ON THE COMPLEXITY OF REDUCTIONS BY RESTARTING AUTOMATA

Martin Plátek[(A)] Markéta Lopatková[(A)]
Dana Pardubská[(B)]

[(A)]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

[(B)]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 analysis by reduction (AR). It concentrates on a study of the whole mechanism through a class of restarting automata with meta-instructions using pebbles, delete, and shift operations (*DS*-automata). The complexity of* DS*-automata is naturally measured by the number of* deletions*, and the number of* word order shifts *used in a single meta-instruction. We study reduction languages, backward correctness preserving AR, correctness preserving AR, and show unbounded hierarchies (scales) for various classes of reduction languages by the same finite witness languages. The scales make it possible to estimate relevant complexity issues of analysis by reduction for natural languages.*

1 Introduction

Analysis by reduction (AR) plays an important role in lexicalized syntax ([1]) of many natural languages. It consists in a stepwise simplification of a sentence, which serves for identifying the syntactic structure of the sentence, and the corresponding lexical categories as well.

We study formal models for the analysis by reduction, where a *delete* operation may be accompanied with a *word order shift*, an operation reflecting the word order freedom of natural languages [3]. Section 2 provides linguistic motivation and an informal description of the process.

[(A)]The paper reports on the research supported by the grants of GAČR No. P202/10/1333, and No. P103/10/0783.

[(B)]The third author was supported by the Slovak Grant Agency for Science (VEGA) under the contract No. 1/0979/12.

The core sections, sections 3 and 4 provide a formal study of the whole mechanism through a refined class of restarting automata with meta-instructions (DS-automata), and their descriptional complexity based on the number of deletions, and on the number of word order shifts used within a single meta-instruction. With the help of these measures, we are able to show that several features of natural languages (e.g., Czech) can be simply described using AR.

Four types of (in)finite sets defined by DS-automata in [6] are relevant: (basic) languages on word forms marked with their (linguistic) categories, sets of reductions on basic languages (reduction languages), proper languages on unmarked word forms, and categorial languages on pure categories. The equivalence of proper languages can be considered as the *weak equivalence* (close to the weak equivalence by formal automata and grammars), and the equivalence between reduction languages as the (linguistically finest) *strong equivalence* between DS-automata. This paper enhances the technical results from [6] with results about backward correctness preserving AR, correctness preserving AR, and some new results about reduction languages.

Formal parts of this article are based on descriptional and editing complexity of selected classes of reduction languages. We focus on those aspects of complexity of meta-instructions that are shown by finite witness (reduction) languages. The obtained results give the theoretical background for the incremental transfer from finite (linguistic) observations (as in [3]) to adequate, fully lexicalized, formal descriptions of (potentially) infinite natural languages based on sentence reductions.

2 Analysis by Reduction

(Surface) analysis by reduction (AR) helps one to identify a sentence syntactic structure and the corresponding grammatical categories of an analyzed language. AR is based upon a stepwise simplification of an analyzed sentence, see [3]. 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 shortening the sentence); here, we allow deleting to be accompanied by a *shift* of (a) word(s) to another position(s) in the sentence.

Let us stress the basic constraints imposed on reduction steps of surface 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 principle of correctness

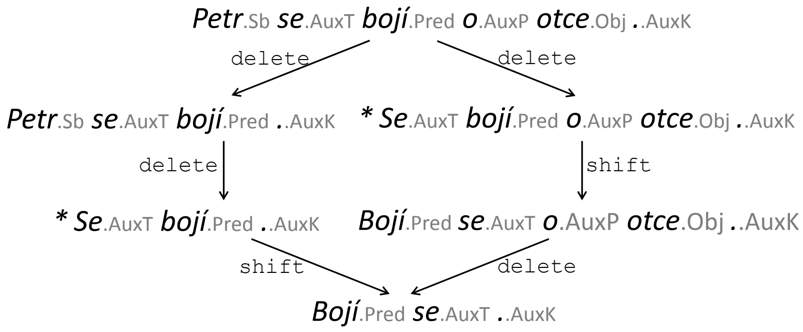
(iv) a sentence which contains a correct sentence (or its permutation) as a subsequence, must be further reduced;

(v) specially, an application of the shift operation is limited only to cases when a shift is enforced by the correctness preserving principle (ii), i.e., simple deletions would result in an incorrect word order.

Note that the possible order(s) of reductions reflect(s) 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 2.1
(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

Figure 1: The schema of AR for sentence (1).

The analysis by reduction of sentence (1) can be summarized with the scheme in Fig.1.

Our example sentence can be simplified in two ways (for simplicity, we do not make distinction between upper and lower case letters at the beginning of the sentence; when deleting and shifting individual word forms, also corresponding categories are deleted and shifted, respectively):
(i) either by deleting the prepositional group *o otce* 'for father' (according to the correctness constraint on the simplified sentence (ii), the pair of word forms must be deleted in a single step; see the left branch of the scheme);
(ii) or by deleting the subject *Petr* (the right part of the scheme); however, this simplification results 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.'.
As these possible reductions are independent of each other, we can conclude that the words *Petr* and *o otce* 'for father' are independent – in other words, both of them depend on (or modify) some word(s) remaining in the simplified sentence, i.e., the verb and its clitic *bojí se* '(he) worries'.

Then, the reduction proceeds in a similar way in both branches of AR until the minimal correct simplified sentence $\rightarrow$ *Bojí se.* '(He) worries.' is obtained. This sentence cannot be further reduced.

[1]For the simplicity, only original word forms (as they appear in the sentence) and their syntactic categories (like predicate (Pred), subject (Sb), object (Obj), auxiliary words (AuxT, AuxP)) are displayed in the examples. For a more detailed description, see [7].

