Strongly Regular Grammars and Regular Approximation of Context-Free Languages

Ömer Eğecioğlu

Department of Computer Science
University of California, Santa Barbara, CA 93106
omer@cs.ucsb.edu

Abstract. We consider algorithms for approximating context–free grammars by regular grammars, making use of Chomsky's characterization of non–self–embedding grammars as generating regular languages and a transformation by Mohri and Nederhof on sets of mutually recursive nonterminals. We give an exposition of strongly regular grammars and this transformation, and use it as a subprocedure to obtain tighter regular approximations to a given context-free grammar. In another direction, the generalization by a 1–lookahead extends Mohri and Nederhof's transformation by incorporating more context into the regular approximation at the expense of a larger grammar.

1 Introduction

The approximation of context-free languages with regular languages is a problem which has been extensively studied because of its importance in a number of applications [6,5,4]. A general framework for the approximation of formal languages by regular languages was studied by Shallit [7]. We consider the case in which a given context-free grammar is approximated from above by a regular grammar.

The algorithms discussed here make use of a transformation introduced by Mohri and Nederhof [4] as a subprocedure to provide tighter regular approximations. As in [4], the approximating grammar obtained is non–self–embedding. Such grammars generate regular languages by a result of Chomsky [2].

We assume that the grammar is in appropriate normal form, although for real-life problems discussed in [4] normal forms would already incur a quadratic increase in the size of the grammar, and may not be desirable. The starting point of normal forms is not a necessary assumption but simplifies the exposition: the resulting regular grammars are easier to keep track of because of the simplicity of their transition diagrams, for example.

We start with an exposition of the transformation of Mohri and Nederhof [4] and then discuss its variants that which provide tighter regular approximations. Regular approximation by two *cycle-breaking* based methods is presented in section 6 and approximation by *1–lookahead* is discussed in section 7.

V. Diekert and D. Nowotka (Eds.): DLT 2009, LNCS 5583, pp. 207–220, 2009.

2 Notation and Definitions

A *context–free grammar* (CFG) G is a 4–tuple $G = (N, T, P, S)$, where N and T are disjoint finite sets of *nonterminals* and *terminals*, respectively. P is a finite set of productions (rules); each production is of the form $A \to \alpha$, where A is a nonterminal and α is a string of symbols (sentential forms) from V^* where $V = N \cup T$. S is the *start symbol*. The relation $\to$ on $N \times V^*$ is extended to a relation on $V^* \times V^*$ as usual. The transitive and reflexive closure of $\to$ is denoted by $\stackrel{*}{\to}$. The language generated by an $A \in N$ is $\{w \in T^* \mid A \stackrel{*}{\to} w\}$. The language generated by G is $L(G) = \{w \in T^* \mid S \stackrel{*}{\to} w\}$. A context–free language (CFL) is a language generated by a CFG. The number of rules in the grammar G is denoted by $|G|$. We use the commonly used convention of denoting the set of nonterminals in N by capital letters $A, B, C, \ldots$, the set of terminals T with $a, b, c, \ldots$, strings of terminals in T^* with $u, v, w, \ldots$, strings of nonterminals and terminals in V^* by $\alpha, \beta, \gamma, \ldots$. The empty string is denoted by ε. Productions with left-hand side $A \in N$ are referred to as the *rules of* A or A*-rules*. The union of rules of $A \in M$ for $M \subseteq N$ are the *rules of* M.

If all productions of G are of the form $A \to wB$ or $A \to w$ then G is called a *right–linear* grammar. If all productions are of the form $A \to Bw$ or $A \to w$ then G is a *left–linear* grammar. G is a *regular* grammar if it is either right–linear or left–linear. Regular grammars characterize regular languages. In addition to regular grammars, regular languages can be represented in many forms such as finite automata (1NFA, 1DFA, 2NFA, 2DFA), and regular expressions, each giving a different insight into the structure of the language. In the Chomsky hierarchy of languages, context–free languages properly contain regular languages. Thus context–free grammars can generate languages which are non–regular, and in fact many languages of interest are context–free but non–regular.

