RESTARTING AUTOMATA, MARCUS GRAMMARS AND CONTEXT-FREE LANGUAGES

PETR JANČAR[1]
University of Ostrava, Dept. of Computer Science, Dvořákova 7
701 03 OSTRAVA, Czech Republic
e-mail: jancar@osu.cz

and

FRANTIŠEK MRÁZ, MARTIN PLÁTEK, MARTIN PROCHÁZKA
Charles University, Dept. of Computer Science, Malostranské nám. 25
118 00 PRAHA 1,Czech Republic
e-mail: mraz,platek@kki.mff.cuni.cz

and

JÖRG VOGEL
Friedrich Schiller University, Computer Science Institute
07740 JENA, Germany
e-mail: vogel@minet.uni-jena.de

ABSTRACT

We investigate the computational power of several types of restarting automata - nondeterministic, deterministic; monotonic; normal - and compare them to (deterministic) context-free languages. The relation between restarting automata and Marcus contextual grammars is discussed and a class of contextual grammars which can be characterized by normal restarting automata is defined.

1 Introduction

In [1] we studied forgetting automata, which can be viewed as special linear bounded automata with limited rewriting capability. In [2] we introduced the notion of restarting automata, which are special forgetting automata with a lucid way of computation. This paper directly continues on [2].

Our motivation partly comes from the area of natural language analysis and resembles the motivation for Marcus contextual grammars. Another part of motivation comes from the area of grammar checking of natural as well as programming languages (cf. [2]).

A restarting automaton (R-automaton) can be roughly described as follows: it has a finite control unit, a head with a lookahead window attached to a tape, and

[1]The first author was supported by the Grant Agency of the Czech Republic, Grant-No. 201/93/2123

it works in certain cycles. In a cycle, it moves the head from left to right along the input word on the tape; according to its instructions, it can at some point "cut off" some symbols out of the lookahead and "restart" – i.e. reset the control unit to the initial state and place the head on the left end of the (shorter) word, and it starts again. The computation halts in an accepting or a rejecting state.

In fact, to make the mentioned "cutting off" more natural, the restarting automaton uses a doubly linked list of items (cells) rather than the common tape.

As usual, we define nondeterministic and deterministic versions of this type of automata. In [2] we defined the (natural) property of *monotonocity* of these automata (that means, that during any computation the places of "cutting off" do not increase their distances from the right end) and showed that deterministic monotonic R-automata nicely characterize the class of deterministic context-free languages ($DCFL$) – monotonic deterministic R-automata recognize exactly $DCFL$. There was also shown the decidability of the monotonocity property and the paper was completed by showing a language outside $DCFL$ which is recognized by both a deterministic nonmonotonic R-automaton as well as a nondeterministic monotonic one.

Here we newly introduce the notion of *normality* which puts a restriction on the "cutting off" before the "restart" – at most two substrings from lookahead can be removed (not any subsequence as in the general case). Even normal deterministic monotonic R-automata recognize exactly $DCFL$. Moreover normal R-automata provide a natural possibility to study relations between R-automata and Marcus grammars. We define a new type of contextual grammars which correspond to normal restarting automata - regular prefix contextual grammars with bounded infix.

We present several results concerning the computational power of different types of restarting automata obtained by monotonicity, normality and determinism and compare them with the class of context-free languages (CFL).

2 Definitions and Results

We present the definitions informally; the formal technical details could be added in a standard way of automata theory.

A *restarting automaton*, abbr. *R-automaton*, M with bounded lookahead is a device with a finite state control unit and one head moving on a finite linear doubly linked list of items. The first item is reserved for a special symbol ¢, the last one for another special symbol \$, and each other item contains a symbol from a finite alphabet (not containing ¢, \$). The head has a lookahead "window" of fixed length k, for some $k \geq 0$; that means – besides the current item, M also scans the next k right neighbour items (or simply the end of the word when the distance to \$ is less than k). In the *initial configuration* on an input *word* w, the control unit is in a fixed, initial, state and the head is attached to that item which contains the left sentinel ¢ (scanning also the first k symbols of the input word) and the list contains ¢w\$.