As was stated above, the order of reductions reflects the dependency relations, which are usually encoded in the form of a so called dependency tree, see [4]. Informally, the words are 'cut from the bottom of the tree'; i.e., a governing node must be preserved in a simplified sentence until all its dependent words are deleted.[2]

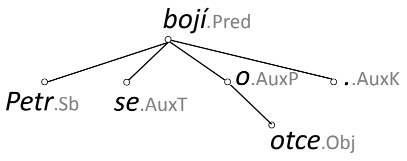


Figure 2: The dependency tree of sentence (1).

The following two sentences illustrate the fact that – despite relatively free word-order in Czech – a meaning of a Czech sentence may depend on word-order (at least for certain phenomena). Consequently, some additional constraints concerning word order must be put on the corresponding analysis by reduction.

Example 2.2
(2) *Petr*.Sb *se*.AuxT *bojí*.Pred *otce*.Obj *matky*.Atr *(svého) souseda*.Atr..AuxK
Peter – REFL – is scared of – father$_{gen}$ – mother$_{gen}$ – (his) neighbor$_{gen}$
'Peter is scared of the father of the mother of (his) neighbor.'

Example 2.3
(3) *Petr*.Sb *se*.AuxT *bojí*.Pred *souseda*.Obj *otce*.Atr *(své) matky*.Atr..AuxK
Peter – REFL – is scared of – neighbor$_{gen}$ – father$_{gen}$ – (his) mother$_{gen}$
'Peter is scared of the neighbor of the father of his mother.'

The meaning of the previous two similar sentences is obviously different (both sentences are correct despite the word order changes). The difference is caused by the different order in the sequence of nouns in genitive case. In such cases, we set additional technical constraints on the analysis by reduction which allows AR to preserve a meaning of the simplified sentence: individual genitives are obligatory deleted in the prescribed direction from the right to the left.

Using this additional constraint, the reduction (of the second part of) sentence (2) obligatory starts with deleting the word *(svého) souseda* '(his) neighbor'; then, deletion of the word *matky* 'mother' proceeds; finally, the word *otce* 'father' is reduced. On the other hand, in (3) analysis by reduction starts with *(své) matky* '(his) mother', then *otce* 'father' is reduced and AR is completed with deletion of the word *souseda* 'neighbor'. In both sentences, the word *Petr* 'Peter' can be deleted any time in the course of the analysis by reduction; however, this reduction enforces the shift operation correcting the resulting word order (as (ii) in sentence (1), right branch of the scheme).

[2]As described in the cited article, the relation between the preposition and its 'head' noun as well as the verb and this type of clitic is rather technical from our point of view as they must be reduced in a single step, despite being represented as two nodes in the dependency tree. Here we adhere to the practice used in the large collection of linguistically annotated data, the Prague Dependency Treebank [2]). In other words, AR makes it possible to identify the dependency tree for sentence (1):

The change in the meaning in (2) and (3) is reflected in the corresponding dependency trees, see Fig. 3 and 4.

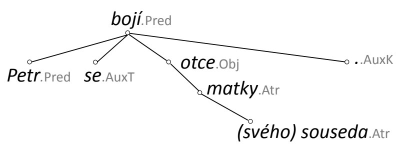


Figure 3: The dependency tree of sentence (2).

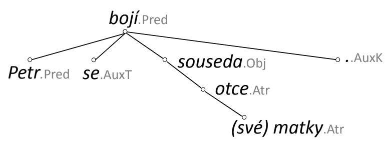


Figure 4: The dependency tree of sentence (3).

3 Automata with Delete and Shift Operations

In what follows, λ denotes the empty word, $\mathbb{N}_+$ and $\mathbb{N}$ denote the set of positive and the set of nonnegative integers, respectively.

In order to model the analysis by reduction with shifts, we introduce a restarting automaton that is allowed to use a limited number of pebbles, and that can perform several deletions and shifts within one meta-instruction – a DS-automaton. The DS-automaton is a refinement of the so called simple restarting sRL-automaton in [7]; the automaton is enriched with the shift operation here, and with a composed basic alphabet, which is important for analysis by reduction.

The class of DS-automata is introduced for modeling (more advanced) AR – these automata allow for checking the whole input sentence (and marking selected words with pebbles) prior to any changes. It resembles a linguist who can read the whole sentence first, and who can reduce the sentence in a correct way. To enable simulation of various orders of reductions, we choose a nondeterministic model of the automaton.

A DS-*automaton* $M = (\Sigma_p, \Sigma_c, \Gamma, ¢, \$, R, A, k)$ is (in general) a nondeterministic machine with a finite proper alphabet Σ_p (modeling individual word forms), an alphabet of categories Σ_c, and a composed tape (or basic) alphabet $\Gamma \subseteq \Sigma_p \times \Sigma_c$. The DS-automaton M works on a flexible tape (i.e., a string of symbols from Γ) delimited by the left sentinel ¢ and the right sentinel \$ ($¢, \$ \notin \Gamma$). M is controlled by finite sets of restarting meta-instructions R and accepting meta-instructions A, respectively, and makes use of k pebbles $p_1, \cdots, p_k$.

To realize a projection from Γ^* to Σ_p^* and Σ_c^*, respectively, we define two homomorphisms, a *proper homomorphism* $h_p : \Gamma \to \Sigma_p$ and a *categorial homomorphism* $h_c : \Gamma \to \Sigma_c$ in the obvious way: $h_p([a, b]) = a$, and $h_c([a, b]) = b$ for each $[a, b] \in \Gamma$.

