Towards a Hierarchy of Visual Languages

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

Abstract

Formalisms for visual language specification have been investigated for more than two decades now. However, there has been little attempt to develop a systematic and comprehensive hierarchy of visual languages based on formal properties. Given the importance of the Chomsky hierarchy in the theory of textual languages and the difficulty of comparing the different visual language formalisms, it is clear that there is a need for such a hierarchy. We develop a hierarchy for VLs and investigate the expressiveness and cost of parsing for classes defined therein. Although the hierarchy is based on the constraint multiset grammar formalism, we sketch how other visual language specification formalisms can be mapped into constraint multiset grammars. One consequence of our work is that a large class of "naturally occurring" visual languages are inherently context-sensitive, so that the core of such a hierarchy has to be built around different forms of context-sensitivity.

1 Introduction

Many different formalisms for the specification of visual languages have been investigated for more than two decades now. Grammar-like formalisms range from early approaches like web and array grammars [15], and shape grammars [6] to recent formalisms like positional grammars [3], relational grammars [5], unification grammars [18, 17], attributed multiset grammars [7], constraint multiset grammars [12], and several types of graph grammars [4]. There are also a variety of non grammar-like formalisms, including algebraic approaches [16] and logic-based approaches [10, 14, 9].

In order to understand the tradeoffs and comparative advantages of these different formalisms, there is a need for a systematic and comprehensive taxonomic hierarchy of visual languages. Such a taxonomy should specify properties of visual languages which allow us to classify them and the position of a particular class of languages in the taxonomy should correspond to the expressiveness of the class and the cost or decidability of parsing and related problems. In the world of textual languages the Chomsky hierarchy serves exactly this purpose. Here we present such a hierarchy for visual languages.

Ideally a taxonomic hierarchy for visual languages should be general enough to include all of the different visual language formalisms. This, however, seems unachievable because there is too much variation in the basic notion of what the building blocks of a visual language are. This is true even if we only consider grammatical formalisms. For example, some formalisms use only lines or points as atomic objects, while others allow almost arbitrary complex symbols.[2] Some do not use attributes at all, while others allow arbitrary computations on attributes. Some support the notion of spatial relationships as abstract, uninterpreted relations, while others strictly treat them as arithmetically interpreted relations. Some handle objects as arbitrary (multi-)sets, others require them to be arranged on some grid. This contrasts to the case of textual languages in which it is clear that the basic object to be manipulated is a sequence of symbols.

Given this variation, it seems the only reasonable way to obtain a general classification of visual languages is to base the taxonomy on some particular but flexible specification method and to clarify how the other formalisms can be mapped into this classification. These mappings in turn are likely to give rise to new, finer grained classes of languages. This is the approach we have taken.

Our main technical contributions are threefold: Firstly, we introduce a taxonomy for visual languages based on different classes of constraint multiset grammars (CMGs). Secondly, we support the choice of CMGs as the basis for the taxonomy by showing how several other representative formalisms – postional grammars, relational grammars and unification grammars – can be mapped into CMGs. Thirdly, we study the relative expressiveness of classes in this taxonomy. We characterize the cost of parsing with each class in the language.

To our knowledge this is the first taxonomic hierarchy for visual languages apart from some work along these lines for early approaches like web grammars and array grammars [1, 15]. Array grammars, however, lack expressiveness. Later research based on web grammars can be found in the graph grammar community. A very systematic survey is [4], 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 previously mentioned visual language formalisms and vice versa. The main reason is that in the graph and web grammar approach to visual language specification, 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.

[1]supported by DFG Grant ME11/94

[2]So the borders between lexical and syntactic recognition are shifting.

2 Constraint Multiset Grammars

We have chosen constraint multiset grammars (CMGs) as the basis for our taxonomy. CMGs have informally been introduced in [11] and have been used for a variety of reasonably complex tasks like analysis of state diagrams and geometric diagrams. In [12] a precise formal treatment is given, and we review only the basic notions here. CMGs are rewrite systems for multisets of objects. Spatial relations are handled by defining arithmetical constraints on object attributes, i.e., there is no explicit representation for spatial relations. A distinguishing feature of CMGs is that they can either be regarded as normal rewrite systems or as a special form of constraint logic programs. In fact, one parsing algorithm for CMGs is a variant of bottom-up derivation for constraint logic programs.