A context–free grammar G is *self–embedding* (SE), if there exists a derivation $A \stackrel{*}{\to} \alpha A \beta$, with both α, β non–empty. G is *non–self–embedding* (NSE) if it is not self–embedding. By a result of Chomsky [2], any NSE grammar generates a regular language. For more details on notation and basic properties of CFGs and CFLs, the reader is referred to Hopcroft and Ullman [3].

3 Mohri and Nederhof's Transformation

In this section we describe the transformation of Mohri and Nederhof [4]. First, consider *strongly regular* CFGs which are defined as follows. Let $\Re$ be the relation defined on the set of nonterminals N of G by:

$$A \Re B \Leftrightarrow (\exists \alpha, \beta \in V^* \text{ s.t. } A \stackrel{*}{\to} \alpha B \beta) \wedge (\exists \alpha, \beta \in V^* \text{ s.t. } B \stackrel{*}{\to} \alpha A \beta) \ .$$

Note that α and β are not required to be nonempty. $\Re$ defines an equivalence relation on N, and partitions N into equivalence classes of nonterminals called *mutually recursive nonterminals*. Strongly regular grammars are grammars in which the rules of each set M of mutually recursive nonterminals are either all left–linear or all right–linear. In determining whether a rule of M is right–linear

or left–linear, the nonterminals that do not belong to M are treated as if they are terminals. The class of languages generated by strongly regular grammars coincide with the class of languages generated by NSE grammars and therefore these languages are regular.

There are efficient algorithms to construct finite automata from strongly regular grammars. An offline construction was given by Nederhof in [6]. One may also construct an alternative, compact representation of the regular language generated, from which a finite automaton for it may be constructed, as shown by Mohri and Pereira in [5]. Briefly, the algorithm is as follows:

1. Determine sets of mutually recursive nonterminals by computing the strongly connected components of the graph of the grammar[1].
2. Construct a the automaton $\mathcal{K}(M)$ for each equivalence class M of mutually recursive nonterminals with unspecified initial state (in case M is right–linear) or unspecified final states (in case M is left–linear). For any $A \in M$, the automaton $\mathcal{N}(A)$ accepting terminals generated from A can be obtained from $\mathcal{K}(M)$.
3. For each input string w, obtain $\mathcal{N}(S)$ from the $\mathcal{K}(M)$ that satisfies $S \in M$. This automaton is then expanded in a lazy way by substituting other automata $\mathcal{N}(A)$ for occurrences of A in $\mathcal{N}(S)$ that are encountered while processing w.

In [4], Mohri and Nederhof describe a transformation that yields a strongly regular grammar from a given context-free grammar: for each class of mutually recursive nonterminals M such that the corresponding rules are not all right–linear or not all left–linear with respect to the nonterminals of M, the following transformation is applied:

1. For each nonterminal $A \in M$, introduce $A' \notin N$ and add the production $A' \to \varepsilon$ to the grammar.
2. For each production of the form: $A \to \alpha_0 B_1 \alpha_1 B_2 \alpha_2 \ldots B_m \alpha_m$ with $m \geq 0$, $B_1, \ldots, B_m \in M, \alpha_0, \ldots, \alpha_m \in (T \cup (N - M))^*$, replace it with

$$\begin{aligned} A &\to \alpha_0 B_1 \\ B'_1 &\to \alpha_1 B_2 \\ B'_2 &\to \alpha_2 B_3 \\ &\vdots \\ B'_{m-1} &\to \alpha_{m-1} B_m \\ B'_m &\to \alpha_m A' \end{aligned}$$

   If $m = 0$, this set of productions only contains $A \to \alpha_0 A'$.

[1] The graph of the grammar has a node for each nonterminal, and an edge from node A to node B iff B appears on the right hand side of a production having A on the left hand side.

All of the rules for M in the transformed grammar are right–linear. Therefore the resulting grammar is strongly regular. We will refer to this transformation as the *MN-transformation*, and the resulting regular approximation as the *MN-approximation*. The MN-approximation $\mathcal{L}(G)$ is a superset of $\mathcal{L}(G)$.

