Contextual Hypergraph Grammars – A New Approach to the Generation of Hypergraph Languages*

Adrian-Horia Dediu[1,4], Renate Klempien-Hinrichs[2], Hans-Jörg Kreowski[3], and Benedek Nagy[1,5]

[1] Research Group on Mathematical Linguistics, Rovira i Virgili University
Pl. Imperial Tàrraco 1, 43005 Tarragona, Spain
adrian.dediu@urv.net
[2] University of Bremen, Department of Production Engineering
P.O.Box 33 04 40, 28334 Bremen, Germany
kh@biba.uni-bremen.de
[3] University of Bremen, Department of Computer Science
P.O.Box 33 04 40, 28334 Bremen, Germany
kreo@informatik.uni-bremen.de
[4] Faculty of Engineering in Foreign Languages
University "Politehnica" of Bucharest, Romania
[5] Faculty of Informatics, University of Debrecen, Hungary
nbenedek@inf.unideb.hu

Abstract. In this paper, we introduce contextual hypergraph grammars, which generalize the total contextual string grammars. We study the position of the class of languages generated by contextual hypergraph grammars in comparison with graph languages generated by hyperedge replacement grammars and double-pushout hypergraph grammars. Moreover, several examples show the potential of the new class of grammars.

Keywords: Contextual hypergraph grammars, contextual grammars, hypergraphs, hyperedge replacement grammars, double-pushout hypergraph grammars.

1 Introduction

Solomon Marcus introduced the *external* variant of contextual grammars in 1969. Initially designed to generate languages without nonterminal rewriting, only by

* This work was possible for the first author thanks to the grant 2002CAJAL-BURV4, provided by the University Rovira i Virgili. The third author's research was partially supported by the EC Research Training Network SegraVis (Syntactic and Semantic Integration of Visual Modeling Techniques) and by the Collaborative Research Centre 637 (Autonomous Cooperating Logistic Processes: A Paradigm Shift and Its Limitations) funded by the German Research Foundation (DFG). The fourth author acknowledges the financial support provided through the European Community's Human Potential Programme under contract HPRN-CT- 2002-00275, SegraVis and by the grants from the Hungarian National Foundation for Scientific Research (OTKA F043090 and T049409).

O.H. Ibarra and Z. Dang (Eds.): DLT 2006, LNCS 4036, pp. 327–338, 2006.

adjoining contexts using a selection procedure, contextual grammars were discovered to have some limitations. Yet, as the term contextual seems to be very appropriate to model linguistic aspects, soon a large variety of such grammars appeared. We mention here only several types, like the *internal* case, *total contextual*, *grammars with choices* respectively, etc. For more details, the reader is referred to [1]. In the basic case of a contextual grammar, a context is a pair (u, v) of strings that are to be inserted into axioms or derived strings. More explicitly, a string x directly derives a string y using the context (u, v) if $y = x_1ux_2vx_3$ for some decomposition $x = x_1x_2x_3$.

Despite the large variety of contextual grammars, it is difficult to put together strings and structures, a very important intrinsic quality of natural languages. There were several proposals to introduce bracketed contextual grammars [2, 3, 4] in order to enhance the words in the generated languages with a tree structure, or to add a dependency relation to contexts, axioms and to generated words. However, we believe that hypergraphs provide a more general structure that could enhance a textual string representation, and therefore we propose a new approach to the generation of hypergraph languages.

In this paper (Section 3), we introduce the concept of contextual hypergraph grammars as a generalization of contextual grammars by considering hypergraphs instead of strings as underlying data structures. The insertion operation is taken over by a merging operation. Two hypergraphs, each with a sequence of external vertices of the same length, are merged by identifying corresponding external vertices whereas all other items are kept disjoint. We can see the external vertices as gluing points, each external node "waiting for" another external node from another hypergraph in order to become an internal node.

A contextual hypergraph grammar is given by finite sets of axioms and contexts both being hypergraphs. Starting with an axiom, derivations are composed of iterated merging with contexts. While the contexts are equipped with external vertices by definition, axioms and intermediately derived hypergraphs do not have external vertices of their own. Therefore, before they can be merged with some context, some preparation is necessary that equips them with external vertices in a suitable way. For this purpose, a contextual hypergraph grammar provides an operator Θ that depends on the contexts and associates each hypergraph without external vertices with a set of hypergraphs each with a proper sequence of external vertices. The idea is that $\Theta_C(H)$ for some context C and some hypergraph H yields variants of H that can be merged with C in particular. Such a merging defines a derivation step if H is an axiom or some already derived hypergraph. In this way, the Θ-operator plays on the level of hypergraphs the role of decomposition on the level of strings.

As shown in Section 4, hyperedge replacement grammars in the sense of [5, 6] can be simulated as contextual hypergraph grammars using a suitable kind of variants which are constructed by the removal of single hyperedges. By means of a more sophisticated kind of variants that are constructed by the removal of homomorphic images of hypergraphs up to a certain set of vertices, one can also translate arbitrary hypergraph grammars in the double-pushout approach

(see, e.g., [7]) into contextual hypergraph grammars. This proves in particular that all recursively enumerable sets of hypergraphs can be generated by contextual hypergraph grammars. In this sense, our new approach is computationally complete.