Each computation of a DS-automaton consists of several phases called cycles, and a last halting phase called 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 and third passes, it performs the shift and delete operations, respectively, as described by the chosen meta-instruction

I; the operations are applied (only) on the symbols marked by pebbles. In each (accepting) tail, the automaton – according to a selected 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 (Σ_c part) of tape (basic) symbols from Γ.[3]

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

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

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 pebble p_i;
- $O_{sh} = o_1, \cdots, o_{p_s}$, $o_j \in \{sh[i,l] \mid 1 \leq i, j \leq s, \ i \neq l\}$, 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;
- $O_d = o_1, \cdots, o_{p_d}$, $o_j \in \{\mathrm{dl}[i] \mid 1 \leq i \leq s\}$, is a sequence of delete operations performed in the third phase: if $o_j = \mathrm{dl}[i]$ then it deletes the tape symbol marked by p_i;
- ¢ is neither deleted nor shifted within I.

Let us note that $E_0 = \lambda$ and $a_1 = ¢$ correspond to a pebble p_1 put on the left sentinel ¢. 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 the 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. Thus, 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 nondeterministically be identified within one pass over $¢w\$$.

At the end of each cycle, the Restart operation removes all pebbles from the new tape inscription w' and places the head on the left sentinel. We write $w \vdash_I w'$, $h_c(w) \vdash_I^c h_c(w')$, and $h_p(w) \vdash_I^p h_p(w')$.

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

[3]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).

Accepting meta-instructions. Tails of accepting computations are described by a set of accepting meta-instructions A, each of the form: $I_{\text{acc}} = (a_1 \ldots a_s, \mathsf{Accept})$, 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 is accepted as well. Otherwise, the computation halts with rejection.

We denote by $u \vdash_M v$ the reduction of u to v by M 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 $u_1, u_2, \ldots, u_n, \mathsf{Accept}$ is an *accepting computation* of M if $u \vdash_M u_1, u_1 \vdash_M u_2, \cdots, u_{n-1} \vdash_M u_n, u_n \vdash_M \mathsf{Accept}$. We can also write $h_c(u) \vdash^c_M h_c(v)$, and $h_p(u) \vdash^p_M h_p(v)$ if $u \vdash_M v$ holds.

A string $w \in \Gamma^*$ is *accepted* by M, if $w \vdash^*_M u$ for some word u such that $(h_c(u), \mathsf{Accept}) \in A$. By $L(M)$ we denote the language consisting of all words accepted by M; we say that M recognizes (accepts) the *basic language* $L(M)$. Let $L(M,p) = \{h_p(w) \in \Sigma_p^* \mid w \in L(M)\}$. We say that $L(M,p)$ is the *proper language* of M. Let $L(M,c) = \{h_c(w) \in \Sigma_p^* \mid w \in L(M)\}$. We call $L(M,c)$ *categorial language* of M.

Since the number of reductions performed within an accepting computation is of (linguistic) interest, we denote by $L_0(M)$ the set of sentences accepted directly by accepting meta-instructions; similarly, $L_n(M)$ denotes the language of all sentences accepted by at most n cycles of M. Obviously, $L(M) = \bigcup_n L_n(M)$.
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 $L_n(M,p)$, $L_n(M,c)$ denote the proper and categorial variants of $L_n(M)$, respectively. Moreover, the notations $RED(M,p)$, $RED(M,c)$ denote the proper and categorial variants of $RED(M)$, respectively.

Backward Correctness Preserving Property (bcp) Realize, that each meta-instruction I of a DS-automaton M is *backward correctness preserving*: $(v \in L(M) \text{ and } u \vdash_I v) \Rightarrow (u \in L(M))$. We will see that *(bcp)* 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, map- and mrp- properties reflect the linguists' preference on as simple reductions as possible.

Minimal Accepting Property (map-): If $w \in L_0(M)$, then neither any proper subsequence of w nor any permutation of such a 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 $O = O_{sh}O_d = o_1, o_2, \cdots, o_p$. Let $\tilde{v}$ be obtained from u by a (new) restarting meta-instruction controlled by a proper subsequence $\tilde{O}$ of O. Then $\tilde{v} \notin L(M)$. (In a sense, the meta-instructions are minimal as none of the shifts/deletions performed by $u \to v$ can be left out in order to obtain a reducible or acceptable sentence.)

If M fulfils the conditions map-, mrp-, then it is said that M is *normalized.*

Correctness Preserving Property (cp) We say that a meta-instruction I of a DS-automaton M is *correctness preserving* if $(u \in L(M)$ and $u \vdash_I v) \Rightarrow (v \in L(M))$. We say that M is correctness preserving if all its meta-instructions are correctness preserving.
Realize that any DS-automaton M which for any $v \in L(M)$ defines at most one reduction, and for $v \in L_0(M)$ defines no reduction, is correctness preserving. On the other hand there are DS-automata which are not correctness preserving. The correctness preserving property (cpp) ensures in general a transformation to a weakly equivalent deterministic restarting automaton (e.g., see [5]). Note that the correctness preserving property ensures an adequate simulation of manual analysis by reduction with the help of DS-automata.

While the definition of the meta-instruction of DS-automata provides a tool for the description of individual syntactic phenomena of natural languages, we will choose complexity measures of meta-instructions that will help us to classify the complexity of such phenomena.

4 Results

The focus of the paper is on the (global) power of DS-automata and the complexity of individual meta-instructions. In particular, several results from [6] are shown to hold also for correctness preserving DS-automata and new hierarchies for reduction languages are formulated. For hierarchies, we consider the number of pebbles, deletions and/or shifts in individual restarting meta-instructions.

We use particular abbreviations for automata/languages with restrictions on these complexity measures. In particular, prefix DS- is used to identify the delete-shift automata without any restriction, and D- is used for automata with deletions only. Further, the prefix (k)- is used to indicate that at most k pebbles are available in one meta-instruction. As a special case, (0)- means that the automaton contains only accepting meta-instructions and thus accepts in tail computations only. We use the syllable d(i)- for automata with at most i deletions in one meta-instruction and s(j)- for automata with at most j shifts in a single meta-instruction. The basic requirements for normalized reduction languages are denoted by map-, and mrp-, and cp- means correctness preserving property. The prefix nr- is used for normalized automata and languages.

