Recursive Matrix Systems (RMS) and TAG

Tilman Becker
DFKI GmbH
Stuhlsatzenhausweg 3
D-66123 Saarbrücken
becker@dfki.de

Dominik Heckmann
Universität des Saarlandes
D-66123 Saarbrücken
dheck@studcs.uni-sb.de

Introduction

We define Recursive Matrix Systems (RMS), a highly parameterizable formalism that allows for a clear separation of various kinds of recursion. One instance of RMS, namely context–free RMS with two rows and a specific reading interpretation turns out to be weakly equivalent to TAG. This allows for the transfer of results from TAGs to this class of RMS. Furthermore, the equivalence proof is constructive and exhibits a very close relationship between the structures of the two formalism, namely trees and matrices. This allows to transfer interesting restrictions which can easily be defined in RMS to TAG. In particular, the obvious restriction of context–free RMS to regular RMS results in a restricted form of TAG which appears sufficient for natural language processing, albeit being less complex than regular TAG.

Recursive Matrix Systems

A *Recursive Matrix* is a finite matrix whose elements are either terminal symbols or again recursive matrices (see Figure 1). Recursive matrices are created by grammars (in particular by regular and context–free grammars) that have vectors as their terminal symbols. Strings are derived from a recursive matrix by a *reading interpretation* which reads the terminal symbols of a matrix line-by-line either from left-to-right or right-to-left and recursively descends for elements that are recursive matrices. In the following, we consider only Recursive Matrices with a constant number of rows in all (sub-) matrices. This number n is an important parameter. We denote the set of all recursive matrices as RM.

a	b	c	ε
a	b	c	d e f d e f d e f
d	ε	a	c

In this example the element in the second row, fourth column is the recursive (sub-) matrix

d	e	f
d	e	f
d	e	f

. The other elements are terminals.

Figure 1: A recursive matrix.

A *regular (context–free) Recursive Matrix System (reg-RMS, cf-RMS)* is a tuple $\langle G,I \rangle$ where G is a grammar that generates recursive matrices and I is an interpretation to read a string from each recursive matrix. $L(G)$ is the set of all recursive matrices derived by the grammar G. $L(G,I)$ is the set of all strings derived from the recursive matrices in $L(G)$ by the interpretation I.

A regular (context–free) grammar G that generates recursive matrices is a grammar with terminal symbols Vec[1], nonterminals N, a start symbol S from N and a set P of regular (context–free) rules. All vectors $v \in Vec$ have constant size n; the elements of v are either symbols from a set T (these are called the terminal symbols of the RMS) or non-terminals from N. T, V, N, P are finite but non–empty sets, $N \cap T = \emptyset$.

The derivation relation $\Rightarrow$ is defined over *Extended Recursive Matrices*, i.e., concatenations

[1] not to be confused with T, the terminal symbols of the RMS.

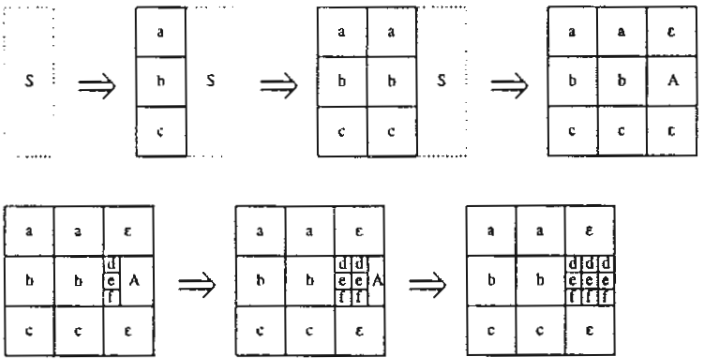


Figure 2: A derivation with RMS grammar G_1.

of vectors and non-terminals, where the elements of a vector are either terminal-symbols of the RMS, non-terminals of G or Extended Recursive Matrices. Each derivation step rewrites exactly one non-terminal according to a rule in P. The language $L(G)$ is defined as $L(G) := \{r | S \stackrel{*}{\Rightarrow} r, r \in RM\}$.

