A Grammatical View of Language Evolution

Gemma Bel-Enguix[1], Henning Christiansen[2] and M. Dolores Jiménez-López[1]

[1]Research Group on Mathematical Linguistics, Universitat Rovira i Virgili
Avd. Catalunya 35, 43002 Tarragona, Spain

[2]Research Group on Programming, Logic and Intelligent Systems
Roskilde University, P.O.Box 260, DK-4000 Roskilde, Denmark

Abstract. Language evolves gradually through its use: over time, new forms come into fashion and others become obsolete. While traditionally a grammar provides a snapshot of an individual's or a society's linguistic competence at a given point in time, our aim is to extend grammars to incorporate competences related to evolution. This paper shows how language evolution can be modeled using Adaptable Grammars, which may be defined as logically based transformational grammars in which the grammar itself may be affected in a derivation step.

1 Introduction

During the last decade, approaches to natural language have undergone a deep transformation thanks to a new interdisciplinary paradigm that integrates artificial intelligence, physics, evolutionary biology and computer science [1, 2]. Under such influences, a new interest has arisen for diachronic change of natural language, based on the understanding of language as a complex adaptive system [3] and evolutionary system [4]. This interest is supported by computational models that provide simulations to understand the dynamics of human language and its adaptation to the environment [5]. However, grammatical and formal approaches, as we suggest in the present paper, are still to be applied. The main problems approached in language evolution are the origins and emergence of language [6], language acquisition [2], and language change [7]. Moreover, in the broad area of formal languages, there exists from the sixties a growing interest in grammatical inference or grammar induction [8, 9]. This is a specialized subfield of machine learning that deals with the learning of formal languages from a set of data. Grammar induction refers, therefore, to the process of learning grammars and languages from a given corpora. In order to solve a grammar induction problem we require, on one hand, a teacher that provides data to a learner, and on the other hand, a learner that from that data must identify the underlying language. In the field of grammatical inference, it is worth noting the contribution of [10], who introduces algorithms for inferring reversible languages from positive data, [11] who developed a grammar induction method that produces stochastic context-free grammars, and [12] presenting an algorithm that generates grammars from positive and negative examples in an incremental way; the relation between this and our work is discussed in more details below.

Bel-Enguix G., Christiansen H. and Dolores Jiménez-López M..
A Grammatical View of Language Evolution.
DOI: 10.5220/0003309200570066
In *Proceedings of the 1st International Workshop on AI Methods for Interdisciplinary Research in Language and Biology* (BILC-2011), pages 57-66
ISBN: 978-989-8425-42-3

Our approach is related to both natural language evolution and grammatical inference. Roughly speaking, we search for a mechanism able to infer the grammar of a system that is constantly changing. We apply a logically based transformational grammar formalism in which the grammar itself may be affected in a derivation step. We intend to model the linguistic competences of a silent listener having the reflective capability of being able to inspect and revise its competences according to current usages. In contrast to some of the models mentioned above, it is not based on neither artificial intelligence nor simulation of communicating agents. Instead we propose a strictly formal approach to the problem of language evolution, showing how a grammar can adapt to new words and ways of building phrases without any external means.

In order to reach this objective, we apply a formalism called *Adaptable Grammars*, based on an earlier proposal of [13–15]. This grammar formalism was invented in the 1980ies, originally for describing phenomena in software systems and programming languages. Later, it has been known under the name Christiansen grammars in work by [16, 17]. It has been applied to formal linguistics only recently [18]. The work of [16] have applied these grammars for grammatical, evolutionary programming; the authors motivate their approach by the observation that with such grammars they can do with shorter derivations of target programs.

In [18], such grammars are demonstrated to capture standard non-context-free languages used in the literature, that represent the central natural language properties of *reduplication, crossed dependencies,* and *multiple agreements*. In the present work, we take this a step further considering language evolution. A surprisingly simple implementation of Adaptable Grammars in Prolog was shown in [18], which, with a few extensions, have been used for the experiments shown in the present paper.