A concrete example of a CMG production taken from the specification of state diagrams is

```
TR:transition ::= A:arrow, T:text
   where exists R:state, S:state where
   T.midpoint close_to A.midpoint,
   R.radius = distance(A.startpoint, R.midpoint),
   S.radius = distance(A.endpoint, S.midpoint)
   and TR.from=R.name, TR.to=S.name.
```

In the standard notation for CMGs each production has the form

$$x ::= X_1, \ldots, X_n \ \textit{where exists}\ X'_1, \ldots, X'_m \ \textit{where}\ C \ \textit{and}\ \vec{v} = E$$

meaning that the non-terminal x can be rewritten to the multiset $X_1, \ldots, X_n$ if the sentence contains symbols $X'_1, \ldots, X'_m$ (the context) such that the attributes of these symbols satisfy the constraint C and that the attributes of x, $\vec{v}$, are given by the expression E.

In the present paper it will be convenient to represent productions in a CMG using a slightly different syntax. The above production will be represented by:

$$xX'_1 \cdots X'_m \Leftarrow X_1 \cdots X_n X'_1 \cdots X'_m \mid\mid C \wedge \vec{v} = E$$

In general, x is a non-terminal symbol with attributes $\vec{v}$, $X_1, ..., X_n$ and $X'_1, ..., X'_m$ are terminal or non-terminal symbols with $n \geq 0$ and the constraint C and expression E are over the attributes of $X_1, ..., X_n$ and $X'_1, ..., X'_m$.

Definition 1 *A* constraint multiset grammar (CMG) *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 of the above form.*

The language $\mathcal{L}(G)$ generated or recognised 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.

2.1 Comparison to Other Formalisms

In order to make sure that a hierarchy based on CMGs can be used to relate results obtained from other formalisms, we must verify that other grammar formalisms can be mapped to CMGs. An analysis of three sufficiently different formalisms shows that this is the case. Only the basic ideas can be illustrated here.

2.1.1 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).

Theorem 1 *For every PG, G, there is a CMG, G', with $\mathcal{L}(G) = \mathcal{L}(G')$. G' does not use any context symbols.*

The production set of G' is simply obtained by replacing each PG production of the above form with a CMG production

$$\begin{aligned} A \Leftarrow x_1 \ldots x_m p(dx_1, dy_1, x_1.x, x_1.y, x_2.x, x_2.y) \\ \wedge \ldots \wedge \\ p(dx_{m-1}, dy_{m-1}, x_{m-1}.x, x_{m-1}.y, x_m.x, x_m.y) \wedge \\ A.x = x_1.x \wedge A.y = x_1.y \end{aligned}$$

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

2.1.2 Relational Grammars

Relational grammars (RGs) are rewrite systems which simultaneously work on a set of unattributed symbols and a set of relations over these symbols. The relations are purely symbolic and are not interpreted arithmetically. There are two types of productions: s-rules for rewriting symbol sets and r-rules for rewriting relation sets. Each s-rule has the form

$$[n]A ::= \{m_1, \ldots, m_k\}, \{p_1(x_1, y_1), \ldots, p_l(x_l, y_l)\}$$

where n is a production index, A is a non-terminal, m_i are terminals or non-terminals, and $p_i(x_i, y_i)$ are relations over some m_i. Each r-rule has the form $r(x_1, x_2) ::= [n, j]P$ where $r(x_1, x_2)$ is a relation, n, j are indices, and P is a set of relations. The symbol set is rewritten by replacing A in the object set by $\{m_1, \ldots, m_k\}$ and adding $\{p_1(x_1, y_1), \ldots, p_l(x_l, y_l)\}$ to the relation set. Each time an s-rule with index n rewrites a symbol A, every relation $r(A, x)$ in the relation set is rewritten by the r-rule with index $[n, 1]$ and every relation $r(x, A)$ is rewritten by the r-rule with index $[n, 2]$. An RG is called a 1NS-RG if every production involves only relations containing at most one nonterminal [5].

Theorem 2 *For every 1NS-RG, G, there is a CMG, G', with $\mathcal{L}(G) = \mathcal{L}(G')$.*

The basic idea how to simulate the two coupled rewrite systems is to drop the set of relations and to use a list-valued attribute *prop* for each symbol instead. 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 rewriting the *prop* attributes of $m_1, \ldots, m_k$ according to the r-rule with the appropriate index. Of course, the CMG obtained in this way is rather unnatural, since it does not make any use of arithmetic constraints. The translation becomes much more natural, 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, This is, however, not guaranteed for arbitrary 1NS-RGs.

2.1.3 Unification Grammars

Visual unification grammars have been introduced in [18]. Their successor, atomic relational grammars (ARGs), is presented in [17]. ARGs 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. All arguments to relations have to be atomic objects.

