Learning Deterministic DEC Grammars Is Learning Rational Numbers

Pieter Adriaans

Department of Computer Science,
University of Amsterdam,
Kruislaan 419, 1098VA Amsterdam
pietera@science.uva.nl
http://www.uva.nl

Abstract. Below I show that the class of strings that can be learned by a deterministic DEC grammar is exactly the class of rational numbers between 0 and 1. I call this the class of semi-periodic or rational strings. Dynamically Expanding Context (Dec) grammars were introduced by Kohonen in order to model speech signals ([8]). They can be learned in quadratic time in the size of the grammar. They have been used successfully in the automatic generation and analysis of music ([7], [5], [6]).

1 DEC Grammars

Kohonen [8] originally developed the idea of Dynamically Expanding Context (DEC) grammars. DEC grammars are interesting because they are very easy to learn (polynomial complexity) and they have interesting potential applications. They have been applied successfully to, among other things, musical composition ([7], [6]). Adriaans en van Dungen [5] have developed an implementation of k-DEC grammars that can be used for real time online learning[1]. k-DEC grammars are powerful enough to produce a subjectively acceptable representation of the aspects of the style of a composer like Bach. Yet, it is not even clear whether DEC grammars in their pure deterministic form are grammars in the proper sense of the word, since each grammar only recognizes one infinite string. In this context the DEC languages are much more restricted than the so-called ω-languages that have been studied by various authors ([4], [3], [2], [1]). The basic idea behind DEC-grammars is the notion that the occurrence of a symbol in a string is determined by its predecessors. A DEC-grammar consists of rules that determine which symbol can occur in which context. A context is a string of symbols that precedes a symbol. A rule of a DEC-grammar has the following form:

$$X \Rightarrow y$$

Here X is a context, y is a symbol. These rules can be used to generate a string of symbols: in context X write symbol y. We will study a limited set of languages

[1] Musical compositions and improvisations made with the Instant Composer Tool (ICT), based on DEC grammars, created by Adriaans and Van Dungen can be downloaded http://www.pieter-adriaans.com/music.html

Y. Sakakibara et al. (Eds.): ICGI 2006, LNAI 4201, pp. 320–326, 2006.

generated by deterministic DEC-grammars. These grammars generate finite or infinite strings. Use S for a start symbol. The infinite string:

$$Sababababababab\ldots$$

is described by the DEC-grammar:

$$S \Rightarrow a,\ a \Rightarrow b,\ b \Rightarrow a$$

The string:

$$Sabdbebab$$

is described by the DEC-grammar:

$$S \Rightarrow a,\ a \Rightarrow b,\ ab \Rightarrow d,\ d \Rightarrow b,\ db \Rightarrow e,\ e \Rightarrow b,\ eb \Rightarrow a$$

If we use this grammar to generate a string starting from the symbol S we get:

$$Sabdbebabdbebabdbebabdbebabdbebabdbebabdbeb\ldots$$

The algorithm to learn a DEC-grammar (i.e. to find a DEC-Grammar which describes a particular string) is very simple. Say you want to find a DEC-Grammar for the string $Sabae$. You start to read the first symbol of the string, and you form the rule: $S \Rightarrow a$. Then you read the next symbol (which is b) and you form the rule: $a \Rightarrow b$ You proceed along these lines, generating rules, until you obtain the grammar:

$$S \Rightarrow a,\ a \Rightarrow b,\ b \Rightarrow a,\ a \Rightarrow e$$

This grammar is not deterministic anymore, because there are two possible successors of a, namely b and e. In order to repair this you update the rules in the grammar that violate the deterministic structure by expanding their context as necessary: $(a \Rightarrow b)$ is replaced by $(Sa \Rightarrow b)$ and $(a \Rightarrow e)$ is replaced by $(ba \Rightarrow e)$. The final grammar becomes:

$$S \Rightarrow a,\ Sa \Rightarrow b,\ b \Rightarrow a\ \ ba \Rightarrow e$$

This grammar only generates a finite string, because there is no rule of the form $\ldots e \Rightarrow \ldots$. From these examples it is clear that DEC grammars do not always distinguish between finite and infinite strings. In order to avoid problems with finite strings I will in the rest of the paper only deal with infinite strings. Finite strings can be modelled using a special blank symbol $-$. The string $Sabae$ then becomes $Sabae - - - \ldots$ and the corresponding grammar is:

$$S \Rightarrow a,\ Sa \Rightarrow b,\ b \Rightarrow a\ \ ba \Rightarrow e\ \ e \Rightarrow -,\ - \Rightarrow -$$