Since we are interested in how well the resulting regular language approximates the given one, we will consider the effect of the transformation on an individual equivalence class of mutually recursive set of nonterminals.

Example 1. As an example of the MN-transformation, consider the grammar G with productions

$$\begin{aligned} A &\to aBa \\ B &\to bA \mid b \end{aligned}$$

in which A is the start state. This grammar generates the nonregular language $\{(ab)^n a^n \mid n > 0\}$. We can show that the MN-transformation approximates this language by the regular language $(ab)^+a^*$. In G, A and B form a mutually recursive set of nonterminals. The transformed grammar G' consists of the productions

$$\begin{aligned} A &\to aB \\ A' &\to B' \mid \varepsilon \\ B &\to bA \mid bB' \\ B' &\to aA' \mid \varepsilon \ . \end{aligned}$$

The following derivation in G' simulates the derivation of $ababaa$: $A \to aB \to abA \to abaB \to ababB' \to ababaA' \to ababaB' \to ababaaA' \to ababaa$. For the nonterminal B, the newly introduced nonterminal B' serves two purposes:

1. It allows the termination of a derivation from B by replacing B with the terminals that B derives. In our example, $B \to b$ in G is simulated using the productions $B \to bB', B' \to \varepsilon$ from G'.
2. Since the productions are all right–linear, it provides a mechanism to return back to the branching point from the original production and continue the derivation.

However, this last point also introduces ambiguities in the grammar. Nonterminal pairs B and B' mark the beginning and end of strings generated by B in the original grammar. This can be used to compile the transformed grammar into a finite-state transducer that outputs bracketed strings equivalent to parse trees [4]. At the same time by making use of B', it is possible to continue the derivation from the right of B in a current sentential form by any production that has B on its right hand side, not necessarily the next nonterminal in the sentential form (see Example 5).

4 The Automaton for the MN-Approximation

Assume that N itself is a mutually recursive set of nonterminals. The structure of the transition diagram of the automaton constructed from the right-linear

grammar G' in the standard way [3] allows us to quickly determine a regular expression For the MN-approximation, especially when the given grammar is in Chomsky Normal Form (CNF).

The transition diagram is organized as two rows of states where each state is labeled with a nonterminal in G', grouped as follows. (see Figure 1 (a) for the automaton corresponding to G' of Example 1.)

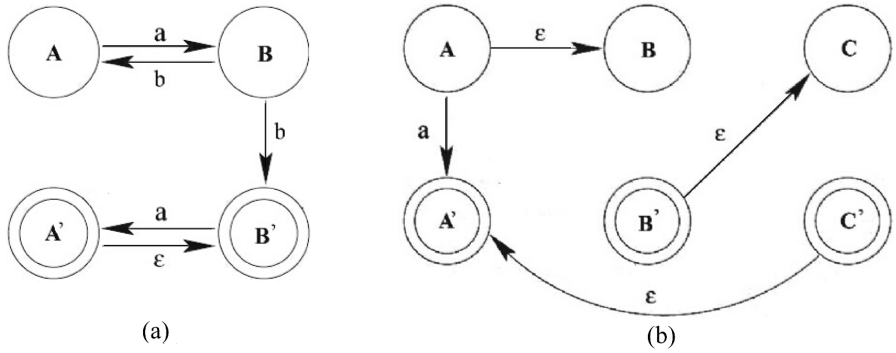

Fig. 1. (a) Automaton corresponding to the transformation of the grammar in Example 1. (b) Transformed CNF rules $A \to BC \mid a$.