2 Preliminaries

In this section, we recall all the notions and notations of hypergraphs as needed in this paper.

We denote by $\mathbb{N}$ the set of natural numbers. For $n \in \mathbb{N}$, $[n]$ represents the finite set $\{1, \ldots, n\}$, with $[0] = \emptyset$.

For a given finite set A, $|A|$ denotes the number of elements in A. We use the notation $\mathcal{P}(A)$ for the powerset of A, i.e. the set of all subsets of A. A function $w : [n] \to A$ is called a *string* over A, also denoted by $w = a_1 \ldots a_n$ with $w(i) = a_i$ for $i \in [n]$. The set A is also called an *alphabet* and strings over A are *words*. For a given string $w = a_1 \ldots a_n$, $|w| = n$ is the length of the string and $w(i) = a_i$ denotes the i-th symbol of the string. We denote by λ the empty string with $|\lambda| = 0$. The set of all strings over A is denoted by A^*.

For a given function $f : A \to B$, we may define the canonical extension to strings as the function $f^* : A^* \to B^*$ defined as $f^*(\lambda) = \lambda$, and $f^*(aw) = f(a)f^*(w)$ for $a \in A$ and $w \in A^*$. By convention if $A = \emptyset$, then $\emptyset^* = \{\lambda\}$ and the canonical extension to strings is also defined.

A relation $R \subseteq A \times A$ is called an *equivalence relation* if R is reflexive, symmetric and transitive. For $x \in A$, the set $\langle x \rangle = \{z \in A \mid xRz\}$ of all elements related to x by R is called the *equivalence class* of x. The set of all equivalence classes $A/R = \{\langle x \rangle \mid x \in A\}$ is the *quotient set* of A by R.

Let Σ be an alphabet. A *hypergraph* over Σ is a system $H = (V, E, att, lab, ext)$ where V is a set of *vertices*, E is a set of *hyperedges*, $att : E \to V^*$, called the *attachment function*, is a mapping that assigns a sequence of vertices to every hyperedge, $lab : E \to \Sigma$ is a mapping that assigns a *label* to every hyperedge, and $ext \in V^*$ is a sequence of *external vertices* of H. For notational convenience, the components of a hypergraph H will often be written with index H, i.e. $H = (V_H, E_H, att_H, lab_H, ext_H)$.

The length $|ext_H|$ is called the *type* of H, denoted also by $type(H)$. The class of all hypergraphs of type $n \in \mathbb{N}$ over Σ is denoted by $\mathcal{H}_{\Sigma,n}$. The class of *type-0* hypergraphs $\mathcal{H}_{\Sigma,0}$ is also denoted by $\mathcal{H}_\Sigma$.

Analogously, for $e \in E_H$, the length $|att_H(e)|$ is called the type of e. A hypergraph with all hyperedges of type 2 is an ordinary directed graph. To denote this special case, we use $\mathcal{G}_\Sigma$ and $\mathcal{G}_{\Sigma,n}$ instead of $\mathcal{H}_\Sigma$ and $\mathcal{H}_{\Sigma,n}$ respectively. If the alphabet is not important we use simply the notations $\mathcal{H}$ or $\mathcal{G}$. Sometimes, to represent unlabelled hypergraphs we use the alphabet $\Sigma = \{*\}$.

A hypergraph $H \in \mathcal{H}_{\Sigma,n}$ is a *subhypergraph* of a hypergraph $\overline{H} \in \mathcal{H}_{\Sigma,n}$, denoted by $H \subseteq \overline{H}$, if $V_H \subseteq V_{\overline{H}}$, $E_H \subseteq E_{\overline{H}}$, $att_H(e) = att_{\overline{H}}(e)$ and $lab_H(e) = lab_{\overline{H}}(e)$ for all $e \in E_H$, and $ext_H = ext_{\overline{H}}$.

Given two hypergraphs $H, H' \in \mathcal{H}_{\Sigma,n}$, a *hypergraph morphism* $f : H \to H'$ is a pair $f = (f_V, f_E)$ of functions $f_V : V_H \to V_{H'}$ and $f_E : E_H \to E_{H'}$ such that

we have $lab_H(e) = lab_{H'}(f_E(e))$ and $f_V^*(att_H(e)) = att_{H'}(f_E(e))$ for all $e \in E_H$, and $f_V^*(ext_H) = ext_{H'}$. The morphism is *injective* if the functions f_V, f_E are injective.

The following two operations on hypergraphs, namely disjoint union and merge, are similar to the hypergraph operations studied by Courcelle [8].

For two hypergraphs $H, \overline{H}$ we define the *disjoint union* denoted as $H + \overline{H}$ that yields a hypergraph $(V_H \uplus V_{\overline{H}}, E_H \uplus E_{\overline{H}}, att, lab, ext)$, where $\uplus$ denotes the disjoint union of sets,

$$att(e) = \begin{cases} att_H(e) \text{ if } e \in E_H, \\ att_{\overline{H}}(e) \text{ otherwise,} \end{cases} \quad lab(e) = \begin{cases} lab_H(e) \text{ if } e \in E_H, \\ lab_{\overline{H}}(e) \text{ otherwise,} \end{cases} \quad \text{and}$$

$$ext(i) = \begin{cases} ext_H(i) & \text{if } i \leq |ext_H|, \\ ext_{\overline{H}}(i - |ext_H|) & \text{otherwise} \end{cases} \quad \text{for } i \in [|ext_H| + |ext_{\overline{H}}|].$$

The disjoint union of hypergraphs is associative. It is commutative only if one component is of type 0, since the external sequence is the concatenation of the external sequences of the component hypergraphs.