The *computation* of M is controlled by a finite set of *instructions* of the following two types:

1) $(q, au) \rightarrow (q', MOVE\text{-}R)$

2) $(q, au) \rightarrow RESTART(v)$

The left-hand side of an instruction determines when it is applicable – q means the current state of the finite control, a means the current symbol which is scanned by the head, and u means the contents of the lookahead window (u being a string of length k or less if it ends with \$). The right-hand side describes the activity to be performed. In case 1) – move-right instruction – M changes the current state to q' and moves the head to the right neighbour item. In case 2) – restart instruction – the activity consists of deleting some items of the just scanned part of the list (containing au) so that v is left (au is replaced by v where v is a proper subsequence of au) and restarting – i.e. setting the initial state and placing the head on the first item of the list (containing ¢).

We suppose that the states of the finite control are divided into two classes – the *nonhalting states* (there is at least one instruction which is applicable when the unit is in such a state) and the *halting states* (any computation finishes by entering such a state); the halting states are further divided into the *accepting states* and the *rejecting states*.

In general, an R-automaton is *nondeterministic*, i.e. there can be two or more instructions with the same left-hand side (q, au). If this is not the case, the automaton is *deterministic*.

An input *word w is accepted by M* if there is a computation which starts in the initial configuration on w (bounded by sentinels ¢,\$) on the list and finishes in an *accepting configuration* where the control unit is in one of the accepting states. $L(M)$ denotes the language consisting of all words accepted by M; we say that M *recognizes the language $L(M)$*.

Any computation of an R-automaton consists of certain *cycles*: in one cycle, the head moves right along the input list (with a bounded lookahead) until a halting state is entered or something from the bounded space is deleted – in that case the computation is resumed in the initial configuration on a new – shorter – word (i.e., a new cycle starts). This immediately implies that any computation of any R-automaton is finite (finishing in a halting state).

The next two claims (with obvious, and hence omitted, proofs) express a certain lucidness of computations of R-automata. The notation $u \longrightarrow_M v$ means that there exists a cycle of M starting in the initial configuration on the word u and finishing in the initial configuration on the word v; the relation $\longrightarrow^*_M$ is the reflexive and transitive closure of $\longrightarrow_M$.

Claim 2.1 (The error preserving property (for all R-automata))
Let M be an R-automaton, and u, v any pair of words over the input alphabet of M.
If $u \longrightarrow^*_M v$ and $u \notin L(M)$, then $v \notin L(M)$.

Claim 2.2 (The correctness preserving property (for *det*-R-automata))
Let M be a deterministic R-automaton and let holds $u \longrightarrow^*_M v$ for some words u, v.
Then $u \in L(M)$ iff $v \in L(M)$.

Considering a deterministic R-automaton M, w.l.o.g., we can suppose (cf. [2]) it being in the *strong cyclic form*; it means, that the words of length less than k, k being the length of lookahead, are immediately (hence in the first cycle) accepted or rejected, and that M performs at least two cycles (at least one restarting) for any longer word. For a nondeterministic R-automaton M, we can suppose the *weak cyclic form* (cf. [2]) – any word of $L(M)$ longer than k can only be accepted by performing two cycles at least.

By a *monotonic R-automaton* we mean a R-automaton where the following condition holds for all computations: all items which appeared in the lookahead window (and were not deleted) during one cycle will appear in the lookahead in the next cycle as well – if it does not finish in a halting state. (I.e., during any computation, the places of deleting do not increase their distances from the right endmarker \$).

An R-automaton is said to be *normal* if for each of its instructions of the form $(q, au) \rightarrow RESTART(v)$ there exist words x_1, x_2, x_3, x_4, x_5 such that $au = x_1x_2x_3x_4x_5$ and $v = x_1x_3x_4$; that is, at most two substrings of au are "cut off" (some of x_i can be empty).

