On Minimalism of Analysis by Reduction by Restarting Automata[*]

Martin Plátek[1], Dana Pardubská[2], and Markéta Lopatková[1]

[1] Charles University in Prague, Faculty of Mathematics and Physics
Malostranské nám. 25, 118 00 Prague, Czech Republic
martin.platek@mff.cuni.cz, lopatkova@ufal.mff.cuni.cz
[2] Comenius University in Bratislava,
Faculty of Mathematics, Physics and Informatics,
Mlynská dolina, 842 48 Bratislava, Slovak Republic
pardubska@dcs.fmph.uniba.sk

Abstract. The paper provides linguistic observations as a motivation for a formal study of an analysis by reduction. It concentrates on a study of the whole mechanism through a class of restarting automata with meta-instructions using pebbles, with delete and shift operations (DS-automata). Four types of (in)finite sets defined by these automata are considered as linguistically relevant: basic languages on word forms marked with grammatical categories, proper languages on unmarked word forms, categorial languages on grammatical categories, and sets of reductions (reduction languages). The equivalence of proper languages is considered for a weak equivalence of DS-automata, and the equivalence of reduction languages for a strong equivalence of DS-automata.

The complexity of a language is naturally measured by the number of *pebbles*, the number of *deletions*, and the number of *word order shifts* used in a single reduction step. We have obtained unbounded hierarchies (scales) for all four types of classes of finite languages considered here, as well as for Chomsky's classes of infinite languages. The scales make it possible to estimate relevant complexity issues of analysis by reduction for natural languages.

1 Introduction

The method of analysis by reduction (AR) plays an important role in a lexicalized syntax of natural languages. It consists in a stepwise simplification of a sentence, which profits from the integration of the sentence syntactic structure and the corresponding grammatical categories.

To model AR, various types of restarting automata can be found in literature [6, 7], which allow one to study dependencies in a natural language. Unfortunately, these types of automata are not able to adequately cope with a word

[*] The paper reports on the research supported by the grants of GAČR No. P202/10/1333 and partially P103/10/0783. The third author is supported by the Slovak Grant Agency for Science (VEGA) under the contract No. 1/0979/12.

order freedom frequently present in Czech sentences. In this paper we present a formal model of the analysis by reduction that is, in addition to a *deletion*, enriched with a *word order shift*, an operation reflecting a word order freedom of natural languages [4]. Section 2 provides a linguistic motivation and informal description of the process.

The core sections 3 and 4 provide a formal study of the whole mechanism through a refined class of restarting automata (DS-automata), and their descriptional complexity based on the number of pebbles, on the number of deletions, and on the number of word order shifts used within a single meta-instruction. Using these measures, we are able to argue that natural languages (e.g. Czech) can be described using rather simple reductions. Our paper refines the notion of window size [5, 7] by the number of pebbles.

Four types of (in)finite sets defined by DS-automata are the most relevant: basic languages on word forms marked with their linguistic categories, sets of reductions on basic languages forming reduction languages, proper languages on unmarked word forms, and categorial languages on pure categories. Inspired by Chomsky [2], we consider the equivalence of proper languages as the *weak equivalence* (close to the weak equivalence by formal automata and grammars), and the equivalence of reduction languages as the linguistically finest *strong equivalence* between DS-automata.

Formal parts of this article are based on the descriptional complexity of selected classes of finite languages and traditional Chomsky classes of infinite languages. Note that the concentration on finite languages comes from the domain of interest; we can to a certain extent claim that the core vocabulary of a natural language may be considered finite and that in a normal everyday use of a language the comprehensible/understandable sentence may be considered finite as well. We introduce the map- and mrp- properties of restarting automata, resp. meta-instructions – these properties in some sense characterize the minimality of reductions. If a correct sentence of a natural language undergoes the process of analysis by reduction, we require that it remains correct in each step of the reduction. The obtained results give the theoretical background for an incremental transfer from finite (linguistic) observations (as in [4]) to adequate, fully lexicalized, formal descriptions (models) of natural languages based on sentence reductions that are applicable on infinite languages.

2 Analysis by Reduction

Analysis by reduction (AR) helps to identify syntactic structure and the corresponding grammatical categories of the analyzed language. AR is based upon a stepwise simplification of an analyzed sentence, see [4]. It defines possible sequences of reductions in the sentence – each step of AR consists in *deleting* at least one word of the input sentence and thus its shortening. Here we allow that a deletion of a word is accompanied by a *shift* of some word(s) to another word order position(s) in the sentence.

Let us stress the basic constraints imposed on each reduction step of AR:

(i) individual words (word forms), their morphological characteristics and/or their syntactic categories must be preserved in the course of AR;
(ii) a grammatically correct sentence must remain correct after its simplification;
(iii) shortening of any reduction would violate the correctness principle (ii);
(iv) a sentence which contains a correct sentence (or its permutation) as a subsequence, must be further reduced;
(v) an application of the shift operation is limited only to cases when a shift is enforced by the correctness principle (ii); i.e., a simple deletion would result in an incorrect word order.

Note that the possible order of reductions reflects dependency relations between individual sentence members, i.e., relations between governing and dependent nodes, as it is described in [7].

Let us illustrate the basic principles of AR on the following example. The sentence undergoing AR is represented as a string of word forms (words and punctuation) enriched with their disambiguated lexical, morphological and syntactic categories.[1]

Example 1.
(1) [*Petr*,Sb] [*se*,AuxT] [*bojí*,Pred] [*o*,AuxP] [*otce*,Obj] [.,AuxK]
‘Peter – REFL – worries – about – father – .’
‘Peter worries about his father.’

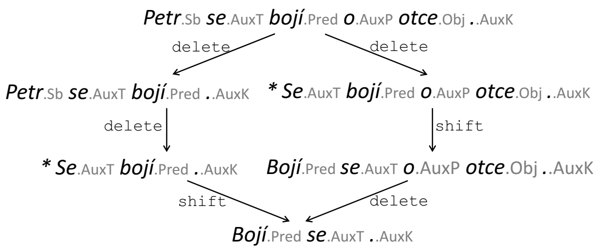
Petr.Sb *se*.AuxT *bojí*.Pred *o*.AuxP *otce*.Obj ..AuxK
delete
delete
Petr.Sb *se*.AuxT *bojí*.Pred ..AuxK
* *Se*.AuxT *bojí*.Pred *o*.AuxP *otce*.Obj ..AuxK
delete
shift
* *Se*.AuxT *bojí*.Pred ..AuxK
Bojí.Pred *se*.AuxT *o*.AuxP *otce*.Obj ..AuxK
shift
delete
Bojí.Pred *se*.AuxT ..AuxK

Fig. 1. The schema of AR for sentence (1)

Our example sentence can be simplified in two ways:
(i) either by deletion of the prepositional group *o otce* ‘about father’ (according to the correctness constraint on the simplified sentence, the pair of word forms must be deleted in a single step, see the left branch of the scheme);

[1] For the simplicity, only lexical categories (i.e., original word forms and punctuation like full stop in the example) as they appear in the sentence and syntactic categories (like predicate (Pred), subject (Sb), object (Obj), auxiliary words (AuxT, AuxP, AuxK)) are displayed in the examples; see [7] for more detailed description.