The Adaptive Grammars of the present paper, explained as extensions to Definite Clause Grammars (DCGs) [19], are inherently related to Abductive Logic Programming (ALP); see, e.g., [21] for an overview. Normally, the process of finding new rules (logical, grammatical, ...) is associated with induction and, in our context, Inductive Logic Programming (ILP); see, e.g., [22] for overview. ILP differs from ALP by considering a larger set of observations, and use powerful machine learning techniques, including generalization steps and statistics to produce rules that cover as many cases as possible. The work by [12] can be seen as an algorithmic counterpart to our work. They describe an algorithm for induction of context free grammars which is incremental in the sense that it takes one sample of a time and presents a well-defined grammar after each step. It is explained as an extension of the classical CYK parsing algorithm that it works bottom-up in a breadth-first way. In case it gives up, an induction step inspects the store of unreduced items to suggest reductions which then are collected to new grammar rules. Obviously such methods must be tested out for any practical application of Adaptable Grammars for larger corpora, although they have not, to our knowledge, been tested for natural language corpora.

In section 2 we give an introduction to adaptable grammars and indicate the fundamental principles for how they may be used for describing language evolution. Section 3 introduces additional notation, which is applied in section 4 that demonstrates grammars for evolution in simplified natural language setting. Finally, section 5 gives some concluding remarks and ideas for future work.

2 Adaptable Grammars or Christiansen Grammars

A grammar formalism is *adaptable* when it allows a grammar to mutate dynamically according to context. Thus, the selection of available grammar rules may change throughout a discourse and that the formalism must provide a way to specify these grammar changes. Intuitively, the notion of discourse may refer also to linguistic samples collected over centuries, or even longer when language development is seen in a biological evolution perspective.We may compare with DCGswhich encode context in dynamically changing attributes that determine the set of available context-free instances of a fixed set of parameterized rules. Adaptable grammars go one step further, treating the current grammar itself as an attribute that can be elaborated in arbitrary ways.

We have developed an adaptable version of the Prolog based DCGs, which give several advantages: they provide a direct representation of grammar rules as data and anticipate straightforward implementations based on well-understood meta-interpretation techniques (see, e.g., [20]). The terminology of logic programs and first-order logic is assumed, including *logical variables*, *terms* and their *(ground) instances*. We use the notion of a *denotation function* $[\![-]\!]$, which is a partial mapping from ground terms into grammars. At this level, we do not need to specify the actual denotation function. The formalism includes also a reification of syntactic derivation; this is merely a practical device which is not essential for grammar adaptation.

Definition 1. *An* adaptable grammar *is a quintuple* $\langle \Sigma, N, \Pi, [\![-]\!], R\rangle$ *where* Σ *is a finite* alphabet *of* terminals, N *a set of* nonterminals *which are function symbols of arity at least* 1*,* Π *a logic program,* $[\![-]\!]$ *the* denotation function, *and* R *is a set of grammar rules (below). As a convenient usage, the notion of a nonterminal applies also for a term whose top symbol belongs to* N.

A nonterminal has a distinguished argument called its grammar argument. *We distinguish* reflexive predicates *of the forms* $\mathit{deriv}(N, S)$ *and* $\mathit{nderiv}(N, S)$*, where* N *is a nonterminal and* S *a term. A* grammar rule *is of the form lhs* $\to$ *rhs, where lhs is a nonterminal and rhs a finite sequence of terminal and nonterminal symbols, reflexive predicates, and first-order predicates of* Π.