Theorem 3 *For every ARG, G, there is a CMG, G', with $\mathcal{L}(G) = \mathcal{L}(G')$.*

As every relation on two atomic objects can be computed from some of their geometric attributes, each ARG rule of the form above can be replaced with a CMG production

$$A \Leftarrow \alpha \mid\mid \beta' \wedge F'$$

ARGs use constraints on symbols, but a CMG must express them by constraints on attributes. Since every ARG constraint $r(x, y)$ must have some computable (i.e., normally arithmetical) interpretation depending on the attributes of x and y, it is easy to modify β into β' such that β' directly uses the attributes of x and y. The only remaining problem is that ARG attributes can be symbols (objects) as well as values. Objects cannot be used as attribute values in a CMG. So F has to be modified into F' such that each assignment $a = x$, where x references an object, is rewritten as a sequence of assignments such that all attributes of x are transfered to A.

As the rules for the application of productions in ARGs and CMGs are virtually the same, even the structure of the derivation is the same in both cases.

3 Copy-restricted Grammars and their Hierarchy

Unfortunately it is not possible to base an interesting hierarchy directly on the definition of CMGs. This is because the arbitrarily complex attribute computations allowed in CMGs are too powerful to allow us to build a hierarchy with sharp class borders. The problem is that we can simulate the derivation mechanisms by accumulating relevant attributes of potential context symbols. For instance, witness the encoding used for embedding relational grammars in CMGs. This difficulty is not surprising, since CMGs over the domain of integers with functions $\{+, -, = 0\}$ are computationally adequate [12].

We therefore need to investigate CMGs in which the type of attribute computation allowed is restricted. The most natural (albeit draconian) restriction is not to allow computations at all – attributes of the LHS symbol must be *copied* from the attributes of the RHS symbols.

Definition 2 *A* copy-restricted CMG (CCMG) *is a CMG in which every production has the form*

$$xX'_1 \cdots X'_m \Leftarrow X_1 \cdots X_n X'_1 \cdots X'_m \mid\mid C \wedge \vec{v} = E$$

where E is a vector of attributes of $X_1 \cdots X_n, X'_1 \cdots X'_m$.

Copy-restricted CMGs give rise to slightly simpler parsing algorithms and lend themselves to a formal treatment as all computation is moved into the derivation process proper, rather than attribute computation. Though the copy restriction seems to be a severe limitation, virtually all of the examples that have been presented for CMGs before are in fact CCMGs. For example, recall the production for transitions. So CCMGs are not only of theoretical value.

Using CCMGs as the basis, we will now develop a hierarchy of visual languages. There are two natural directions in which we can extend or restrict CCMGs. The first direction is to consider the analogue of the restrictions used in the Chomsky hierarchy on the form and number of symbols in the LHS and RHS of the productions.

The above (context-sensitive) production used in the definition of CCMGs is termed *type 1*.

Definition 3 *A type x CCMG (denoted as $CCMG_x$) is a CCMG in which every production is of type x.*

The first natural restriction is to disallow any context symbols:

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

This type can be further restricted to an even simpler class similar to regular grammars:

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 $B \in T_{NT}$.*

It is also natural to generalise type 1 CCMGs as they are not expressive enough to allow the specification of all naturally occurring visual languages. For example, they cannot

specify constraints that are quantified universally and so it is impossible to specify complete graphs using a type 1 CCMG. Analogously to the Chomsky hierarchy we can introduce type 0 grammars which drop the monotonicity criterion of type 1 grammars.

Definition 6 *A CCMG production is* type 0 *if it is of the form* $uA \Leftarrow v \mid\mid C \wedge F$ *, where* $u \in (T_T \cup T_{NT})^*$ *and* $A \Leftarrow v \mid\mid C \wedge F$ *is of type 1 or* v *is empty.*

The reader may also be wondering if there is a class between type 0 and type 1 CCMGs, namely the monotonic type 0 CCMGs. However, as for string grammars the type 1 CCMGs are actually equivalent to the monotonic type 0 grammars. For this reason we will not consider monotonic type 0 CCMGs further.

Lemma 1 *For every* $CCMG_0$ G *containing only productions of the form* $u \Leftarrow v \mid\mid \varphi$ *where* $\mid u \mid \leq \mid v \mid$ *there exists a* $CCMG_1$ G' *with* $\mathcal{L}(G) = \mathcal{L}(G')$*. (Like for textual grammars, every non-shortening production can be replaced by a sequence of type 1 productions.)*

The other dimension in which we can extend CCMGs is to modify the form of the constraint in the production. This dimension is particularly interesting since it has no analogue for string grammars. In particular, negative contexts (i.e., contexts that must not be present in order for a certain production to be applicable) add expressive power [11]. We therefore investigate the following extended form of CCMGS:

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*). A grammar is type* $CCMG_x^{\forall}$ *if all its its productions are of type* $x\forall$*.*

Note that we are assuming that the underlying constraints involving the attributes can be negated, so that a negative context can be rewritten to a universal context.

More generally, we can consider the case in which the constraint is allowed to be an arbitrary first order formula.