(ii) or by deleting the subject *Petr* (the right part of the scheme); however, the simple simplification would result in the incorrect word order variant starting with the clitic *se* (such position of a clitic is forbidden in Czech); thus the shift operation is enforced $\rightarrow_{shift}$ *Bojí se o otce.* '(he) worries about his father.'. The reduction proceeds in a similar way in both branches of AR until the minimal correct simplified sentence *Bojí se.* '(He) worries.' is obtained. This sentence cannot be further correctly reduced.

3 Restarting Automata with Delete and Shift Operation

In order to model the analysis by reduction with shifts, we introduce a restarting automaton that uses a limited number of pebbles, and that performs several deletions and shifts within one meta-instruction – a DS-automaton. The DS-automaton is a refinement of the so called sRL-automaton in [7]; here, the automaton is enriched with the shift operation, and with categorial and basic alphabets.

DS-automata are suitable for modeling AR – these automata make it possible to check the whole input sentence and mark selected words with pebbles prior to any changes. It resembles a linguist who can read the whole sentence first, and then reduce the sentence in a correct way. To enable simulation of various orders of reductions, we choose a nondeterministic model of the automaton. We distinguish three alphabets (or vocabularies): a proper alphabet Σ_p that is used to model individual word forms, an alphabet Σ_c of categories and a basic alphabet Γ. Since only symbols from the basic alphabet can appear on a tape of a DS-automaton, Γ is also called a tape alphabet. (In what follows, λ denotes an empty word, $\mathbb{N}_+$ and $\mathbb{N}$ denote the set of positive and the set of nonnegative integers, respectively.)

More formally, a DS-*automaton* is a tuple $M = (\Sigma_p, \Sigma_c, \Gamma, \text{¢}, \$, R, A, k)$, where Σ_p, Σ_c, and $\Gamma \subseteq \Sigma_p \times \Sigma_c$ are finite alphabets, R and A are finite sets of restarting and accepting meta-instructions, respectively, and $k \in \mathbb{N}$ is the number of pebbles available. M works on a flexible tape (i.e., on a string of symbols from Γ) delimited by the left sentinel ¢ and by the right sentinel \$ ($\text{¢}, \$ \notin \Gamma$). Its computation is controlled by finite sets of meta-instructions R and A, and it makes use of k pebbles $p_1, \cdots, p_k$.

A projection from Γ^* to Σ_p^* and Σ_c^*, respectively, is convenient – we define two homomorphisms, a *proper homomorphism* $h_p : \Gamma \rightarrow \Sigma_p$ and a *categorial homomorphism* $h_c : \Gamma \rightarrow \Sigma_c$ in the obvious way: $h_p([a, b]) = a$, and $h_c([a, b]) = b$ for each $[a, b] \in \Gamma$. For technical reasons, we define $h_p(\text{¢}) = h_c(\text{¢}) = \text{¢}$.

Each computation of a DS-automaton consists of several phases called *cycles*, and a last halting phase called a *tail*. In each cycle, the automaton performs three passes through the tape with symbols from Γ. During the first pass, it marks certain symbols of a processed sentence with pebbles according to some meta-instruction I; then during the second pass, it performs the shift operations as described by the chosen meta-instruction I; and during the third pass, it performs the delete operations as described by the meta-instruction I. The operations are applied only on symbols marked by pebbles. Moreover, we allow the left sentinel to be pebbled as well since we sometimes need to shift some symbols just behind it. However, although pebbled, ¢ is treated differently from other tape symbols; it can neither be deleted nor can it be shifted.

In each accepting tail, the automaton according to a meta-instruction I_{acc} from A halts and accepts the analyzed sentence.

In accordance with the linguistic motivation, the meta-instructions check only the categorial part of tape symbols from Γ.[2]

Restarting Meta-instructions. Each cycle of the DS-automaton M is controlled by a single restarting meta-instruction $I \in R$ of the form

$$I = (E_0, a_1, E_1, \dots a_s, E_s; O_{sh}; O_d; \text{ Restart}) \tag{1}$$

where:

- each E_i, $0 \leq i \leq s$ is a regular language over $\Sigma_c \cup \{\lambda\}$, E_i is called the i-th context of I;
- $a_i \in \Sigma_c \cup \{¢\}$ (for $1 \leq i \leq s \leq k$) indicates that each a_i is marked with the pebble p_i;
- $O_{sh} = o_1, \cdots, o_{p_{sh}}$, $o_j \in \{sh[i,l] \mid 1 \leq i,j \leq s,\ i \neq l\}$, is a sequence of shifting operations performed in the second phase: if $o_j = sh[i,l]$ then it shifts the tape symbol marked with p_i to the position behind the symbol with p_l;
- $O_d = d_1, \cdots, d_{p_d}$, $d_j \in \{\mathrm{dl}[i] \mid 1 \leq i \leq s\}$, is a sequence of delete operations performed in the third phase: if $d_j = \mathrm{dl}[i]$ then it deletes the tape symbol marked by p_i;
- ¢ is neither deleted nor shifted within I.

We require an ‘exclusivity’ of the shift operation: each symbol a_i can be shifted only once (as a maximum); moreover, if a_i is shifted then it cannot be deleted within the same meta-instruction. Formally, if O_{sh} contains the shift operation $\mathrm{sh}[i,l]$ for some i then no other $\mathrm{sh}[i,r]$ can be in O_{sh}; moreover, O_d cannot contain the $dl[i]$ operation.

Each computation of M on the input $w \in \Gamma^*$ starts with the *tape inscription* $¢w\$$. After a nondeterministic choice of a cycle C realizing the guessed restarting meta-instruction I, M nondeterministically marks tape symbols $b_1, \dots, b_s$ by pebbles in accordance with I: it finds a factorization $w = v_0 b_1 v_1 b_2 \dots v_{s-1} b_s v_s$, $0 \leq i \leq s$, $1 \leq j \leq s$ such that $h_c(v_i) \in E_i$, $h_c(b_j) = a_j$ in the first pass. Then M applies the implied sequence of shifts O_{sh} during the second pass, and the implied sequence of deletions during the third pass. If the factorization is not found within the first pass, the automaton gets stuck (and thus it rejects w). Notice that due to regularity of individual E_i's the instruction I can be (nondeterministically) identified within one pass over $¢w\$$.

[2] When considering only categorial symbols as a context we avoid both the problem of data sparsity and the problem of a very large alphabet Σ_p (i.e., lexicon with hundred of thousands word forms for a natural language).

At the end of each cycle, the $\mathsf{Restart}$ operation removes all pebbles from the new tape inscription w' and places the head on the left sentinel. We write $w \vdash^I w'$.

Remember that none of the sentinels can be deleted/shifted and that M is required to execute at least one delete operation during each (restarting) cycle.

If no further cycle is performed, each accepting computation necessarily finishes in a tail performed according to one of the accepting meta-instructions.

Accepting Meta-instructions. Tails of accepting computations are described by a set of accepting meta-instructions A, each of the form:

$$I_{\mathrm{acc}} = (a_1 \dots a_s, \mathsf{Accept}), \tag{2}$$

where a_i are symbols from Σ_c.