The meaning of $\mathit{deriv}(N, S)$ (definition 2 below) is that a string S is derivable from the nonterminal N; $\mathit{nderiv}(N, S)$ represents the opposite, namely that no derivation is possible. The use of an explicit denotation function eliminates any potential confusion of variables at the different levels [20]. The program component Π is included for convenience only, as it can be embedded as a set of grammar rules producing the empty string. For clarity, the grammar argument is moved outside the standard parentheses and attach by a hyphen, e.g., instead of $n(x, y, G)$, we write $n(x, y)$-G when n is a nonterminal of arity 3 with grammar argument G. The reflexive predicates represent a version of the general derivation relation, limited to a non-adaptable fragment. To this end, we define a *static rule* as one without reflexive predicates and in which all grammar arguments coincide with the same logical variable e.g., p-$G \to q$-G, but not p-$G_1 \to \{t(G_1, G_2)\}, q$-$G_2$. The *static restriction* of a grammar G, written $\sigma(G)$, coincides with G except that any non-static rule is removed. The following simultaneous definition of derivation and meaning of reflexive predicates is sound as the latter are characterized by derivation in statically restricted grammars that do not it turn contain reflexive predicates.

Definition 2. *A* production instance *of rule r of $\langle \Sigma, N, \Pi, [\![-]\!], R\rangle$ is found by*

1. *selecting a ground instance r' of r in which any reflexive predicate is satisfied (def. below) and $\Pi \vdash A$ for any other logical predicate A in r',*
2. *removing any such predicates from r', so that only grammar symbols remain.*

Whenever $\alpha(N\text{-}G)\gamma$ is a sequence of ground grammar symbols and $N\text{-}G \rightarrow \beta$ a production instance of a rule in $[\![G]\!]$, the following derivation step *applies,*

$$\alpha(N\text{-}G)\gamma \Rightarrow \alpha\beta\gamma.$$

The derivation relation $\Rightarrow^*$ *refers to the reflexive, transitive closure of $\Rightarrow$.*

A ground reflexive predicate deriv$(A\text{-}G, S)$ is satisfied *whenever $\sigma[\![G]\!] \Rightarrow^* S$; nderiv$(A\text{-}G, S)$ is satisfied when deriv$(A\text{-}G, S)$ is not satisfied. The* language *given by a ground nonterminal $N\text{-}G$ is the set of terminal strings S with $N\text{-}G \Rightarrow^* S$.*

The available implementation in Prolog can analyze text in terms of queries posed as follows, where G represents a grammar,

```
?- parse(N(Atts)-G,text).
```

The obtained answer may be either 1) a substitution for variables in $Atts$ under which the derivation succeeds, or 2) "no" if parsing is not possible.

Example 1. General grammar induction is not the main focus of the present paper, but we can use it to illustrate the power of adaptable grammars. Grammar induction is the problem of finding a good grammar that can recognize a given discourse. This can be specified in terms of an adaptable grammar as shown in figure 1. The predicate good-grammar specifies a class of grammars in which the induced grammar is to be found. Thus the rule specifies that variable G_{new} should be instantiated to a grammar that makes it possible to parse the given input as discourse.

$\langle \Sigma, N_0, \Pi_0, [\![-]\!], R_0\rangle$, where

$$\begin{aligned} N_0 &= \{\text{induce}/2\} \\ \Pi_0 &= \{\text{good-grammar}(G)\text{:- } \ldots,\ \ldots\} \\ R_0 &= \{\text{induce}(G_{new})\text{-}G \rightarrow \\ &\qquad\qquad \{\text{good-grammar}(G_{new})\}, \\ &\qquad\qquad \text{discourse-}G_{new}\} \end{aligned}$$

Fig. 1. Grammar induction formulated as an adaptable grammar.

We do not intend to present an implementation capable of handling example 1; this would require more advanced techniques (e.g., of Inductive Logic Programming) that those we introduce below for an incremental adaptation. For grammars with explicit synthesis of new grammar rules, it is straightforward to extend a general DCG parser written in Prolog for adaptable grammars, cf. [18]; the following "constructive" grammar represents the essence of our approach to characterize language evolution.