For two hypergraphs $H, \overline{H} \in \mathcal{H}_{\Sigma,n}$ we denote by EXT the equivalence relation on $(V_H \uplus V_{\overline{H}}) \times (V_H \uplus V_{\overline{H}})$ induced by $ext_H(i) = ext_{\overline{H}}(i)$ for all $i \in [n]$. Then the *merge* operation $H \circ \overline{H}$ of H and $\overline{H}$ is defined by

$$H \circ \overline{H} = ((V_H \uplus V_{\overline{H}})/EXT, E_H \uplus E_{\overline{H}}, att, lab, \lambda) \in \mathcal{H}_\Sigma$$

where $att(e) = \begin{cases} att_H(e) \text{ if } e \in E_H, \\ att_{\overline{H}}(e) \text{ otherwise,} \end{cases}$ and $lab(e) = \begin{cases} lab_H(e) \text{ if } e \in E_H, \\ lab_{\overline{H}}(e) \text{ otherwise.} \end{cases}$

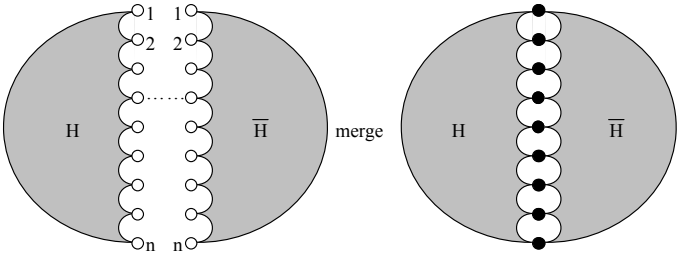

Fig. 1. The merging of H and $\overline{H}$

Figure 1 illustrates the merge operation. It should be noted that the merging is commutative, but not associative.

3 Contextual Hypergraph Grammars

In this section we introduce the new concept of contextual hypergraph grammars and their generated hypergraph languages. Besides a label alphabet and

a subalphabet of terminal labels, a contextual hypergraph grammar consists of two finite sets of hypergraphs: a set of axioms and a set of contexts, as well as an operator that specifies how each context can be used to derive hypergraphs from hypergraphs. Moreover, various examples are given to illustrate the derivation mechanism of contextual hypergraph grammars.

Definition 1 (Contextual Hypergraph Grammar). *A* contextual hypergraph grammar *is a system* $CHG = (\Sigma, T, \mathcal{A}, \mathcal{C}, \Theta)$ *where* Σ *is a finite set of* labels, $T \subseteq \Sigma$ *is a set of* terminal labels, $\mathcal{A} \subset \mathcal{H}_\Sigma$ *is a finite set of* axioms, $\mathcal{C}$ *is a finite set of hypergraphs of various types called* hypergraph contexts, *and* $\Theta = (\Theta_C)_{C \in \mathcal{C}}$ *is a family of mappings where every* $\Theta_C : \mathcal{H}_\Sigma \to \mathcal{P}(\mathcal{H}_{\Sigma, type(C)})$ *is a* selection function *for every hypergraph context* $C \in \mathcal{C}$. *The elements of* $\Theta_C(H)$ *are called* variants *of the hypergraph* H.

A derivation relation *is defined on* $\mathcal{H}_\Sigma \times \mathcal{H}_\Sigma$ *and the hypergraph* G directly derives *the hypergraph* H, *denoted by* $G \Rightarrow H$, *if there are* $C \in \mathcal{C}$ *and* $G' \in \Theta_C(G)$ *such that* $H = C \circ G'$, *i.e.* H *is a merging of a variant of* G *with some hypergraph context. We denote by* $\Rightarrow^*$ *the reflexive and transitive closure of the derivation relation* $\Rightarrow$. *The* language *generated by a contextual hypergraph grammar* $CHG = (\Sigma, T, \mathcal{A}, \mathcal{C}, \Theta)$ *consists of all terminal hypergraphs derived from some axiom, i.e.* $L(CHG) = \{H \in \mathcal{H}_T \mid Z \Rightarrow^* H \text{ for } Z \in \mathcal{A}\}$.

In this paper, we assume that the function Θ is computable, so that the generated languages are recursively enumerable. It should be noted that we assume terminal labels in contrast to the usual definition of centextual grammars in the string case. But this allows us more flexibility from the very beginning. The task of the Θ function is to provide variants of a type-0 hypergraph that can be merged with a chosen context. While in all following examples the originals and their variants are closely related, the very general definition of Θ admits also much more sophisticated constructions.