The tail performed by the meta-instruction I_{acc} starts with the inscription on the tape ¢z\$; if $h_c(z) = a_1 \cdots a_s$ then M accepts z (we write $z \vdash^{I_{acc}} \mathsf{Accept}$), and the whole computation as well. Otherwise, the computation halts with rejection.

We denote by $u \vdash_M v$ the reduction of u into v performed during one cycle of M (that begins with the tape inscription ¢u\$ and ends with the tape inscription ¢v\$) and by $\vdash_M^*$ the reflexive and transitive closure of $\vdash_M$. We say that $u_1, u_2, \dots, u_n, \mathsf{Accept}$ is an *accepting computation* of M if $u \vdash_M u_1, u_1 \vdash_M u_2, \cdots, u_{n-1} \vdash_M u_n, u_n \vdash_M \mathsf{Accept}$.

A string $w \in \Gamma^*$ is *accepted* by M, if $w \vdash_M^* u$ with $(u, \mathsf{Accept}) \in A$. By $L(M)$ we denote the language of words accepted by M; we say that M recognizes (accepts) the *basic (tape) language* $L(M)$. We say that $L(M,p) = \{h_p(w) \in \Sigma_p^* \mid w \in L(M)\}$ is the *proper language* of M. Analogously, $L(M,c) = \{h_c(w) \in \Sigma_p^* \mid w \in L(M)\}$ is called the *categorial language* of M.

Since the number of reductions performed within an accepting computation is of (linguistic) interest, we denote by $L_n(M)$ the language of all sentences accepted by at most n cycles of M; $L_0(M)$ is the set of sentences accepted directly by accepting meta-instructions.

Further, we define the *reduction language of* M as $RED(M) = \{u \to v \mid u \vdash_M v, u, v \in L(M)\}$, and $RED_n(M) = \{u \to v \in RED(M) \mid v \in L_{n-1}(M)\}$. Note that $L_n(M)$ and $RED_n(M)$ are finite for any $n \in \mathbb{N}$.

The notations $L_n(M,p)$ and $L_n(M,c)$ denote the proper and categorial variants of $L_n(M)$, respectively.

Backward Correctness Preserving Property (bcpp) Realize that each meta-instruction I of the DS-automaton M is *backward correctness preserving*:

$$\big(v \in L(M) \text{ and } u \vdash^I v\big) \Rightarrow (u \in L(M))$$

We will see that this bcpp property plays a crucial role in the study of analysis by reduction.

We naturally suppose that any restarting meta-instruction of M can be associated with some reduction from $RED(M)$. Two conditions formulated below, the map- and mrp- properties, reflect the linguists' preference of reductions being as simple as possible.

Minimal Accepting Property (map-): If $w \in L_0(M)$ then neither proper subsequence of w nor any permutation of such subsequence belongs to $L_0(M)$.

Minimal Reduction Property (mrp-): Let $u \vdash_M v$ for a cycle C realizing the restarting meta-instruction controlled by the sequence of operations $O = o_1, o_2, \cdots, o_p$. Let $\tilde{v}$ be obtained from u by a (new) restarting meta-instruction controlled by a proper subsequence $\tilde{O}$ of O. Then $\tilde{v} \notin L(M)$. The property implies that a meta-instruction is in some sense minimal as none of the deletions/shifts performed by $u \to v$ can be left out in order to obtain a reducible or acceptable sentence.

If M fulfils the map- and mrp- conditions then $RED(M)$ is said to create a *normalized reduction language of* M, and that M is *normalized.* We will show in Section 4 that every DS-automaton can be transformed to a normalized one that recognizes the same proper language.

Example 2. The notions of restarting and accepting meta-instructions are illustrated on the analysis of our example sentence (1) from Section 2. The respective DS-automaton M_{ex} is described by two restarting meta-instructions I_{r_1} and I_{r_2}, and one accepting meta-instruction I_{acc}. It formalizes both branches of AR of the sentence [*Petr*,Sb] [*se*,AuxT] [*bojí*,Pred] [*o*,AuxP] [*otce*,Obj] [.,AuxK] 'Peter worries about his father.', (see the scheme in Section 2).

$I_{r_1} = (E_0^1, a_1^1, E_1^1, a_2^1, E_2^1, a_3^1, E_3^1; o_1^1, o_2^1; \mathsf{Restart})$, where $o_1^1 = \text{dl}[2]$, $o_2^1 = \text{dl}[3]$, $E_0^1 = \{\mathsf{Sb\ AuxT}, \lambda\}$, $a_1^1 = \mathsf{Pred}$, $E_1^1 = \{\lambda, \mathsf{AuxT}\}$, $a_2^1 = \mathsf{AuxP}$, $E_2^1 = \{\lambda\}$, $a_3^1 = \mathsf{Obj}$, $E_3^1 = \{\mathsf{AuxK}\}$;

$I_{r_2} = (E_0^2, a_1^2, E_1^2, a_2^2, E_2^2, a_3^2, E_3^2; o_1^2; o_2^2; \mathsf{Restart})$, where $o_1^2 = \text{sh}[2,3]$; $o_2^2 = \text{dl}[1]$, $E_0^2 = \{\lambda\}$, $a_1^2 = \mathsf{Sb}$, $E_1^2 = \{\lambda\}$, $a_2^2 = \mathsf{AuxT}$, $E_2^2 = \{\lambda\}$, $a_3^2 = \mathsf{Pred}$, $E_3^2 = \{\mathsf{AuxK}, \mathsf{AuxP\ Obj\ AuxK}\}$;

$I_{acc} = (\mathsf{Pred\ AuxT\ AuxK},\ \mathsf{Accept})$.

The computation corresponding to the left branch consists in two cycles. Within the first cycle driven by the meta-instruction I_{r_1}, M_{ex} puts pebbles p_1, p_2, and p_3 on the symbols containing the words [*bojí*,Pred] (with the left (tape) context [*Petr*,Sb][*se*,AuxT] and empty right context), on [*o*,AuxP], and on [*otce*,Obj], respectively; the operations dl[2] and dl[3] delete the words [*o*,AuxP] (marked with the pebble p_2) and [*otce*,Obj] (marked with the pebble p_3), respectively. Then the automaton restarts and removes the pebbles from the processed sentence. Similarly in the second cycle realizing restarting meta-instruction I_{r_2}, M_{ex} marks the respective words and then in the second pass the operation sh[2, 3] shifts the word [*se*,AuxT] with the pebble p_2 to the right of the word [*bojí*,Pred] (with the pebble p_3); in the third pass, the operation dl[1] deletes the word [*Petr*,Sb] (marked with p_1). Finally, accepting instruction I_{acc} just accepts the remaining words. Similarly for the right branch (starting with I_{r_2} and followed by I_{r_1} and I_{acc} instructions).

For the sake of simplicity, let α_0, α_1, α_2, and β_1 be defined as follows:

α_0= [*bojí*,Pred] [*se*,AuxT] [.,AuxK];
α_1= [*Petr*,Sb] [*se*,AuxT] [*bojí*,Pred] [.,AuxK];

$\beta_1 =$ [*boj́ı*,Pred] [*se*,AuxT] [*o*,AuxP] [*otce*,Obj] [.,AuxK], and
$\alpha_2 =$ [*Petr*,Sb] [*se*,AuxT] [*bojí*,Pred] [*o*,AuxP] [*otce*,Obj] [.,AuxK].

