A Grammatical View of Language Evolution

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

[1] Research Group on Mathematical Linguistics
Universitat Rovira i Virgili
Avd. Catalunya 35, 43002 Tarragona, Spain
{gemma.bel,mariadolores.jimenez}@urv.cat

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

Abstract. In this paper, we show an application of Adaptable Grammars to language evolution. An adaptable grammar may be defined as a logically based transformational grammar formalism in which the grammar itself may be affected in a derivation step. This grammar formalism was originally intended for describing software systems and programming languages. For the field of natural language analysis, the main advantage of adaptable grammars over other types of formal grammars is the idea of evolution. Adaptable grammars are dynamic entities in which novelties appearing in lexical units or language structure can create new or modify existing grammar rules. Taking into account this idea of 'dynamicity', we suggest the possibility of applying adaptable grammars to natural language change.

1 Introduction

During the last decade, approaches to language have undergone a deep transformation thanks to a new interdisciplinary paradigm that integrates artificial intelligence, physics and evolutionary biology [14, 6, 35, 44]. Currently, the research in diachronic change is based on the understanding of language as a complex adaptive system [55] and an evolutionary system [16, 5]. Moreover, it is supported by computational models and simulation to fully understand the dynamics of human language and its adaptation to the environment [7]. Some consistent contributions to the topic have also been introduced from mathematics [45–47].

The main problems approached by language evolution are the origins and emergence of language, language acquisition and language change. The origins and emergence of language was first tackled from a computational and formal perspective by the pioneering work of Steels [53, 54], and summarized by Knight, Studdert-Kennedy & Hurford [27].

Language evolution through language acquisition has several key contributions in the works of Briscoe [6], Kirby [25], Komarova, Niyogi & Nowak [28] and Komarova & Nowak [29].

Finally, language change has been approached by a number of articles in recent years. One of the most recent mathematical models of language change has been developed by Baxter et al. [3]. The authors introduce parameters for both evolution by communicative interaction and generational change. Some formalisms have been used to explain particular cases, like Baxter et al. [4] for the English of New Zealand and Lieberman et al. [32] for the emergence of the "-ed" suffix Closely related to the problem of language change is the formal representation of language death [1, 38, 37], bilingualism [36, 8, 49] and creolization / dialectalization [42, 43].

The approach we are introducing in this paper is a very simple model for dealing with language change. It is based on 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 equipped with reflective capabilities, i.e., who is capable of inspecting and revising its competences according to current usages. In contrast to some of the models mentioned above, it is not based on artificial intelligence or 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.

We apply a formalism called *Adaptable Grammars*, based on an earlier proposal of Christiansen [9–11], which only recently have been applied to formal linguistics. In [13], the author shows how such grammars may capture standard non-context-free languages used in the literature (e.g., [34]) as prototypical representatives for 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 [13], which will be extended below with additional auxiliaries for rule generation; this implementation makes adaptable grammars also a powerful tool for experimentation.

In section 2 we give an introduction to adaptable grammars and indicate principles for how they may be used for describing language evolution, and in section 3 we show an adaptable grammars for a simple language evolution problem. A realistic example is outside the scope of this paper, so the mentioned grammar should be taken as a proof of concept. Section 4 compares with related work with a focus on machine learning and grammar induction, and finally section 5 gives some concluding remarks, including some plans for future work. The appendix shows an implementation in Prolog for parsing with adaptable grammars.

2 Adaptable Grammars

This grammar formalism was originally intended for describing software systems and programming languages; other authors have used the name "Christiansen grammars" [51, 22, 48, 56]. Recently, Ortega et al. [48] have used 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, thus shorter chromosomes and therefore faster convergence. See also [10, 11, 13] for comparison with other attempts to adaptable or extensible grammar formalisms. The inclusion of reflective atoms in the definition below is new.

We assume the terminology of first-order logic, including logical variables, and logic programs (pure Prolog programs) as well as related notions such as (ground) instances for formulas and terms.