Example 1 (All Graphs). As a first example, we define a contextual hypergraph grammar that generates the set of all directed (unlabeled) graphs:

$$CHG_{all} = (\{*\}, \{*\}, \{empty\}, \{Vertex, Edge\}, \Theta_{all})$$

where *empty* denotes the empty graph and $Vertex$, $Edge$ and Θ_{all} are given as follows.

1. $Vertex$ is the type-0 graph with a single vertex without edges.
2. $Edge$ is the graph with two vertices and a connecting edge whose attachment defines also the sequence of external vertices.
3. The only $Vertex$-variant of a hypergraph is the hypergraph itself, that is $\Theta_{all,Vertex}(H) = \{H\}$.
4. The $Edge$-variants of a type-0 hypergraph are given by all choices of a sequence of two distinct vertices, i.e. $\Theta_{all,Edge}(H) = \{(H, v_1 v_2) \mid v_1, v_2 \in V_H,\ v_1 \neq v_2\}$ where $(H, v_1 v_2) = (V_H, E_H, att_H, lab_H, v_1 v_2)$.

Given a hypergraph H (of type 0), $Vertex \circ H$ adds a single vertex disjointly to H, and $Edge \circ (H, v_1 v_2)$ adds a new edge to H connecting v_1 and v_2 for each choice

of v_1v_2. If a graph G has n vertices, V_G can be generated by $Vertex^n \circ empty$ up to the naming of vertices. Then every edge of G can be added successively, connecting the proper vertices by means of the context $Edge$. In other words, CHG_{all} generates the set of all graphs.

Example 2 (Eulerian Graphs). Our second example is a contextual hypergraph grammar that generates the sets of all Eulerian graphs (where a graph is Eulerian if it has a cycle passing each edge exactly once):

$$CHG_{Euler} = (\{*\}, \{*\}, \{2Cycle^0\}, \{Vertex, 2Cycle, 2Path\}, \Theta_{Euler})$$

where $Vertex$ is the same graph as in the previous example, $2Cycle^0$ is the type-0 graph consisting of a cycle of length 2, $2Cycle$ is the same graph with its two vertices as external vertices, and $2Path$ is a graph with three vertices, say v_1, v_2, and v_3, two edges, one from v_1 to v_2 and the other from v_2 to v_3, and $v_1v_2v_3$ as sequence of external vertices. Θ_{Euler} is defined as follows.

1. $\Theta_{Euler,Vertex} = \Theta_{all,Vertex}$,
2. $\Theta_{Euler,2Cycle}(H) = \{(H, v_1v_2) \in \Theta_{all,Edge} \mid v_1 \text{ or } v_2 \text{ not isolated}\}$,
3. $\Theta_{Euler,2Path}(H) = \{(H - e, v_1v_2v_3) \mid e \in E_H,\ v_1, v_2, v_3 \text{ pairwise distinct}\}$ where $H - e$ is the subhypergraph of H obtained by removing the hyperedge e, and $(H-e, v_1v_2v_3)$ is $H-e$ with $v_1v_2v_3$ instead of λ as sequence of external vertices.

It is not difficult to see that the merging of $Vertex$, $2Cycle$ or $2Path$ with an admitted variant of the axiom or a derived graph preserves connectivity up to isolated vertices as well as the property that indegree equals outdegree for each vertex. Conversely, every graph with these properties can be obtained from $2Cycle^0$ by a sequence of such mergings. Altogether, the grammar generates the language of Eulerian graphs according to the well-known characterization of Eulerian graphs by these two properties.

In a similar manner, we can generate the sets of all Hamiltonian graphs or all non-Hamiltonian graphs.

Example 3 (Square Grids Graphs).

$$CHG_{SG} = (\{a, c, N\}, \{a, c\}, \{E, F\}, \{Stop, Cont, Corner, Tile\}, \Theta_{SG})$$

In Figure 2 we see the grammar's hypergraphs. Θ_{SG} is defined as follows.

1. $\Theta_{SG,Stop}(H) = \{(V_H, E_H \setminus \{e_1\}, att_H|_{E_H\setminus\{e_1\}}, lab_H|_{E_H\setminus\{e_1\}}, v_1v_2v_4) \mid (v_1, v_2, v_3, v_4 \in V_H, \text{ pairwise distinct nodes}),\ e_1 \in E_H, condStop(H, v_1, v_2, v_3, v_4, e_1)\}$; $condStop(H, v_1, v_2, v_3, v_4, e_1)$ is a boolean function that is *true* when the following conditions hold:
   - $att_H(e_1) = v_2v_1$, $lab_H(e_1) = N$,
   - $\exists$ edges $e_2, e_3 \in E_H$, $att_H(e_2) = v_2v_3$, $lab_H(e_2) = a$, $att_H(e_3) = v_3v_4$, $lab_H(e_3) = a$,

