

## Chapter 11

---

# Recursive Matrix Systems (RMS)

## A highly parameterizable formal rewriting system

DOMINIK HECKMANN

**ABSTRACT.** A new formal rewriting system is defined. It is capable of describing regular, context-free, mildly-context-sensitive and context-sensitive languages as well as languages in between. It allows to compare formal language properties in one uniform framework. The main idea of RMS is adding a vertical dimension to the usual string deriving grammars.

## Introduction

Small extensions of context-free languages (CFL) are interesting for natural language processing. In [Joshi et al,1991] mildly-context-sensitive languages are characterized and tree-adjoining languages (TAL) are described as a representative of this family.

The main subject of this work is the introduction and definition of "Recursive Matrix Systems" (RMS), a newly developed formalism capable of describing mildly-context-sensitive languages. We present a restricted recursive matrix system which is slightly stronger than context-free grammars, in such a way that it allows cross-serial and nested dependencies together. But it is still a proper subset of tree-adjoining grammars.

In a first example the idea of using matrices is motivated. RMS, the derivation and the interpretation function are defined. Some properties of RMS and comparisons with other formalisms are shown. A relation between tree-adjoining grammars and recursive matrix systems is demonstrated.

## Motivation

$S \rightarrow \begin{bmatrix} a \\ b \\ c \end{bmatrix} S$  applied as a regular rule creates matrices like  $\begin{bmatrix} a \\ b \\ c \end{bmatrix} \begin{bmatrix} a \\ b \\ c \end{bmatrix} \begin{bmatrix} a \\ b \\ c \end{bmatrix} \dots$

If the letters a,b and c are read row by row strings of the form  $aaa...bbb...ccc...$  are generated. The language  $L_1 = \{a^n b^n c^n | n \in \mathbb{N}\}$  is described by this simple rule (together with an adequate termination condition).  $L_1$  is context-sensitive but the application of the rule is independent from the context. Matrices are created column by column but read row by row.

## The Definition of Recursive Matrix Systems

Definition: A **Recursive Matrix System (RMS)** is a tuple  $\langle G, I \rangle$  where  $G$  is an arbitrary grammar formalism to create recursive matrices and  $I$  is an interpretation function to read a string out of each recursive matrix.



Definition: A **Recursive Matrix** is a matrix consisting of terminal symbols or again recursive matrices as elements.

|   |   |   |   |   |   |   |   |   |   |   |   |
|---|---|---|---|---|---|---|---|---|---|---|---|
| a | a | $\epsilon$  |   |   |   |   |   |   |   |   |   |
| b | b | <table border="1"><tr><td>d</td><td>d</td><td>d</td></tr><tr><td>e</td><td>e</td><td>e</td></tr><tr><td>f</td><td>f</td><td>f</td></tr></table> | d | d | d | e | e | e | f | f | f |
| d | d | d   |   |   |   |   |   |   |   |   |   |
| e | e | e   |   |   |   |   |   |   |   |   |   |
| f | f | f   |   |   |   |   |   |   |   |   |   |
| c | c | $\epsilon$  |   |   |   |   |   |   |   |   |   |

For example 

|   |   |   |
|---|---|---|
| d | d | d |
| e | e | e |
| f | f | f |

 has the recursive matrix 

|   |   |   |
|---|---|---|
| d | d | d |
| e | e | e |
| f | f | f |

 as an element.

Definition: A **Recursive Matrix Grammar** is the four-tuple  $\langle T, N, S, P \rangle$ . Let  $T, N, P$  be finite but non-empty sets where  $T$  is the *terminal-alphabet*,  $N$  is the *nonterminal-alphabet* with  $N \cap T = \emptyset$  and  $S \in N$  is the *starting variable*.  $P$  is the *set of production rules* (not necessarily in a normal form):

$$\left. \begin{array}{l} A \rightarrow \beta \\ A \rightarrow \beta C \\ A \rightarrow \beta C \delta \\ A \rightarrow \beta C D \\ AB \rightarrow CD \end{array} \right\} \text{regular} \left. \right\} \text{linear} \left. \right\} \text{context-free}^1 \left. \right\} \text{context-sensitive}^2$$

The recursive matrix grammar is either **regular**, **linear**, **context-free** or **context-sensitive**. The difference between the normal string-deriving formalisms and the recursive matrix grammars is:  $\beta$  and  $\delta$  are not just terminals but vertical vectors with terminals or non-terminals as elements.