Definition 1 (Adaptable grammars). *An* adaptable grammar *is a quintuple* $\langle \Sigma, N, \Pi, [\![-]\!], R\rangle$ *where* Σ *is a finite* alphabet*,* N *a set of* nonterminal *symbols which are logical predicate symbols of arity at least 1,* Π *is a logic program,* $[\![-]\!]$ *is the* denotation function *which is a (partial) function from terms to grammars, and* R *is a set of grammar rules (below). Nonterminals and predicates of* Π *are assumed to be disjoint. Each nonterminal symbol has a distinguished argument called its* grammar argument*. A* reflective atom *is of the form* `derive_static(`N`,`S`,`G`)` *or* `not_derive_static (`N`,`S`,`G`)`*; the arguments indicated by "G" are also called* grammar arguments*. A* grammar rule *is of the form:*

$$lhs \texttt{-->} rhs.$$

where lhs is a nonterminal and rhs a finite sequence of elements which may be terminal or nonterminal symbols, reflective atoms, or first-order atoms defined by Π*.*

The use of an explicit denotation function means that we avoid potential circularity problems, so that the definition can rely on standard sets of first-order terms. Furthermore, it allows us to introduce arbitrary auxiliary notation as we will do below. We apply a notation inspired by definite clause grammars so that terminals in grammar rules are written in square brackets, and atoms referring the program component in curly brackets. We assume that the grammar argument for a nonterminal always is the *last* one, and we drag it outside the standard parentheses and attach it by a hyphen, e.g., instead of `n(x,y,G)`, we prefer `n(x,y)-G` when `n` is a nonterminal of arity 3. Terms within rules that represent grammars are generally written using Prolog's list syntax but may be written in other ways depending on the actual denotation function. The program component can also change over time and serve as a dynamic knowledge base, although we do not employ this in the present paper.

A *static rule* is one without reflective atoms and in which all grammar arguments coincide with the same logical variable (e.g., `p-G-->q-G`, but not `p-G1-->{t(G1,G2)},q-G2`. The *static version* of a grammar G, denoted $\sigma(G)$ coincides with G except that any non-static rule is removed. Reflective atoms can be applied for explicitly checks whether a string is derivable in a "current" grammar.

Definition 2 (derivation). *Given an adaptable grammar* $\langle \Sigma, N, \Pi, [\![-]\!], R\rangle$*, an* application instance *of one of its rules* $r \in R$ *is a ground instance* r' *of* r *in* *which grammar arguments denote grammars and any logical atom in r' is satisfied in Π; the reflective atom* `derive_static(`N-G`,`S`)` *is satisfied whenever N-G is a nonterminal, S a sequence of terminals, G a term that denotes a grammar, and N-$G' \Rightarrow^* S$, where G' is a grammar term such that $[\![G']\!] = \sigma([\![G']\!])$;* `not_derive_static(`$N$-$G$`,`$S$`)` *is defined in a similar way, but with N-$G' \not\Rightarrow^* S$.*[3] *Whenever $\alpha\beta\gamma$ is a sequence of ground grammar symbols, β being a nonterminal of the form N-G with an application instance β*`-->`*δ of a rule in $[\![G]\!]$, we write:*

$$\alpha\beta\gamma \Rightarrow \alpha\delta'\gamma$$

where δ' is a copy of δ with any atom referring to Π and reflective atoms taken out. The relation $\Rightarrow^*$ refers to the reflexive, transitive closure of $\Rightarrow$.

The language *defined by a given ground nonterminal N-G is the set of terminal strings S for which N-$G \Rightarrow^* S$.*

Notice the difference between grammars and terms that represent them. A grammar argument may be a list of rule representations containing duplicates, and the "processing" of it may take multiplicity into account. Even without the program components, our adaptive grammars are obviously Turing complete as they embed pure Prolog programs, which can be rewritten as grammar rules that generate the empty sequence. Thus the computational complexity for text analysis with an adaptable grammar can be arbitrary high, but the examples we show here has linear time complexity in the length of the discourse being analyzed (assuming a maximum length for periods).

The following example represents the essence of our approach to characterize language evolution.

Example 1. In [13] we show a ten line Prolog implementation of a basic version of adaptable grammars, and in the appendix of the present paper we give a version that also supports reflection so that it can interpret (i.e., parse with) a grammars as the one we show below. Grammars are represented in the so-called non-ground representation (i.e., new variables are denoted by variables that do not occur elsewhere [21]), the rules of a grammar represented as a list, and the other components left implicit.