Definition 8 *A production is* type xFOL *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 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* $Q_i \in \{\exists, \forall\}$*. A grammar is of type* $CCMG_x^{FOL}$ *if all its productions are of type xFOL.*

Note that it does not make sense to consider type 3FOL and type 2FOL grammars as the extension to first order formula allows existentially quantified symbols and so type 3FOL and type 2FOL CCMG grammars are equivalent to type 1FOL CCMG grammars.

4 Expressiveness

In the previous section we have introduced a hierarchy of formalisms based on CCMGs. In this section we will investigate their expressiveness and in particular their relative expressiveness. It will turn out that even context-sensitive (i.e. type 1) CCMGs are not powerful enough to specify several interesting languages.[3] The full set of classes under investigation is given in Figure 1.

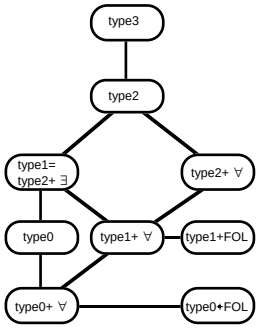

Figure 1: Complete CCMG Hierarchy

Strict inclusions hold between type 3, type 2 and type 1 CCMGs. Type 3 CCMGs can only specify languages in which the constraints are in a certain sense local. They can, for example, not specify the language T of trees. In the following $\mathcal{L}(G)$ denotes the language defined by a CCMG G or the class of languages defined by a CCMG grammar class G.

Theorem 4 $\mathcal{L}(CCMG_3) \overset{\subset}{\neq} \mathcal{L}(CCMG_2)$.

Proof The inclusion is trivial. It is strict, because $\mathcal{L}(CCMG_3) \not\ni T \in \mathcal{L}(CCMG_2)$. For the sake of simplicity we use a graphical tree representation without explicit nodes, i.e., nodes are assumed where edges meet. The following $CCMG_2$ generates the language of binary trees. The extension to trees of arbitrary degree is straight forward:[4]

$$
\begin{aligned}
&start \Leftarrow tree\\
&tree \Leftarrow \varepsilon\\
&tree_1 \Leftarrow line, tree_2 \mid\mid\\
&\qquad tree_2.pos = line.end, tree_1.pos = line.start\\
&tree_1 \Leftarrow line_1, line_2, tree_2, tree_3 \mid\mid\\
&\qquad line_1.end = tree_2.pos, line_2.end = tree_3.pos,\\
&\qquad tree_1.pos = line_1.start = line_2.start
\end{aligned}
$$

A $CCMG_3$ cannot generate L. Every $CCMG_3$-derivation imposes a total order on the terminal symbols. (1) By definition of tree structures there must be a position constraint between every father and each of its sons. (2) Constraints on some $x \in V_T$ can only be stated at its father node in the derivation tree or some node above it to which the attributes of x are passed. (3) Due to the copy restriction the attributes of not more than a constant number c of attributes can be accumulated at any node $n \in V_N$.

[3] The same applies to other multi-dimensional context-sensitive grammar formalisms, if they are restricted to copy-attributes.

[4] The ε-production is used for clarity only and can obviously be removed to obtain a proper $CCMG_2$.

Lemma 2 *For any total order of the nodes of trees there is a tree $T = t_1, \ldots, t_n$ for any given c that contains a continuous subsequence $S = t_l, \ldots, t_k$ such that at least $2c+1$ nodes in S have sons or parents in $T - S$.*

Let T be a such a tree. By (1): In every regular derivation D for T there exists a subsequence S of nodes such that at least $2c+1$ nodes must have a position constraint in relation to some node outside this subsequence. By (2): the relevant attributes of at least $c+1$ attributes have to be passed through t_l or t_k. This contradicts (3). □

Theorem 4 would intuitively be expected extrapolating from knowledge about string grammars. On the other hand, some languages in $\mathcal{L}(CCMG_3)$ may be unexpected: (1) The string language $L = a^i b^i$ is in $\mathcal{L}(CCMG_3)$ if the terminal symbols have an index attribute giving their position in the string. (2) $L = \{p \mid p \text{ is a set of lines forming a closed polygon}\}$ is in $\mathcal{L}(CCMG_3)$. This is due to the fact that not only the structure of the grammar but also constraints on attributes provide means for limited forms of context-handling.

Context-free (type 2) CCMGs, in turn, are less expressive than context-sensitive (type 1) CCMGs, because they can only specify languages with a bounded number of spatial relations for each symbol. So they can, e.g., not specify the language G of graphs:

Theorem 5 $\mathcal{L}(CCMG_2) \overset{\subseteq}{\neq} \mathcal{L}(CCMG_1)$.