Applying the relevant definitions we get:

$L_0(M_{ex}) = \{\alpha_0\}$ as $\alpha_0 \vdash^{I_{acc}} \mathsf{Accept}$;
$L_1(M_{ex}) = \{\alpha_0, \alpha_1, \beta_1\}$, as $\alpha_1 \vdash^{I_{r_2}} \alpha_0$, and $\beta_1 \vdash^{I_{r_1}} \alpha_0$;
$L_2(M_{ex}) = \{\alpha_0, \alpha_1, \beta_1, \alpha_2\}$ since α_2 is the only word not in $L_1(M_{ex})$ for which $\alpha_2 \vdash_{M_{ex}} \alpha, \alpha \in L_1(M)$ (as $\alpha_2 \vdash^{I_{r_1}} \alpha_1$, or $\alpha_2 \vdash^{I_{r_2}} \beta_1$);
$L(M_{ex}) = L_2(M_{ex})$ as no $\alpha \in L_2(M_{ex})$ and $\beta \notin L_2(M_{ex})$ exist for which $\beta \vdash_{M_{ex}} \alpha$ holds.

The proper homomorphism of M_{ex} removes linguistic categories Sb, AuxT, Pred, AuxP, Obj, and AuxK; so the sentence [*Petr*,Sb] [*se*,AuxT] [*bojí*,Pred] [*o*,AuxP] [*otce*,Obj] [.,AuxK] (from the basic language of M_{ex}) is mapped onto the sentence *Petr se bojí o otce.* 'Peter worries about his father.' (from the proper language of M_{ex}). Similarly, the string $\mathsf{Sb\ AuxT\ Pred\ AuxP\ Obj\ AuxK}$ is a sentence of the categorial language of M_{ex}.

4 Results

With respect to the linguistic motivation, we focus on the number of deletions and/or shifts in individual restarting meta-instructions and we use particular abbreviations for automata/languages with a restriction on these complexity measures. In particular, prefix DS- is used to identify the delete-shift automata without any restrictions, and D- is used for automata with deletions only. Further, the prefix $\mathsf{(k)}$- is used to indicate that at most k pebbles are available in one meta-instruction. As a special case, $\mathsf{(0)}$- means that the automaton contains only accepting meta-instructions and thus it accepts in tail computations only. We use the syllable $\mathsf{d(i)}$- for automata with at most i deletions in one meta-instruction and $\mathsf{s(j)}$- for automata with at most j shifts in a single meta-instruction. The requirements for normalized reduction languages are denoted by map-, and mrp-.

For each type X of restarting automata, we use $\mathcal{L}(\mathsf{X})$, $\mathcal{LP}(\mathsf{X})$, $\mathcal{LC}(\mathsf{X})$ to denote the class of all basic, proper and categorial languages, recognizable by automata of this type. Analogously, $\mathcal{RED}(\mathsf{X})$ denotes the class of all reduction languages of these automata. Further, $\mathcal{L}_n(\mathsf{X})$, $\mathcal{LP}_n(\mathsf{X})$, $\mathcal{LC}_n(\mathsf{X})$ denote the classes of basic, proper, and categorial languages defined by at most n reductions of X-automata; $\mathcal{RED}_n(\mathsf{X})$ denotes analogical notion for reduction languages. Proper inclusions are denoted by $\subset$.

FIN, REG, $\mathsf{(D)CFL}$, and CSL are used for classes of finite, regular, (deterministic) context-free, and context-sensitive languages, respectively, and FINR, REGR, $\mathsf{(D)CFR}$, and CSR for the classes of reduction languages defined by DS-automata for which their basic languages are from FIN, REG, $\mathsf{(D)CFL}$, and CSL, respectively.

First part of our results is devoted to the map- and mrp- properties, their influence on the delete and shift complexities and on the computational power. Then we show delete, shift and pebble hierarchies.

Let M_{ex} be the automaton (implicitly) given in Example 2. Let us recall that $L(M_{ex}) = L_2(M_{ex})$. Analyzing its meta-instructions we see that automaton M_{ex} is in fact $\mathsf{map\text{-}mrp\text{-}s(1)\text{-}d(2)\text{-}DS}$-automaton implying $L(M_{ex}) = L_2(M_{ex}) \in \mathsf{FIN}$. Relaxing the conditions map- and mrp-, the finite language $L(M_{ex})$ can obviously be recognized by accepting tails only. The situation changes when the automaton recognizing $L(M_{ex})$ is required to fulfil the condition map-. Then, not only that it cannot accept this finite language by accepting tails only; it requires (at least) two pebbles and two deletions in one restarting meta-instruction. Thus, it can be used for separation.

Proposition 1. *Let M_{ex} be the automaton implicitly given in Example 2. Then*

1. $L(M_{ex}) \in \mathcal{L}_2(\mathsf{map\text{-}d(2)\text{-}DS}) \setminus \mathcal{L}(\mathsf{map\text{-}d(1)\text{-}DS})$,
2. $RED(M_{ex}) \in \mathcal{RED}_2(\mathsf{map\text{-}d(2)\text{-}DS}) \setminus \mathcal{RED}((\mathsf{1})\text{-}\mathsf{DS})$,
3. $L_1(M_{ex}) \in \mathcal{L}_1(\mathsf{map\text{-}s(1)\text{-}d(1)\text{-}DS}) \setminus \mathcal{L}(\mathsf{map\text{-}(1)\text{-}DS})$,
4. $L(M_{ex}) \in \mathcal{L}((\mathsf{0})\text{-}\mathsf{D})$.

Proof. Take the sentence [*Petr*,Sb] [*se*,AuxT] [*boží*,Pred][.,AuxK] from $L_1(M_{ex})$. The single reduction of the sentence follows the map-principle, as *boží se* is a permutation of a correct subsequence *se boží*. The reduction consists of one delete and one shift operations; they are unambiguously given by the backward correctness preserving property (bcpp). Two operations on two different positions imply automaton with at least two pebbles, thus $L_1(M_{ex}) \notin \mathcal{L}(\mathsf{map\text{-}(1)\text{-}DS})$, and $RED(M_{ex}) \notin \mathcal{RED}((\mathsf{1})\text{-}\mathsf{DS})$. Similarly for assertions *3*, and *4* is obvious. □

Consider shifts and deletions being the only operations allowed on (a set of) words. Based on these operations, we can naturally define the partial order $\succ_L$ on the set L of words. We say that u *syntactically precedes* v in L and write $u \succ_L v$ iff:

1. $u, v \in L$, $|u| > |v|$;
2. v can be obtained from u by a sequence O of deletions and shifts applied on u; $u \overset{O}{\to} v$;
3. the application of any proper subsequence O' of O on u would end up with a word outside L.

By $\succ_L^+$ we denote the transitive, nonreflexive closure of $\succ_L$. Obviously, for a DS-automaton M, $u \to v \in RED(M)$ implies $u \succ_{L(M)}^+ v$ and $h_p(u) \succ_{L(M,p)}^+ h_p(v)$.