The following grammar describes simple `d`iscourses, consisting of `s`entences, and initially only sentence "`a`" is recognized. The last rule extends the grammar in case a new usage such as "`b`" comes into fashion.

```
(1) d( G)-G --> []
(2) d(G2)-G --> s(G1)-G, d(G2)-G1
(3) s( G)-G --> [a]
(4) s(G1)-G --> [X], not_derive_static(s(_)-G,[X]),
                {G1=[(s(Gx)-Gx-->[X])|G]}
```

[3] Notice that there is no circularity problem here as satisfaction of reflective atoms is defined in terms of derivations using static rules only. In fact, reflective atoms are included for conceptual clarity only, as they could have been defined in the program component by a straightforward meta-interpreter (here: a parser).

When this grammar is applied for parsing a text given as `[a,b,b]`, it will produce, as argument of the topmost `d` node, a grammar which is extended with the rule `s(Gx)-Gx-->[b]`. The occurrence to `not_derive_static` in line (4) prevents the creation of rules that are already present.

This grammar contains some patterns that will recur often in grammars that capture language evolution, and we will introduce more convenient notation later.[4] Rule (2) is a sequencing operator which explicitly states that the current grammar `G` is imported for the start of the lefthand side, it is possible modified by the constituents from left to right and finally exported as `G2` to whatever context the rule was applied in; we will take this as a default pattern and leave out the language arguments. Furthermore, rule (4) indicates an addition of a new rule to the current grammar; we will indicate this by writing the rule, prefixed by `+`. Finally, we define two infix operators for the reflexive predicates, `=>` and `\=>`, referring implicitly to the current grammar.

Example 2. With the notation introduced, we can now write the grammar of example 1 as follows.

```
(1) d --> []
(2) d --> s, d
(3) s --> [a]
(4) s --> [X], d \=> [X], +(s-->[X])
```

For the specific aims of the present paper, we introduce a final convention that actually changes the meaning of the notation for grammars used in example 2. We may assign a numerical weight to each rule in a grammar; each time a rule is applied, it gets weight 1, all other rule weights are multiplied by some constant f, $0 < f < 1$, called the *obsolescence factor*, and as soon as a rule's weight becomes less that some *treshold* t, $0 < t < f$, it is discarded from the grammar. The initial weight of each rule will be either 1 or the special value `null`, indicating that the particular rule is never discarded. In the example above, we would expect rules (1), (2), (4) to be of that kind and any other rule, whether existent from the start or created dynamically, has the risk of being forgotten. We will introduce notation in the following sections to distinguish.

So with reasonable choices of f and t, we expect the sample grammar, when analyzing the discourse `[a,b]`, leads to a revised grammar that knows sentences "`a`" and "`b`". In case of a discourse `[a,b,b,...,b,c]`, we may expect the rule for "`a`" to vanish and rules for "`b`" and "`c`" to remain at the end.

However, it must be emphasized that we did no extend the formalism introduced in definitions 1 and 2, we just introduced a convenient notation. The weighting information can be represented as an additional argument to each

[4] Be aware that Adaptable Grammars can describe much more advanced grammar manipulation that what is shown here, so the notation we introduce here indicates a restriction to a subset of Adaptable Grammars.

grammar symbol and each rule extended with an initial call to a weight adjustment predicate, that possibly also removes rules from the grammar.[5]

3 Modeling Language Evolution by Adaptable Grammars

A version of adaptable grammars have been implemented in Prolog for experiments with language evolution, and we illustrate this below for a small sample grammar that accommodates, in an incremental way, to new usages coming up. We take this demonstration as a proof of concept and use the experience gained to discuss possible improvements of the approach. The following grammar describes a small subset of Catalan, and shows the extensions we have made to the grammar format explained so far. It must be stressed, however, that the new notations introduced do not extend the formal framework given by definitions 1 and 2; they are included for convenience only. Figure 1 shows this adaptable grammar represented as a Prolog term so it can be directly input to an analysis program, which is a mere extension of the one shown in the appendix.

```
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  --> s,['.'],d):null,
  ( d  --> []):null,
  ( s  --> substring_until_period(Tokens),
           s \=> Tokens,
           new_rules(Tokens,R),
           +R,
           s => Tokens):null,
  ( substring_until_period([T|Ts]) -->
      [T], {T \= '.'}, substring_until_period(Ts) ):null,
  ( substring_until_period([]) --> [], look_ahead(['.']) ):null ]
```

Fig. 1. An adaptable grammar for a simple language evolution problem

[5] From an implementation point of view, it will be easier to merge the weights into the grammar representation, but this still does not change the formalism.