$$\beta, \delta \in \left\{ \begin{bmatrix} x_1 \\ \vdots \\ x_k \end{bmatrix} \mid x_i \in N \cup T \cup \{\epsilon\} \text{ for all } 1 \leq i \leq k \in \mathbb{N} \right\}$$

The grammar is called **lexicalized** if at least one of the  $x_i$  in every produc-

<sup>1</sup>This is the "minimal" Greibach normalform [Schöning,1995] p.54

<sup>2</sup>Context-sensitive grammars are only mentioned for completeness.

tion rule is a terminal symbol. It is called **terminalized** if all the  $x_i$  are terminal symbols. If all vectors have the same size  $k$  the grammar has the **index  $k$** . It reflects the number of rows in the matrix. Here we consider only systems with index  $k$  grammars and denote them  $RM S_k$ .

## The Derivation of Recursive Matrices

Let  $RM$  be the set of all recursive matrices with terminals or recursive matrices as elements. Let  $ERM$  be the set of all extended recursive matrices with terminals, nonterminals or extended recursive matrices as elements.

The derivation-step relation is described informally: In each step an arbitrary nonterminal is locally replaced by the right-hand-side of an adequate production rule. The surrounding submatrices are not changed.

A complete derivation of a recursive matrix  $r$  is a list of extended recursive matrixes where all neighbours hold the derivation-step relation.

$$\boxed{S} \Rightarrow e_1 \Rightarrow e_2 \Rightarrow \dots \Rightarrow r \quad , \text{with all } e_i \in ERM \text{ and } r \in RM$$

The set of recursive matrices which can be derived by a grammar  $G$  is:

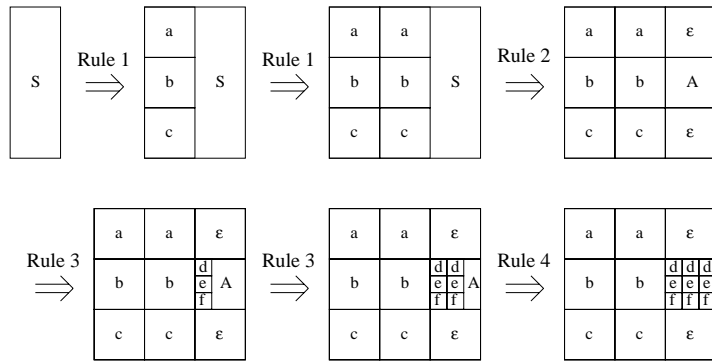
$$\boxed{L(G) := \{r \mid S \xRightarrow{*} r, r \in RM\}}$$

The following example shows the derivation process:

$$G_1 = \langle \{a,b,c,d,e,f\}, \{S,A\}, S, \{S \rightarrow \begin{bmatrix} a \\ b \\ c \end{bmatrix} S, S \rightarrow \begin{bmatrix} \epsilon \\ A \\ \epsilon \end{bmatrix}, A \rightarrow \begin{bmatrix} d \\ e \\ f \end{bmatrix} A, A \rightarrow \begin{bmatrix} d \\ e \\ f \end{bmatrix} \} \rangle$$

When applying the first or third of the four rules a vector is added to the actual matrix. When applying the second rule a descend into the next "matrix-level" takes place. Only the last rule is a terminating one. All vectors have the size 3 and all rules are regular.  $G_1$  is an "index-3-regular grammar".

A possible derivation with the grammar  $G_1$ :



Remark: the horizontal dimension of the recursive matrices are not limited. It happened to occur only 3x3 recursive matrices in this example.