1.1 Numbers and Strings

Intuitively an infinite string with a finite DEC grammar must have a repeating pattern. To formalize this intuition we turn our attention to number theory. We use the standard number classes:

Definition 1. **N** (**Z**+) *is the set of natural numbers* $1, 2, 3 \ldots$
Z *the set of whole numbers or integers* $\ldots -3, -2, -1, 0, 1, 2, 3 \ldots$.
Q *the set of rational numbers (expressible as* p/q *where* p *and* q *are integers),*
R *the set of real numbers. An algebraic number is expressible as the root of a polynomial with integer coefficients. Non-algebraic numbers are transcendental.*

We need corresponding notions for strings. Let A be a finite alphabet. We define:

- A^*: the set of all finite strings consisting of elements of A.
- A^k: the set of all strings of length k consisting of elements of A.
- $A^{<k}$: the set of all strings of length less than k consisting of elements of A.
- A^∞: the set of all infinite strings consisting of elements of A.
- $[0, 1) \subset \mathbf{R}$ The interval between 0 (included) and 1 (not included).

Important are the set of binary strings $\{0, 1\}^*$ and the set of decimal strings $\{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}^*$.

Fact 1. *There is a one to one correspondence between elements of* A^∞ *and* $[0, 1) \subset \mathbf{R}$. *Each element of* A^∞ *corresponds with a unique fraction in a number system with base (radix)* A.[2]

Take the set of decimal fractions x such that $0 \leq x < 1$ and the associated set of strings $\{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}^\infty$. There are *terminating decimals* like $5/8 = 0.625$ and *non-terminating decimals* like $1/3 = 0.333\ldots$ With each terminating decimal I will associate an infinite string with a tail of zero's: like $5/8 = 0.625000$ The non-terminating decimals can be periodic, i.e. $2/7 = 285714\ldots$ where the block of digits 285714 repeats itself indefinitely.

We will call a decimal fraction $0.n$ semi-periodic if it is either finite or periodic with an initial arbitrary segment.

Theorem 2. $0.n$ *is a semi-periodic decimal fraction if and only if* $0.n$ *is rational.*

Proof: (*If*) We first take the simple periodic case. Suppose $0.n$ is a periodic decimal with a repeating block d with length $|d| = l$. In this case $0.n = d/(10^l - 1)$ which is a rational number. The semi-periodic case follows from the fact that **Q** is closed for addition and subtraction.
(*Only if*) Suppose $0.n$ is rational. We have $p/q = 0.n$. Consider the standard division algorithm. A division by q will have at most $q - 1$ rest values. Dividing by q therefore is either terminating or has a repeating block of at most $q - 1$ digits, i.e. it is semi-periodic.

Definition 2. *Suppose* $0.n$ *is a semi-periodic decimal fraction. The string* n *in* $\{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}^\infty$, *with possibly an infinite tail of zero's, is called a semi-periodic or rational string.*

Definition 3. *A deterministic connected DEC grammar is a triple* $< \Sigma, P, S >$ *where:*

[2] Strictly spoken the one to one correspondence is between representations of elements of A^∞ and $[0, 1) \subset \mathbf{R}$. Note that 0.9999... = 1.0.