Example 2. Consider sequences of letters in which the letter a is the only one known by the initial grammar. Instead of failing for an unknown letter, a new rule should be added. Let $Gram_1 = \langle \Sigma_1, N_1, \emptyset, [\![-]\!]_1, R_1 \rangle$ be a grammar with $\Sigma = \{a, b, \ldots\}$, $N_1 = \{d/2, s/2\}$ (for **d**iscourse and **s**entence) and R_1 the rules shown in figure 2; the denotation function $[\![-]\!]_1$ represents a grammar as a list of terms for each rule; each symbol stands for itself except variables where $\widehat{X}$ denotes variable X. The reflexive predicate in the last rule ensures that new rules are only considered when necessary. The answer to a query `?- parse(`$d(G)$-$Gram_1, [a, b, c]$`)` should provide a value for G that denotes a grammar whose static restriction is isomorphic to a CFG for sequences of exactly the letters a, b and c.

$$\begin{aligned} d(G)\text{-}G &\rightarrow [\,] \\ d(G_2)\text{-}G &\rightarrow s(G_1)\text{-}G, d(G_2)\text{-}G_1 \\ s(G)\text{-}G &\rightarrow [a] \\ s(G_1)\text{-}G &\rightarrow [X], \text{nderiv}(s(\_)\text{-}G, [X]), \{G_1 = [(s(\widehat{G}))\text{-}\widehat{G} \rightarrow [X])|G]\} \end{aligned}$$

Fig. 2. The rules of an adaptable grammar for a single letter sentence language.

3 Adaptable Grammars for Language Evolution

A specialized notation for Adaptable Grammars has been implemented in Prolog for experiments with language evolution; the implementation is a straightforward extension of the 10 line Prolog program given in [18] and is not described further.

Grammars are given a more conventional appearance, having the grammars arguments implicit, so that language changes appear as side-effect on a "current" grammar. A weighting of each rule is introduced for measuring how recently it has been applied. With this we can model both how new forms come into fashion and obsolete forms disappear from the language user's memory. Furthermore, the grammar notation allows to indicate which syntactic categories can be changed and which can be used in the right hand side of new rules. The new notation is introduced by the following example.

```
(structural categories --d) //
(lexical categories ++s)    //
[ (d --> []):null,
  (d --> s, *forget, d):null,
  (s --> [a]):1,
  (s --> [X], d \=> [X], +(s-->[X]):1):null ]
```

Fig. 3. Adaptable grammar for a single letter sentence with rule weights.

Example 3. The grammar of figure 3 is identical to the one of example 2, with the difference that some rules that have not been used, may be removed. A grammar has three parts, structural and lexical nonterminals and the set of rules. The distinction between structural and lexical categories (or nonterminals) may be utilized in more advanced patterns for creation of new rules and has no influence for this grammar. The pluses and minuses in front of a nonterminal determine how it may be used in when forming new rules. The first sign determines whether it can appear in the left-hand side of new rules, i.e., whether or not this category can be extended with new rules; the second whether the nonterminal can be applied in right-hand sides in new rules. The "`:`n" is the current weight assigned to each rule; the special value `null` indicates that the given rule is not degraded by not being used. The `*forget` operation degrades each rule (with non null weight) multiplying its weight by 0.99, and deleting those with a weight < 0.5.[3] New rules are added to the current grammar using the prefix plus operator as shown in the last rule. Finally, reflexive predicates *deriv* and *nderiv* are represented by the operators `=>` and `\=>`, referring to (non-) derivation in the current grammar.

When the grammar of example 3 is applied for the discourse `[a,b]`, the final grammar has rules for sentences '`a`' and '`b`'; for `[a,b,c,b,c,`$\ldots$`,b,c]` with '`b,c`' repeated sufficiently many times, there will be rules '`b`' and '`c`' for but not for '`a`' .

For the examples below, the distinction between lexical and structural categories is made as follows. A lexical category C allows only rules of the form C`-->[`*Terminal*`]` that we refer to as lexical rules. Structural categories can have rules whose right-hand side is composed of one or more nonterminals.