- The nonterminals of the original grammar are represented in the upper part.
- The newly introduced nonterminals are represented in the lower part.
- The final states are the states in the lower part of the automaton.
- Every production of the type $A \to w$ induces a transition from the upper part to the lower part. The transition from B to B' in Figure 1 (a) that comes from the rule $B \to b$ demonstrates this.
- For every production of the type $A \to \alpha_0 B_1 \alpha_1 B_2 \alpha_2 \ldots B_m \alpha_m$ in G, the first rule $A \to \alpha_0 B_1$ in G' induces a transition strictly within the upper part of the automaton. The transition from A to B in Figure 1 (a) that comes from the rule $A \to aBa$ in this way demonstrates this.
- The last production in $B'_m \to \alpha_m A'$ in G' induces a transition strictly within the lower part of the automaton. The transition from B' to A' in Figure 1 (a) that comes from $B' \to aA'$ demonstrates this.
- All other intermediate productions in G' induce transitions from the lower part of the automaton to the upper part.

In CNF, the productions of G are of the form $A \to BC$ or $A \to a$. Assuming that A, B, C are all in the same set of mutually recursive nonterminals, the transformation for the above mentioned rules yields:

$$\begin{aligned} A &\to B \\ B' &\to C \\ C' &\to A' \\ A &\to aA' \ . \end{aligned}$$

The first production leads to an ε-transition in the upper part of the automaton. The second production leads to an ε-transition from the lower part to the upper part. The third production leads to an ε-transition within the lower part. Its only productions of the fourth kind that actually derive all the terminals, and they result in transitions from the upper part of the automaton to the lower part. This is illustrated in Figure 1 (b).

5 NSE Grammars

We will assume for the rest of the discussion, that G is in CNF and that N is a mutually recursive set.

Recall that G is SE if for some nonterminal A, there is a derivation $A \stackrel{*}{\to} \alpha A \beta$, with both α, β non–empty. G is NSE if for any nonterminal A and a derivation $A \stackrel{*}{\to} \alpha A \beta$, either $\alpha = \varepsilon$ or $\beta = \varepsilon$. In general, it is undecidable if a context–free grammar generates a regular language [8], or even if $\mathcal{L}(G) = T^*$. However whether a context–free grammar is NSE is decidable [1]. By Chomsky's result, if G is NSE then $\mathcal{L}(G)$ is regular. Of course this leaves open the possibility that G is SE, but $\mathcal{L}(G)$ is nevertheless regular. The property $A \stackrel{*}{\to} \alpha A \beta$, with $\alpha, \beta \neq \varepsilon$ enables the grammar to generate terminal strings of the form $u^i x v^i$. If u and v are sufficiently complex, then the language has a counting property and cannot be regular. Therefore the nature of the terminal strings derivable by the self-embedding in G is the thin line that separates the decidable question of "Is G NSE?" and the undecidable question of "Is $\mathcal{L}(G)$ regular?".

We make use of some of the ideas from [1]. Define the edge-colored production graph $CP(G)$ for a grammar G by starting with the nonterminals as vertices. Since G is in CNF, all productions are of the form: $A \to BC$ or $A \to a$. In $CP(G)$, we are only concerned with productions of the form $A \to BC$. For every production $A \to BC$, $CP(G)$ has an edge from node A to node B colored l, and an edge from A to C colored r. We note that in $CP(G)$ self-loops are possible, and if we ignore the colors on the edges, then the graph is strongly connected. Also, an l-colored edge can arise from more than one rule, e.g. $A \to BC \mid BD$. Similarly for r-colored edges. Therefore the number of l-colored edges is not necessarily equal to the number of r-colored edges.

Theorem 1. *G is NSE iff all cycles in $CP(G)$ are monochromatic.*

Proof. Any derivation $A_1 \stackrel{*}{\to} \alpha A_1 \beta$ in G corresponds to a cycle in $CP(G)$. If the cycle containing A_1 is monochromatic with color l, then this a derivation is of the form $A_1 \to A_2 B_2 \to A_3 B_3 B_2 \to \cdots \to A_k B_k B_{k-1} \cdots B_2 \to A_1 B_1 B_k \cdots B_2$ with $\alpha = \varepsilon$. Similarly, if the cycle containing A_1 is monochromatic with color r, then $\beta = \varepsilon$. Conversely, any cycle with an edge $A_1 \to A_2$ colored l followed by an edge $A_2 \to A_3$ colored r gives a derivation of the form $A_1 \to A_2 B_2 \to B_3 A_3 B_2 \to \cdots \to \alpha A_1 \beta$ where α starts with B_3 and β ends with B_2. Therefore $\alpha, \beta \neq \varepsilon$, and G is SE.