- Σ *is a finite set of terminals.*
- $S \in \mathbf{N}$ *is a start symbol.*
- P *is a finite set consisting of the initial rule* $S \Rightarrow \xi$ *and a finite set of concatenation rules of the form* $\eta \Rightarrow \xi$*, where* $\xi \in \Sigma$ *and* $\eta \in \Sigma^*$*.*

If we have a string $\delta \bullet \eta$ *then the rule* $\eta \Rightarrow \xi$ *allows us to create* $\delta \bullet \eta \bullet \xi$ *(*$\delta \in \Sigma^*$*). The rules* $\eta \Rightarrow \xi$ *are deterministic, connected and minimal.*

- *Deterministic: no* η *is a prefix of any other* η' *nor identical.*
- *Connected: for each finite* s *string that can be constructed from the start symbol there is a rule* $\eta \Rightarrow \eta \bullet \xi$ *such that* η *is a postfix of* s*.*
- *Minimal: there are no superfluous rules.*

Each DEC grammar generates a unique infinite string from the initial symbol. Let $\bullet$ be a concatenation operator and let S_i $(i \in \mathcal{N})$ be the finite string $S \bullet s_1 \bullet \ldots \bullet si$.

```
An algorithm for learning deterministic DEC grammars
Start
   Input: An infinite string s = S • s1 • s2 • s3 • ...
   P =: {S ⇒ S • s1}
   c =: 1
   Loop
        s = Sc−1 • sc • sc+1 • ... (Sc−1 possibly empty)
        P′ =: P ∪ {sc ⇒ sc+1}
        P =: make-deterministic(P′, Sc+1)
        c =: c + 1
   EndLoop
End

make-deterministic(P, Sc)
   If
        There are two rules η ⇒ ξ and η′ ⇒ ξ' such that
        either η′ = η or η′ is a proper prefix of η.
   then
        Do
        If
           η′ is a proper prefix of η
        then
           P=: expand-context(P, Sc, η′ ⇒ ξ')
        If
           η′ = η
        then
           P=: expand-context(P, Sc, η′ ⇒ ξ')
           P=: expand-context(P, Sc, η ⇒ ξ)
        Od
   else
```

```
        Do nothing

expand-context(P, S_c, η ⇒ ξ)
   Do
        S_c = S_i • s_{i+1} • η • ξ • δ
        (ξ and s_{i+1} in Σ, S_i and δ in Σ* and possibly empty)
        P' =: (P − (η ⇒ ξ)) ∪ (s_{i+1} • η ⇒ ξ)
        make-deterministic(P', S_c)
   Od
```

Definition 4. *An infinite string s can be learned by a deterministic DEC grammar if there is a constant c such that after scanning s_c the rule set P stabilizes, i.e. P has a finite set of rules with finite heads.*

Theorem 3. *A string can be learned by a deterministic DEC grammar if and only if it is semi-periodic.*

Proof: (*If*) Suppose a string s is deterministic DEC learnable. Since the rules in P in the limit are finite deterministic and connected there must be a point in s where the rules start to loop, i.e. s must be semi-periodic.
(*Only if*) Suppose s is semi-periodic with an indefinitely repeating block d after some s_c for a constant c. Take $k = 2c$. Run k loops of the learning algorithm. Suppose that in the k-th loop we construct a rule $\eta \Rightarrow sk$.
Case 1) $|\eta| \leq c$. In this case η is a suffix of the periodic part of Sk. Therefore it will not be updated in consecutive loops.
Case 2) $|\eta| > c$. Here η consists of three parts $\eta = si \bullet \mu \bullet \nu$. ($\mu, \nu \in \Sigma^*$, $si \in \Sigma$) such that ν lies in the periodic part of Sk and μ in the non-periodic part. By its construction P will also contain a second rule $sj \bullet \mu \bullet \nu \Rightarrow sl$ ($sk \neq sl$, $si \neq sj$). This is a contradiction since $si \bullet \mu \bullet \nu$ is a suffix of $Sk - 1$ that is longer than $|Sk|/2$. So after k loops only rules with heads in the periodic part of the string will be created. The number of prefix free rules of this form that can exist is d. Therefore P will stabilize after at most k loops.

Lemma 1. *Suppose s is semi-periodic with an indefinitely repeating block d after some s_c for a constant c. An upperbound for the complexity of learning a DEC grammar for s is $O(c'k^2)$ where c' is constant and $k = 2c$.*

The proof of this lemma is immediate clear from the proof of theorem 3 and the observation that the algorithm has to scan a prefix of a length of maximally k of the string s at each pass of the central loop.

