Formal Classification of Visual Languages

Kim Marriott & Bernd Meyer[1]
{marriott | berndm}@cs.monash.edu.au
Department of Computer Science
Monash University
Clayton, Victoria 3168, Australia

1 Introduction

In terms of technical usability, the field of visual language specification has achieved a lot of progress since research into "syntactical pattern recognition" started. Several specification methods have been proposed and have proven useful in practical applications. This progress, however, came at a certain price. VL specification is now faced with a Babel of different formalisms that are very hard to compare. Most of them are grammar-like (such as Web and Array Grammars [15], Shape Grammars [7], Positional Grammars [4], Relational Grammars [6], Unification Grammars [18, 17], Attributed Multiset Grammars [8], Constraint Multiset Grammars [12], and several types of graph grammars [5]), but logic-based methods [10, 14, 9] and algebraical formalisms [16] are also used. Even when the discussion is restricted to grammar-like formalisms, there is such a diversity in the underlying assumptions that a comparison is made difficult.

A systematic, comprehensive hierarchy of VLs based on formal properties – an analogue to the Chomsky hierarchy for textual languages — is still missing. Such a taxonomy is of great importance, because it does not only provide common ground for the comparison of results obtained from different approaches, but it can also yield new insights into the general structure of VLs. This paper presents the core of such a hierarchy based on Constraint Multiset Grammars, and some dimensions for extensions are discussed. Several classes of VLs are defined and their formal properties are analyzed. It is shown how the classification can serve to relate different specification methods to each other.

2 Constraint Multiset Grammars

Constraint Multiset Grammars (CMGs) have been introduced in [11] and have been used in a number of complex tasks including real-time editing [2]. A formal treatment of the original context-sensitive form is given in [12], and we only review the basic notions briefly. The choice of CMGs is justified by the fact that they are powerful enough to emulate a broad spectrum of other specification formalisms, so that these can be mapped into a hierarchy based on CMGs (cf. Sec. 5). Of course, any other choice of an equally powerful approach would have the same justification. CMGs are, however, particularly interesting, because their semantics is not only closely linked to grammars but also to constraint logic programs. CMGs are rewrite systems working on multisets of typed attributed symbols. Spatial relations are handled by arithmetical constraints on attributes, i.e., there is no explicit representation of spatial relations. The original definition allows arbitrary computations on attributes. This, however, is too powerful for the purpose of building a hierarchy, as even CMGs over the domain of integers with functions $\{+, -, = 0\}$ are computationally adequate.[2] Therefore the usage of attributes has to be restricted properly. A natural way is obviously not to allow computations at all but to restrict attribute manipulation to

[1]supported by DFG Grant ME11/94

[2]Other uses of complex attribute structures will be illustrated in Sec. 5.

copying without using any function symbols. A further discussion of attribute computations will be given in Sec. 6.

Definition 1 *A* copy-restricted constraint multiset grammar (CCMG) *over a computational domain D is a quadruple (T_T, T_{NT}, S, P) consisting of a set of terminal symbols T_T, a set of non-terminal symbols T_{NT}, $T_T \cap T_{NT} = \emptyset$, a start symbol $S \in T_{NT}$ and a set of productions P that have the form*

$$\begin{array}{rcl} T(\vec{x}) & ::= & T_1(\vec{x}_1), \ldots, T_n(\vec{x}_n) \\ & & \textit{where exists } T'_1(\vec{x}'_1), \ldots, T'_m(\vec{x}'_m) \\ & & \textit{where } C \textit{ and } F \end{array}$$

where $T \in T_{NT}$, $T_1, \ldots, T_n, T'_1, \ldots, T'_m \in T_T \cup T_{NT}$ with $n \geq 0$ are type symbols, C is a constraint over $\vec{x}_1, \ldots, \vec{x}_n, \vec{x}'_1, \ldots, \vec{x}'_m$ and F is a sequence of equalities $T.a_i = T^{(\prime)}_{k_i}.b_i$ such that a_i (b_i) is an attribute name of T ($T^{(\prime)}_{k_i}$).