6 Regular Approximation by Cycle-Breaking

Rather than replacing the rules of the grammar with the appropriate approximations, an alternative approach is to only use the approximation for non–monochromatic cycles in $CP(G)$, and leave the rest of the graph intact. We present an example to demonstrate this approach.

Example 2. Let $T = \{a, b\}$ and consider the CFG G:

$$A_1 \to A_2A_3 \mid b,\ A_2 \to A_3A_4,\ A_3 \to A_4A_5,\ A_4 \to A_5A_1,\ A_5 \to A_1A_2 \mid a\ . \quad (1)$$

Applying the MN-transformation, the resulting regular grammar G' is:

$$\begin{array}{l} A'_1 \to \varepsilon, \quad A'_2 \to \varepsilon, \quad A'_3 \to \varepsilon, \quad A'_4 \to \varepsilon, A'_5 \to \varepsilon \\ A_1 \to bA'_1 \\ A_5 \to aA'_5 \\ A_1 \to A_2,\ A'_1 \to A_2, A'_1 \to A'_4 \\ A_2 \to A_3,\ A'_2 \to A_3, A'_2 \to A'_5 \\ A_3 \to A_4,\ A'_3 \to A_4, A'_3 \to A'_1 \\ A_4 \to A_5,\ A'_4 \to A_5, A'_4 \to A'_2 \\ A_5 \to A_1,\ A'_5 \to A_1, A'_5 \to A'_3 \end{array} \quad (2)$$

To get a sense of the approximation, note that G is equivalent to the grammar

$$\begin{array}{l} A_1 \to A_5A_1A_5A_5A_1A_5A_1A_5 \mid b \\ A_5 \to A_1A_5A_1A_5A_5A_1 \mid a\ , \end{array}$$

and in particular $\mathcal{L}(G)$ contains no word of length $2, 3, \ldots, 7$. The automaton corresponding to G' is shown in Figure 2. The language accepted is T^+. Therefore