4 Modeling Language Evolution by Adaptable Grammars

Example 3 above demonstrated the overall principles of how a grammar may adapt to current usages. Here we show two linguistically inspired examples. The grammar shown in figure 4 shows how new rules can created be by a naive oracle to cope with new usages in very simple Catalan sentences. The period symbol serves here as a unique demarkation of where a sentence can end, and thus reduces the search space drastically. We allow, for simplicity of writing, that structural rules of the initial grammar may contain terminals (such as "."), although this is not allowed in automatically generated rules. Nonterminal `d` stands for discourse, `p` for period `s` for sentence and `to_period` for arbitrary sequences until and including the period symbol. The device `*new_rules` generates rules in a nondeterministic fashion according to the following heuristics.[4]

– A first attempt is made adding lexical rules only.
– If this is not sufficient, combinations of structural and (perhaps) lexical rules are tried out.
– Only a limited number of rules can be introduced in one go and only a limited number of nonterminals are allowed in the right-hand side of a structural rule; here both are arbitrarily limited to a maximum of two.
– Grammar extensions are not allowed to introduce left-recursion.[5]
– A rule already in the grammar will not be created again.
– Any word belongs to at most one lexical category (admittedly too simple for realistic applications).

[3] The indicated degradation factor and threshold are arbitrary and should of course be refined for any practical applications.

[4] At the level of implementation, our Prolog implementation works in a generate-and-test manner, `*new_rules` generating candidates until one is accepted by `p=>Tokens`.

[5] Due the inherent top-down parsing strategy, a left recursive grammar will lead to infinite loops.

```
(structural categories  --d, +-s, ++np, ++vp) //
(lexical categories ++a, ++pr, ++n, ++v)       //
[ ( np --> pr):1,
  ( np --> a, n):1,
  ( pr --> [ella]):1,
  ( a  --> [una]):1,
  ( n  --> [poma]):1,
  ( v  --> [menja]):1,
  ( vp --> v):1,
  ( vp --> v, np):1,
  ( s  --> np, vp):1,
  ( d  --> p,d):null,
  ( d  --> []):null,
  ( p  --> s,['.'],*forget):null,
  ( p  --> to_period(Tokens), period \=> Tokens,
           *new_rules(Tokens,R), +R, p => Tokens):null,
  (to_period([T|Ts]) --> [T], {T \= '.'}, to_period(Ts)):null,
  (to_period(['.']) --> ['.'] ):null]
```

Fig. 4. An adaptable grammar for a simple language evolution problem.

	Sentence(s)	New rule(s)
(1)	`ella menja una poma.`	(no new rules)
(2)	`blabla menja una poma.`	`pr-->[blabla]`

(3)	`la blabla menja una poma.`	`a-->[la]`
		`n-->[blabla]`

(4)	`menja una poma.`	`s-->vp`
(5)	`ella una poma menja.`	`vp-->np,vp`
(6)	`menja.`	`s-->vp`

(7)	`beu.`	`s-->a`
		`a-->[beu]`

(8)	`menja. beu.`	`s-->vp`
		`v-->[beu]`

Fig. 5. Sample sentences and discourse plus the rules created to accommodation.

In the section for future work below, we discuss possible improvements of this strategy. Figure 5 shows the new rules that are created for sample sentences that contain new words and usages. Example (1) illustrates that no rules are generated when the existing ones are sufficient; (2–4) shows examples of new usages that are accommodated by means of the rules that we might expect; in (5), a new structural rule is created, which may or may not be the desired one; in (6), the perfect rule for a new type of sentences is created, whereas in (7), a new sort of sentence consisting of one new word generates some unnatural rules. We may relate the problem in (7) to the observation that any competent and reflexive language user may have difficulties when too many novelties are introduced at the same time; here both a new word and new sentence form is introduced in a one word sentence. Sample (8) is perhaps the most interesting: it analyzes a discourse consisting of two sentences, the first one shows a new sentence form that is accommodated by the rule `s-->vp`, which is feasible as `menja` is known to be a verb;

in the next sentence this rule is applied when accommodating the new word `beu` which is now classified in an intuitively correct way.