A production is applied by rewriting the non-terminal T into the multiset $T_1, \ldots, T_n$ if the sentence contains symbols $T'_1, \ldots, T'_m$ (the context) such that the attributes of $T_1, \ldots, T_n, T'_1, \ldots, T'_m$ satisfy the constraint C. The language $\mathcal{L}(G)$ generated or recognized by a CMG G is the set of all multisets of attributed terminal symbols that can be derived from the start symbol by repeated application of productions in G.

3 Expressiveness

How can we use CMGs to describe different classes of VLs? A natural way to start is to investigate, whether the various grammar classes whose meta-structure correponds to the Chomsky hierarchy result in meaningful classes of languages. Thus we start by analyzing regular (type 3), context-free (type 2), and context-sensitive (type 1) CCMGs. In fact, all three classes characterize different classes of VLs and form a hierarchy of strict inclusions. The class of grammars defined in Sec. 2 are type 1 grammars. In the remainder we will use a slightly modified notation, where context symbols appear on the LHS of a production in order to facilitate consistent notation of different classes.

Definition 2 *A CCMG production is* type 1 *if it has the form: $uA \Leftarrow uv \mid\mid C \wedge F$ where $u \in (T_T \cup T_{NT})^*$, $v \in (T_T \cup T_{NT})^+$, $A \in T_{NT}$. Such a production is equivalent to $A ::= v$* where exists u where C and F.

Definition 3 *A type x CCMG is a CCMG in which every production is of type x.*

We can then derive "context-free" (type 2) and regular (type 3) grammars by imposing approporiate syntactic restrictions on permissible productions:

Definition 4 *A CCMG production is* type 2 *if it is a type 1 production $A \Leftarrow v \mid\mid C \wedge F$.*

Definition 5 *A CCMG production is* type 3 *if it is a type 2 production of one of the forms $A \Leftarrow b \mid\mid C \wedge F$ or $A \Leftarrow bB \mid\mid C \wedge F$ where $b \in T_T$ and $B \in T_{NT}$.*

Let $\mathcal{L}(G)$ denote the language specified by a grammar G or the class of languages specified by a class of grammars G, respectively. The three classes defined above form a strictly monotone hierarchy of expressiveness.

We can only illustrate the general ideas of proofs in this abstract. Most proofs are based on the fact that *copy-restricted* CMGs allow only a bounded amount of information to be shared by two nodes in a derivation tree for a given grammar G.

Theorem 1 $\mathcal{L}(3 - CCMG) \subsetneq \mathcal{L}(2 - CCMG)$.

The inclusion is strict, because $\mathcal{L}(3-CCMG) \not\ni T \in \mathcal{L}(2-CCMG)$ for the language T of trees. It is simple to give a *2-CCMG* for T. Obviously a *3-CCMG* G defines a total ordering of the terminal symbols in a sentence. It can be shown that given any total ordering of the nodes in a tree T and some constant c, we can always construct a tree such that there is a subsequence $S = s_1, \ldots, s_n$ in the ordered sequence of nodes such that at least $2c+1$ symbols in S have parents or sons in $T - S$. As every parent node must share a spatial constraint with its sons, at least $c+1$ attributes must pass through s_1 or s_n. Since G is copy-restricted, this is a contradiction.

Theorem 2 $\mathcal{L}(2-CCMG) \subsetneq \mathcal{L}(1-CCMG)$.

We can show that $\mathcal{L}(3-CCMG) \not\ni G \in \mathcal{L}(2-CCMG)$ for the language G of graphs represented as sets of edges and nodes. It is easy to give a *1-CCMG* for graphs.

Every *2-CCMG* can be rewritten such that all productions have one of the forms $x \Leftarrow yz \mid\mid C \wedge F$ or $x \Leftarrow a \mid\mid C \wedge F$ where $x, y, z \in V_N$ and $a \in V_T$. Consider the general structure of a type 2 derivation given in Figure 2.[3] Each edge in the graph has to be spatially constrained to two nodes. Let the graph to be parsed be complete and have $2k$ nodes. Every node is connected to $2k-1$ edges. Consider pairs of nodes (p_i, p_j) with $p_i \in L$, $p_j \in R$. At least k arguments must pass through the root of L in order to constrain all the edges touching p_i. Since k is not bounded, this contradicts the copy restriction.