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, and $\mathcal{REC}(\mathsf{X})$, $\mathcal{REP}(\mathsf{X})$ mean classes of categorial and proper variants of reduction languages. 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 $\subset$.

Here, FIN, REG, $\mathsf{(D)CFL}$, and CSL are used for classes of finite, regular, (deterministic) context-free, and context-sensitive languages, respectively, and $\mathsf{FINR(X)}$, $\mathsf{REGR(X)}$, and $\mathsf{(D)CFR(X)}$, are used for the classes of reduction languages defined by X-automata (where $\mathsf{X} \in \{\mathsf{D}, \mathsf{DS}\}$), for which their categorial (and basic) languages are from FIN, REG, and $\mathsf{(D)CFR}$, respectively. We can easily see that $\mathsf{FINR(X)} \subseteq \mathsf{REGR(X)} \subseteq \mathsf{DCFR(X)} \subseteq \mathsf{CFR(X)} \subseteq \mathcal{RED}(\mathsf{X})$ for $\mathsf{X} \in \{\mathsf{D}, \mathsf{DS}\}$. We will show that all inclusions are in fact proper.

Consider shifts and deletions being the only operations allowed on (a set of) strings. The delete operation is determined by the place of the deletion. By shift we mean a transfer of a symbol within a string from one position in the string to a different one. Based on these operations, we can naturally define the partial order $\succ_L$ on the set L of strings. We say that u DS-*precedes* v in L and write $u \succ_L v$ iff:

1. $u, v \in L$, $|u| > |v|$;
2. v is a permutation of a proper subsequence of u that results from the application of a sequence O of shifts and deletions on u; $u \overset{O}{\to} v$;
3. the application of any proper subsequence O' of the sequence of operations O on u would end up with a word v' such that $v' \notin L$.

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

The partial order $\succ_L$ naturally defines the set $L_{\succ_L}^0$ of minimal words in L:
$L_{\succ_L}^0 = \{v \in L \mid \neg\exists u \in L : v \succ_L u\}$, and further by induction: $L_{\succ_L}^{n+1} = \{v \in L \mid \exists u \in L_{\succ_L}^n : v \succ_L u\}$. In what follows, we will alternatively use $\succ_L$ as abbreviation for a set of DS-precedences $\{u \to v \mid u \succ_L v\}$. The meaning will always be clear from the context.

For $w \in L$ we denote by $\sigma(w)$ any sequence $\sigma(w) = w_0,\ w_1, \ldots,\ w_n$ such that $w = w_0$, $w_{i-1} \succ_L w_i$, $1 \leq i \leq n$ and $w_n \in L_{\succ_L}^0$. We call $\sigma(w)$ the $\succ_L$-sequence of w, and write $w \succ_L^+ w_n$. Notice that a $\succ_L$-sequence of w needs not be uniquely given by L, w.

Note that every pair u, v with $u \succ_L v$ implicitly defines some (one or more) sequences O of deletions and shifts that transforms u to v ($u \overset{O}{\to} v$). 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 to 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
$D(\sigma(w)) = \max_i\{D(w_{i-1} \succ_L w_i)\}$; further, we define the *delete lower bound* of w with respect to L by $D_\ell(L, w) = \min_{\sigma(w)}\{D(\sigma(w))\}$. The *shift lower bounds* $S(\sigma(w))$ and $S_\ell(L, w)$ are

defined analogously. Further, let $L_1 \subseteq L_2$. The technically useful 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-instructions uses exactly as many pebbles as are needed to realize the involved sequence of operations.

The reduction languages correspond to possible steps by which the individual sentences were reduced until they fell to a finite set of correct words. The number of steps by which an individual w is reduced to one accepted in a tail computation can in fact vary; the reduction can be detailed or can "skip" some intermediate steps. Thus, the reduction languages are in general only an approximation of the linguistic notion of an analysis by reduction. A more precise definition of surface analysis by reduction follows. It reflects the linguists interest in the formulation of as simple as possible reduction rules that within a reduction process allow as large as possible sets/sequences of correct sentences.

Refined variants of AR. Let M be a DS-automaton. We say that $RED(M)$ is a *bcp-analysis by reduction (bcp-AR)* of $L(M)$ (or M) if $L_0(M) = L^0_{\succ_{L(M)}}$, and $RED(M)$ is a subset of $\succ_{L(M)}$. If M is correctness preserving, then (bcp-AR) $RED(M)$ is a *cp-analysis by reduction (cp-AR)* of $L(M)$.

Our approach formalizes the linguistic effort to gradually prepare sets of categories for correctness preserving analyzes by reduction. A close relation between bcp-analysis by reduction, the complexity of meta-instructions of nr-DS-automata and the above defined upper and lower bounds of DS-complexity are formulated in Theorem 4.1 which follows from a similar theorem from [6] and its proof.

Theorem 4.1. *Let M be a reduced* nr-s(i)-d(j)-DS-*automaton. Then the following holds:*

1. *$RED(M)$ is a bcp-AR of M;*
2. *$L_n(M) \subseteq L^n_{\succ_{L(M)}}$;*
3. *[6] $S_\ell(L(M), L(M)) \leq i \leq S_\omega(L(M))$, and $D_\ell(L(M), L(M)) \leq j \leq D_\omega(L(M))$;*
4. *[6] ($L \subseteq L(M)$ and $r \leq D_\ell(L, L(M)) - 1$) implies $L(M) \notin \mathcal{L}($map-d(r)-DS$)$;*
5. *[6] ($L \subseteq L(M)$ and $r \leq S_\ell(L, L(M)) - 1$) implies $L(M) \notin \mathcal{L}($map-s(r)-DS$)$.*

   Note that assertions 2,3,4 and 5 hold also for categorial languages.