For investigating the long-term behavior of the suggested adaptation mechanism, an artificial corpus of 300 sentences has been generated in a probabilistic way from two grammars, one (a) for simplistic English sentences and another (b) for a kind of Yoda-like German. In the beginning of the corpus, only rules from grammar (a) can be used, and gradually rules of (b) sneaks and replaces (a) via changing probabilities so that only rules of (b) are used at the end. For example, grammar (a) allows sentences such as `[peter,and, mary,likes,pluto]`, grammar (b) for example `[peter,und,mary,pluto,liebst]`.[6] Midway, we find hybrids such as `[peter,und,mary,liebst,pluto,and,mary]` and `[pluto,peter,and, mary,liebst]` Grammar (a) is then given as Adaptable Grammar; figure 6 shows the language specific rules, and the those for periods, `p`, and `to_period` are as in figure 4; the lexical category of proper names `pn` are defined to be fixed throughout the discourse. The final grammar resulting from the successive adaptations is shown in figure 7.

```
s --> np,vp           vp --> v,np       pn --> [mary]
np --> pn             vp --> v          pn --> [pluto]
np --> pn,more_np     con --> [and]     v --> [eats]
more_np --> con,np    pn --> [peter]    v --> [likes]
```

Fig. 6. Simplistic English grammar prior to adaptation.

```
s --> np,vp           vp --> np,vp      pn --> [mary]
np --> pn             vp --> v          pn --> [pluto]
np --> pn,more_np     con --> [und]     v --> [liebst]
more_np --> con,np    pn --> [peter]    v --> [isst]
```

Fig. 7. The result of adapting the grammar of fig. 6 via 300 sentences to Simplistic Yoda-like German.

The verbs and the conjunctions have been replaced by German ones as expected, and the change of order of object and verb is accommodated by the replacement of `vp-->v,np` by `vp-->np,vp`. The last one of these rules is intuitively not the desired one as it is too general; the best rule would, of course, be `vp-->np,v`.

5 Conclusions and Future Work

Language evolution is an aspect of natural language that is especially difficult to model by means of formal grammars. We have approached it by adaptable grammars, that can evolve during the processing and modify their own rules, changing gradually in order to adapt themselves to the evolving language.

If the main ideas collected here are shown to be expressive enough to be developed, then a new system should be designed taking into account many aspects that have been dismissed up to now, in order to approach language evolution in a more realistic way. We characterize language evolution from the viewpoint of a passive listener, but it seems

[6] As it appears, we ignore inflection for singular and plural; this is, however, trivial to add.

possible to apply these ideas also in the setting of communicating agents. An agent may test a proposed grammar extension by generating sentences that are presented to other agents for evaluation (a pattern often observed between children and parents). The multi-agent perspective may also be used to model how different languages develop and influence each other over time.

Summing up, the principle of having the grammar dynamically modifying or adapting itself along a discourse may be seen as a universal mechanism that may be incorporated in other grammatical frameworks as well. The advantage is that original and novel language constructs are represented in an equal manner so that, at any stage, the current grammar may be read out. In most traditional grammar formalisms, that are capable of expressing some context-dependencies, this need to be modeled by highly over-general rules whose application is controlled by an encoding of the linguistic context.

We are considering to improve the adapt-to-one-sentence-at-a-time principle by giving preference to least general rules when adapting to a single sentence, complemented by a generalization step that groups often used rules into a more general rule when possible and judged suitable. It will also be interesting to apply the principle of Adaptable Grammars to see if it can reveal new bits of knowledge of how different languages may descent from each other or influence each other. There may be other and much faster evolution processes that also may be interesting to characterize, for example to trace the spreading of topics in electronic and social networks.

Acknowledgements

This work is supported by the Spanish Ministry of Science and Education, project MTM2007-63422, and the NABIIT committee under the Danish Strategic Research Council, project “Logic-statistic modelling and analysis of biological sequence data”.