Attributes give certain means to handle contexts within these limit. E.g., $a^i b^i \in \mathcal{L}(3-CCMG)$. As both of the above are very simple structural creteria that are contained in a lot of "naturally occuring" VLs (like road maps, graph-like organizational diagrams etc.), we cannot escape context-sensitive VLs, if we want to manage VLs that have not artificially been designed for computer-based work. But even *1-CCMG*s have severe limitations, e.g., they cannot specify the language of complete graphs. Thus more complex forms of context-handling are required. We investigate extended forms of CCMGS by (a) relaxing the monotonicity criterion for *1-CCMG*s, (b) introducing universally quantified contexts (i.e., negative contexts), and (c) arbitrary first order context formulas.

Definition 6 *A CCMG production is* type 0 *if it is of the form* $uA \Leftarrow v \mid\mid C \wedge F$ *, where* $u, v \in (T_T \cup T_{NT})^*$*,* $A \in T_{NT}$*, C is constraint on attributes in uv, and F is a sequence of equalities of the form $A.x = U.y$ where $U \in uv$ and x, y are attribute names of A and U, respectively.*

Definition 7 *A production is* type $x\forall$ *if it is of the form* $u \Leftarrow v \mid\mid C \wedge F$ *, such that* $u \Leftarrow v \mid\mid F$ *is type x and C is a universally quantified formula* $\forall x_1, ..., x_l : \varphi(x_{k_1}.a_1, \ldots, x_{k_n}.a_n, y_1.b_1, \ldots, y_m.b_m)$ *where* $x_i \in T_T \cup T_{NT}$*,* $k_i \in \{1, \ldots, l\}$*, $y_i \in v$, and a_i (b_i) are attribute names of x_i (y_i).*

Definition 8 *A production is* type 0FOL *if it is of the form* $u \Leftarrow v \mid\mid C \wedge F$ *, such that* $u \Leftarrow v \mid\mid F$ *is type 0 and C is a first order formula* $Q_1 x_1, ..., Q_l x_l : \varphi(x_{k_1}.a_1, \ldots, x_{k_n}.a_n, y_1.b_1, \ldots, y_m.b_m)$ *where* $x_i \in T_T \cup T_{NT}$*,* $k_i \in \{1, \ldots, l\}$*, $y_i \in v$, a_i (b_i) are attribute names of x_i (y_i), and $Q_i \in \{\exists, \forall\}$. If $\mid v \mid \geq \mid u \mid$ the production is of* type 1FOL*.*

The full hierarchy of all these classes is given in Figure 1.

Theorem 3 $\mathcal{L}(2-CCMG) \subsetneq \mathcal{L}(2\forall - CCMG)$.

A *2-CCMG* cannot specify constraints that are quantified over tuples. Consider the language that is the set of all sets of pairwise disjoint circles. It is easy to specify this language in a *2∀-CCMG* by adding the condition $(\forall c_1, c_2 : circle) disjoint(c_1, c_2)$ to the grammar. By the same logic as in Theorem 2 we can show that this language is not in $\mathcal{L}(2-CCMG)$.

Theorem 4 $\mathcal{L}(2\forall - CCMG) \subsetneq \mathcal{L}(1\forall - CCMG)$.

As we can specify graphs with *1-CCMG*s, it is obvious how complete graphs can be specified with *1∀-CCMG*s. They are not in $\mathcal{L}(2\forall - CCMG)$, since we still cannot check the connectedness of lines to nodes in a *2∀-CCMG*. A refined version of the argument for Theorem 2 continues to hold.

[3] disregard the dotted lines here, they represent information passing for context-symbols in *1-CCMG*s.

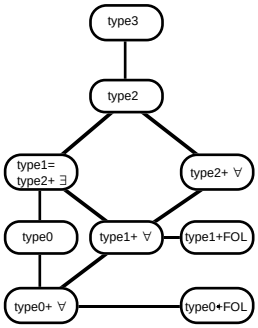

Figure 1: The Complete CCMG Hierarchy

Theorem 5 $\mathcal{L}(1-CCMG) \subsetneq \mathcal{L}(1\forall - CCMG)$.