For brevity, we use the following obvious notation. R denotes the class of all (nondeterministic) restarting automata (with some lookahead). Prefix *det-* denotes the deterministic version, similarly *mon-* the monotonic version and *norm-* the normal version. $\mathcal{L}(A)$, where A is some type of automata (or of grammars), denotes the class of languages recognizable by automata (or generated by grammars) of type A.

E.g., the class of languages recognizable by deterministic monotonic R-automata is denoted by $\mathcal{L}(det\text{-}mon\text{-}R)$.

Exactly this type of automata allows a characterization of $DCFL$. The following theorem is proven in [2].

Theorem 2.1 $DCFL = \mathcal{L}(det\text{-}mon\text{-}R)$.

Indeed, the proof which we have given there shows even that $DCFL = \mathcal{L}(norm\text{-}det\text{-}mon\text{-}R)$.

The monotonicity of restarting automata seems to be a very natural restriction and here we consider the effect of monotonicity for nondeterministic R-automata. Monotonicity restricts also the computational power of nondeterministic R-automata.

Ad first, languages which are recognizable by monotonic R-automata are context-free (Theorem 4.1).

Ad second, R-automata (*norm-R*-automata, *det-R*-automata and *norm-det-R*-automata, resp.) are more powerful than the monotonic ones (Theorem 4.2).

Ad third, not all context-free languages can be recognized by R-automata (Corollary 4.2, 4.3, 4.4, resp.).

These and some further detailed relations are given in Section 4.

To illustrate the closeness between restart automata and contextual grammars, next we introduce Marcus contextual grammar from [4]. A *contextual grammar* (CG) is a quadruple $G = (V, B, C, f)$, where V is a finite nonempty alphabet, B (*basis set*) is a finite language over V, C (set of *contexts* over V) is a finite subset of $V^* \times V^*$, and f is a mapping described by $f : V^* \times V^* \times V^* \to 2^C$ (*selection mapping* of G).

The language generated by G, denoted by $L(G)$, is the smallest subset L of V^*, which includes B and has the following property: if x is in L, $x = x_1x_2x_3$, and (u, v) is in $f(x_1, x_2, x_3)$, then $x_1ux_2vx_3$ is in L (we write $x \to_G x_1ux_2vx_3$).

The context (u, v) is added to x to obtain $x_1ux_2vx_3$ if and only if it is selected by the mapping f. The words x_1,x_2,x_3 for which $f(x_1, x_2, x_3)$ is nonempty are called *prefix selector*, *infix selector* and *suffix selector*, respectively.

The reflexive and transitive closure of the relation $\to_G$ is denoted by $\to_G^*$.

Let $G = (V, B, C, f)$ be a contextual grammar and k be a positive integer. If
$f(x_1, x_2, x_3) = f(x_1', x_2, x_3')$ for any $x_1, x_1', x_2, x_3, x_3' \in V^*$ whereby
$f(x_1, x_2, x_3) = \emptyset$ for any x_2 with length $|x_2| > k$
then G is said to be an *internal contextual grammar with bounded choice* ($ICGBC$). In words, the adjoing of a context depends only on a bounded string, namely that embraced by the context.

In the next section we define a special class of contextual grammars - regular prefix contextual grammars with bounded infix ($RPCGBI$). They are more powerful than $ICGBC$, but less powerful than CG, and they generate exactly $\mathcal{L}$(*norm-R*).