The above theorem is often used for separation results. Notice that without the nr-conditions the assertions 1. and 2. of Theorem 4.1 would not hold.

Let M be a DS-automaton and $RED(M)$ its bcp-analysis by reduction ; then it is not hard to see that M is an nr-DS-automaton. On the other hand, reduction languages can substantially differ from DS-precedences and bcp-analyzes by reductions in case of unrestricted DS-automata.

Consider the language $L_1 = \{a^n b^n | n \in \mathbb{N}_+\}$. For any $j \in \mathbb{N}_+$ a D-automaton M_j can easily be constructed such that $L(M_j, c) = L_1$, $L_0(M_j, c) = \{a^n b^n | 1 \leq n \leq j\}$, and $RED(M_1, c) = \{a^n b^n \to a^{n-j} b^{n-j} | n > j\}$. Realize that $L^0_{\succ_{L_1}} = \{ab\}$ and $\succ_{L_1} = \{a^n b^n \succ_{L_1} a^{n-1} b^{n-1} | n > 1\}$

implying that, for $j > 1$, automaton M_j categorially accepts words $ab, a^2b^2, \ldots, a^jb^j$ in an accepting tail. Obviously, $L_0(M_j, c)$ differs from $L^0_{\succ_{L_1}}$ illustrating that we are able to construct arbitrarily many different D-automata categorially recognizing L_1 with mutually different reduction languages which do not create a bcp-analysis by reduction of these automata.

Classes of AR. We use cp-$\mathcal{AR}(\mathsf{X})$ and (bcp-$\mathcal{AR}(\mathsf{X})$) to denote the class of all (b)cp-AR of DS-automata of the type X. Further, (b)cp-$\mathcal{AR}_n(\mathsf{X})$ denote the classes of (b)cp-AR defined with at most n reductions of X-automata.

4.1 On the Power of DS-Automata and Their Normalization

This subsection demonstrates the power of DS-automata by categorial, proper and reduction languages, and their relation to the Chomsky hierarchy. It is therefore natural that presented separation results are (and have to be) based on infinite witness languages. As shown in Corollary 4.3, $\mathsf{cp\text{-}map}$-D-automata are powerful enough for categorial recognition of deterministic CF-languages and proper recognition of all CF-languages. Theorem 4.2 and Corollary 4.3 are stronger variants of results from [6]. With almost the same proofs the correctness preserving property of simulating automata can be achieved.

Theorem 4.2. *Let* $X \in \{\mathsf{DS}, \mathsf{D}, \mathsf{cp\text{-}DS}, \mathsf{cp\text{-}D}\}$. *Then* $\mathcal{LP}(\mathsf{X}) = \mathcal{LP}(\mathsf{nr\text{-}X})$.

Corollary 4.3. $\mathsf{DCFL} \subset \mathcal{LC}(\mathsf{cp\text{-}nr\text{-}D})$, $\quad \mathsf{CFL} \subset \mathcal{LP}(\mathsf{cp\text{-}nr\text{-}D})$, $\quad \mathcal{REP}(\mathsf{DS}) = \mathcal{REP}(\mathsf{nr\text{-}DS})$

Remark. Realize, that the above given results document the usefulness of the model, as the linguistic requirement of normalization preserves the recognizing power of DS-automata as well as analysis of proper languages. The correctness preserving property and regular languages are dealt with in the next theorem.

Theorem 4.4. *1. Let M be a DS-automaton with $L(M, c)$ that is a regular language. Then there is a $\mathsf{cp\text{-}DS}$-automaton M_1 such that $RED(M) = RED(M_1)$, i.e., M_1 is strongly equivalent to M.*

2. For any regular language L there is $n \in \mathbb{N}$ and a $\mathsf{cp\text{-}D}$-automaton M such that $L = L(M, c)$, $RED(M)$ is a cp-AR of M, where for any $u \in L(M, c)$, $|u| > n$ there is $u \to v \in RED(M, c)$ such that $u = u_1u_2$, $v = v_1u_2$, where $u_1 \leq n$.

3. Let M be a $\mathsf{cp\text{-}D}$-automaton, and $n \in N$ such that for any $u \in L(M, c)$, $|u| > n$ there is $u \to v \in RED(M, c)$ such that $u = u_1u_2$, $v = v_1u_2$, where $|u_1| \leq n$. Then $L(M), L(M, c)$ are regular.

Proof. At first we outline the proof of the first assertion. Since one transition over the analyzed word is sufficient for recognition of regular languages, the applicability of a candidate meta-instruction I can easily be combined with the computation of a DFA A recognizing $L(M, c)$. As a result, each meta-instruction I of M can be transformed to a correctness preserving I_1 of M_1 such that I_1 fulfills both the I and A requirements.

Outline of the proof of 2. Let us suppose that A is a deterministic reduced minimal finite automaton recognizing L, n be the number of its states. Every regular language $L(A)$ can be seen as a finite set of cycle free words $F(A)$ and a finite set of cycles $Cyc(a)$ of length at most n such that every word $w \in L(A)$ either belongs to $F(A)$ or can gradually be reduced by replacement of the leftmost $y \in Cyc(A)$ by λ to some $v \in F(A)$. Straightforward simulation of described process results in bcp-D-automaton, combination with assertion 1 gives cp-D-automaton whose reduction language might not form an AR. Thus, let us outline how the cp-D-automaton M should reduce in one cycle a word $w \in L$, $|w| > n$. Let $w = xyz$, where y corresponds to the leftmost cycle in a computation of A on w. Then M will substitute y by some of its shortest scattered subwords v such that $xvz \in L$; realize that the decision is in fact done on the right sentinel after w. Similarly if $n \geq |w|$ then M substitutes w by shortest scattered subword v of w such that $v \in L$. If no such v exists then M accepts $w \in L(M, c)$. Note that M rejects $w \notin L(M, c)$ in the first cycle.