We know that complete graphs are in $\mathcal{L}(1\forall - CCMG)$. They cannot be specified by *1-CCMG*s. In a copy-restricted *1-CCMG*, a grammar-dependent number of k attributes can be used at each node. Additionally the attributes of a fixed number e of context nodes can be used. Thus the total amount of information used at a node can not be larger than $O((1+e)k)$. Given a complete graph of size n and choosing L and R as above, we know that each line in L has to be checked against n points in R. As n is not bounded in the size of the grammar, this amount of information cannot be passed between L and R using a type 1 derivation.

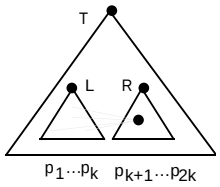

Figure 2: Type 1 Derivation Structure

Theorem 6 $\mathcal{L}(1-CCMG) \not\supset\not\subset \mathcal{L}(2\forall - CCMG)$.

General graphs are not in $\mathcal{L}(2\forall - CCMG)$, since they include complete graphs. By the same logic that was used for complete graphs, we can show that the language of pairwise disjoint circles is not in $\mathcal{L}(1-CCMG)$.

Conjecture 1 $\mathcal{L}(1\forall - CCMG) \not\supset\not\subset \mathcal{L}(0-CCMG)$.[4]

Let G be the language of complete graphs: $\mathcal{L}(0-CCMG) \not\ni G \in \mathcal{L}(1\forall - CCMG)$. Let R be the language that consists of all sets of axis-parallel lines which form a rectangle that is completely filled by other rectangles. The lines in the input sentence may but need not be segmented at points where two segments meet in a right angle. $\mathcal{L}(1\forall - CCMG) \not\ni R \in \mathcal{L}(0-CCMG)$. R can obviously be generated by the *0-CCMG* sketched in Figure 3. It is highly unlikely that R can be specified by a *1*$\forall$*-CCMG*, since some derivations need to "construct" a number of merged line segments exceeding the size of the input.

From this conjecture and the theorems above we can immediately derive two other conjectures:

Conjecture 2 $\mathcal{L}(1-CCMG) \subsetneq \mathcal{L}(0-CCMG)$.

[4]The authors expect to be able to provide a proof in the final version of the paper.

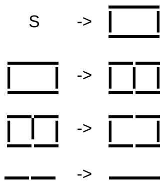

Figure 3: Tiling Grammar

Conjecture 3 *$\mathcal{L}(0 - CCMG) \subsetneq \mathcal{L}(0\forall - CCMG)$ and $\mathcal{L}(1\forall - CCMG) \subsetneq \mathcal{L}(0\forall - CCMG)$.*

Finally, we have to look at arbitrary first order quantification of contexts. Surprisingly, they do not give any additional expressiveness:

Theorem 7 *$\mathcal{L}(0FOL - CCMG) = \mathcal{L}(0\forall - CCMG)$.*

Theorem 8 *$\mathcal{L}(1FOL - CCMG) = \mathcal{L}(1\forall - CCMG)$.*

In both cases, nested quantification in productions can inductively be removed, until no $\forall\exists\ldots$ chains are left. For each production in P: if the leading quantification in C is $\exists x$ then x can be handled as a normal context symbol. If the leading quantification is $\forall x$ then this quantification is removed and replaced by a "proof procedures", i.e., a sequence of productions that explicitely iterates over all symbols of type x and tests the appropriate condition. The proof procedure contains only productions with $n - 1$ levels of quantification, when C has n levels.

4 Complexity

Apart from expressiveness, we also have to consider the complexity of parsing. Therefore the *membership problem* has been analyzed for the language classes defined above. Given a sentence S and a grammar G the membership problem is to determine if $S \in \mathcal{L}(G)$. For a particular class of grammars $\mathcal{G}$, there are really two complexity questions we are interested in. The first question is, what is the complexity of the membership problem for all $G \in \mathcal{G}$ and for all sentences S. The second question is, given a fixed grammar $G \in \mathcal{G}$ what is the complexity of the membership problem for G for all sentences S. We assume throughout this section that constraint testing takes polynomial time.

In comparison with string grammars, it is surprisingly expensive to solve the membership problem for regular CMGs. Only the basic results can be summarized here:

Theorem 9

- *The complexity of the membership problem for a fixed grammar $G \in 3 - CCMG$ is NP-complete.*
- *The complexity of the membership problem for $3 - CCMG$ is PSPACE-complete.*

The first part follows from a 3-SAT encoding in a *3-CCMG* and the following theorem. The second part is proven by encoding a context-sensitive string grammar into a *3-CCMG*.