3 R-automata and Marcus contextual grammars

Definition 3.1 (Regular-Prefix Contextual Grammar with Bounded Infix) *Let k be a positive integer. A contextual grammar $G = (V, B, C, f)$ is said to be a regular-prefix contextual grammar with bounded infix ($RPCGBI$) of degree k if the mapping f has the following properties:*

(1) for any words y, z the set $\{x | f(x, y, z) \neq \emptyset\}$ is a regular language,

(2) $\{y | f(x, y, z) \neq \emptyset$ for some $x, z \in V^*\}$ is a finite language,

(3) for any $x, y, z_1, z_2 \in V^*$ the following condition holds:
if the prefixes of z_1 and z_2 of the length k are equal, then $f(x, y, z_1) = f(x, y, z_2)$

Theorem 3.1 $\mathcal{L}(RPCGBI) = \mathcal{L}$(*norm-R*).

Proof: a) Let $G = (V, B, C, f)$ be an arbitrary $RPCGBI$ of degree $k \geq 1$. We describe a *norm-R*-automaton M which recognizes the language $L(G)$.

The size of the lookahead window of M is $k_M = max(k + k_C, k_B)$, where k_C is the maximal length of a context together with its infix selector in G, i.e.

$$k_C = max\{|uyv|; (u, v) \in f(x, y, z) \text{ for some } x, z \in V^*\}$$

and k_B is the maximal length of a word of the basis set B.

If there is an input word w of length not greater than k_M then M either accepts it immediately (if it is either contained in the basis set B or it is a further but short word in $L(G)$) or rejects it immediately (if $w \notin L(G)$).

Let us assume that the length of the input word is greater than k_M. Then M will move along the input list to the right. The set of all prefix selectors of grammar G is the union of a finite number of different regular languages – each of them determined by an infix selector from a finite language (according to (2)) and by a finite prefix of length k of a suffix selector (according to (3)). Thus, the automaton M is able to recognize whether the part of the input list, which has been already scanned, is in the set of prefix selectors of the grammar G or not. If a prefix selector x is successfully recognized then the automaton M checks its lookahead window. If in the window there is a concatenation of a left context u, an infix selector y, a right context v and a word z of the length at least k and $(u, v) \in f(x, y, z)$ then the automaton M is allowed to execute an operation $RESTART(yz)$ (according to (3) without any other checking). It is obvious that, if the word on the list after a cycle is in $L(G)$ then also the word at the beginning of that cycle is in $L(G)$. Thus every word accepted by M is in $L(G)$. On the other side, for every word w in $L(G)$ there exists an accepting computation of M.

b) The opposite inclusion can be demonstrated in the following way. Let M be a *norm-R*-automaton with lookahead k in the weak cyclic form. We will describe a regular-prefix grammar with bounded infix $G = (V, B, C, f)$ of degree k which generates the language $L(M)$. M is a normal R-automaton so it may "cut off" at most two substrings. This fact enables us to express the behaviour of automaton M in terms of a selection mapping:

We define $(u, v) \in f(x_1x_2, y, z)$ for any $u, v, x_1, x_2, y, z \in V^*$ iff M executes a restart instruction $RESTART(x_2yz_1)$ after move-right instructions reading x_1 and with the lookahead window containing x_2uyvz_1, where z_1 is a prefix of z.

f fulfils all properties of a selection mapping:

(1) Because of the fixed length of the lookahead, the length of x_2 is bounded by k. For any $y, z \in V^*$ there is a finite number of words x_2 such that there exists some x_1 satisfying the definition for f. For any $y, z, x_2 \in V^*$ the set $\{x_1 | f(x_1x_2, y, z) \neq \emptyset\}$ is regular (determined by the state in which M can restart). Thus, for any $y, z \in V^*$ the set $\{x | f(x, y, z) \neq \emptyset\}$ is a finite union of regular languages (concatenations of a regular language and a word), which is again regular language.

(2) The length of y is bounded by k, thus there are only finitely many words $y \in V^*$ for which $f(x, y, z)$ is nonempty for some $x, z \in V^*$.

(3) Since the length of the relevant prefix of z is also bounded by k, thus as well (3) is satisfied.