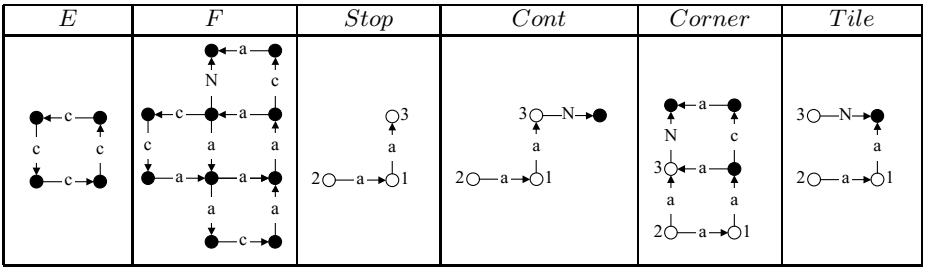

Fig. 2. CHG_{SG} hypergraphs

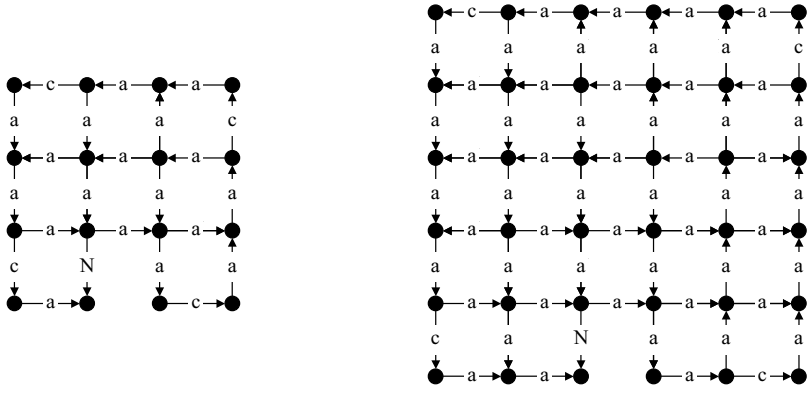

Fig. 3. $\Theta_{SG,Stop}$ returns nonempty sets for these hypergraphs

- $|\{e \in H \mid att_H(e) \in V_H^* v_3 V_H^*, lab_H(e) = a\}| = 4$,
- $|\{e \in H \mid att_H(e) \in V_H^* v_4 V_H^*\}| = 2$.

In order to understand the conditions checked by $\Theta_{SG,Stop}$ we present in Figure 3 two hypergraphs containing vertices and edges such that *condStop* is true.

2. $\Theta_{SG,Cont} = \Theta_{SG,Stop}$.
3. $\Theta_{SG,Corner}(H) = \{(V_H, E_H \setminus \{e_1, e_2\}, att_H|_{E_H \setminus \{e_1,e_2\}}, lab_H|_{E_H \setminus \{e_1,e_2\}}, v_1v_2v_3) \mid (v_1, v_2, v_3 \in V_H$ pairwise distinct vertices), $(e_1, e_2 \in E_H$, distinct edges), $att_H(e_1) = v_2v_1$, $lab_H(e_1) = N$, $att_H(e_2) = v_2v_3$, $lab_H(e_2) = c\}$.
4. $\Theta_{SG,Tile}(H) = \{(V_H, E_H \setminus \{e_1\}, att_H|_{E_H \setminus \{e_1\}}, lab_H|_{E_H \setminus \{e_1\}}, v_1v_2v_4) \mid (v_1, v_2, v_3, v_4 \in V_H$ pairwise distinct nodes), $e_1 \in E_H$, NOT $condStop(H, v_1, v_2, v_3, v_4, e_1)$, $att_H(e_1) = v_2v_1$, $lab_H(e_1) = N$, $\exists$ an edge $e_2 \in E_H$, $att_H(e_2) = v_2v_3$, $lab_H(e_2) = a\}$.

It is not difficult to see the existence of only one edge labelled by N to all derived hypergraphs except the ones from the language as an invariant for all the derivations. Also all nonempty Θ sets contain hypergraphs with the external nodes in the neighborhood of the edge labelled by N.

Our example actually generates only odd sizes square grids; to get all even square grids we should add the corresponding axioms.

The graph languages generated by Eulerian, Hamiltonian, Square Grid Grammars are particularly interesting because they cannot be generated by any context-free (hyper)graph grammar (c.f., e.g., [5, 6] where the generative power of hyperedge replacement grammars is discussed).

4 Simulation Results

In this section, we show that contextual hypergraph grammars generalize total contextual grammars on strings. Moreover, contextual hypergraph grammars can simulate in a natural way two well-known types of graph grammars: hyperedge replacement grammars [5, 6], which provide a context-free generation mechanism for (hyper)graph sets, and hypergraph grammars in the double-pushout approach [7], which is one of the most frequently used graph transformation frameworks.

4.1 Contextual String-Hypergraph Grammars

The notion of contextual grammars evolved starting from the initial paper proposed by Solomon Marcus up to the current definition [1].