We define the set $L_{\succ_L}^{min} = \{v \in L \mid \neg\exists u \in L : v \succ_L u\}$ as the set of minimal words in L. Then, for the map-DS-automaton M:

$$L_0(M) = L_{\succ_{L(M)}}^{min}.$$

For $w \in L$ we denote by $\sigma(w)$ any sequence $\sigma(w) = w_0,\ w_1, \ldots,\ w_n$ such that $w = w_0$, $w_{i-1} \succ_L w_i$, $1 \le i \le n$ and $w_n \in L_{\succ_L}^{min}$. We call $\sigma(w)$ the $\succ_L$-sequence of w. Realize that $\succ_L$-sequence of w needs not be uniquely given by L, w.

Note that every pair u, v with $u \succ_L v$ implicitly defines one or more sequences O of deletions and shifts that transforms u into v. For technical reasons, we will only work with sequences of minimal length and will, for every pair $u \succ_L v$, denote one of them as $O(u,v)$. Since the number of deletions and shifts in $O(u,v)$ is determined unambiguously by the length of it, we denote by $S(u \succ_L v)$ the number of shifts, and by $D(u \succ_L v)$ the number of deletions of $O(u,v)$. Let us stress that the minimal number of shifts and deletions needed to transform u into v is a kind of edit-distance between words u,v. Not surprisingly, it will be shown that these numbers are related to the numbers of deletions and shifts used in meta-instructions of the corresponding DS-automaton. For that, we introduce several delete and shift complexities.

By $D_\omega(L) = \max\{D(u \succ_L v); u,v \in L\}$ we denote the *delete upper bound* of L (or of $\succ_L$). Analogously, $S_\omega(L) = \max\{S(u \succ_L v); u,v \in L\}$ denotes the *shift upper bound of L.*

For any word w and its $\succ_L$-sequence $\sigma(w) = w_0, w_1, \ldots, w_n$ we define $D(\sigma(w)) = \max_i\{D(w_{i-1} \succ_L w_i)\}$ and the *delete lower bound* of w with respect to L as $D_\ell(L,w) = \min_{\sigma(w)}\{D(\sigma(w))\}$. For shifts, $S(\sigma(w))$ and $S_\ell(L,w)$ are defined analogously.

For $L_1 \subseteq L_2$ the delete and shift lower bounds of L_1 with respect to L_2 are defined in the following way: $D_\ell(L_1,L_2) = max_{w\in L_1}\{D_\ell(L_2,w)\}$ and $S_\ell(L_1,L_2) = max_{w\in L_1}\{S_\ell(L_2,w)\}$.

We call the DS-automaton *reduced* if each of its meta-instruction uses exactly as many pebbles as it is needed to realize the involved sequence of operations.

A close relation between the complexity of meta-instructions of map- and/or mrp-DS-automata and the above defined upper and lower bounds is formulated in the following Theorem 1.

Theorem 1. *Let M be a reduced* map-mrp-s(i)-d(j)-DS-*automaton, and $LM \in \{L(M), L(M,c)\}$. Then the following holds:*

1. *$u \to v \in RED(M)$ implies $u \succ_{L(M)} v$;*
2. *$S_\ell(LM,LM) \le i \le S_\omega(LM)$, and $D_\ell(LM,LM) \le j \le D_\omega(LM)$;*
3. *($L \subseteq LM$ and $r \le D_\ell(L,LM) - 1$) implies $LM \notin \mathcal{L}$(map-d(r)-DS);*
4. *($L \subseteq LM$ and $r \le S_\ell(L,LM) - 1$) implies $LM \notin \mathcal{L}$(map-s(r)-DS).*

Notice that without the mrp- condition the assertion 1. of Theorem 1 would not hold.

Proof. Assertion 1. directly follows from the definition of the mrp- condition; together with the map- property it implies the lower bounds in assertion 2. The upper bounds in 2. follow from the definition of D_ω and S_ω. To get 3. and 4., realize that M is reduced map-DS-automaton and that $u \vdash_M v$ implies $u \succ_M^+ v$. □

As shown in Theorem 2, map-D-automata are powerful enough for categorial recognition of deterministic CF-languages and proper recognition of all CF-languages. As a corollary of Theorem 3 we even get that every CF-language is a proper language of some map-mrp-DS-automaton.

Theorem 2. DCFL $\subset \mathcal{LC}$(map-D), CFL $\subset \mathcal{LP}$(map-D).

Proof. To make use of two useful results from [6], we need to introduce an extended class of D-automata. By D_{reg}, we denote a class of D-automata whose accepting meta-instructions are of the form $I_a = (L_r, \mathsf{Accept})$, where L_r is a regular language. I_a accepts each $v \in L_r$.

1. Proof of $\mathsf{DCFL} \subset \mathcal{LC}(\mathsf{map\text{-}D})$. The relevant assertion from Proposition 3.8. in [6] can be reformulated as $\mathsf{DCFL} \subset \mathcal{LC}(\mathsf{D}_{reg})$. To prove our proposition it is therefore sufficient to transform the given D_{reg}-automaton to a map-D-automaton. Thus, let M be a D_{reg}-automaton such that $L(M,c) \in \mathsf{DCFL}$. Based on the size of $|L_0(M,c)|$ we distinguish two cases:

(a) If $L_0(M,c)$ is a finite language then it is easy to check whether it fulfils the condition map-. If not, then there obviously exists map-D-automaton A_0 recognizing exactly the language $L_0(M,c)$. The automaton A_0 simulates the leftmost branch of the syntactic precedence $\succ_{L_0(M,c)}$. Then, it suffices to substitute original accepting meta-instructions of M by all of A_0's restarting and accepting meta-instructions.

(b) If $L_0(M,c)$ is not a finite language then $L_0(M,c)$ is still $\in \mathsf{REG}$ and there is a deterministic finite automaton A recognizing $L_0(M,c)$. From A we can construct deterministic D-automaton A_0 whose restarting meta-instructions always delete by the simulating of a rightmost cycle of A and accepting meta-instructions accepts only cycle-free words. Here, by a cycle in a word w we mean such subword u of w that – based on pumping lemma – could be iterated (within the computation on w, the automaton starts reading u in the same state as it leaves it). Replacing accepting meta-instructions of M by all meta-instructions of A_0, we get a D-automaton that according to (a) can be modified to an equivalent map-D-automaton recognizing $L(M,c)$.

2. Proof of $\mathsf{CFL} \subset \mathcal{LP}(\mathsf{map\text{-}D})$. It directly follows from Proposition 3.4. in [6] that $\mathsf{CFL} \subset \mathcal{LP}(\mathrm{D}_{reg})$ which – based upon the above given construction for DCFL – implies $\mathsf{CFL} \subset \mathcal{LP}(\mathsf{map\text{-}D})$. □

Adopting the above given constructions we show in the next theorem that as for the proper languages, the linguistic requirement of normalization preserves the power of DS-automata.

Theorem 3. *Let* $X \in \{\mathsf{DS}, \mathsf{D}\}$. *Then* $\mathcal{LP}(\mathsf{X}) = \mathcal{LP}(\mathsf{map\text{-}mrp\text{-}X})$.

Proof. Let us prove the theorem by explaining how a DS-automaton for a proper language can be transformed onto one with the map- and mrp- properties. We will show the construction for D-automata first and secondly we will explain how the construction can be adopted for automata with shifts. Note that based on the proof of Theorem 2 we can suppose that the original D-automaton already posseses the map- property.