Two different languages are generated in this example:

$$RML_3(G_1, \vec{\Rightarrow}) = \{a^n b^n d^p e^p f^p c^n \mid n, p \in \mathbb{N}\}$$

$$RML_3(G_1, \overleftarrow{\Rightarrow}) = \{a^n d^p e^p f^p b^n c^n \mid n, p \in \mathbb{N}\}$$

## RMS in Comparison with CFG, TAG and LCFRS

- According to [Schöning,1995] each CFG can be transformed into an equivalent CFG with rules of the form  $A \rightarrow bCD$  only. Every rule of that type can directly be transformed into the RMS rule of the form:  $A \rightarrow \begin{bmatrix} b \\ D \end{bmatrix} C$ .

$$CFL = RML_2(\text{reg}\&\text{lex}, \vec{\rightarrow})$$

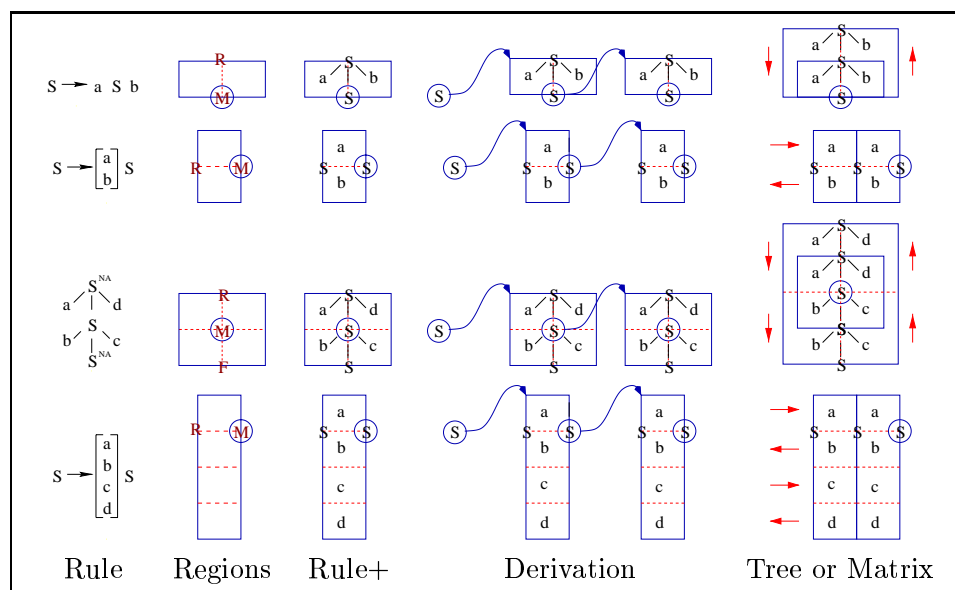
- TAGs that are limited to allow auxiliary trees only which consist of a root and a foot which must not be adjoined to, and only one middle knot that may be adjoined to are weaker than normal TAGs:

$$TAL(\overset{\vec{\rightarrow}}{\underset{\vec{\leftarrow}}{\text{reg}}}) = RML_4(\text{reg}, \overset{\vec{\rightarrow}}{\underset{\vec{\leftarrow}}{\text{cf}}}) \subset TAL = RML_2(\text{cf}, \vec{\rightarrow})$$

- A description of linear context-free rewriting systems can be found in [Becker,1994]. The recursive matrix structure and the interpretation function of the RMS(reg), RMS(lin) and RMS(cf) can directly be translated into index-functions and sort-functions of the LCFRS formalism.

$$RML_k(\text{reg}) \subset RML_k(\text{lin}) \subset RML_k(\text{cf}) \subset LCFRS$$

Here are some illustrations on *CFG*,  $RML_2(\text{reg})$ , *TAG* and  $RMS_4(\text{reg})$ :



These pictures above exhibit the construction of RMS from CFG and TAG.

## Recursive Matrix Languages

These languages are defined to help distinguishing between the RMSs:

|   |   |
|---|---|
| $Copy(k)$   | $:= \underbrace{\{ww\dots\}}_{k\text{-times}} \mid k \in \mathbb{N}, w \in T^*$   |
| $Count(k)$  | $:= \underbrace{\{a^n b^n \dots\}}_{k\text{-times}} \mid k, n \in \mathbb{N}$   |
| $Depend\left(\begin{smallmatrix} d_1 \\ \vdots \\ d_k \end{smallmatrix}\right)$ | $:= \underbrace{\{(a_1..a_n)^{d_1} (b_1..b_n)^{d_2} \dots\}}_{k\text{-times}} \mid k, n \in \mathbb{N}, \text{all } d_i \in \{\rightarrow, \leftarrow\},$<br>$(a_1..a_n)^\rightarrow = a_1..a_n, (a_1..a_n)^\leftarrow = a_n..a_1 \}$ |