Proof The inclusion is trivial. $\mathcal{L}(CCMG_2) \not\ni G \in \mathcal{L}(CCMG_1)$. It is easy to give a $CCMG_1$ for G:

$$
\begin{array}{ll}
start & \Leftarrow point, start \\
start, point_1, point_2 & \Leftarrow start, line, point_1, point_2 \mid\mid \\
 & \quad line.start = point_1.pos, \\
 & \quad line.end = point_2.pos, \ldots \\
start & \Leftarrow \varepsilon
\end{array}
$$

By definition a graph consists of a set of node markers (points) and a set of edge markers (lines) and each line has to be connected to exactly two points. Thus each line must be constrained to end at two points. Let the graph G to be parsed be a complete graph with $2k$ nodes.

Lemma 3 *Every $CCMG_2$ 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 $a \in V_T$. (This can be done by unfolding analogously to textual grammars).*

Let T be a context-free derivation tree generated by a $CCMG_2$ in normal form. It must be of the form given in Picture 2.[5] A constraint for an object can only be stated at some ancestor of its leaf node and the attribute to be constrained must percolate to this ancestor.

Select L and R such that L contains the leftmost k nodes (points) and R the rightmost k nodes. Consider a set of pairs (p_i, p_j) that contains each point p_i, $1 \leq i \leq k$, paired with some $p_j, k+1 \leq j \leq 2k$, such that no point is used in more than one tuple. Since G is complete each pair must be connected by a line l_{ij}. (a) If l_{ij} is in L then the coordinates of p_j must obviously be passed through the root of L in order for l_{ij} to be checked, for p_j is in R. (b) Analogously, if l_{ij} is outside of L the coordinates of p_i have to be passed through the root of L. Thus at least k coordinates have to be passed through the root of L. As the grammar is copy-restricted, only a fixed amount c of information can be passed in the attributes of each node in the derivation tree. As k is linear in the input size we can always find a graph such that the derivation cannot be generated by a $CCMG_2$. □

Adding universal quantification to type 2 CCMGs increases their power because we can now specify conditions quantified over tuples of objects. Thus it is, e.g., possible to specify the language C consisting of all sets of pairwise disjoint circles.

Theorem 6 $\mathcal{L}(CCMG_2) \overset{\subseteq}{\neq} \mathcal{L}(CCMG_2^{\forall})$.

Proof The inclusion is trivial. $\mathcal{L}(CCMG_2) \not\ni C \in \mathcal{L}(CCMG_0)$. The following $CCMG_2^{\forall}$ generates C:

$$
\begin{array}{ll}
start \Leftarrow diag & \\
diag_1 \Leftarrow circle_1, diag_2 \mid\mid & (\forall circle_2) : \\
 & disjoint(circle_1.area, circle_2.area) \\
diag \Leftarrow \varepsilon &
\end{array}
$$

Again, a type 2 derivation must have the structure given in Figure 2. Let the sentence consist of $2k$ circles and let L and R contain k circles each. By definition of C it has to be checked for every circle in L (R) that it is disjoint from every circle in R (L). All circles in L must have different coordinates, so k attributes must pass through the root of L (R). Since k depends only on the size of the input sentence and thus has no bound, there cannot be a $CCMG_2$ that defines C. The problem here is not the quantification over a single symbol type, but the quantification ranging over tuples. $(\forall C_1, C_2 : circle) : disjoint(C_1, C_2)$. □

Adding universal quantification to type 1 CCMGs also extends their power. It allows the specification of the language G of complete graphs. This language is not in $CCMG_2^{\forall}$ because the connectedness of edges cannot be expressed despite of universal quantification.

Theorem 7 $\mathcal{L}(CCMG_2^{\forall}) \overset{\subseteq}{\neq} \mathcal{L}(CCMG_1^{\forall})$.

Proof The inclusion is trivial. $\mathcal{L}(CCMG_2^{\forall}) \not\ni G \in \mathcal{L}(CCMG_1^{\forall})$.

The following $CCMG_1^{\forall}$ specifies complete graphs:

$$
\begin{array}{ll}
\widehat{start} & \Leftarrow point, \widehat{start} \\
\widehat{start} & \Leftarrow \widehat{point}, cont \mid\mid \widehat{point}.pos = cont.pos \\
\widehat{start} & \Leftarrow \varepsilon \\
point, cont & \Leftarrow \widehat{point}, cont, line \mid\mid point.pos = \widehat{point}.pos, \\
 & \quad line.start = point.pos, line.end = cont.pos \\
cont & \Leftarrow res \mid\mid \nexists point \\
\widehat{point}, res & \Leftarrow point, res \mid\mid point.pos = \widehat{point}.pos \\
res & \Leftarrow \widehat{start} \mid\mid \nexists \widehat{point}
\end{array}
$$

A refined version of the argument in Theorem 5 can be used to show that $CCMG_2^{\forall}$s cannot specify G, because they cannot check whether edges are connected to nodes. □

Both type 1 and $\forall$ type 1 CCMGs, can define the connectedness criteria for a graph, but a $CCMG_1$ cannot specify the

[5] ignore the dotted lines for now.

completeness criteria, because it corresponds to a universally quantified condition over tuples $(\forall x \forall y : connected(x, y))$.