Definition 2 (Total Contextual Grammars). *A (string)* total contextual grammar *is a construct* $TCG = (\Sigma, A, C, \varphi)$*, where* Σ *is an alphabet,* A *is a finite subset of* Σ^**,* C *is a finite subset of* $\Sigma^* \times \Sigma^*$ *and* $\varphi : \Sigma^* \times \Sigma^* \times \Sigma^* \to \mathcal{P}(C)$*. The elements of* A *are called* axioms*, the elements of* C *are called* contexts*, and* φ *is a* choice function*.*

A derivation relation $\underset{TC}{\Rightarrow}$ *is defined as* $x \underset{TC}{\Rightarrow} y$ *if and only if* $x = x_1x_2x_3$*,* $y = x_1ux_2vx_3$*, for* $x_1, x_2, x_3 \in \Sigma^*$*, and* $(u, v) \in C$*, s.t.* $(u, v) \in \varphi(x_1, x_2, x_3)$*. The transitive closure of the relation* $\underset{TC}{\Rightarrow}$ *is denoted by* $\overset{*}{\underset{TC}{\Rightarrow}}$*.*

The generated language is $L(TCG) = \{w \in \Sigma^* \mid a \overset{*}{\underset{TC}{\Rightarrow}} w$*, for* $a \in A\}$*.*

In order to relate string contextual grammars with contextual hypergraph grammars, we need the notion of a *string hypergraph* $\overrightarrow{s}$ for a string s so that a string language L can be considered as a hypergraph language $\overrightarrow{L}$.

We will use the "$\cdot$" operator to denote the concatenation of numeric symbols.

Definition 3 (string-hypergraph). *Let us consider an alphabet* Σ *and a finite nonempty string* $s = a_1 \dots a_n$ *over* Σ*.*

A string-hypergraph context *denoted by* $\underline{s} \in \mathcal{H}_{\Sigma,n}$ *is the hypergraph* $([n], \{ec_i \mid i \in [n]\}, (att(ec_i) = i,$ *for all* $i \in [n]), (lab(ec_i) = a_i,$ *for all* $i \in [n]), 1 \cdot \ \dots \ \cdot n)$ *with all its vertices as external nodes; and let* $\underline{\lambda} = empty$*.*

A string-hypergraph *denoted by* $\overrightarrow{s} \in \mathcal{H}_\Sigma$ *is the hypergraph* $([n+1], \{ec_i \mid i \in [n]\} \cup \{es_i \mid i \in [n]\}, (att(ec_i) = i, att(es_i) = i \cdot (i+1),$ *for all* $i \in [n]), (lab(ec_i) = a_i, lab(es_i) = *,$ *for all* $i \in [n]))$ *having no external nodes but some sequential edges in order to reconstruct a string form a string-hypergraph.* $\overrightarrow{\lambda} = Vertex$*.*

For a given string language L *we denote by* $\overrightarrow{L} = \{\overrightarrow{s} \mid s \in L\}$*.*

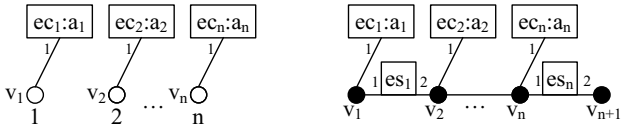

Fig. 4. A string-hypergraph context and a string-hypergraph

We can see a representation of a string-hypergraph context and a string-hypergraph in Figure 4, the rectangles are the hyperedges with their names and labels, while the numbers next to them represent the attachments to nodes.

A total contextual (string) grammar $TCG = (\Sigma, Ax, Ctx, \varphi)$ can be transformed into a contextual hypergraph grammar $CHG(TCG) = (\Sigma,\ \Sigma,\ \mathcal{A},\ \mathcal{C},\ \Theta_{TCG})$ in the following way. $\mathcal{A} = \{\overrightarrow{a} \mid a \in Ax\}$ that is the axioms are transformed into string-hypergraphs, $\mathcal{C} = \{\underline{u}+\underline{w} \mid (u,w) \in Ctx\}$, that is each context is transformed into the disjoint union of the left and right string-hypergraph contexts. The function $\Theta_{TCG,\underline{u}+\underline{w}}(\overrightarrow{xyz})$ takes a string-hypergraph and prepares it for the merging operation with a string-hypergraph context. Figure 5 shows the Θ-preparation, where the *ec* hyperedges took labels from the original string.

$\Theta_{TCG,\underline{u}+\underline{w}}(\overrightarrow{xyz}) = \{([|xuywz|+1], EC \cup ES, att, lab, (|x|+1)\cdot\ \ldots\ \cdot|xu|\cdot(|xuy|+1)\cdot\ \ldots\ \cdot|xuyw|) \mid$ for $(u,w) \in Ctx$, $x,y,z \in \Sigma^*, s.t.(u,w) \in \varphi(x,y,z)\}$, where

- $ES = \{es_i \mid i \in [|xuywz|]\}$,
- $EC = \{ec_i \mid i \in [|xyz|]\}$,
- $att(es_i) = i \cdot (i+1)$, $lab(es_i) = *$, for i in $[|xuywz|]$,
- $att(ec_i) = i$ for $1 \leq i \leq |x|$, $att(ec_i) = i + |u|$ for $|x|+1 \leq i \leq |xy|$, $att(ec_i) = i + |uw|$ for $|xy|+1 \leq i \leq |xyz|$, $lab(ec_i) = xyz(i)$ for i in $[|xyz|]$.

Using this construction, it is easy to prove the following theorem.