Theorem 10

- *The complexity of the membership problem for a fixed grammar $G \in 2 - CCMG$ is NP-complete.*
- *The complexity of the membership problem for $2 - CCMG$ is PSPACE-complete.*

An NP-Algorithm to guess and check a derivation for a fixed grammar is straight forward, since the derivation length is bounded polynominal by

(# of non-terminals in G) $\times$ (# of different attributes in S)$^{(\max \#\text{ of attributes of non-terminals in } G)}$
$\times$(# of tokens in S)

An NPSPACE-Algorithm for the second part is also easy to construct. It guesses each sentence in the derivation and the applied production. Clearly not more than polynominal space is required to check it.

Theorem 11 *Let $\mathcal{G}$ be one of the grammar classes of $1-CCMG$, $1\forall-CCMG$, or $1FOL-CCMG$.*

- *The complexity of the membership problem for a fixed grammar $G \in \mathcal{G}$ is PSPACE-complete.*
- *The complexity of the membership problem for $\mathcal{G}$ is PSPACE-complete.*

Since a *1-CCMG* which is a universal grammar for context-senstitve string grammars can be defined, the membership problem is PSPACE-hard. It is PSPACE-complete, because we can use an NPSPACE-algorithm similar to the one used in Theorem 10.

Theorem 12

- *The membership problem for a fixed grammar $G \in 0-CCMG$ is EXP-SPACE hard.*
- *The membership problem for $0-CCMG$ is decidable.*

Both problems can be reduced to the word problem for commutative Semi-Thue Systems, which is known to be decidable and EXP-SPACE hard [13].

Theorem 13 *Let $\mathcal{G}$ be either $0\forall-CCMG$ or $0FOL-CCMG$. The membership problem for a fixed grammar $G \in \mathcal{G}$ is undecidable.*

5 Integrating Other Formalisms

A hierarchy as the one discussed in the previous sections, though giving insight in formal properties of VLs, would not serve the purpose of providing common ground for VL research, if it was confined to a single formalism. Therefore mappings of other formalisms to CMGs have to be established. We have analyzed some considerably different well-established grammar formalisms, and all of them can be mapped to CMGs. The full paper will discuss mappings for Positional Grammars [4], Relational Grammars [6], and Unification Grammars [18, 17] in detail.

Theorem 14 *Let G be either a Positional Grammar, a 1NS Relational Grammar, or an Atomic Relation Grammar. There is a Constraint Multiset Grammar G' such that $\mathcal{L}(G) = \mathcal{L}(G')$.*

5.1 Positional Grammars

Positional grammars (PGs) are rewrite systems working on sets of symbols that are located at a position on some grid. A PG production has the form

$$A \rightarrow x_1 R_1 x_2 R_2 \ldots\ R_{m-1} x_m$$

where A is a non-terminal, x_i is a terminal or non-terminal and R_i is a spatial relation of the form (k, l) indicating that x_{i+1} must be found at the position $(i+k, j+l)$ if x_i was located at (i, j). The corresponding production set of G' is obtained by replacing each PG production of the above form with a CMG production

$$\begin{array}{lll} A(\bar{x}, \bar{y}) & ::= & x_1(\bar{x}_1, \bar{y}_1), \ldots, x_m(\bar{x}_m, \bar{y}_m) \\ \textit{where} & & p(dx_1, dy_1, \bar{x}_1, \bar{y}_1, \bar{x}_2, \bar{y}_2), \ldots, \\ & & p(dx_{m-1}, dy_{m-1}, \bar{x}_{m-1}, \bar{y}_{m-1}, \bar{x}_m, \bar{y}_m) \\ \textit{and} & & \bar{x} = \bar{x}_1, \bar{y} = \bar{y}_1 \end{array}$$

when $R_i = (dx_i, dy_i)$ and $p(m, n, x_1, y_1, x_2, y_2) \Leftrightarrow x_1 + m = x_2 \wedge y_1 + n = y_2$.