Outline of the proof of 3. It is not hard to see that M can be simulated by a one way automaton with a finite buffer. That is, M can be simulated by a finite automaton. □

Theorem 4.5. *[6] Let $X \in \{\mathsf{DS}, \mathsf{D}\}$. Then* $\mathcal{LC}(\mathsf{X}) \subset \mathcal{LP}(\mathsf{X}) \subset \mathsf{CSL}$.

Lemma 4.6. *[6] Let M be a DS-automaton. Then there is a constant p such that, for each of its restarting meta-instructions I, and for each of its contexts E the following holds: each $w \in E$, $|w| > p$ can be written as $w = xyz$, where $0 < |yz| < p$, and $xy^iz \in E$ for $i \geq 0$.*

For linguists, the correctness preserving property is desired within the analysis as no realized reduction can destroy the analyzed sentence. To ensure this property, the reduction of the analyzed sentence should proceed by a sequence of very careful steps. Our first example of a categorial language for which correctness preserving property cannot be guaranteed follows.

Proposition 4.7. *Let $L_c = \{a^n b^m | n, m \in \mathbb{N}, n \leq m \leq 2n\}$. Then no DS-automaton M such that $L(M, c) = L_c$ is correctness preserving.*

Proof. Suppose for a contradiction that there is a correctness preserving (k)-DS-automaton M such that $L(M, c) = L_c$. Its categorial alphabet needs to be $\{a, b\}$, the proper, and the basic alphabet are irrelevant for the proof. Without loss of generality, suppose that the categorial and proper languages of M are the same.

Let us suppose that M contains a meta-instruction $I = (E_0, a_1, E_1, \ldots a_s, E_s; O_{sh}; O_d; \ \mathsf{Restart})$, $s \leq k$, which deletes twice as many b's as a's. Let p be the number ensured by Lemma 4.6 for I, m be the length of the longest word accepted by an accepting meta-instruction. Consider n divisible by $p!$, $n > \max\{k + (k+1)p, m\}$ and word $w = a^n b^{2n} \in L(M, c)$. Then, the above mentioned meta-instruction I is applicable to w realizing a reduction $w = a^n b^{2n} \vdash^I a^{n_1} b^{2n_1}$. More precisely, let $n - n_1 = \Delta$. Then I reduces $w = x_0 a_1 x_1 \ldots a_s x_s$ to $a^{n_1} b^{2n_1} = x_0 b_1 x_1 b_1 \ldots b_s x_s$, $b_i \in \{a_i, \lambda\}$ for all $1 \leq i \leq k$. The choice of n together with Lemma 4.6 guarantee the existence of $0 \leq j_a \leq s$ such that $x_{j_a} = \alpha_a \beta_a \gamma_a$, $\beta_a \in a^+$, $0 < |\beta_a| < p$, $\alpha_a \beta_a^t \gamma_a \in E_{j_a}$ for all $t \geq 0$. The choice $t = 1 + n/|\beta_a|$ implies the applicability of I on $a^{2n} b^{2n}$; i.e., $a^{2n} b^{2n} \vdash_I^c a^{2n-\Delta} b^{2n-2\Delta}$.

Since $a^{2n}b^{2n} \in L(M,c)$, it follows from the correctness preserving property that $a^{2n-\Delta}b^{2n-2\Delta} \in L(M,c)$, which is a contradiction.

The assumption $L(M,c) = L_c$, and the correctness preserving property of M ensures the existence of such a meta-instruction I in a similar way as the proof above. □

Corollary 4.8. *For* $X \in \{\mathsf{D}, \mathsf{DS}\}$ *ve have*
(a) $\mathsf{FINR}(\mathsf{X}) \subset \mathsf{REGR}(\mathsf{X}) \subset DCFR(\mathsf{X}) \subset CFR(\mathsf{X}) \subset \mathcal{RED}(\mathsf{X})$,
(b) $cp\text{-}\mathcal{AR}(\mathsf{X}) \subset bcp\text{-}\mathcal{AR}(\mathsf{X}) \subset \mathcal{RED}(\mathsf{X})$.
Further, for $\mathcal{RX} \in \{\mathsf{FINR}, \mathsf{REGR}, \mathcal{DCFR}, \mathcal{CFR}, \mathcal{RED}\}$ *we have* $\mathcal{RX}(\mathsf{D}) \subset \mathcal{RX}(\mathsf{DS})$.

4.2 Scales

In this subsection we focus on DS-automata which are not necessarily normalized. In order to show some delete and shift hierarchies related to the Chomsky hierarchy, we use two families of infinite sample languages; as before, the finite witness languages are not enough. Let $j, m \in \mathbb{N}_+$, $i \in \mathbb{N}$, $\Sigma = \{P, b, s\}$, $\Delta_j = \{a_1, a_2, \ldots, a_j\}$, $\Lambda = \{\lambda\}$ then we define:
$LS(j,i) = \{\ Ps^i\{b^j\}^+, \{b^j\}^+s^i, s^i\}$, and $Le(j) = \{\, a_1^n a_2^n \cdots a_j^n \mid n > 0\,\}$.

The construction of relevant DS-automata and delete and shift complexities of these languages are given in the following lemma. The lemma is slightly stronger than a similar lemma from [6], where the correctness preserving property was not required. Note that the languages $LS(j,i)$ are infinite regular, $Le(2) \in \mathsf{CFL} \setminus \mathsf{REG}$, and for $j \geq 3$ it holds $Le(j) \in \mathsf{CSL} \setminus \mathsf{CFL}$.

Lemma 4.9. *For* $\mathcal{LX} \in \{\mathcal{LC}, \mathcal{LP}\}$, $j \in \mathbb{N}_+$, $i \in \mathbb{N}$ *we have:*
(a) $LS(j,i) \in \mathcal{LX}(\text{d(j)-s(i)-cp-nr-DS})$ *(b)* $Le(j) \in \mathcal{LX}(\text{d(j)-cp-nr-D})$.