The automaton M is in weak cyclic form, that means, the set of words accepted by M without restart is finite and this will be the basis set B of grammar G. Finiteness of the set of contexts follows directly from the bounded lookahead of M, which limits also the size of deleted parts by any *RESTART* operation. □

Internal contextual grammars with bounded choice are weaker then regular-prefix contextual grammar with bounded infix. We can show it by using *norm-R*-automata.

Theorem 3.2 $\mathcal{L}(ICGBC) \subset \mathcal{L}(norm\text{-}R)$.

Proof: a) From the definitions of *ICGBC* and *RPCGBI* it follows that $\mathcal{L}(ICGBC)$ is contained in $\mathcal{L}(RPCGBI)$ and – because of Theorem 3.1 – therefore in $\mathcal{L}(norm\text{-}R)$.

b) We use $L = \{c_1a^nb^{2n} | n \geq 1\} \cup \{c_2a^{2n}b^n | n \geq 1\}$ as a witness language in $L(norm\text{-}R)$ which cannot be generated by any *ICGBC*.

Clearly L is in *DCFL*, therefore it is in $\mathcal{L}(norm\text{-}R)$.

Let us suppose that L is generated by an *ICGBC* G. The selection mapping f of G has the following property:

(*) there are y,u,v such that for every x,z the set $f(x, y, z)$ contains (u, v), where $y = a^rb^s, u = a^i, v = b^{2i}$ and $r, s, i \geq 1$

Otherwise, G does not generate all words of the form $c_1a^nb^{2n}$ (for all $n \geq 1$) or G generates words out of L.

We consider $w_1 = c_2a^{2(r+s)})b^{r+s}$. Obviously, w_1 is in L, but therefore by (*) also $w_2 = c_2a^{2(r+s)+i}b^{r+s+2i}$ is in L – a contradiction. □

4 Relations of R-automata to CFL

Monotonicity is a natural restriction not only for *det-R*-automata (Theorem 2.1) but also for nondeterministic *R*-automata. It restricts the computational power of restart automata. On the one hand side we have

Theorem 4.1 $\mathcal{L}(mon\text{-}R) \subseteq CFL$. □

On the other hand side the assumption of monotonicity is substantial in both theorems, Theorem 2.1 as well as Theorem 4.1. Even *det-R*-automata are able to recognize non context-free languages. The proof is similar to the proof of the fact that there is an *ICGBC* which generates a non context-free language (cf. [4]).

Theorem 4.2 *a) There is a norm-det-R-automaton recognizing a non context-free language.*

b) R-automata are more powerful than mon-R-automata and norm-R-automata are more powerful than norm-mon-R-automata.

Proof: a) We describe a *norm-det-R*-automaton M which recognizes a non context-free language. In fact, we show that the language $L(M)$ is not context-free by showing that

$$L(M) \cap \{(ab)^n \mid n \geq 1\} = \{(ab)^{2^k} \mid k \geq 0\} \tag{1}$$

Thus the language $L(M)$ cannot be recognized by a deterministic monotonic automaton (Theorem 2.1).

The automaton M works as follows:

1. reading $\cent abab$ or $\cent abba$ it moves to the right;
2. reading $ababa$ or $babab$ it moves to the right;
3. reading $abab\$$ it deletes second a and restarts;
4. reading $ababb$ it deletes the first a and restarts;
5. reading $abbab$ or $bbabb$ or $babba$ it moves to the right;
6. reading $babb\$$ it deletes the second b and restarts;
7. reading $bbab\$$ or $bbaba$ it deletes the first b and restarts;
8. reading $\cent abb\$$ it deletes the first b and restarts;
9. reading $\cent ab\$$ it accepts;
10. in all other cases the automaton halts in a nonaccepting state.