Let us start with the proof of $\mathcal{LP}(\mathsf{D}) = \mathcal{LP}(\mathsf{map\text{-}mrp\text{-}D})$. It is obvious that $\mathcal{LP}(\mathsf{map\text{-}mrp\text{-}D}) \subseteq \mathcal{LP}(\mathsf{D})$. To show the opposite inequality we start with a map-D-automaton $M = (\Sigma_p, \Sigma_c, \Gamma, ¢, \$, R, A, k)$ and we construct a map-mrp-D-automaton $M_D = (\Sigma_p, \Sigma_c^D, \Gamma_D, ¢, \$, R_D, A_D, k_D)$ with an enriched set of categories Σ_c^D and a new basic alphabet Γ_D.

Based on any individual accepting computation $\mathcal{C}_\mathcal{A}$ of M on w, where $h_c(w) = w_1 \dots w_n$, each symbol w_j can be unambiguously associated with either restarting or accepting meta-instruction $I(j)$ and with an index p_j identifying either the pebble that is within realization of the restarting meta-instruction $I(j)$ put on it or that identifies the position of w_j within the accepting meta-instruction $I(j)$; if w_j is deleted within $\mathcal{C}_\mathcal{A}$ then $I(j)$ is that restarting meta-instruction that has deleted it, otherwise $I(j)$ is the accepting meta-instruction applied in $\mathcal{C}_\mathcal{A}$. To be able to insert this information into w we enrich the categorial alphabet Σ_c of M: $\Sigma_c^D = \{[x,i,r] \ : \ x \in \Sigma_c, i \in \{1, \dots |R \cup A|\}, r \leq \ell$, where $\ell = \max\{k, \max\{s; (a_1, \dots a_s;\ \mathsf{Accept}) \in A\}\}$.

The order of meta-instructions in $R \cup A$: If $I = (E_0, a_1, E_1, \dots a_s, E_s; O_{sh}; O_p;$ $\mathsf{Restart}) \in R$ is i-th in the ordering then put $I(i) = (E_0, [a_1, i, 1], E_1, \dots [a_s, i, s],$ $E_s;\ O_{sh}; O_p\ \mathsf{Restart})$ into R_D. Analogously, if $I_{acc} = (a_1, \dots a_s;\ \mathsf{Accept})$ is j-th then put the accepting meta-instruction $I_{acc}(j) = ([a_1, j, 1], \dots [a_s, j, s];\ \mathsf{Accept})$ to A_D.

Realize that the proper alphabet Σ_p has not been changed, thus $L(M,p) = L(M_2,p)$:

– if $u \vdash_{M_D} v$ then there are x, y such that $x \vdash_M y$, and $h_p(x) = h_p(u), h_p(y) = h_p(v)$;
– $\forall x, y \in L(M)$, $x \vdash_M y$ there are $x_2, y_2 \in L(M_D)$ such that $x_2 \vdash_{M_D} y_2$, and $h_p(x_2) = h_p(x), h_p(y_2) = h_p(y)$

Due to the index of the instruction and the position in that instruction associated with individual symbols as described above, the obtained D-automaton M_2 fulfills both the mrp- and map- conditions.

To finish the proof of the theorem $\mathcal{LP}(\mathsf{DS}) = \mathcal{LP}(\mathsf{map\text{-}mrp\text{-}DS})$, we explain how information about shifts can be handled using the idea of the above given construction. Let $M = (\Sigma_p, \Sigma_c, \Gamma, ¢, \$, R, A, k)$ be the $\mathsf{map\text{-}DS}$-automaton. We will construct $\mathsf{map\text{-}mrp\text{-}DS}$-automaton $M_{DS} = (\Sigma_p, \Sigma_c^{DS}, \Gamma^{DS}, ¢, \$, R_{DS}, A_{DS},$ $k_{DS})$ with an enriched set of categories $\Sigma_c^{DS} = \{[x,i,r] \ : \ x \in \Sigma_c, i \in \{1, \dots |R \cup$ $A|\}, r \leq \ell$, where $\ell = \max\{k, \max\{s; (a_1, \dots a_s;\ \mathsf{Accept}) \in A\}\}$.

Within a restarting meta-instruction I some pebbles are put on symbols that are to be deleted and some of them on those moved by I. As described earlier, the index r in the triple $[x,i,r]$ is associated either with pebble put on the original symbol x within the meta-instruction that deletes the symbol or with the position of x in the accepting meta-instruction. Thus, to realize restarting meta-instruction I that involves shifts, we will simply ignore the index in those triples that are moved by I.

Fix any ordering of meta-instructions in $R \cup A$. If $I_{acc} = (a_1, \dots a_s;\ \mathsf{Accept})$ is j-th in that ordering then put accepting meta-instruction $I_{acc}(j) = ([a_1, j, 1],$ $\dots [a_s, j, s];\ \mathsf{Accept})$ to A_{DS}.

Let $I = (E_0, a_1, E_1, \dots a_s, E_s; O_{sh}; O_d;\ \mathsf{Restart}) \in R$ be a restarting meta-instruction of M and $O_d = \{dl[j_1], \dots, dl[j_d]\}$ is the set of all delete operations involved in I. If I is j-th in the ordering then add the set of restarting meta-instructions $I(j) = \{(E_0, [a_1, i_1, 1], E_1, \dots [a_s, i_s, s], E_s; O_{sh}; O_d;\ \mathsf{Restart}) \mid i_j =$ j for $j \in \{j_1, \dots, j_d\}\}$ into R_{DS}.

Realize that the set of restarting meta-instructions of M_{DS} is larger than that of M if at least one of M's restarting meta-instructions involves shift. However, fixing the accepting computation $\mathcal{C}_{\mathcal{A}}$ of M on w, the transformation of $w = w_1 \dots w_n \in \Gamma^*$ to $W = W_1 \dots W_n = [w_1, i_1, r_1] \dots [w_n, i_n, r_n] \in \Gamma_{DS}^*$ such that $h_p(w) \in L(M,p) \iff h_p(W) \in L(M_{DS}, p)$ can be done by a deterministic algorithm:

1. initialize $W = w$;
2. let $I_1, I_2, \dots, I_t$ be a sequence of meta-instructions corresponding to $\mathcal{C}_{\mathcal{A}}$, $I_1, \dots, I_{t-1} \in R, I_t \in A$;
3. let $I_t = (a_1 \dots a_s;\ \mathsf{Accept})$ be j_t-th in the ordering of $R \cup A$, where $a_1 = w_{i_1}, \dots, a_s = w_{i_s}$; set W_{i_m} to $[w_{i_m}, j_t, m]$;
4. for $q = t-1$ *downto* 1 let $I_q = (E_0, a_1, E_1, \dots a_s, E_s; O_{sh}; O_d;\ \mathsf{Restart})$ be j_q-th in the ordering of $R \cup A$, where $a_1 = w_{i_1}, \dots, a_s = w_{i_s}$ and $p_1, \dots, p_d$ are the pebbles put on those symbols deleted by I_q; set $W_{i_{p_i}}$ to $[w_{i_{p_i}}, q_t, p_i]$.