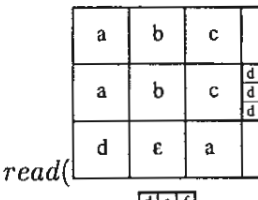
The following example grammar is used to show the derivation process:

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

All vectors have the size 3 and all rules are regular. G_1 is a reg-RMS. When applying the first or third rule, a vector is added to the matrix. When applying the second rule, a descend into the next recursive "matrix-level" takes place. Only the last rule is a terminating one. A possible derivation with the grammar G_1 is shown in figure 2. Note that the horizontal dimension of the recursive matrices is unbound.

The *reading interpretation* of a recursive matrix is derived from a vector of directions for each row of the matrix, i.e., an n-dimensional vector $I = \begin{bmatrix} i_1 \\ \vdots \\ i_k \end{bmatrix}$ of elements $i_j \in \{\rightarrow, \leftarrow\}$. It is recursively defined as shown in figure 3.

For example, with $I = \begin{bmatrix} \rightleftarrows \\ \rightleftarrows \end{bmatrix}$, we get



$read(\quad) =$

$abc \circ read(\quad) \circ cba \circ dac =$

$abc \circ def \circ fed \circ def \circ cba \circ dac =$

$abcdeffeddefcbadac.$

The Equivalence of CF-RMS$_{\rightleftarrows}$ and TAG

Although a TAG can be directly transformed into a weakly equivalent RMS, it is easier to demonstrate if we assume a normal form for TAG where no adjunction is possible into root and foot nodes, the root node has only one daughter, and there are no more than two inner nodes dominating the foot node. Figure 4 shows how such an auxiliary tree β can be directly mapped into a rule P of a context-free RMS$_{\rightleftarrows}$. The details for mapping the subtrees s, t, u, v to submatrices of the right-handside of P are omitted here.

Note the close resemblance of the notation of a TAG as an RMS to the notation of a TAG as a Linear Context-Free Rewriting System (LCFRS, Weir 1988). Even though in general, RMS can be captured as LCFRS, the particular structure of RMS which separates different dimensions of recursion has lead us to a number of observations which are not obvious

$$
\begin{aligned}
read(recursive\ matrix, I) &:= read(row_1, i_1) \circ \ldots \circ read(row_k, i_k) \\
read(row[1..m], \rightarrow) &:= read(row[1], I) \circ read(row[2..m], \rightarrow) \\
read(row[1..m], \leftarrow) &:= read(row[m], I) \circ read(row[1..m-1], \rightarrow) \\
read(terminal\ symbol, I) &:= terminal\ symbol
\end{aligned}
$$

Figure 3: Definition of the reading interpretation *read* for *recursive matrix* $= \begin{bmatrix} row_1 \\ \vdots \\ row_k \end{bmatrix}$.

when looking at TAGs or even at LCFRS.

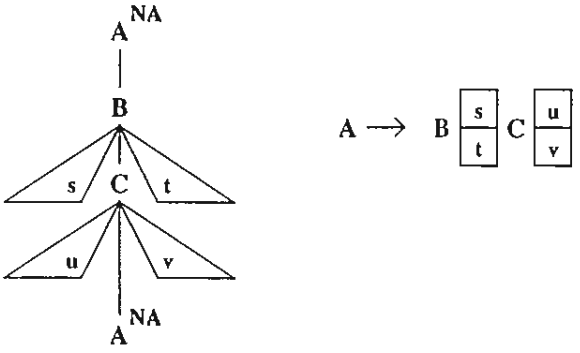


Figure 4: Tranforming a TAG into a weakly equivalent RMS.

Like contect–free grammars, context–free RMS can be transformed into a normal form resembling Chomsky normal form. In such a transformed cf-RMS$\rightleftarrows$, all rules are of the form shown in figure 5.

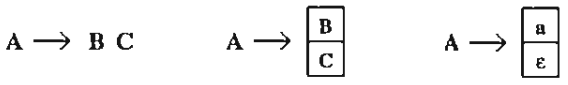


Figure 5: A normal form for cf-RMS$\rightleftarrows$.

Figure 6 sketches how a TAG grammar is constructed from such a cf-RMS that derives the same language.

Given this relation, the question arises whether a TAG can be transformed into a *regular* RMS, i.e., whether the non-terminal B in Figure 4 can be dropped. The answer is no, and it can be seen, e.g., by the fact that the normal form transformation cannot be tightend up to only one inner node dominating the foot node. This implies that regular RMS are a proper subset of context–free RMS[2].

[2]Actually, we found this relation when failing to show the equivalence of regular RMS and TAG, forcing us to extend RMS to context–free RMS.