Theorem 8 $\mathcal{L}(CCMG_1) \subsetneq \mathcal{L}(CCMG_1^\forall)$.

Proof The inclusion is trivial. We have already shown that for the language G of complete graphs $G \in \mathcal{L}(CCMG_1^\forall)$.

Figure 2 shows a context-sensitive derivation tree. The dotted lines indicate the connections added by the existential quantification. The amount of information passed through a certain node is still bounded by a grammar-dependent constant k. In addition there can be up to e existentially quantified symbols used in every production, adding up to e dotted lines to a node. 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, we know that each line in L has to be checked against n points in R. (if all the lines are contained in R we swap L and R). Since n is not bounded by the grammar, this cannot be done. □

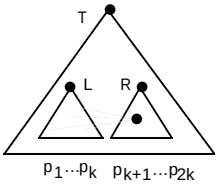

Figure 2: $CCMG_1$ Derivation Structure

Theorem 9 $\mathcal{L}(CCMG_1) \not\supset\not\subset \mathcal{L}(CCMG_2^\forall)$.

Proof The proof that complete graphs are not in $\mathcal{L}(CCMG_2^\forall)$ directly extends to any other kind of graph, as well. Thus $\mathcal{L}(CCMG_1) \not\subset \mathcal{L}(CCMG_2^\forall)$. The proof that complete graphs are not in $\mathcal{L}(CCMG_1)$ can easily be modified to show that the language of disjoint circles is not in $\mathcal{L}(CCMG_1)$, for here an unbounded number of constraints has to be checked, too. So $\mathcal{L}(CCMG_2^\forall) \not\subset \mathcal{L}(CCMG_1)$. □

Adding universal quantification to type 0 CCMGs greatly increases their power, in fact they become *computationally adequate*. Thus we have that:

Theorem 10 $\mathcal{L}(CCMG_0) \subsetneq \mathcal{L}(CCMG_0^\forall)$ *and* $\mathcal{L}(CCMG_1^\forall) \subsetneq \mathcal{L}(CCMG_0^\forall)$.

We conjecture that we gain further expressiveness, when the monotonicity criteria for type 1 grammars is dropped, because derivations do not have to be bounded in the input length then.

Conjecture 1 $\mathcal{L}(CCMG_1^\forall) \not\supset\not\subset \mathcal{L}(CCMG_0)$.[6]

Proof [Sketch] Let G be the language of complete graphs: $\mathcal{L}(CCMG_0) \not\ni G \in \mathcal{L}(CCMG_1^\forall)$. 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. $\mathcal{L}(CCMG_1^\forall) \not\ni R \in \mathcal{L}(CCMG_0)$. R can obviously be generated by the $CCMG_0$ sketched in Figure 3. It is highly

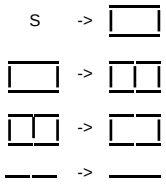

Figure 3: Tiling Grammar

unlikely that R can be specified by a $CCMG_2^\forall$, since some derivations need to "construct" a number of merged line segments that exceeds the size of the input. □

The following conjecture essentially completes the hierarchy. Its validity depends only on Conjecture 1.

Conjecture 2 $\mathcal{L}(CCMG_1) \subsetneq \mathcal{L}(CCMG_0)$.

Proof $\mathcal{L}(CCMG_0) \supset \mathcal{L}(CCMG_1)$ by definition. $\mathcal{L}(CCMG_1^\forall) \not\ni R \in \mathcal{L}(CCMG_0)$. $\mathcal{L}(CCMG_1^\forall) \supset \mathcal{L}(CCMG_1)$. Thus $R \notin \mathcal{L}(CCMG_1)$. □

All that remains is to consider grammars in which the context can be specified with a full first order logical formula. Surprisingly this does not lead to greater expressiveness.

Theorem 11 $\mathcal{L}(CCMG_0^{FOL}) = \mathcal{L}(CCMG_0^\forall)$.