5.2 Unification Grammars

Atomic Relational Grammars (ARGs), the successor of Unification Grammars, are rewrite systems operating on a single set of symbols and (evaluable) relations. An ARG production has the form

$$A \rightarrow \ \alpha/\beta/F$$

where A is a non-terminal, α a string of non-terminals and terminals, β a set of constraints of the form $r(x, y)$ where x and y refer either to a symbol in α or to an attribute of such a symbol. F is a set containing exactly one assignments $a = x$ for each attribute a of A where x is again either a symbol in α or an attribute of such a symbol. Attribute values and arguments to relations can be data values or atomic objects. As every relation on two atomic objects can be computed from some of their geometric attributes, the corresponding CMG production set consists of a production

$$A ::= \alpha \ \textit{where} \ \beta' \ \textit{and} \ F'$$

for every production in the ARG where F' transfers the same attributes as F in the case of data values or all of the attributes of the symbol that F transfers in the case of objects, respectively. β' is obtained from β by replacing all relations on objects with relations on the attributes of these objects. Given F' this is neccessarily a finite (one-step) expansion of β.

In the case of PGs and ARGs the structure of the derivation is virtually the same as the structure of the corresponding CMG derivation.

5.3 Relational Grammars

Relational grammars (RGs) are rewrite systems working on a set of unattributed symbols and a set of relations over these symbols simultaneously. The relations are purely symbolical and not interpreted arithmetically. There are two types of productions: s-rules for rewriting symbol sets and r-rules for rewriting relation sets. Both production sets are coupled, so that each time some s-rule is applied certain corresponding r-rules have to be applied. This allows to drop the relation set alltogether. Instead, a list-valued attribute *prop* is used for each symbol. Given a relational sentence (S, R) with symbol set S and relation set R, the corresponding CMG sentence is S such that for each symbol $s \in S$, $s.prop$ contains exactly one item $p_1(X)$ for each $p(s, X) \in R$ and exactly one item $p_2(X)$ for each item $p(X, s) \in R$. For every s-rule an equivalent CMG productions is created that rewrites the object in the same way verifying the spatial relations by testing if the appropriate relations are contained in the attribute *prop*. Additionally it computes the *prop* attribute for its left-hand side symbol by rewritting the *prop* attributes of the rewritten objects according to the r-rule with the appropriate index. Of course, the CMG obtained in this way is rather unnatural, since it cannot make any use of arithmetic constraints, but uses symbolic derivations of list-valued attributes instead.[5] Also, the derivation structure of both grammars is much different. These differences have to be attributed to the fact that the interpretation of a relation is not neccessarily context-free in an RG. The mapping becomes much more natural[6], if the reasonable assumption is made that terminal symbols have geometric attributes and that all admissible relation symbols have a geometric interpretation such that they can be tested by computations on these attributes.

6 Extensions: Confluence and Attribute Handling

The hierarchy skeleton presented above opens several ways for extensions. We are looking into three areas that are suggested by the analysis made so far:

- **Confluence:** The analysis of parsing complexity shows that CMGs are surprisingly expensive to parse. Though a theoretical result like $\mathcal{NP}$ completeness does not automatically render a formalism unusable — *1-CCMG*s are already in use in practical applications — it is important to look at more efficient, restricted

[5] This, of course, cannot be done in a copy-restricted CMG.

[6] and can be done with copy-restricted CMGs

versions. Significant efficiency improvements can be obtained by requiring confluence. The word membership problem for a fixed grammar for a significant set of CMG classes (*3-CCMG*, *2-CCMG*, *2∀-CCMG*, *1-CCMG*, *1∀-CCMG*, *1FOL-CCMG*) is in $\mathcal{P}$ when only confluent cycle-free grammars are permitted.

- **Attribute Handling:** Copy-restricted grammars are a reasonable concept for theoretical investigations, but they are not sufficient in general for two reasons: (1) Not all other formalisms can be mapped to CCMGs (cf. RGs in Sec. 5). (2) Not all spatial concepts can be expressed in CCMGs (it is, e.g., impossible to compute the intersection point of two non axis-parallel lines). Moreover, certain forms of context sensitive grammars can probably be replaced by simpler grammar classes plus attribute handling. We therefore have to investigate extensions. A natural extension is to consider continuous regions with $\cup, \cap$ as a generalization of bounding boxes. It should also be possible to integrate more complex operations on regions derived from the relations in [3]. In the latter case we even obtain a topologically complete set of operations. Both domains are special cases of subset selection and unions of a finite set of sets, and for such domains most of the complexity and decidability results outlined above continue to hold.
- **Sets vs. Multisets:** CMGs are based on multisets, but often only sets are used. Restricting CMGs completely to sets can make some classes more efficient without loosing expressive power.