Proof. The proof is done by an informal construction of DS-automata. Although similar to that from [6], it is given here for better understanding the results given below.

(a) We describe a d(j)-s(i)-cp-nr-DS-automaton $MS(j,i)$ such that $LS(j,i) = L(MS(j,i),c) = L(MS(j,i),p)$, and which uses $\max\{j, i+2\}$ pebbles. The automaton $MS(j,i)$ 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 DS-precedence of any word of $LS(j,i)$); we have two possibilities for one cycle, and one possibility for a tail:

- 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 symbol $[P,P]$, all symbols $[s,s]^i$ and the last symbol $[b,b]$ first; then $[P,P]$ is deleted and $[s,s]^i$ are shifted after $[b,b]$'s;
- 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)-cp-nr-D-automaton $Me(j)$ such that $Le(j) = L(Me(j),c) = L(Me(j),p)$, and it uses j pebbles. The automaton $Me(j)$ works with the basic alphabet $\{[a_1,a_1],...,[a_j,a_j]\}$ and categorial and proper alphabets equal to Δ_j. It simulates always the

leftmost DS-precedence for any word from $Le(j)$; in one cycle, the automaton marks by pebbles and deletes the first copy of $[a_1, a_1], [a_2, a_2], \ldots, [a_j, a_j]$ from a tape word longer then j. The automaton $Me(j)$ 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 cp-nr- condition. □

Theorem 4.10. *For* $i > 2, j \geq 0$, $Y \in \{\mathsf{REGR}, \mathsf{CFR} \smallsetminus \mathsf{REGR}, \mathcal{RED}(\mathsf{DS}) \smallsetminus \mathsf{CFR}\}$, $\mathcal{RX} \in \{\mathcal{RED}$, *bcp-*$\mathcal{AR}$,*cp-*$\mathcal{AR}\}$, *we have the following proper inclusions:*

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

Proof. To separate $\mathcal{RED}(\mathsf{d(i)\text{-}s(j)\text{-}DS})$ from $\mathcal{RED}(\mathsf{d(i{+}1)\text{-}s(j)\text{-}DS})$ and $\mathcal{RED}(\mathsf{d(i)\text{-}s(j{+}1)\text{-}DS})$ we analyze the automata for witness languages $LS(i, j)$ and their reduction languages. The proper inclusion can be shown applying the technique proposed in Theorem 4.1.

– $LS(j, i)$; for corresponding automaton $MS(i, j)$ its reduction language $RED(MS(i, j))$ belongs to REGR.
– $L_{\mathsf{CFL} \setminus \mathsf{REG}}(i, j) = LS(i, j) \cup Le(2)$; the corresponding automaton $M_{CR}(i, j)$ contains all the meta-instructions from $MS(i, j)$, and all meta-instructions from $Me(2)$; obviously, $RED(M_{CR}(i, j))$ is from CFR and from *cp-*$\mathcal{AR}$, and is not from REGR.
– $L_{\mathsf{CSL} \setminus \mathsf{CFL}}(i, j) = LS(i, j) \cup Le(3)$, the correspoding automaton $M_{CSCF}(i, j)$ contains all the meta-instructions from $MS(i, j)$, and all meta-instructions from $Me(3)$; obviously, $RED(M_{CSCF}(i, j))$ is from *cp-*$\mathcal{AR}$, and is not from CFR.

We can see that all DS-automata above described use reductions with i deletions, and reductions with j shifts; further, the number of deletions does not depend on the number of shifts, and vice versa. The second automaton uses also reductions with two deletions, and the third one with three deletions. Thus, the presented separation results are valid for $i > 2$, $j \geq 0$. □

Separations by finite witness languages. The last results of this section deal with languages of reductions and analysis by reduction. Unlike the results concerning categorial and basic languages (see [6]), these hierarchical results can be achieved by (rather small) finite witness languages. Such small witness languages are closer to the technique of linguistic (syntactic) observation.

Proposition 4.11. *For* $j \in \mathbb{N}_+$, $n > 1$, $i \in \mathbb{N}$ *we have:*
$RED_n(MS(j, i)) \in$ *cp-*$\mathcal{AR}(\mathsf{d(j)\text{-}s(i)\text{-}DS})$, $RED_n(MS(j+1, i)) \notin \mathcal{RED}(\mathsf{d(j)\text{-}s(i)\text{-}DS})$, $RED_n(MS(j, i+1)) \notin \mathcal{RED}(\mathsf{d(j)\text{-}s(i)\text{-}DS})$.

Proof. The proof follows from Lemma 4.9 and Theorem 4.1. □

To show the next corollary, it suffices to consider $RED_2((MS(j, i))$, i.e., a sequence of reduction languages consisting only of two reductions.

Corollary 4.12. *For $i > 2, j \geq 0, n > 0$, $\mathcal{RX} \in \{\mathcal{RED}$, bcp-$\mathcal{AR}$,cp-$\mathcal{AR}\}$, we have the following proper inclusions:*

(a) $\mathcal{RX}(\mathsf{d(i)\text{-}s(j)\text{-}DS}) \subset \mathcal{RX}(\mathsf{d(i{+}1)\text{-}s(j)\text{-}DS})$, (b) $\mathcal{RX}(\mathsf{d(i)\text{-}s(j)\text{-}DS}) \subset \mathcal{RX}(\mathsf{d(i)\text{-}s(j{+}1)\text{-}DS})$,

(c) $\mathcal{RX}_n(\mathsf{d(i)\text{-}s(j)\text{-}DS}) \subset \mathcal{RX}_n(\mathsf{d(i{+}1)\text{-}s(j)\text{-}DS})$, (d) $\mathcal{RX}_n(\mathsf{d(i)\text{-}s(j)\text{-}DS}) \subset \mathcal{RX}_n(\mathsf{d(i)\text{-}s(j{+}1)\text{-}DS})$.