The collection of syntactic categories (nonterminals) are defined in the first two lines, distinguished into structural and lexical ones which gives a difference in which rules that can be generated for them. The syntactic categories stay fixed throughout the analysis of a discourse being analyzed. A prefix "`+`" means that the category can be extended and "`-`" that a category cannot be used in new rules being generated. Discourses "`d`" and sentences "`s`" are considered high-level notions that cannot appear as substructures inside new constructs; sentences can be extended with new forms, but discourses cannot, i.e., we simplify the problem by having periods to serve as definite steering marks. The initial weights, as explained above, are made explicit as to have a way to state that some of the rules will stay forever in the current grammar (those with weight `null`). The nonterminal `substring_until_period` is a generic device that collects the string up to and not including the next period; this string is made available in the variable `Tokens` so it can be processed further. We use here a `look_ahead` device to ensure the `Tokens` are expanded as far as possible; it must match a given character, but does not "consume" it. An oracle is used for generating new rules in an attempt to make the `Tokens` parseable.

This oracle appears as a new reflexive construct, `new_rules`, that includes some heuristics designed for the specific example, but which is expected to work at a larger scale. It generates in a nondeterministic way different collections of rules and commits to the first one that solves the problem, so to speak. In case of new words being applied, they must be placed in lexical categories; if that is sufficient, no more rules are generated. Otherwise, the oracle starts adding new structural rules based on the following heuristics:

- Only a limited number of rules can be introduced in one go; here limited to a maximum of two.
- The body of a structural rule may contain only a limited number of nonterminals; here limited to a maximum of two.
- Grammar extensions that introduce left-recursion to the current grammar are not allowed.[6]
- 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).

In the section for future work below, we discuss possible improvements of this strategy. Figure 2 shows the new rules that are created for some sentences that contains 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)

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

	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. 2. Sample sentences and discourse plus the rules created to accommodation.

to the observation that any competent and reflective grammar 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.

Example (8) is perhaps the most interesting: it analyzes a discourse consisting of two sentences, the first on 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.

Examples for longer discourses that we do not show here, demonstrate how the obsolescence principle removes, after a while, rules that are not used, either because the usage they represent goes out of fashion or it may be a strange rule generated from a single sample that a normative linguist would classify as incorrect.

4 Related Work

Adaptable grammars can be related to work done in fields as, for example, *grammar induction*, *inductive logic programming* and *iterated learning*.

Grammar induction, e.g, [20, 15, 19, 26], also known as grammatical inference, automatic language acquisition or automata induction, 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 (or learning algorithm) that from that data must identify the underlying language. Adaptable grammars share with grammar induction the idea of 'learning' from data, but while in the field of grammatical inference the goal is to learn a new language/grammar from the given data, in our model the goal is to adapt (to slightly modify) the grammar we start with in order to fit the changes that happen in language evolution. In grammatical inference, the learner must *infer* the grammar from the data, in adaptable grammars the speaker must *adapt* his grammar in order to be able of parsing the new/unknown structures. In what refers to the applications to natural language, while grammatical inference seems to be easily applicable to the area of natural language acquision, our model fits better the field of natural language evolution/change. Also, [30] has developed a grammar induction method that produces Stochastic Context Free Grammars.