7 Related Work & Conclusions

Although there is a lot of work on VL specification formalisms, systematic comparisons or taxonomies are rare. Such a common basis, however, seems to be indispensable to clarify similarities and differences between the various lines in VL research and to gain the most from their insights.

Some work along these lines has been done for early approaches like web grammars and array grammars [1, 15]. Array grammars, however, lack expressive power. Later work related to web grammars can be found in the graph grammar community. A very systematic survey is [5], where connections between first order logic, monadic second order logic, algebraic specification, and graph grammars are discussed. It is, however, not easy to see, how different types of graph grammars can be mapped into the above discussed VL grammar formalisms and vice versa. Graph and web grammars have a different approach to VL specification. There, spatial relations are treated as uninterpreted, thus in a certain sense "meaningless" relations. It seems natural to interpret spatial relations, be it arithmetically or deductively, since otherwise their *natural spatial properties and interdependencies* are lost.

We have discussed such an approach and have identified basic classes forming the skeleton of a hierarchy. We have also illustrated how other formalisms can be mapped into it. Refinements along the various dimensions outlined above will hopefully be flexible enough to yield finer grained classes and to integrate an increasing number of different formalisms.

References

[1] N. Abe, M. Mizumoto, J.-I. Toyoda, and K. Tanaka. Web grammars and several types of graphs. *Journal of Computer and System Sciences*, 7, 1973.

[2] S.S. Chock and K. Marriott. Automatic construction of user interfaces from constraint multiset grammars. In *IEEE Symposium on Visual Languages*, 1995.

[3] E. Clementini, P. Di Felice, and P. van Oosterom. A small set of formal topological relationships suitable for end-user interaction. In *Advances in Spatial Databases: SSD-93*, 1993.

[4] G. Costagliola, S. Orefice, G. Polese, G. Tortora, and M. Tucci. Automatic parser generation for pictorial languages. In *IEEE Symposium on Visual Languages*, 1993.

[5] B. Courcelle. Graph rerwiting: An algebraic and logic approach. In J. van Leeuwen, editor, *Handbook of Theoretical Computer Science*. Elsevier, 1990.

[6] F. Ferrucci, G. Tortora, M. Tucci, and G. Vitiello. A predictive parser for visual languages specified by relation grammars. In *IEEE Symposium on Visual Languages*, 1994.

[7] J.E. Gips. *Shape Grammars and their Uses*. PhD thesis, Stanford University, 1974.

[8] E. Golin and S.P. Reiss. The specification of visual language syntax. In *IEEE Symposium on Visual Languages*, 1989.

[9] V. Haarslev. Formal semantics of visual languages using spatial reasoning. In *IEEE Symposium on Visual Languages*, 1995.

[10] R. Helm and K. Marriott. A declarative specification and semantics for visual languages. *Journal of Visual Languages and Computing*, 2, 1991.

[11] R. Helm, K. Marriott, and M. Odersky. Building visual language parsers. In *ACM Conf. Human Factors in Computing*, 91.

[12] K. Marriott. Constraint multiset grammars. In *IEEE Symposium on Visual Languages*, 1994.

[13] E.W. Mayr. An algorithm for the general petri net reachability problem. *SIAM Journal of Computing*, 13, 1984.

[14] B. Meyer. Pictures depicting pictures. In *IEEE Symposium on Visual Languages*, 1992.

[15] Azriel Rosenfeld. Array and web grammars: An overview. In A. Lindenmayer and G. Rosenberg, editors, *Automata, Languages, and Development*. North-Holland, 1976.

[16] S. Üsküdarli. Generating visual editors for formally specified languages. In *IEEE Symposium on Visual Languages*, 1994.

[17] K. Wittenburg. Predictive parsing for unordered relational languages. In *Recent Advances in Parsing Technologies*, to appear.

[18] K. Wittenburg and L. Weitzmann. Visual grammars and incremental parsing for interface languages. In *IEEE Symposium on Visual Languages*, 1990.