To prove (1) let us consider a word of the form $(ab)^n$ for some $n \geq 1$. Then holds

$$\text{(i)} \quad (ab)^n \longrightarrow_M^* (ab)^{\frac{n}{2}} \qquad \text{when } n \text{ is even,}$$

$$\text{or (ii)} \quad (ab)^n \longrightarrow_M^* b(abb)^{\frac{n-1}{2}} \qquad \text{when } n \text{ is odd.}$$

In the case (i) the automaton makes $\frac{n}{2}$ cycles (using points 1, 2–4), and gets the word $(abb)^{\frac{n}{2}}$, then after another $\frac{n}{2}$ cycles it gets the word $(ab)^{\frac{n}{2}}$ (using points 1, 5–8)

In the case (ii) the automaton makes $\frac{n-1}{2}$ cycles (using points 1, 2–4), and gets the word $b(abb)^{\frac{n-1}{2}}$, which will be rejected in the next cycle.

Thus let the input word be of the form $(ab)^n$, where $n = l2^k$ for some integer $k \geq 0$ and some odd integer $l \geq 1$.

- If n is a power of 2 (i.e. $l = 1$ and $k \geq 1$), then according (i) $(ab)^n \longrightarrow_M^* (ab)^{2^{k-1}} \longrightarrow_M^* (ab)^{2^{k-2}} \longrightarrow_M^* \ldots \longrightarrow_M^* ab$ and according point 9 the input word will be accepted.
- If n is not a power of 2 (i.e. $l > 1$), then according (i) $(ab)^n \longrightarrow_M^* (ab)^l$, $l > 1$ and odd. Further according (ii) $(ab)^l \longrightarrow_M^* b(abb)^{\frac{l-1}{2}}$ and this word will be rejected by 10.

b) The *norm-det-R*-automaton M described in a) accepts a non context-free language, $\mathcal{L}(mon\text{-}R)$ is a subset of $\mathcal{L}(R)$ and $\mathcal{L}(mon\text{-}R)$ is a subset of *CFL* (Theorem 4.1). From this follows that $\mathcal{L}(mon\text{-}R)$ is a proper subset of $\mathcal{L}(R)$ and also $\mathcal{L}(norm\text{-}mon\text{-}R)$ is a proper subset of $\mathcal{L}(norm\text{-}R)$. □

In the previous theorem we have shown that *det-R*-automata can accept non context-free languages, but they are stronger than *det-mon-R*-automata even inside the class *CFL* .

Theorem 4.3 *There exists a context-free language recognized by a norm-det-R-automaton, which cannot be recognized by any mon-R-automaton.* □

The witness language constructed in the proof of Theorem 4.3 can be slightly modified in order to proof the following

Corollary 4.1 *There is a context-free language in*
$$(\mathcal{L}(det\text{-}R) \cap \mathcal{L}(mon\text{-}R)) - \mathcal{L}(det\text{-}mon\text{-}R).$$ □

(Nondeterministic) R-automata are stronger than deterministic R-automata and (nondeterministic) *norm-R*-automata are stronger than deterministic *norm-R*-automata. This is shown in

Theorem 4.4
$$(\mathcal{L}(norm\text{-}R) - \mathcal{L}(det\text{-}R)) \cap \mathcal{L}(norm\text{-}mon\text{-}R) \neq \emptyset$$
$$(\mathcal{L}(norm\text{-}R) - \mathcal{L}(det\text{-}R)) \cap (CFL - \mathcal{L}(mon\text{-}R)) \neq \emptyset$$
$$(\mathcal{L}(norm\text{-}R) - \mathcal{L}(det\text{-}R)) \cap co\text{-}CFL \neq \emptyset$$ □

As a simple consequence we have

Corollary 4.2 $\mathcal{L}(det\text{-}R)$ *and* $\mathcal{L}(norm\text{-}det\text{-}R)$ *are incomparable with CFL .* □