Lemma 2. *An infinite string is deterministic DEC learnable if and only if there exists a rational number that describes that string.*

This last lemma, that is a direct consequence of theorem 3, is a beautiful example of learning by compression. Each deterministic DEC grammar can be described by a rational number in a number system with the lexicon as its base.

To show that there are strings with short descriptions that are non periodic one only has to consider irrational numbers. The decimal expansion of the square

root of 2 will be non-periodic. This holds in general for the class of so-called algebraic numbers: numbers that can be expressed as the root of a polynomial with integer coefficients. In this case $x^2 - 2 = 0$. The decimal expansions of transcendental numbers like e and π are also non-periodic but they have no algebraic definition.

1.2 Conclusion and Further Work

I have shown that DEC grammars can be learned easily in quadratic time and I have shown that the class of strings that can be learned in this way is equal to the class of rational numbers between 0 and 1. I believe that DEC grammars are interesting because they represent one of the simplest string analysis methods that I know. Further work would involve a deeper analysis of the possibilities to model musical styles. Another line of research would be aiming at a better understanding of the relation between various ω-languages and variants of DEC grammars. A specific deterministic DEC grammar only recognizes one infinite string. The whole class of strings that is recognized by the whole class of deterministic DEC grammars is exactly $\mathbf{Q}+$. The concept of DEC grammars might be weakened in various way in order to make the recognition of richer language classes possible. All these definitions focus on the structure of the rule set P:

Definition 5. *A non-deterministic DEC grammar is a DEC grammar in which the rule set P is non-deterministic, i.e. P is a finite set consisting of the initial rule $S \Rightarrow \xi$ and a finite set of concatenation rules of the form $\eta \Rightarrow \xi$, where $\xi \in \Sigma$ and $\eta \in \Sigma^*$, and where we can have two rules $\eta \Rightarrow \xi$ and $\eta \Rightarrow \xi'$ such that $\xi \neq \xi'$*
A stochastic DEC grammar is a tuple $< \Sigma, P, \mathcal{P}, \mathcal{S} >$ such that $< \Sigma, P, S >$ is a non-deterministic DEC grammar and $\mathcal{P}$ is a probability distribution over P.
A k-DEC grammar is a non-deterministic DEC grammar for which the rules in P of the form $\eta \Rightarrow \xi$ satisfy $|\eta| \leq k$, i.e. the expansion of the contexts is limited to a ceiling k.

A detailed analysis of how these variants of DEC grammars behave in Büchi automata would be useful.

Acknowledgments. This project is supported by a BSIK grant from the Dutch Ministry of Education, Culture and Science (OC&W) and is part of the ICT innovation program of the Ministry of Economic Affairs (EZ).

References

1. de La Higuera C. and Janodet J. C., Inference of W-languages from prefixes, Theoretical Computer Science, vol. 313, 2, (2004), 295–312
2. Thomas, D. G., Humrosia Begam M., Subramanian K. G., Gnanasekaran S., Learning of Regular Bi-omega Languages, ICGI '02: Proceedings of the 6th International Colloquium on Grammatical Inference, (2002), 283–292

3. Thomas W., Automata on infinite objects, Handbook of theoretical computer science (vol. B): formal models and semantics, (1990), 133–191, MIT Press, Cambridge, MA, USA
4. Staiger L., ω-languages, Handbook of formal languages, vol. 3: beyond words, (1997), 339–387, Springer-Verlag New York, Inc., New York, NY, USA
5. Adriaans P., van Dungen M., A method for automatically controlling electronic musical devices by mean of real-time construction and search of a multi-level data structure, European Patent no. 1.062 656, United States Patent 6313390, (2001)
6. Kohonen T., Method for controlling an electronic musical device by utilizing search arguments and rules to generate digital code sequences, United States Patent 5418323, (1995)
7. Kohonen T., Laine P., Tiits K., Torkkola K., A Nonheuristic Automatic Composing Method, Music and Connectionism, MIT press, Peter M. Todd, D. Gareth Loy (eds.), (1991), 229–242
8. T. Kohonen, Dynamically Expanding Context, with application to the correction of symbol strings in the recognition of continuous speech, Proceedings of the 8th International Conference on Pattern Recognition (8th ICPR), Paris France, (1986), 1148–1151