Inductive logic programming [39, 31, 41, 40] is also a subfield of machine learning particularly useful in bioinformatics and natural language processing, which also represent powerful methods for grammar induction, e.g., [2, 17]. The term inductive logic programming was first introduced by Stephen Muggleton in 1991 [39]. Inductive logic programming has been defined as a technology that combines principles of inductive machine learning with the representation of logic programming. The aim of this technology is to induce *general rules* starting from *specific observations* and *background knowledge*. It is considered more powerful than traditional techniques that learn from examples because it uses an expressive first order logic framework instead of the traditional attribute-value framework and because it facilitates the use of background knowledge. Those two features are very important because many domains of expertise cannot be formulated in an attribute-value framework and because background knowledge is very important in artificial intelligence applications. Roughly speaking, a problem in inductive logic programming is concerned with finding a hypothesis (a logic program) from a set of positive and negative examples. It is required that the hypothesis covers all positive examples and none of the negative examples. There is a background knowledge (a logic program) that is provided in the inductive logic programming system and fixed during the learning process. Adaptable grammars work in a similar fashion. The background knowledge in inductive logic programming can be equivalent to the grammar we start with in adaptable grammars. This initial grammar must be adapted/fixed during the parsing process in order to fit new examples. Of course, in our adaptable grammars we deal also with hypothesis, the system make different hypothesis in order to find the new rule that better fits the new data. The weights introduced in our adaptable grammars relate this framework to the so-called *probabilistic inductive logic programming* [50] or statistical relational learning, a model that shows an integration of probabilistic reasoning.

Iterated learning [25, 52] is a model for language evolution through language acquisition that was introduced by Kirby in 2001. In this model, agents are able to create a compositional language starting from a holistic language by means of inferring a grammar from the language they learn every generation. Both models share the idea that the grammar evolves and provide a method to capture such evolution. However, their theoretical principles are widely different. First of all, we have to place iterated learning model in the field of simulation and artificial intelligence, whereas adaptable grammars offer an strictly formal model for language evolution. Moreover, while iterated learning takes into account agents and generational change, adaptable grammars are not concerned about the population nor the reasons of the evolution. They only account for changes and are able to adapt to them. Finally, iterated learning deals with the emergence of language structure. It is a model of language change, that is also able to explain the origins of compositionality. However, adaptable grammars are a model of language change that can describe the death of languages or the processes of splitting and dialectalization, but are not able to suggest an explanation for language emergence. Finally, we mention that genetic algorithms also have been suggested for grammar induction [24].

It may be suggested that our aim could be fulfilled even better by running a grammar induction procedure on the previous n sentences whenever a next sentence cannot be parsed. However, in this way we would loose the property of incrementality that we find central in the modeling of how a competent language user gradually changes his or her understanding of the grammar as the language evolves over time. In fact, our approach is more closely related to *abductive reasoning* in logic programmming (see [18] for an overview), as it proceeds by adding one rule or as few new rules as possible in order to accommodate to one sample at a time, rather that revising everything. We may refer to [12] that describes a quite analogous system for abductive reasoning in which a constraint-logic implementation attempts to guess new Prolog rules that makes a given top-level goal provable; such techniques might also apply to the present application. We have not seen the obsolescence principle applied in abductive logic programming.

Finally, we compare our approach to the tradition of mildly context-sensitive grammars [23], which are capable of capturing (essential aspects of) natural language, while maintaining a polynomial time complexity for analysis, thus also limiting its formal expressibility. Our approach is different as we propose a formalism which, in this sense, is far too powerful, but we obtain an ease of understanding and naturalness of the actual grammars which (under suitable assumptions) can analyze in linear time. This can be in seen contrast to the mildly context-sensitive Contextual Grammars [33] in which grammars for even small and artificial languages convey (our claim!) very little intuition about the languages described. We have not notices any attempts to apply mildly context-sensitive grammars for characterizing language evolution.

5 Conclusions and Future Work

From the origins of formal language theory, in the middle of the 20th century, models coming from the formal grammars research field have been applied to the description, explanation and processing of natural language. However, there is a one aspect of natural language that is specially difficult to be modelled by using formal grammars: *language evolution.*

In this paper, we have shown the possible applications of adaptable grammars to language evolution. The main advantage of the formalism introduced here is precisely, that it can account for natural language change. Adaptable grammars are able to evolve during the processing, they are dynamic entities that can modify their rules, changing gradually (as natural languages do) in order to adapt themselves to the evolution of language.

The implementation introduced in the paper has the aim to show how the mechanism can actually explain natural language evolution. 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.

Overall, language evolution is tackled here from the viewpoint of a passive listener, but there is nothing that limits of using a (current) grammar for sentence generation. In other words, using the same type of grammars, an agential perspective could be taken, designing different individuals that exchange utterances in a population, taking into account generational take over, mutation rates and social parameters.

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 motivation is to provide a "natural" way of modelling those context-dependent aspects of language that essentially are related to the introduction of new linguistic potential or – in a broader perspective – the development of language. 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 modelled by highly over-general rules whose application is controlled by an encoding of the linguistic context.