Theorem 4.13. *Let $n > 0, i > 0, j \geq 0$, $\mathcal{RX} \in \{\mathcal{RED}$, bcp-$\mathcal{AR}$, cp-$\mathcal{AR}\}$. Then*
(a) $\mathcal{RX}_{n-1}(\mathsf{d(i)\text{-}s(j)\text{-}DS}) \subset \mathcal{RX}_n(\mathsf{d(i)\text{-}s(j)\text{-}DS})$, *(b)* $\mathcal{RX}_{n-1}(\mathsf{DS}) \subset \mathcal{RX}_n(\mathsf{DS})$.

Proof. To prove the theorem, we use the DS-precedence of $Le(j) = \{\, a_1^n a_2^n \cdots a_j^n \mid n > 0\,\}$. For any word $w = a_1^n \cdots a_j^n \in Le(j)$, $|w| > j, n > 0$ there is exactly one word α such that $w \succ_{Le(j)} \alpha$, namely $a_1^n \cdots a_j^n \succ_{Le(j)} a_1^{n-1} \cdots a_j^{n-1}$. That is why $RED(Me(j), c) = \{u \to v | u \succ_{Le(j)} v\}$ for D-automaton $Me(j)$ described above.

On the other hand, any DS-automaton M with the reduction language $RED(Me(j), c)$ needs to perform exactly the reductions $a_1^n \cdots a_j^n \succ_{Le(j)} a_1^{n-1} \cdots a_j^{n-1}$ for any $n > 1$. Obviously $RED_{n-1}(M, c)$ is a proper subset of $RED_n(M, c)$, i.e., $RED_{n-1}(M, c)$ and $RED_n(M, c)$ are different, and serve for separation of cp-$\mathcal{AR}_n$ from $\mathcal{RED}_{n-1}$. That proves the proposition. □

5 Conclusion and Perspectives

In this paper we have enhanced and deepened results from [6]: we have refined the results by the correctness preserving property and we have introduced and studied two new formal variants of analysis by reduction. We have (in a uniform way) obtained several infinite hierarchies of all variants of analyses by reduction by (rather small) finite witness languages. That demonstrates the similarity to linguistic observations about basic syntactic phenomena. We have also added hierarchies based on the number of reductions. These new results on reduction languages differ in some aspects from the results on basic and categorial languages from [6], as e.g., the hierarchies on reduction languages use finite witness languages more often. Motivated by linguistic techniques, we defined four types of languages in [6] – basic, proper, categorial, and reduction languages. The reduction languages (linguistically most important) allow for an explicit description of the integration of individual disambiguated word forms into the sentence structure. While proper languages play a 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 [3], we estimate that roughly seven deletions in one reduction step suffice to analyze adequately any sentence from the Prague Dependency Treebank (PDT), a collection of 50.000 Czech sentences annotated with rich morphological and syntactic information, see [2]. 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, AR of natural languages is quite simple. The information which is obtained from dictionaries of individual natural languages is in fact modeled by the information contained in the basic (tape) alphabet of DS-automata.

Moreover, the average number of reductions necessary for processing (any branch of) analysis by reduction of a sentence from PDT can be estimated on 20; the upper bound reaches approx. 120 reductions for PDT.

We have already used the analysis by reduction for explaining the basics of dependency syntax of Czech (e.g., see [7]), but it can as well be used for explanation of the basic issues of lexicalized syntax based on (even discontinuous) constituents. We propose a type of strong equivalence which 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. In the future we plan to study the models of restarting automata which simulate at the same time analysis by reduction and dependency analysis of sentences of natural languages.

References

[1] K. AJDUKIEWICZ, Die syntaktische Konnexität. *Studia Philosophica* I (1935), 1–27.

[2] J. HAJIČ, J. PANEVOVÁ, E. HAJIČOVÁ, P. SGALL, P. PAJAS, J. ŠTĚPÁNEK, J. HAVELKA, M. MIKULOVÁ, Z. ŽABOKRTSKÝ, M. ŠEVČÍKOVÁ-RAZÍMOVÁ, *Prague Dependency Treebank 2.0*. Linguistic Data Consortium, Philadelphia, 2006.

[3] V. KUBOŇ, M. LOPATKOVÁ, M. PLÁTEK, On Formalization of Word Order Properties. In: A. GELBUKH (ed.), *Theoretical Computer Science and General Issues, Computational Linguistics and Intelligent Text Processing, CICLing 2012*. LNCS 7181, Springe Berlin Heidelberg, 2012, 130–141.

[4] M. LOPATKOVÁ, M. PLÁTEK, V. KUBOŇ, Modeling Syntax of Free Word-Order Languages: Dependency Analysis by Reduction. In: V. MATOUŠEK ET AL. (ed.), *Proceedings of TSD 2005*. LNCS 3658, Springer, Berlin Heidelberg, 2005, 140–147.

[5] F. MRÁZ, F. OTTO, M. PLÁTEK, 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 (2009) 37, 3530–3538.

[6] M. PLÁTEK, M. LOPATKOVÁ, D. PARDUBSKÁ, On Minimalism of Analysis by Reduction by Restarting Automata. In: G. MORRILL ET AL. (ed.), *Formal Grammar 2014*. LNCS 8612, Springer, Berlin Heidelberg, 2014, 155–170.

[7] M. PLÁTEK, F. MRÁZ, M. LOPATKOVÁ, (In)Dependencies in Functional Generative Description by Restarting Automata. In: H. BORDIHN ET AL. (ed.), *Proceedings of NCMA 2010*. books@ocg.at 263, Österreichische Computer Gesellschaft, Wien, Austria, 2010, 155–170.