Janet L. Kolodner. An introduction to case-based reasoning. *Artificial Intelligence Review*, 6:3–34, 1992.

[Kol93] Janet L. Kolodner. *Case-Based Reasoning.* Morgan Kaufmann, 1993.

[Kos83] K. Koskienniemi. Two-level morphology: A general computational model for word-form recognition and production. Technical report, Department of General Linguistics, University of Helsinki, Publication No. 11, 1983.

[KW80] Reinhard Klette and Rolf Wiehagen. Research in the theory of inductive inference by GDR mathematicians - a survey. *Information Sciences*, 22:149–169, 1980.

[Lak87] George Lakoff. *Women, Fire, and Dangerous Things.* The University of Chicago Press, 1987.

[Lee94] John R. Lee. The formal basis of "reasoning with shapes". In *Third International Conference on Artificial Intelligence in Design, Lausanne, Switzerland, August 15–18, 1994, Workshop on Reasoning with Shapes in Design, Workshop Notes*, pages 30–34, 1994.

[PAG92] David S. Prerau, Mark R. Adler, and Alan S. Gunderson. Eliciting and using experiential knowledge and general expertise. In Robert R. Hoffman, editor, *The Psychology of Expertise. Cognitive Research of Empirical AI*, pages 137–148. Springer-Verlag, 1992.

[RH92] Stephen B. Regoczei and Graeme Hirst. Knowledge and knowledge acquisition in the computational context. In Robert R. Hoffman, editor, *The Psychology of Expertise. Cognitive Research of Empirical AI*, pages vii–ix. Springer-Verlag, 1992.

[Ros75] Eleanor Rosch. Cognitive reference points. *Cognitive Psychology*, 7:531–547, 1975.

[RS89] Christopher K. Riesbeck and Roger C. Schank. *Inside Case-Based Reasoning.* Lawrence Erlbaum Assoc., 1989.

[Sch95] Ralf Schulz. Handling similarity in case-based systems for ill-structured problems. Arbeitsbericht 7, Universität Leipzig, Wirtschaftswissenschaftliche Fakultät, Lehrstuhl Wirtschaftsinformatik, January 1995.

[SJL94] Yasubumi Sakakibara, Klaus P. Jantke, and Steffen Lange. Learning languages by collecting cases and tuning parameters. In Setsuo Arikawa and K.P. Jantke, editors, *Algorithmic Learning Theory, AII'94 & ALT'94*, volume 872 of *LNAI*, pages 533–547. Springer-Verlag, 1994.

[Vid94] Enrique Vidal. Grammatical inference: An introductory survey. In Rafael C. Carrasco and Jose Oncina, editors, *Grammatical Inference and Applications, Second International Colloquium, ICGI-94, Alicante, Spain, September 21–23, 1994*, volume 862 of *Lecture Notes in Artificial Intelligence*, pages 1–4. Springer–Verlag, 1994.