Our plans for future work include a more careful selection of new rules by considering a suitable generalization-specialization hierarchy, so that we generate only most specific rules to accommodate new usages, and add a generalization or induction step that clashes similar rules into a more general one when certain criteria are met, and which takes into account also the weights that reflect how often and recent the different rules have been applied. We are considering applying the approach to developments in Latin and the languages derived from it. Another possible application is to trace the development of topics in social networks.

Appendix: An Implementation of Adaptable Grammars in Prolog

The following Prolog program provides an implementation of adaptable grammars given in the format applied in example 1; only a few trivial lines of code have been left out.

```
derive(NG,S):- derive(NG,S,[]).

derive([],S,S).

derive([T|Ts],[T|S1], S2):- derive(Ts,S1,S2).

derive({Code},S,S):- Code.

derive((A,B),S1,S3):- derive(A,S1,S2), derive(B,S2,S3).

derive(derive_static(N-G,X),S,S):-
      make_static(G,GS), derive(N-GS,X).

derive(not_derive_static(N-G,X),S,S):-
      make_static(G,GS), \+ derive(N-GS,X).

derive(N-G,S1,S2):-
      renameVars(G,Gx), member((N-G-->B),Gx), derive(B,S1,S2).

renameVars(X,Y):- assert(quax(X)), retract(quax(Y)).

make_static(G,Gx):- Gx contains only the static rules of G.
```

An interpreter for the grammar format indicated in section 3 can be obtained from the one above by adding more or less straightforward code, but needs no essential new inventions.

At the time of publication, we will have announced a website that contains the full code of this interpreter as well as the more elaborate implementation needed for the example in section 3.

Acknowledgment

This work is supported by the project "Logic-statistic modelling and analysis of biological sequence data" funded by the NABIIT programme of the Danish Strategic Research Council.

References

1. D.M. Abrams and S.H. Strogatz. Modelling the dynamics of language death. *Nature*, 424(6951):900, 2003.
2. Pieter W. Adriaans, Marten Trautwein, and Marco Vervoort. Towards high speed grammar induction on large text corpora. In Václav Hlavác, Keith G. Jeffery, and Jirí Wiedermann, editors, *SOFSEM*, volume 1963 of *Lecture Notes in Computer Science*, pages 173–186. Springer, 2000.
3. Gareth J. Baxter, Richard A. Blythe, William Croft, and Alan J. McKane. Utterance selection model of language change. *Physical Review E*, 73:046118, 2006.