Proof We can inductively remove levels of universal quantification up to a single level. (1) Add a new starting symbol $\hat{S}$ and the productions $\hat{S} \Leftarrow S\sharp \,||$ and $\sharp \Leftarrow \varepsilon \,||$ (2) Replace every production $u \Leftarrow v \,||\, C$ by $u\sharp \Leftarrow v\sharp \,||\, C'$, where C' is C in prenex NF. (3) Consider every member (P) of this new production set in turn. If the leading quantifier in C' is existential rewrite it as a normal context symbol. If P has the form $u\sharp \Leftarrow v\sharp \,||\, \forall x : \varphi$ and φ contains further quantifiers, remove the quantification of x and replace the test by an explicit "proof procedure" consisting of the productions:

$u\sharp \Leftarrow uA \,||$
$uAx \Leftarrow uA\bar{x} \,||\, \varphi$
$uA \Leftarrow uB \,||\, \sharp\bar{x}$
$\bar{x}B \Leftarrow xB \,||$
$B \Leftarrow v\sharp \,||\, \sharp\bar{x}$

Where A, B are unique new non-terminals for each production and $\sharp$ is a single new non-terminal shared by all productions. This process is inductively repeated until all $\forall\exists\ldots$ quantification chains are eliminated. In subsequent steps productions of the form or $u \Leftarrow v \,||\, \forall x : \varphi$ (with $\sharp \notin uv$) have to be treated analogously. □

Theorem 12 $\mathcal{L}(CCMG_1^{FOL}) = \mathcal{L}(CCMG_1^\forall)$.

In principle, the same technique as above can be used. However, we have to be careful about shortening productions. Because of that the $\sharp$ marker that prevents original productions from interfering with "proof procedures" cannot be used in a $CCMG_1^\forall$, and the actual productions are much more complicated.

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

5 Complexity of Parsing

Given a sentence S and grammar G the *membership problem* is to determine if $S \in \mathcal{L}(G)$. In this section we investigate the inherent complexity of the membership problem for the various classes in the hierarchy.

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 textual grammars, it is surprisingly expensive to solve the membership problem for regular and context-free CMGs.

Theorem 13 *Let $\mathcal{G}$ be one of $CCMG_3$ or $CCMG_2$. The complexity of the membership problem for a fixed grammar $G \in \mathcal{G}$ is NP-complete. The complexity of the membership problem for $\mathcal{G}$ is PSPACE-complete.*

This follows from the following four lemmas. The first follows from the encoding of 3-SAT in Golin [8].

Lemma 4 *For some $G \in CCMG_3$ the membership problem is NP-hard.*

Lemma 5 *For any fixed $G \in CCMG_2$ the membership problem is in NP.*

Proof The NP-algorithm is to guess a derivation D for S which does not have any repeated sentences and with each stage identifies the tokens reduced at the grammatical rule used. Clearly $S \in G$ iff such a derivation exists. The length of D is bounded by $a \times b^c \times d$ where a is the # of non-terminals in G, b is the # of different attribute values in S, c is the max # of attributes of non-terminals in G, and d is the # of tokens in S which is polynomial in S for a fixed G. It is possible to verify that D is a valid derivation in polynomial time. □

Lemma 6 *For $CCMG_3$ the membership problem is PSPACE-hard.*

Proof The proof is by encoding a context sensitive string grammar G' and string S' into a $CCMG_3$ G. The encoding of S' is as a single token called *string* with essentially an attribute for each symbol in the string. The encoding of G' is to productions in G which reduce a *string* token to another *string* token. Let S be the singleton multiset containing a token of type *string* which encodes the string S'. Then $S \in \mathcal{L}(G)$ iff $S' \in \mathcal{L}(G')$. Now the membership problem for context sensitive string grammars in PSPACE-hard. As the size of the encoding of G' and S' is polynomial in the size of G' and S', it follows that the membership problem for $CCMG_3$ is PSPACE-hard. □

Lemma 7 *For $CCMG_2$ the membership problem is in PSPACE.*

Proof It suffices to show that it is in NPSPACE. Imagine we are trying to determine if $L \in \mathcal{L}(G)$. Our non-deterministic algorithm just guesses each sentence in the derivation from L to the start symbol together with the production used and the mapping from symbols in the production to those in the sentence. At each step the algorithm verifies that this is a valid derivation by checking the constraints. At each stage only two sentences need to be remembered. As the size of each of these sentences is less than or equal to L and the size of the production is less than or equal to G, the algorithm takes only polynomial space. □

Parsing with $CCMG_1$s has roughly the same cost as parsing with context-sensitive string grammars.

Theorem 14 *Let $\mathcal{G}$ be one of the grammar classes of $CCMG_1$, $CCMG_1^{\forall}$, or $CCMG_1^{FOL}$. 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.*

This follows from the following two lemmas.