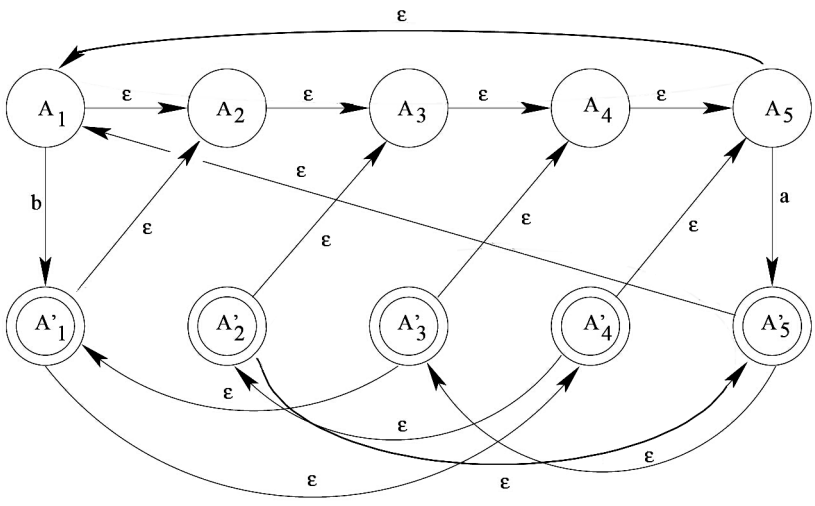

Fig. 2. The automaton for the grammar in Example 2

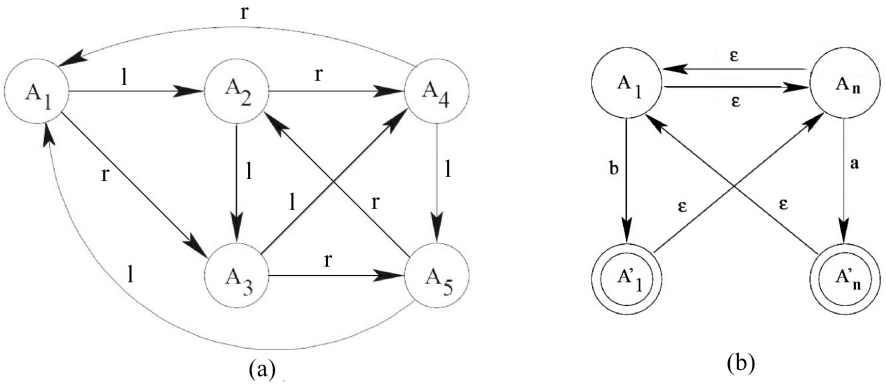

Fig. 3. (a) Edge colored production graph $CP(G)$ for the grammar in Example 2. (b) Transformation of the grammar G_n of Example 4 generates T^+.

the MN-approximation to $\mathcal{L}(G)$ is T^+. We also note from Figure 2 that any nonterminal A_i in this example generates T^+.

For the approximation using cycle-breaking, we first construct $CP(G)$. This is shown in Figure 3 (a) for the grammar in Example 2. Using cycle-breaking, it is possible to devise different regular approximations to $\mathcal{L}(G)$. We can use the MN-approximation itself as a subroutine, for example. Alternatively, we break non–monochromatic cycles by introducing a new nonterminal for each edge eliminated. Depending on what we allow these new nonterminal to derive in the new grammar, we obtain regular approximations that are supersets or subsets of the given language. It is also possible to mix these two ideas.

We consider cycle-breaking by using the MN-transformation first, and then describe cycle-breaking based on introduction of new nonterminals.

6.1 Cycle-Breaking Using the MN-Transformation

To eliminate an l-colored edge $A_i \to A_j$ in $CP(G)$, we proceed as follows. Suppose the A_i-productions of G the form $A_i \to A_j A_k$ are

$$A_i \to A_j A_{k_1} \mid A_j A_{k_2} \mid \cdots \mid A_j A_{k_t} \tag{3}$$

and G' is the grammar obtained from G by the MN-transformation. Make a fresh copy of G' by relabeling each A_k by B_k where a distinct symbol B is is used for every edge eliminated. G'_j be this grammar with start symbol B_j. We replace the rules (3) with

$$A_i \to B_j A_{k_1} \mid B_j A_{k_2} \mid \cdots \mid B_j A_{k_t} \ . \tag{4}$$

Similarly, to eliminate an r-colored edge $A_i \to A_j$, assume that the A_i-rules of the form $A_i \to A_k A_j$ are

$$A_i \to A_{k_1} A_j \mid A_{k_2} A_j \mid \cdots \mid A_{k_t} A_j \ . \tag{5}$$

Let G' be the grammar obtained from G by the MN-transformation. Make a fresh copy of G' by relabeling each A_k by C_k where a distinct symbol C is is used for every edge eliminated. Let G'_j be this grammar with start symbol C_j. Then we replace the rules (5) with

$$A_i \to A_{k_1}C_j \mid A_{k_2}C_j \mid \cdots \mid A_{k_t}C_j \ . \tag{6}$$

Example 3. Let G be the grammar in Example 2. From Figure 3 (b), we see that eliminating the r-colored edges $A_4 \to A_1, A_5 \to A_2$ and the l-colored edge $A_5 \to A_1$ are sufficient to make all cycles monochromatic in $CP(G)$. Using the above idea, we obtain the grammar

$$\begin{aligned} A_1 &\to A_2A_3 \mid b \\ A_2 &\to A_3A_4 \\ A_3 &\to A_4A_5 \\ A_4 &\to A_5B_1 \\ A_5 &\to D_1C_2 \mid a \end{aligned}$$

together with three copies of the productions in (2), one each for nonterminal B, C and D. From the automaton in Figure 2, we see that every nonterminal in G' derives T^+. For this example, the grammar G'' obtained by cycle-breaking using the MN-transformation generates the language

$$(T^*T^3 + aT^*T)(T^*T^2 + a)(T^*T^3 + aT^*T)^2(T^*T^2 + a) + b \ , \tag{7}$$

which generates no word of length $2, 3, \ldots, 7$, and is strictly contained in the MN-approximation.