$RML_3(reg, \begin{smallmatrix} \rightarrow \\ \leftarrow \end{smallmatrix})$  for example can be specified by the subsets:

$$\begin{aligned} Count(3) &= \{a^n b^n c^n \mid n \in \mathbb{N}\} \\ Copy(2) &= \{ww \mid w \in T^*\} \\ Depend\left(\begin{smallmatrix} \rightarrow \\ \leftarrow \end{smallmatrix}\right) &= \{a_1..a_n b_n..b_1 c_1..c_n \mid n \in \mathbb{N}\} \end{aligned}$$

$$\begin{aligned} Depend(\rightarrow) &= \{a_1..a_n b_1..b_n \mid n \in \mathbb{N}\} = \text{cross-serial dependency} \\ Depend(\leftarrow) &= \{x\{a_1..a_n b_n..b_1 \mid n \in \mathbb{N}\} = \text{nested dependency} \end{aligned}$$

$RML_3(reg, \begin{smallmatrix} \rightarrow \\ \leftarrow \end{smallmatrix})$  is between *CFL* and *TAL*:

|  |
|--|
| $CFL = RML(reg, \begin{smallmatrix} \rightarrow \\ \leftarrow \end{smallmatrix}) \subset RML_3(reg, \begin{smallmatrix} \rightarrow \\ \leftarrow \end{smallmatrix}) \subset RML(reg, \begin{smallmatrix} \rightarrow \\ \leftarrow \end{smallmatrix}) \subset RML(cf, \begin{smallmatrix} \rightarrow \\ \leftarrow \end{smallmatrix}) = TAL$ |
|--|

Remark:  $RML_3(reg, \begin{smallmatrix} \rightarrow \\ \leftarrow \end{smallmatrix})$  should be analysed for linguistic relevance.

## Polynomial Parsing

RMS(reg), RMS(lin) and RMS(cf) have the property that they can be parsed in polynomial time. This property is inherited from the fact, that a RMS(cf) can be directly transformed into a weak equivalent LCFRS. Additionally the

following upper time bounds hold:

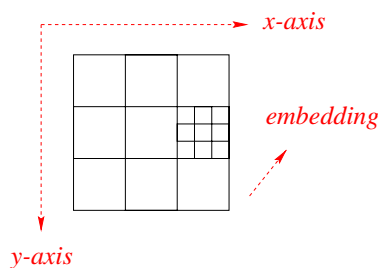
|  |                          |
|--|--------------------------|
| $RMS_k(reg&ter) = O(n)$  | conjecture               |
| $RMS(reg, \vec{\rightarrow}) = O(CFG) \leq \frac{n^3}{\log n}$ | [Harrison,1978]          |
| $RMS(reg, \vec{\leftarrow}) = O(n^5)$                          | [Becker & Heckmann,1998] |
| $RMS(cf, \vec{\rightarrow}) = O(TAG) = O(n^6)$                 | [Nederhof,1997a]         |

## The Structure of Recursive Matrices

The language  $L = \{a^n b^n a^p b^p c^p c^n \mid n, p \in \mathbb{N}\}$  has the following structure:

- $a^p b^p c^p$  is *embedded* in  $a^n b^n c^n$ :  $a^n b^n \boxed{a^p b^p c^p} c^n$
- $a, b, c$  are created in *parallel*.
- $^n, ^p$  indicate how often  $a, b, c$  is *pumped*.

A recursive matrix is a structure or a mathematical object with three different dimensions: *x-axis*, *y-axis* and *embedding*.



This structure fits the requirements of formal languages better than the structure of (recursive-)strings with only two dimensions, namely *x-axis* and *embedding*. With the context-free grammar rule  $S \rightarrow aSb$  for example, it can be pumped only into two directions: to the left or to the right of the nonterminal:  $a \leftarrow S \rightarrow b$ . But *pumping* and *embedding* could be observed and limited independently if a new, vertical dimension were introduced.

In the recursive matrix systems *pumping* takes place in the *x-axis*, *parallel* in the *y-axis* and *embedding* in an element.