Theorem 1. *Let TCG be a total contextual (string) grammar and $CHG(TCG)$ the corresponding contextual hypergraph grammar. Then they generate the same string-hypergraph language, i.e. $\overrightarrow{L(TCG)} = L(CHG(TCG))$.*

4.2 Hyperedge Replacement Grammars

A *hyperedge replacement grammar* is a system $HRG = (N, T, P, S)$ where $N \subseteq \Sigma$ is a set of *nonterminal labels*, $T \subseteq \Sigma$ is a set of *terminal labels*, P is a set of

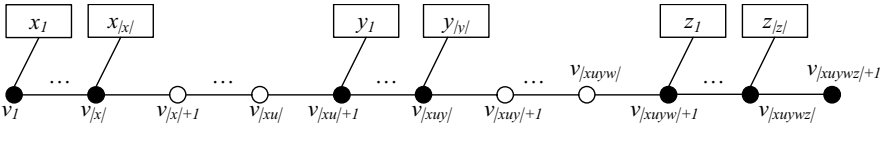

Fig. 5. An object returned by the function Θ_{TCG}

rules having the form $(A ::= R)$ with $A \in N$ and $R \in \mathcal{H}_{\Sigma,n}$ for some $n \in \mathbb{N}$, and $S \in N$ is a start symbol.

For technical simplicity, we assume that the nonterminals are *typed*, meaning that there is a mapping $type : N \to \mathbb{N}$ subject to the following conditions:

i) $type(A) = type(R)$ for each $(A ::= R) \in P$,
ii) $type(e) = type(lab_R(e))$ for each $e \in E_R$ with R being the right-hand side of some rule $(A ::= R) \in P$.

Let $H \in \mathcal{H}_\Sigma$ and $e \in E_H$ with $type(e) = type(lab_H(e))$. Then H *derives directly* $\overline{H}$ through $(lab_H(e) ::= R) \in P$ if $\overline{H} = (H - e) \circ R$. Note that the merging of $(H - e)$ and R is always defined because the types of e and R are equal and e transfers its type to $(H - e)$. Here $H - e$ denotes the *removal* of e from H yielding the hypergraph $H - e = (V_H, E_H \setminus \{e\}, att, lab, att_H(e))$ where att and lab are the restrictions of att_H and lab_H, respectively, to the set $E_H \setminus \{e\}$.

A direct derivation of $\overline{H}$ from H is denoted by $H \to \overline{H}$ and the reflexive and transitive closure of this relation by $\to^*$.

The language generated by a hyperedge replacement grammar $HGR = (N, T, P, S)$ consists of all terminal hypergraphs derivable from the start handle, *i.e.* $L(HRG) = \{H \in \mathcal{H}_T \mid S^\circ \to^* H\}$. Here, a *handle* of a nonterminal $A \in N$ denotes the hypergraph $A^\circ = ([type(A)], \{e_0\}, att, lab, 1 \dots type(A))$ with $att(e_0) = 1 \dots type(A)$ and $lab(e_0) = A$.

A hyperedge replacement grammar $HRG = (N, T, P, S)$ can be translated into a contextual hypergraph grammar $CHG(HRG) = (N \cup T, T, \{S^\circ\}, \{R \mid (A ::= R) \in P\}, \Theta_{HRG})$ with $\Theta_{HRG,R}(H) = \{H - e \mid e \in E_H,\ (lab_H(e) ::= R) \in P\}$ such that $CHG(HRG)$ generates the same language as HRG.

Theorem 2. *Let HRG be a hyperedge replacement grammar and $CHG(HRG)$ the corresponding contextual hypergraph grammar. Then HRG and $CHG(HRG)$ generate the same language, i.e. $L(HRG) = L(CHG(HRG))$.*

4.3 Hypergraph Grammars

In this subsection, we introduce hypergraph grammars in the so-called double-pushout approach. They generalize hyperedge replacement grammars in that a direct derivation does not replace a hyperedge only, but a subgraph (up to external nodes) which is a matching of a left-hand side of a rule.

A *hypergraph grammar* is a system $HG = (T, P, Z)$ where $T \subseteq \Sigma$ is a set of *terminal labels*, P is a finite set of *rules* of the form $L \supseteq K \subseteq R$ with $L, K, R \in \mathcal{H}_\Sigma$, and $Z \in \mathcal{H}_\Sigma$ is an *axiom*.

Without loss of generality, we may assume that the *gluing hypergraph* K of each rule $L \supseteq K \subseteq R$ is totally disconnected, i.e. $E_K = \emptyset$, and $V_K = [n]$ for some $n \in \mathbb{N}$. Therefore, the rule can be represented by the pair $(L, 1 \dots n) ::= (R, 1 \dots n)$.

A hypergraph $H \in \mathcal{H}_\Sigma$ *directly derives* a hypergraph $\overline{H} \in \mathcal{H}_\Sigma$ through the application of the rule $(L, 1 \dots n) ::= (R, 1 \dots n)$ if there is a hypergraph morphism $g : L \to H$ such that

(1) $att_H(e) \in (V_H \setminus (g_V(V_L) \setminus g_V([n])))^*$ for all $e \in E_H \setminus g_E(E_L)$,
(2) g_E is injective, and $g_V(v) = g_V(v')$ for $v \neq v'$ implies $v, v' \in [n]$,
(3) $X = (V_H \setminus (g_V(V_L) \setminus g_V([n]))), E_H \setminus g_E(E_L), att_X, lab_X, g_V(1) \dots g_V(n))$ with $att_X(e) = att_H(e)$ and $lab_X(e) = lab_H(e)$ for all $e \in E_H \setminus g_E(E_L)$, and
(4) $\overline{H} = X \circ (R, 1 \dots n)$.