The derivations in the NSE grammar obtained by breaking cycles by using the MN-transformation can be simulated by the NM-transformation of the original grammar. Thus

Theorem 2. *The regular approximation G'' produced by breaking all non–monochromatic cycles of $CP(G)$ using the MN-transformation is finer than the MN-approximation G' of G. In other words $\mathcal{L}(G) \subseteq \mathcal{L}(G'') \subseteq \mathcal{L}(G')$.*

How close is $\mathcal{L}(G'')$ to $\mathcal{L}(G')$? The following example gives an idea.

Example 4. For $n \geq 3$ and $T = \{a, b\}$, consider the grammar G_n with rules $A_1 \to A_2A_3 \mid b$, $A_2 \to A_3A_4, \ldots, A_{n-2} \to A_{n-1}A_n$, $A_{n-1} \to A_nA_1$, $A_n \to A_1A_2 \mid a$. Suppose we obtain G''_n by breaking all l-colored cycles by the MN-transformation, and let G'_n be grammar of the MN-transformation directly applied to G_n. Then $\mathcal{L}(G'_n) = T^+$ regardless of n, whereas

$$\mathcal{L}(G''_n) = \begin{cases} (\varepsilon + T^*T^m)b & \text{if } n = 2m, \\ (\varepsilon + T^*T^n)(b + T^*T^m a) & \text{if } n = 2m+1 \ . \end{cases}$$

The grammar G_n and the approximations given above are considered in detail next.

6.2 Cycle-Breaking Using New Nonterminals

We can simplify the resulting grammar by bypassing the MN-transformation for cycle-breaking. This done at some expense. We still have $\mathcal{L}(G) \subseteq \mathcal{L}(G'')$ but the inclusion $\mathcal{L}(G'') \subseteq \mathcal{L}(G')$ of Theorem 2 is lost.

To eliminate an l-colored edge $A_i \to A_j$ in $CP(G)$ with new nonterminals, we proceed as follows: Suppose the A_i-rules of G the form $A_i \to A_j A_k$ are $A_i \to A_j A_{k_1} \mid A_j A_{k_2} \mid \cdots \mid A_j A_{k_t}$.

1. Replace these by $A_i \to B_j A_{k_1} \mid B_j A_{k_2} \mid \cdots \mid B_j A_{k_t}$.
2. Add the productions

$$\begin{aligned} B_j &\to B_a B_j, \ \forall a \in T, \\ B_a &\to a, \ \forall a \in T, \\ B_j &\to a, \ \forall a \in T, \end{aligned}$$

where B_j and B_a, $(a \in T)$ are new nonterminals.

Elimination of an r-colored edge is done similarly. Call the resulting grammar G''. This creates no new non–monochromatic cycles, and the edge $A_i \to A_j$ in $CP(G)$ has been eliminated in $CP(G'')$. In effect, we are replacing the terminals derivable from A_j for the A_i-rules that involve A_j, by terminals derivable from B_j. We generously made B_j derive all of T^+, so that the language generated by G'' is a superset of the language generated by G. We note that if it is possible to make each B_j derive a regular language that is contained in what A_j derives in G, then cycle-breaking gives a regular approximation to $\mathcal{L}(G)$ from below.

One obvious way to eliminate non–monochromatic cycles is to break all l-colored edges in $CP(G)$. For the example grammar $G = G_5$, we can write the resulting grammar (using T^+ for any nonterminal that now derives only T^+ to simplify notation) by

$$A_1 \to T^+ A_3 \mid b, \ A_2 \to T^+ A_4, \ A_3 \to T^+ A_5, \ A_4 \to T^+ A_1, \ A_5 \to T^+ A_2 \mid a$$

so that the approximating language is generated by $A_1 \to (T^+)^5 A_1 \mid (T^+)^2 a \mid b$. A regular expression for this language is