Lemma 8 *For some $G \in CCMG_1$ the membership problem is PSPACE-hard.*

Proof This can be proved by giving a $CCMG_1$ grammar G which is a universal grammar for context sensitive string grammars. Now consider a context sensitive string grammar G' and string S'. Let S be the encoding of productions in G', symbols in S' and the token *reducerule*. Then $S \in \mathcal{L}(G)$ iff $S' \in \mathcal{L}(G')$. Now the membership problem for context sensitive string grammars in PSPACE-hard. As the size of the encoding of G' and S' is polynomial in the size of G' and S', it follows that the membership problem for G is PSPACE-hard. □

Lemma 9 *For $CCMG_1^{FOL}$ the membership problem is in PSPACE.*

Proof Similar to Lemma 7. □

Surprisingly, the membership problem for $CCMG_0$s is decideable. This contrasts with type 0 string grammars, where it is undecideabe.

Theorem 15 *The membership problem for a fixed grammar $G \in CCMG_0$ is EXP-SPACE hard. The membership problem for $CCMG_0$ is decidable.*

A constraint multiset grammar in which no symbols have attributes is said to be *propositional*. The membership problem for propositional $CCMG_0^{\forall}$ has been previously studied under the name of the word problem for commutative Semi-Thue Systems. This problem is known to be decidable and EXP-SPACE hard [13]. This result can be used to prove the above theorem.

Proof Consider the sentence S and $CCMG_0$ grammar G. Let the attributes occuring in S be A. Let S' be the set of tokens which can be constructed from the types in G using the attributes in A. Clearly S' is finite. We can map each element t of S' to a unique propositional token $\iota(t)$. Now consider a production P in G. We can map it to a set of productions $\iota(P)$ which rewrite a sentence of propositional

tokens to a new sentence of propostional tokens iff the corresponding tokens in S' can be rewritten with G. Note that $\iota(P)$ is finite. Let G' be the set of $\iota(P)$s where P is a production in G and S' the propositional tokens encoding S. Then we have $S \in \mathcal{L}(G)$ iff $S' \in \mathcal{L}(G')$. As the membership problem for propositional $CCMG_0$ grammars is decidable, it is decidable if $S' \in \mathcal{L}(G')$, so $S \in \mathcal{L}(G)$ is decidable. □

The following theorem holds because the membership problem for propositional $CCMG_0^\forall$ grammars is undecidable. The proof uses a similar encoding to Theorem 15 to give a $CCMG_0^\forall$ grammar which is a universal grammar for propositional $CCMG_0^\forall$ grammars.

Theorem 16 *Let $\mathcal{G}$ be either $CCMG_0^\forall$ or $CCMG_0^{FOL}$. The membership problem for a fixed grammar $G \in \mathcal{G}$ is undecidable.*

6 Conclusion and Extensions

We have developed a hierarchical classification of visual languages based on CMGs. We have investigated the expressiveness and the cost of parsing for the classes in the hierarchy. We have also illustrated how other formalisms can be mapped into CMGs and hence into the hierarchy.

One consequence of our results concerning expressiveness is that context-sensitive specifications are important for visual languages. We note that although the majority of textual languages are context-free, this is *because* they have artificially been designed *to be* context-free. For the same reason virtually all of the artificially designed visual languages are context-free. However, a prime objective of visual language research is to allow intuitive, easy-to-use visual languages, and we feel this necessarily includes visual languages such as graphs that have not particularly been designed for computer usage and are not context-free.

Our results on complexity seem somewhat depressing as they indicate that parsing for even the most restrictive class in the hierarchy is very expensive. This, however, is not necessarily true. Firstly, a theoretical $\mathcal{NP}$-hardness result does not automatically render a method unusable in practice. Context-sensitive CCMGs are already in use in real-time editing applications [2]. Secondly, we can consider restrictions which make the underlying formalism more efficient. Currently we are focusing on two approaches: *confluence* and *sets vs. multisets*.

The main reason for the high complexity of parsing is nondeterminism. If grammars are required to be confluent, then, under reasonable assumptions, the cost of parsing for a given monotonic CCMG becomes polynomial. The reason for the undecidability of parsing in the universally quantified type 0 case is because we can introduce multiple occurrences of the same symbol with the same attribute values. If instead we view parsing as set rewriting, rather than multi-set rewriting then the membership problem is decidable for all elements of the hierarchy.

Another extension we are looking at, is how to introduce limited forms of attribute computation without flattening the hierarchy. This however, requires great care as simple extensions like introducing addition and subtraction can make parsing undecidable, and allowing accumulating data types such as sets or lists can make it possible to simulate arbitrary positive and negative contexts. Nontheless, in practical applications, attribute computation may well be useful and inexpensive to integrate. The most obvious domain to explore are continuous regions (point sets) with union and intersection operations serving as a natural non-rectangular generalisation of bounding-boxes.

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

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

[5] 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.

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

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

[8] E.J. Golin. *A Method for the Specification and Parsing of Visual Languages*. PhD thesis, Brown University, 1991.

[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.