Thanks to this deterministic process the indices i, r in triples $[x, i, r]$ guarantee both the map- and mrp- property of the constructed DS-automaton M_{DS}. $\square$

Corollary 1. $\mathsf{CFL} \subset \mathcal{LP}(\text{map-mrp-D})$

It is easy to see that $\mathcal{LC}(\mathsf{X})$ is a subset of $\mathcal{LP}(\mathsf{X})$ for $X \in \{D, DS\}$ and that every computation of restarting automaton can directly be simulated in linear space implying $\mathcal{LC}(\mathsf{X}) \subseteq \mathcal{LP}(\mathsf{X}) \subseteq \mathsf{CSL}$. These inequalities are shown as valid with the help of separation languages L_{e} and L_{a2b} (see Proposition 2). The lower bound parts of both Propositions are mainly based on counting argument combined with consequences of pumping lemma for regular languages.

Proposition 2. $L_{\mathrm{e}} = \{ a^{2^n} \mid n \in \mathbb{N} \} \in \mathsf{CSL} \setminus \mathcal{LP}(\mathsf{DS})$,
$L_{a2b} = \{ a^n b^n \mid n \geq 1 \} \cup \{ a^n b^m \mid m > 2n > 0 \} \in \mathcal{LP}((3)\text{-D}) \setminus \mathcal{LC}(\mathsf{DS})$

Corollary 2. *Let* $X \in \{\mathsf{DS}, \mathsf{D}\}$. *Then* $\mathcal{LC}(\mathsf{X}) \subset \mathcal{LP}(\mathsf{X}) \subset \mathsf{CSL}$.

In order to show the delete and shift hierarchies, we define two classes of sample languages. Let $j \in \mathbb{N}_+$, $i \in \mathbb{N}$, $\Sigma = \{P, b, s\}$, $\Delta_j = \{c, a_1, a_2, \dots, a_j\}$, $\Lambda = \{\lambda\}$:

$$LS(j,i) = \{\ Ps^i\{b^j\}^+, \{b^j\}^+ s^i, s^i \}, \qquad Le(j) = \{ a_1^n a_2^n \cdots a_j^n \mid n > 0 \}.$$

The construction of relevant DS-automata and delete and shift complexities of these languages are given in Lemma 1. It is easy to see that all above defined classes of languages belong to CSL, languages $LS(j,i)$ are infinite regular, $Le(2) \in \mathsf{CFL}$, and $Le(j) \in \mathsf{CSL} \setminus \mathsf{CFL}$ for $j \geq 3$.

Lemma 1. *Let* $\mathsf{X} = \text{map-mrp}$, $\mathcal{LX} \in \{\mathcal{LC}, \mathcal{LP}\}$, $j \in \mathbb{N}_+$, $i \in \mathbb{N}$. *Then:*

(a) $LS(j,i) \in \mathcal{LX}(\text{d(j)-s(i)-X-DS})$ *(b)* $LS(j,i) \in \mathcal{LX}((\max\{\text{j, i+2}\})\text{-X-DS})$
(c) $Le(j) \in \mathcal{LX}(\text{d(j)-X-D})$ *(d)* $Le(j) \in \mathcal{LX}((\text{j})\text{-X-D})$

Proof. The proof is done by an informal construction of DS-automata.
(a) We describe a d(j)-s(i)-map-mrp-DS-automaton $MS(j,i)$ such that $LS(j,i) = L(MS(j,i),c) = L(MS(j,i),p)$, and it uses max{j, i+2} pebbles. $MS(j,i)$ works with the basic alphabet {[P,P], [b,b], [s,s]}, and categorial and proper alphabets equal to {P, b, s}. The automaton $MS(j,i)$ simulates the leftmost syntactic precedence of any word of $LS(j,i)$); we have three possibilities for one cycle:

– the word $[P,P][s,s]^i\{[b,b]^j\}^n$ is changed to $\{[b,b]^j\}^n[s,s]^i$; for this, $i+2$ pebbles are used to mark the symbol $[P,P]$, all symbols $[s,s]^i$ and the last symbol $[b,b]$ first; then $[s,s]^i$ are shifted after $[b,b]$'s and $[P,P]$ is deleted;
– the prefix $[b,b]^j$ is marked with pebbles and deleted;
– the word $[s,s]^i$ is accepted in a tail computation.

(b) Here we describe a d(j)-map-mrp-D-automaton $Me(j)$ such that $Le(j) = L(Me(j),c) = L(Me(j),p)$, and it uses j pebbles. The automaton $Me(j)$ it simulates always the leftmost syntactic precedence for any word from $Le(j)$; in one cycle the automaton marks by pebbles and deletes one copy of $[a_1,a_1], [a_2,a_2], \ldots, [a_j,a_j]$ in a word longer then j and accepts the word $[a_1,a_1][a_2,a_2]\ldots[a_j,a_j]$ in a tail computation.

It is not hard to see that the described automata fulfill the map- and mrp-conditions. □

Now we are ready for our separation results; the very robust pebble hierarchy is dealt with in the following part, shift and delete hierarchies are given afterwards.

Theorem 4. *Let $i > 0$, $X \in$ {DS, D}, $Y \in \{\lambda,$ map, map-mrp$\}$, $\mathcal{LX} \in \{\mathcal{L}, \mathcal{LC}, \mathcal{LP}, \mathcal{RED}\}$. Then* $\mathcal{LX}$(Y-(i)-X) $\subset \mathcal{LX}$(Y-(i+1)-X).

Proof. To prove the proposition, we consider the sequence of above defined languages $Le(j) = \{\, a_1^n a_2^n \cdots a_j^n \mid n > 0\}$, $j > 1$.

For the upper bound consider the automaton $Me(j)$ from the proof of Lemma 1b. Its categorial and proper languages are both equal to $Le(j)$ and the automaton uses exactly j deletions and pebbles in a cycle.

The lower bound parts are based on Theorem 1. For each DS-automaton M recognizing $Le(j)$, the set $L_0(M)$ is finite, thus $RED(M)$ is nonempty; any reduction from $RED(M)$ is forced by some syntactic precedences from $\succ_{Le(j)}$. Realize that if $u \succ_{Le(j)} v$ then $|u| = |v| + j$; v can be obtained from u by j deletions for which M uses at least j pebbles. This way we get the desired hierarchies for basic, categorial and reduction languages.

For proper languages hierarchy realize that for any DS-automaton M with proper homomorphism h_p, each relation $u \succ_{L(M)} v$ implies the existence of at least one relation of the form $h_p(u) \succ^+_{L(M,p)} h_p(v)$; from this, $D_\ell(L(M), L(M)) \geq D_\ell(L(M,p), L(M,p))$ follows. This means that the pebble complexity of every DS-automaton with the proper language $Le(j)$ is at least $D_\ell(Le(j), Le(j))$ and thus at least j. □

The subsequent results show the existence of infinite pebble hierarchies even on classes of finite sub-languages of DS-automata. Note that the hierarchies for reduction languages are more robust than for the other types of languages; they hold also without the map- and mrp- conditions. The language $Le(j)$ represents a core of the corresponding proofs.