References

1. Christiansen, M. H., Kirby, S., eds.: Language Evolution: The States of the Art. Oxford University Press (2003)
2. Briscoe, E. J., ed.: Linguistic Evolution through Language Acquisition: Formal and Computational Models. Cambridge University Press (2002)
3. Steels, L.: Language as a complex adaptive system. Volume 1917 of Lecture Notes in Computer Science., Springer (2000) 17–26
4. Brighton, H., Smith, K., Kirby, S.: Language as an evolutionary system. Physics of Life Reviews 2 (2005) 177–226
5. Cangelosi, A., Parisi, D., eds.: Simulating the evolution of language. Springer-Verlag (2002)
6. Knight, C., Hurford, J., Studdert-Kennedy, M., eds.: The Evolutionary Emergence of Language: Social Function and the Origins of Linguistic Form. Cambridge University Press (2000)
7. Baxter, G. J., Blythe, R. A., Croft, W., McKane, A. J.: Utterance selection model of language change. Physical Review E 73 (2006) 046118
8. Gold, M.: Language identification in the limit. Information and Control 10 (1967) 447–474
9. Clark, A., Coste, F., Miclet, L., eds.: Grammatical Inference: Algorithms and Applications, Volume 5278 of Lecture Notes in Computer Science, Springer (2008)

10. Angluin, D.: Inference of reversible languages. Journal of the Association for computing Machinery 29 (1982) 741–765
11. Kurihara, K., Sato, T.: Variational bayesian grammar induction for natural language. In Sakakibara, Y., Kobayashi, S., Sato, K., Nishino, T., Tomita, E., eds.: ICGI. Volume 4201 of Lecture Notes in Computer Science., Springer (2006) 84–96
12. Nakamura, K., Matsumoto, M.: Incremental learning of context free grammars bsed on bottom-up parsing and search. Pattern Recognition 38 (2005) 1384–1392
13. Christiansen, H.: Syntax, semantics, and implementation strategies for programming languages with powerful abstraction mechanisms. In: 18th Hawaii International Conference on System Sciences. Volume 2. (1985) 57–66
14. Christiansen, H.: The syntax and semantics of extensible languages. *Datalogiske skrifter* 14 (Tech. rep). Computer Science Section, Roskilde University, Denmark (1988)
15. Christiansen, H.: A survey of adaptable grammars. SIGPLAN Notices 25 (1990) 35–44
16. Ortega, A., de la Cruz, M., Alfonseca, M.: Christiansen grammar evolution: Grammatical evolution with semantics. IEEE Trans. Evolutionary Computation 11 (2007) 77–90
17. de la Cruz Echeandía, M., de la Puente, A.O.: A christiansen grammar for universal splicing systems. Volume 5601 of Lecture Notes in Computer Science., Springer (2009) 336–345
18. Christiansen, H.: Adaptable grammars for non-context-free languages. In Cabestany, J., Hernández, F.S., Prieto, A., Corchado, J.M., eds.: IWANN (1). Volume 5517 of Lecture Notes in Computer Science., Springer (2009) 488–495
19. Pereira, F. C. N., Warren, D. H. D.: Definite clause grammars for language analysis—A survey of the formalism and a comparison with augmented transition networks. Artificial Intelligence 13 (1980) 231–278
20. Hill, P. M., Gallagher, J.: Meta-programming in logic programming. In: Handbook of Logic in Artificial Intelligence and Logic Programming, Oxford Science Publications, Oxford University Press (1994) 421–497
21. Kakas, A., Kowalski, R., Toni, F.: The role of abduction in logic programming. Handbook of Logic in Artificial Intelligence and Logic Programming, vol. 5, Gabbay, D.M, Hogger, C.J., Robinson, J.A., (eds.), Oxford University Press (1998) 235–324
22. Lavrac, N., Dzeroski, S.: Inductive Logic Programming: Techniques and Applications. Ellis Horwood, New York (1994)