### Count(3) written as TAG, CSG and RMS

Context-sensitive grammars can handle the language  $L_1 = \{a^n b^n c^n \mid n \in \mathbb{N}\}$  because they swap the terminals conditionally around. But conditional swapping makes the grammar difficult. Here is a context-sensitive, a tree-adjoining and a recursive matrix grammar for the language  $L_1$ :

TAG:  $\begin{array}{c} a \\ \swarrow \searrow \\ b \quad c \\ \swarrow \searrow \\ b \quad c \end{array}$       RMS:  $S \rightarrow \begin{bmatrix} a \\ b \\ c \end{bmatrix} S$

CSG:  $\{S \rightarrow aSBC, S \rightarrow aBC, CB \rightarrow BC, aB \rightarrow ab, bB \rightarrow bb, bC \rightarrow bc, cC \rightarrow cc\}$ .

The language  $Count(5)$  can easily be expressed as a  $RMS_5(reg)$ , but not at all as a TAG and only very difficultly as a CSG.

## Conclusion

The developed formalism RMS has the facility to compare many rewriting systems in one uniform framework.  $RMS_3(regular, \vec{\rightarrow})$  is a superset of CFG and a subset of TAG. It can express cross-serial and nested dependencies. In our opinion it should be checked for linguistic relevance. *Pumping* and *embedding* are separated into two dimensions. The structure of recursive matrices offers a simple extension to the known rewriting systems. Some ideas of this paper are from [Becker,1994], [Guan,1992], [Reichert,1991]. Further work will concentrate on efficient parsing algorithms. RMS is not to be confound with Matrix Grammars, introduced in [Abraham,1965].

## Acknowledgments

For motivation and helpful discussions I would like to thank Tilman Becker.

## Bibliography

- [Abraham,1965] L. Abraham, *Some questions of phrase-structure grammars*. Comput. Linguistics 4 61-70, 1965
- [Becker,1994] Tilman Becker, *HyTAG: A New Type of TAGs for Hybrid Syntactic Representation of Free Word Order Languages*. Ph.D.thesis, Univ. des Saarlandes, 1994
- [Becker & Heckmann,1998] Tilman Becker, Dominik Heckmann, *Recursive Matrix Systems (RMS) and TAG*. Submitted to TAG+ Workshop, Philadelphia, August 1998
- [Dassow & Paun,1989] Jürgen Dassow, Gheorghe Paun, *Regulated rewriting in formal language theory*. EATCS monographs on theoretical computer science 18. Springer Berlin, 1989
- [Guan,1992] Yonggang Guan, *Klammergrammatiken, Netzgrammatiken und Interpretationen von Netzen*. Ph.D.thesis, Univ. des Saarlandes, 1992
- [Harrison,1978] Michael A. Harrison *Introduction to Formal Language Theory*. Addison-Wesley, 1978
- [Hopcraft & Ullman,1994] John E. Hopcroft, Jeffrey D. Ullman, *Einführung in die Automatentheorie, Formale Sprachen und Komplexitätstheorie*. Addison Wesley 3. Auflage, 1994

- [Joshi,1985] Aravind K. Joshi, *Tree adjoining grammars: How much context-sensitivity is required to provide reasonable structural descriptions?* Editors: D.Dowty,L. Karttunen, A. Zwicky, *Natural Language Parsing* pages 206 - 250 Cambridge University Press, 1985
- [Joshi et al,1991] Aravind K. Joshi, Vijay Shanker, David Weir, *The Convergence of Mildly Context-Sensitive Grammar Formalisms*. Editors: Peter Sells, Stuart M. Shieber, Thomas Wasow, *Foundational issues in natural language processing* pages 31 -82 Massachusetts Institute of Technology, 1991
- [Nederhof,1997a] Mark-Jan Nederhof, *Solving the correct-prefix property for TAGs*. In proceedings of MOL5, Saarbrücken, Germany, 1997
- [Nederhof,1997b] Mark-Jan Nederhof, *Regular Approximations of CFLs: A Grammatical View*. Talk at DFKI, Saarbrücken, Germany, 1997
- [Reichert,1991] Armin Reichert, *Baumgrammatiken mit Multilinearer Interpretation*. Diplomarbeit, Univ. des Saarlandes, 1991
- [Schöning,1995] Uwe Schöning, *Theoretische Informatik - kurzgefasst*. Spektrum Akademischer Verlag, 1995
- [Weir,1988] David K. Weir, *Characterizing Mildly context-Sensitive Grammar Formalisms*. Ph.D. thesis, Univ. of Pennsylvania, 1988