On the other hand, this emphasizes a parameter of TAGs that was not obvious before: Even though the well known example grammars for deriving $L^4 = \{a^n b^n c^n d^n\}$ and $L^{copy} = \{ww | w \in \{a,b\}^*\}$ already exhibit non context–free properties and even cross–serial dependencies, they are restricted in the sense that their trees have only one node dominating the foot node that is available for adjunction. While it is not easy to give an example for the effects that can be achieved with two or more such nodes, when looking at RMS, this parameter becomes obvious (i.e. as the difference between regular or context–free RMS).

Looking at natural languages, it appears that in fact the restriction to TAG with only one adjunction node on the spine (an important restriction of regular RMS) are sufficient since recursive, unbounded dependencies are restricted to one type (e.g., either embedded or cross–serial), but don't occur intertwined with a second type of recursive, unbounded dependencies.

It remains unclear though, whether the second restriction of regular RMS, which in TAG terms means that *no* path from the root to a leaf can have more than one available adjunction node is too strong.

Current Work

We are currently exploring the consequences of the restrictions that reg-RMS have compared to CF-RMS. Exploiting the equivalence of TAGs and RMS allows us to adopt results for TAGs for RMS. A point of special interest is parsing and its time complexity. Taking any of the various known parsing algorithms for TAGs immediately gives us an $O(n^6)$ parsing algorithm

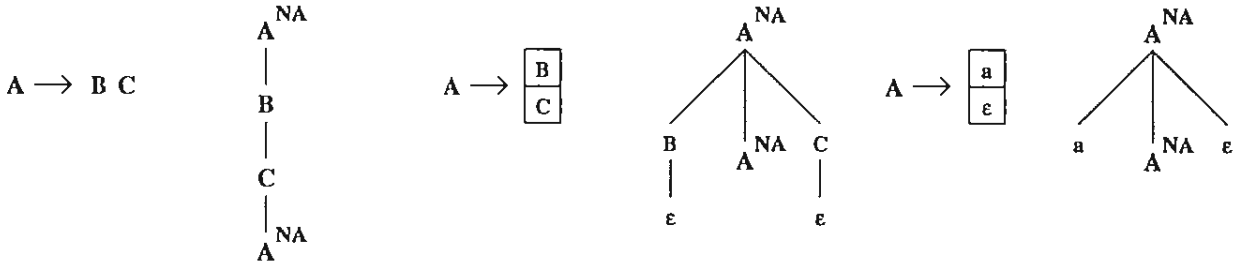


Figure 6: Elementary trees constructed for each rule of a cf-RMS$_{\rightleftarrows}$ in normal form.

for CF-RMS$_{\rightleftarrows}$. Moreover, given the tight coupling between the grammar rules of an RMS and the elementary trees of the equivalent TAG, we can find stronger restrictions on the steps of the TAG parser if the original RMS grammar is regular and not context-free. In particular, using the algorithm by (Nederhof 1997), we conjecture that reg-RMS can be parsed in at most $O(n^5)$ time.

A further avenue of research is the fact that the context-freeness of RMS is not necessary to construct grammars that exhibit cross-serial dependencies, one of the core arguments for TAGs. While 2-dimensional reg-RMS with a reading interpretation of $\rightrightarrows$ ($\rightleftarrows$) are sufficent to exhibit cross-serial dependencies (center-embedded dependencies resp.), they can't exhibit both. However, 3-dimensional reg-RMS are sufficient and therefore a candidate for a further restriction on TAGs for natural language processing which might result in a further reduction of the time complexity of parsing. While such a restriction might not be obvious when looking at TAG trees, the representation as an RMS allows for a very succint formulation.

Bibliography

Heckmann, Dominik. *Recursive Matrix Systems.* In Proceedings of the Student Session at the 11th ESSLLI, Saarbrücken, Germany, 1998.

Nederhof, Mark-Jan. *Solving the Correct-prefix Property for TAGs.* In Proceedings of MOL5, Saarbrücken, Germany, 1997. Available as DFKI document D-97-02.
`http://www.dfki.uni-kl.de/~dfkidok/publications/D/97/02/abstract.html`

Weir, David. *Characterizing Mildly Context-Sensitive Grammar Formalisms.* Ph.D. thesis, University of Pennsylvania, Philadelphia, 1988.