4. Gareth J. Baxter, Richard A. Blythe, William Croft, and Alan J. McKane. Modeling language change: An evaluation of trudgills theory of the emergence of new zealand english. *Language Variation and Change*, 21:257–296, 2009.
5. Henry Brighton, Kenny Smith, and Simon Kirby. Language as an evolutionary system. *Physics of Life Reviews*, 2(3):177–226, September 2005.
6. E. J. Briscoe, editor. *Linguistic Evolution through Language Acquisition: Formal and Computational Models*. Cambridge University Press, 2002.
7. Angelo Cangelosi and Domenico Parisi, editors. *Simulating the evolution of language*. Springer-Verlag, 2002.
8. Xavier Castelló, Víctor M Eguíluz, and Maxi San Miguel. Ordering dynamics with two non-excluding options: bilingualism in language competition. *New Journal of Physics*, 8(12):308, 2006.
9. Henning Christiansen. Syntax, semantics, and implementation strategies for programming languages with powerful abstraction mechanisms. In *18th Hawaii International Conference on System Sciences*, volume 2, pages 57–66, 1985.
10. Henning Christiansen. The syntax and semantics of extensible languages. *Datalogiske skrifter* 14 (Tech. rep). Computer Science Section, Roskilde University, Denmark, 1988.
11. Henning Christiansen. A survey of adaptable grammars. *SIGPLAN Notices*, 25(11):35–44, 1990.
12. Henning Christiansen. Automated reasoning with a constraint-based metainterpreter. *Journal of Logic Programming*, 37(1-3):213–254, 1998.
13. Henning Christiansen. Adaptable grammars for non-context-free languages. In Joan Cabestany, Francisco Sandoval Hernández, Alberto Prieto, and Juan M. Corchado, editors, *IWANN (1)*, volume 5517 of *Lecture Notes in Computer Science*, pages 488–495. Springer, 2009.
14. M. H. Christiansen and S. Kirby, editors. *Language Evolution: The States of the Art*. Oxford University Press, 2003.
15. Alexander Clark, François Coste, and Laurent Miclet, editors. *Grammatical Inference: Algorithms and Applications, 9th International Colloquium, ICGI 2008, Saint-Malo, France, September 22-24, 2008, Proceedings*, volume 5278 of *Lecture Notes in Computer Science*. Springer, 2008.
16. William Croft. *Explaining Language Change: An Evolutionary Approach*. London: Longman, 2000.
17. James Cussens and Saso Dzeroski, editors. *Learning Language in Logic*, volume 1925 of *Lecture Notes in Computer Science*. Springer, 2000.
18. Marc Denecker and Antonis C. Kakas. Abduction in logic programming. In Antonis C. Kakas and Fariba Sadri, editors, *Computational Logic: Logic Programming and Beyond*, volume 2407 of *Lecture Notes in Computer Science*, pages 402–436. Springer, 2002.
19. Heshaam Feili and Gholamreza Ghassem-Sani. Unsupervised grammar induction using history based approach. *Computer Speech & Language*, 20(4):644–658, 2006.
20. K S Fu and T L Booth. Grammatical inference: introduction and survey—part i. *IEEE Trans. Pattern Anal. Mach. Intell.*, 8(3):343–359, 1986.
21. P. M. Hill and J. Gallagher. Meta-programming in logic programming. In *Handbook of Logic in Artificial Intelligence and Logic Programming*, pages 421–497. Oxford Science Publications, Oxford University Press, 1994.
22. Quinn Tyler Jackson. *Adapting to Babel: Adaptivity and Context-Sensitivity in Parsing*. Ibis Publications, Plymouth, Massachusetts, USA, 2006.

23. Aravind K. Joshi. Tree adjoining grammars: how much context-sensitivity is required to provide reasonable structural descriptions? In David R. Dowty, Lauri Karttunen, and Arnold Zwicky, editors, *Natural Language Parsing*, pages 206–250. Cambridge University Press, Cambridge, 1985.
24. Thomas E. Kammeyer and Richard K. Belew. Stochastic context-free grammar induction with a genetic algorithm using local search. In Richard K. Belew and Michael D. Vose, editors, *FOGA*, pages 409–436. Morgan Kaufmann, 1996.
25. S. Kirby. Spontaneous evolution of linguistic structure: an iterated learning model of the emergence of regularity and irregularity. *IEEE Transactions on Evolutionary Computation*, 5(2):102–110, 2001.
26. Dan Klein and Christopher D. Manning. Natural language grammar induction with a generative constituent-context model. *Pattern Recognition*, 38(9):1407–1419, 2005.
27. Chris Knight, Jim Hurford, and Michael Studdert-Kennedy, editors. *The Evolutionary Emergence of Language: Social Function and the Origins of Linguistic Form*. Cambridge University Press, 2000.
28. N. L. Komarova, P. Niyogi, and M. A. Nowak. The evolutionary dynamics of grammar acquisition. *Journal of Theoretical Biology*, 209(1):43–59, 2001.
29. N. L. Komarova and M. A. Nowak. Language, learning, and evolution. In M.H. Christiansen and S. Kirby, editors, *Language Evolution: The States of the Art*. Oxford University Press, 2003.
30. Kenichi Kurihara and Taisuke Sato. Variational bayesian grammar induction for natural language. In Yasubumi Sakakibara, Satoshi Kobayashi, Kengo Sato, Tetsuro Nishino, and Etsuji Tomita, editors, *ICGI*, volume 4201 of *Lecture Notes in Computer Science*, pages 84–96. Springer, 2006.
31. Nada Lavrac and Saso Dzeroski. *Inductive Logic Programming: Techniques and Applications*. Ellis Horwood, New York, 1994.
32. Erez Lieberman, Jean-Baptiste Michel, Joe Jackson, Tina Tang, and Martin A. Nowak. Quantifying the evolutionary dynamics of language. *Nature*, 449(7163):713–716, Oct 2007.
33. Solomon Marcus. Contextual grammars. *Rev. roum. de math. pures et appl*, 14:1473–1482, 1969.
34. Solomon Marcus, Gheorghe Paun, and Carlos Martín-Vide. Contextual grammars as generative models of natural languages. *Computational Linguistics*, 24(2):245–274, 1998.
35. James W. Minett and William S.-Y. Wang, editors. *Language Acquisition, Change and Emergence: Essays in Evolutionary Linguistics*. City University of Hong Kong Press, July 2005.
36. James W. Minett and William S-Y. Wang. Modelling endangered languages: The effects of bilingualism and social structure. *Lingua*, 118(1):19–45, January 2008.
37. J. Mira and A. Paredes. Interlinguistic similarity and language death dynamics. *Europhysics Letters*, 69(6):1031–1034, 2005.
38. Salikoko S. Mufwene. Language birth and death. *Annu. Rev. Anthropol.*, 33:201–222, 2004.
39. Stephen Muggleton. Inductive logic programming. *New Generation Computing*, 8(4):295, 1991.
40. Stephen Muggleton. Inductive logic programming: Derivations, successes and shortcomings. *SIGART Bulletin*, 5(1):5–11, 1994.
41. Stephen Muggleton and Luc De Raedt. Inductive logic programming: Theory and methods. *Journal of Logic Programming*, 19/20:629–679, 1994.