But also the class $\mathcal{L}(R)$ does not contain the whole class *CFL* . The language $L = \{ww^R \mid w \in \{a,b\}^*\}$ cannot be recognized by any R-automaton. Suppose that L is recognized by some R-automaton M in a cyclic form. Consider an accepting computation on a sufficiently long word $a^k b^m b^m a^k$. In the first cycle of the accepting computation, M can only shorten the segment of $b's$. We will get a word of the form $a^k b^{2m'} a^k$, for some $m' < m$, after the first cycle. But M can make the same first cycle in the computation on the word $a^k b^{2m} a^k a^k b^{2m'} a^k \notin L(M)$ and get the word $a^k b^{2m'} a^k a^k b^{2m'} a^k$ which is in $L(M)$. This is a contradiction to the error preserving property of R-automata (Claim 2.1).

On this way we get the following two corollaries.

Corollary 4.3 $\mathcal{L}(R)$ *and* $\mathcal{L}(norm\text{-}R)$ *are incomparable with CFL .* □

Corollary 4.4 *a)* $\mathcal{L}(det\text{-}R)$ *is incomparable with* $\mathcal{L}(mon\text{-}R)$.

b) $\mathcal{L}(norm\text{-}det\text{-}R)$ *is incomparable with* $\mathcal{L}(norm\text{-}mon\text{-}R)$. □

The sketches of proofs of the Theorems 4.1, 4.3 and 4.4, resp., as well as the proofs of the corollaries are given in [3] and the reader will find they also in a full version of the paper.

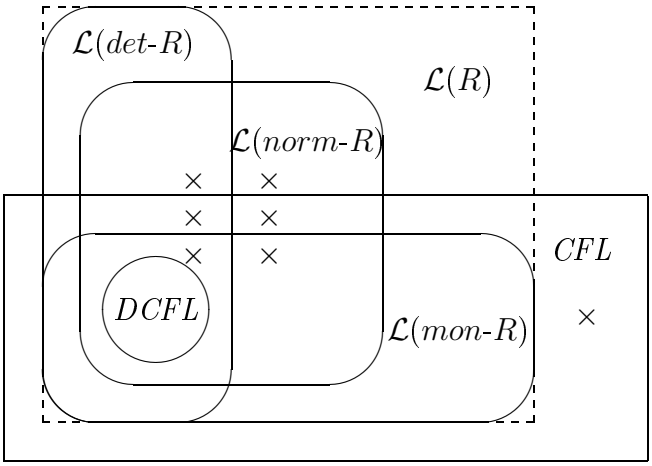

Figure 1: Relations between classes of restart automata and *CFL*.

5 Conclusions

We studied several versions of restarting automata and their relations to *CFL* and *DCFL* concentrating on normal restarting automata. Normal restarting automata recognize a class of languages which can be generated by certain type of contextual grammars, and normal deterministic monotonic restarting automata recognize exactly *DCFL* .

Relations between the classes of languages accepted by various types of restarting automata are depicted in Figure 1. The symbol $\times$ denotes classes which we have proven to be nonempty. Note that we completely separated subclasses of normal R-automata with respect to *CFL* , monotonic and deterministic versions of R-automata.

References

[1] P. Jančar, F. Mráz, M. Plátek: *A Taxonomy of Forgetting Automata*; Proc. MFCS'93, Gdańsk, Poland, LNCS 711, Springer 1993, pp. 527–536

[2] P. Jančar, F. Mráz, M. Plátek, J. Vogel: *Restarting Automata*; Proc. FCT'95, Dresden, Germany, LNCS 965, Springer 1995, pp. 283–292

[3] P. Jančar, F. Mráz, M. Plátek, M. Procházka, J. Vogel: *Restarting Automata and Marcus Grammars*; TR Math/95/1, Friedrich Schiller University, Jena, 1995

[4] G. Păun: *On Some Open Problems about Marcus Contextual Languages*; Intern. J. Computer Math., 1985, Vol. 17, pp. 9–23

[5] G. Păun: *Marcus Contextual Grammars (after 25 years)*; Bull. EATCS 52, February 1994, pp. 263–273