Theorem 5. *Let* $n > 0$, $i > 0$, $X \in \{\mathsf{D}, \mathsf{DS}\}$, $Y \in \{\mathsf{map}, \mathsf{map\text{-}mrp}\}$, $\mathcal{LX} \in \{\mathcal{L}, \mathcal{LC}, \mathcal{LP}, \mathcal{RED}\}$.
Then $\mathcal{LX}_n(\mathsf{Y\text{-}(i)\text{-}X}) \subset \mathcal{LX}_n(\mathsf{Y\text{-}(i{+}1)\text{-}X})$, *and* $\mathcal{RED}_n(\mathsf{(i)\text{-}X}) \subset \mathcal{RED}_n(\mathsf{(i{+}1)\text{-}X})$.

Theorem 6. *For* $i > 2, j \geq 0$, $Y \in \{\mathsf{FIN}, \mathsf{REG} \setminus \mathsf{FIN}, \mathsf{CFL} \setminus \mathsf{REG}, \mathsf{CSL} \setminus \mathsf{CFL}\}$, $X \in \{\mathsf{map}, \mathsf{map\text{-}mrp}\}$, $\mathcal{LX} \in \{\mathcal{L}, \mathcal{LC}, \mathcal{LP}\}$ *we have the following proper inclusions:*

(a) $Y \cap \mathcal{LX}(\mathsf{d(i)\text{-}s(j)\text{-}X\text{-}DS}) \subset Y \cap \mathcal{LX}(\mathsf{d(i{+}1)\text{-}s(j)\text{-}X\text{-}DS})$,

(b) $Y \cap \mathcal{LX}(\mathsf{d(i)\text{-}s(j)\text{-}X\text{-}DS}) \subset Y \cap \mathcal{LX}(\mathsf{d(i)\text{-}s(j{+}1)\text{-}X\text{-}DS})$.

Proof. To separate $\mathcal{LX}(\mathsf{map\text{-}d(i)\text{-}s(j)\text{-}DS})$ from $\mathcal{LX}(\mathsf{map\text{-}d(i{+}1)\text{-}s(j)\text{-}DS})$ and $\mathcal{LX}(\mathsf{map\text{-}d(i)\text{-}s(j{+}1)\text{-}DS})$ we use the languages $LS(i, j)$, and their syntactic precedences. The number of shift operations is forced by the map- property, otherwise the language $LS(i, j)$ could also be recognized with a D-automaton without shifts simply by deleting the sufix $[b, b]^i$ from $[P, P][s, s]^j\{[b, b]^i\}^n$, for $n > 1$, instead. The number of deletions in one restarting instruction is also determined by the map- property that forces the instruction to delete $[b, b]^i$ from the proper prefix of $\{[b, b]^i\}^n[s, s]^j$. Since $LS(i, j)$ is infinite regular, the proof for $Y = \mathsf{REG} \setminus \mathsf{FIN}$ follows.

For remaining classes we use the languages obtained by a small modification of $LS(i, j)$: $L_{\mathsf{FIN}}(i, j) = \{Ps^jb^i, b^is^j, s^j\}$, $L_{\mathsf{CFL}\setminus\mathsf{REG}}(i, j) = LS(i, j) \cup Le(2)$, and $L_{\mathsf{CSL}\setminus\mathsf{CFL}}(i, j) = LS(i, j) \cup Le(3)$. □

The automata from the previous theorem can be used to prove the similar hierarchical results for reduction languages even with the absence of the map- and mrp- conditions. Moreover, hierarchies similar to those in Theorems 5 and 6 hold also for finite languages defined by n reductions.

Conclusion and Perspectives

We have presented a class of restarting automata (DS-automata), which formalize lexicalization in a similar way as categorial grammars (see e.g. [1, 2]). This class of automata – similarly as categorial grammars – allows us to introduce (in a natural way) basic languages (on word forms marked with categories), proper languages (on unmarked word forms), and categorial languages (on grammatical categories). Further, they allow to introduce reduction languages – this concept is quite natural for DS-automata and the analysis by reduction.

We have introduced also the minimalist map- and mrp- properties, which were used for the normalization of DS-automata; these properties ensure similar and transparent hierarchies for classes of finite and infinite languages. Note that relaxation of these minimalist properties often leads to different results for classes of finite and infinite languages. The normalized DS-automata formalize the notion of analysis by reduction.

Reduction languages allow for explicit description of the integration of individual (disambiguated) word forms into a sentence structure. While proper languages play the role of input languages for weak equivalence of DS-automata (and other types of automata or grammars), reduction languages serve for linguistically more relevant strong equivalence of DS-automata.

Based on [4], we estimate that roughly seven deletions in one reduction step suffice to analyze adequately any sentence (not containing coordination in its structure) from the Prague Dependency Treebank [3]. As for the shift complexity, we have only been able to find reductions of Czech sentences with at most one shift in a single reduction step. From this point of view, normalized reductions in natural languages are quite simple. The information stored in morphological lexicons of individual natural languages is in fact modeled by the information contained in the basic (tape) alphabet of DS-automata. On the other hand, meta-instructions model syntactic potential of individual words (i.e., information stored in valency lexicons and a grammar component of a natural language description).

We have already used the analysis by reduction for explaining the basics of dependency syntax of Czech (see e.g. [7]). However, it can be used for explanation of basic issues of lexicalized syntax based on (even discontinuous) constituents as well – in such a case, individual (restarting) meta-instructions of a normalized description correspond to individual types of constituents. The proposed type of strong equivalence and three proposed types of very robust complexity measures can serve for both types of syntactic methods.

Finally, we strongly believe that for linguistic applications, (relatively simple) star-free languages are sufficient as contexts in meta-instructions since the main information contained in a context is that the context cannot contain some special subwords (or some simple symbols as, eg., punctuation symbols).

References

1. Ajdukiewicz, K.: Die syntaktische Konnexität. Studia Philosophica I, 1–27 (1935)
2. Chomsky, N.: Formal Properties of Grammar. In: Handbook of Mathematical Psychology, vol. II, pp. 323–418. John Wiley and Sons, Inc. (1963)
3. Hajič, J., Panevová, J., Hajičová, E., Sgall, P., Pajas, P., Štěpánek, J., Havelka, J., Mikulová, M., Žabokrtský, Z., Ševčíková-Razímová, M.: Prague Dependency Treebank 2.0. Linguistic Data Consortium, Philadelphia (2006)
4. Kuboň, V., Lopatková, M., Plátek, M.: On Formalization of Word Order Properties. In: Gelbukh, A. (ed.) CICLing 2012, Part I. LNCS, vol. 7181, pp. 130–141. Springer, Heidelberg (2012)
5. Mráz, F.: Lookahead Hierarchies of Restarting Automata. Journal of Automata, Languages and Combinatorics 6(4), 493–506 (2001)
6. Mráz, F., Otto, F., Plátek, M.: The degree of Word-Expansion of Lexicalized RRWW-automata: A New Measure for the Degree of Nondeterminism of (Context-Free) Languages. Theoretical Computer Science 410(37), 3530–3538 (2009)
7. Plátek, M., Mráz, F., Lopatková, M. (In)Dependencies in Functional Generative Description by Restarting Automata. In: Bordihn, H., et al. (eds.) Proceedings of NCMA 2010. books@ocg.at, vol. 263, pp. 155–170. Österreichische Computer Gesellschaft, Wien (2010)