A direct derivation from H to $\overline{H}$ through the rule r is denoted by $H \underset{P}{\rightarrow} \overline{H}$ if $r \in P$. The reflexive and transitive closure of the relation $\underset{P}{\rightarrow}$ is denoted by $\overset{*}{\underset{P}{\rightarrow}}$.

Let $HG = (T, P, Z)$ be a hypergraph grammar. Then the *generated language* $L(HG)$ contains all terminal hypergraphs derivable from the axiom through given rules *i.e.* $L(HG) = \{H \in \mathcal{H}_T \mid Z \overset{*}{\underset{P}{\rightarrow}} H\}$.

It should be noted that the hypergraphs H and $\overline{H}$ are *pushouts* of the intermediate hypergraph X and the left-hand side L respectively the right-hand side R using the gluing hypergraph K. H and $\overline{H}$ remain invariant whether K is totally disconnect or has got hyperedges. In the $double - pushout$ approach, the hypergraphs X and $\overline{H}$ are usually constructed as *pushout complement* and *pushout* respectively. We prefer the given version because it is easier related to contextual hypergraph grammars.

A hypergraph grammar $HG = (T, P, Z)$ can be transformed into a contextual hypergraph grammar $CHG(HG) = (\Sigma, T, \{Z\}, \mathcal{C}_{HG}, \Theta_{HG})$ where $\mathcal{C}_{HG} = \{(R, 1 \dots n) \mid ((L, 1 \dots n) ::= (R, 1 \dots n)) \in P\}$ and $\Theta_{HG,(R,1\dots n)}(H)$ contains all hypergraphs X that are constructed as in Point 3 above for all rules $((L, 1 \dots n) ::= (R, 1 \dots n)) \in P$ and hypergraph morphisms $g : L \rightarrow H$ that fulfil Points 1 and 2.

We get $H \Rightarrow \overline{H}$ with $\overline{H} = (R, 1 \dots n) \circ X$ for some $(R, 1 \dots n) \in \mathcal{C}_{HG}$ and $X \in \Theta_{HG,(R,1\dots n)}(H)$ if and only if $H \underset{P}{\rightarrow} \overline{H}$. This proves the following theorem.

Theorem 3. *Let HG be a hypergraph grammar and $CHG(HG)$ the corresponding contextual hypergraph grammar. Then HG and $CHG(HG)$ generate the same language, i.e. $L(HG) = L(CHG(HG))$.*

It is known that hypergraph grammars in the double-pushout approach generate all recursively enumerable hypergraph languages. Using the theorem, this holds for our new concept of contextual hypergraph grammars, too.

5 Conclusion

We have introduced a new type of hypergraph grammars, namely contextual hypergraph grammars, that generalize in a natural way the contextual string grammars. Using the power of our formalism, we are able to simulate already existing devices for the generation of hypergraph languages, like hyperedge replacement grammars and hypergraph grammars in the double-pushout approach. Our approach is useful to model operations with annotated documents even with multiple structures including syntactic-semantic structures, dependencies, multilingual information, etc. Further investigations are needed to study possible

hierarchies within the generated languages. Also, the type and complexity of the Θ function deserve special attention. Possible classes to be studied are arbitrary functions, NP, or P. Furthermore, types of specifying a matching condition may be distinguished, such as global checking, local checking, without labels, without edges limitation, with a maximum number of connections, etc.

Acknowledgements

We would like to thank Victor Mitrana, Claudio Moraga, and the anonymous referees for their helpful comments and observations.

References

1. Păun, G.: Marcus Contextual Grammars. Kluwer Academic Publishers, Norwell, MA, USA (1997)
2. Kudlek, M., Martín-Vide, C., Mateescu, A., Mitrana, V.: Contexts and the concept of mild context-sensitivity. Linguistics and Philosophy (26) (2002) 703–725
3. Marcus, S., Păun, G., Martín-Vide, C.: Contextual grammars as generative models of natural languages. Comput. Linguist. **24**(2) (1998) 245–274
4. Kappes, M.: Combining contextual grammars and tree adjoining grammars. Grammars, A Journal of Mathematical Research on Formal and Natural Languages **3**(2-3) (2000) 175–187
5. Habel, A.: Hyperedge replacement: Grammars and languages. LNCS **643** (1992)
6. Drewes, F., Kreowski, H.J., Habel, A.: Hyperedge replacement graph grammars. In: Handbook of graph grammars and computing by graph transformation: volume I. foundations. World Scientific Publishing (1997) 95–162
7. Corradini, A., Montanari, U., Rossi, F., Ehrig, H., Heckel, R., Löwe, M. In: Algebraic approaches to graph transformation. Part I: basic concepts and double pushout approach. World Scientific, River Edge, NJ, USA (1997) 163–245
8. Courcelle, B.: The expression of graph properties and graph transformations in monadic second-order logic. In Rozenberg, G., ed.: Handbook of graph grammars and computing by graph transformations, vol. 1: Foundations. World Scientific, New-Jersey, London (1997) 313–400