42. Makoto Nakamura, Takashi Hashimoto, and Satoshi Tojo. Prediction of creole emergence in spatial language dynamics. In Adrian Horia Dediu, Armand-Mihai Ionescu, and Carlos Martín-Vide, editors, *LATA*, volume 5457 of *Lecture Notes in Computer Science*, pages 614–625. Springer, 2009.
43. Daniel Nettle. Using social impact theory to simulate language change. *Lingua*, 108(2-3):95–117, June 1999.
44. P. Niyogi. The computational study of diachronic linguistics. In D. Lightfoot, editor, *Syntactic Effects of Morphological Change*. Cambridge University Press, 2002.
45. M. A. Nowak and N. L. Komarova. Towards an evolutionary theory of language. *Trends in Cognitive Sciences*, 5(7):288–295, 2001.
46. M. A. Nowak, N. L. Komarova, and P. Niyogi. Evolution of universal grammar. *Science*, 291:114–118, 2001.
47. M. A. Nowak, N. L. Komarova, and P. Niyogi. Computational and evolutionary aspects of language. *Nature*, 417:611–617, June 2002.
48. Alfonso Ortega, Marina de la Cruz, and Manuel Alfonseca. Christiansen grammar evolution: Grammatical evolution with semantics. *IEEE Trans. Evolutionary Computation*, 11(1):77–90, 2007.
49. Marco Patriarca and Teemu Leppanen. Modeling language competition. *Physica A: Statistical Mechanics and its Applications*, 338(1-2):296–299, July 2004.
50. Luc De Raedt, Paolo Frasconi, Kristian Kersting, and Stephen Muggleton, editors. *Probabilistic Inductive Logic Programming - Theory and Applications*, volume 4911 of *Lecture Notes in Computer Science*. Springer, 2008.
51. John N. Shut. Recursive adaptable grammars. Master's thesis, Computer Science Department, Worcester Polytechnic Institute, Worcester Massachusetts, USA, 1993.
52. Kenny Smith, Simon Kirby, and Henry Brighton. Iterated learning: A framework for the emergence of language. *Artificial Life*, 9(4):371–386, 2003.
53. L. Steels. The synthetic modeling of language origins. *Evolution of Communication*, 1(1):1–34, 1997.
54. L. Steels. *The Talking Heads Experiment. Volume 1. Words and Meanings*. Laboratorium, Antwerpen, 1999.
55. Luc Steels. Language as a complex adaptive system. In Marc Schoenauer, Kalyanmoy Deb, Günter Rudolph, Xin Yao, Evelyne Lutton, Juan J. Merelo Guervós, and Hans-Paul Schwefel, editors, *PPSN*, volume 1917 of *Lecture Notes in Computer Science*, pages 17–26. Springer, 2000.
56. Thomas Weise. Global optimization algorithms – theory and application. Electronic manuscript (e-book), 2007. http://www.it-weise.de/projects/book.pdf; checked June 2009.