$$(\varepsilon + T^* T^5)(b + T^* T^2 a) \ . \tag{8}$$

For the same G, eliminating all r-colored edges from $CP(G)$, we obtain the grammar

$$A_1 \to A_2 T^+ \mid b, \ A_2 \to A_3 T^+, \ A_3 \to A_4 T^+, \ A_4 \to A_5 T^+, \ A_5 \to A_1 T^+ \mid a$$

which is equivalent to $A_1 \to A_1 (T^+)^5 \mid a (T^+)^4 \mid b$. Therefore the approximation is given by the regular expression

$$(\varepsilon + T^* T^5)(b + a T^* T^4) \ . \tag{9}$$

In either case, the resulting approximating language is properly contained in the language $(a + b)^+$ of the MN-approximation.

To eliminate non–monochromatic cycles in $CP(G)$, removing all l-colored edges or all r-colored edges may be an overkill. It suffices to eliminate any set of edges with the property that all the cycles in the resulting graph are monochromatic. It would appear that the fewer edges we remove, the closer the approximation is to the original language, because fewer nonterminals are made to derive T^+ instead of what they originally derive in G. However it is possible that the we make more of an error when breaking a short cycle because T^+ may be far from what each of the eliminated nonterminals for this cycle derives, whereas eliminated edges on a long cycle may be each coming from nonterminals that derive languages much closer to T^+.

Continuing with Example 2, eliminating the r-colored edges $A_4 \to A_1, A_5 \to A_2$ and the l-colored edge $A_5 \to A_1$ are sufficient to make all cycles monochromatic in $CP(G)$. The resulting grammar is

$$A_1 \to A_2A_3 \mid b,\ A_2 \to A_3A_4,\ A_3 \to A_4A_5,\ A_4 \to A_5T^+,\ A_5 \to T^+T^+ \mid a\ .$$

This generates the language denoted by the regular expression in (7). The language in (7) is obtained by eliminating 3 edges of $CP(G)$ whereas the regular expressions in (8) and (9) were both obtained by eliminating 5 edges. Now consider the grammar G_n of Example 4.

Lemma 1. *Let $G = G_n$ be the grammar of Example 4. The regular language $\mathcal{L}(G'')$ obtained from G by eliminating l-colored edges in $CP(G)$ is given by*

$$\begin{aligned} (\varepsilon + T^*T^m)b \quad & \text{if } n = 2m, \\ (\varepsilon + T^*T^n)(b + T^*T^m a) \quad & \text{if } n = 2m+1\ . \end{aligned}$$

Proof. By repeated substitutions, we compute that the G'' is equivalent to the grammar

$$\begin{aligned} A_1 \to (T^+)^m A_1 \mid b \quad & \text{if } n = 2m, \\ A_1 \to (T^+)^n A_1 \mid (T^+)^m a \mid b \quad & \text{if } n = 2m+1\ , \end{aligned}$$

from which we obtain

$$\begin{aligned} ((T^+)^m)^*b \quad & \text{if } n = 2m, \\ ((T^+)^n)^*(b + (T^+)^m a) \quad & \text{if } n = 2m+1\ . \end{aligned}$$

From the identities $(T^+)^m = T^*T^m$ and $(T^*T^m)^* = \varepsilon + T^*T^m$, the regular expressions in (1) follow.

A similar result can be obtained for the left-linear grammars constructed by eliminating r-colored edges.

Remark: In the automaton M of the MN-transformation for the grammar G_n of Example 4 contains ε-transitions from A_n to A_1, and A'_n to A_1; ε-paths from A_1 to A_n, and from A'_1 to A_n. Therefore the automaton in Figure 3 (b) sits inside M, and the language accepted is $\mathcal{L}(G') = T^+$. We have $\mathcal{L}(G) \subseteq \mathcal{L}(G'') \subseteq \mathcal{L}(G')$

where $\mathcal{L}(G)$ is the context-free language generated by the grammar $G = G_n$ of Example 4, $\mathcal{L}(G')$ is the regular approximation from the MN-transformation and $\mathcal{L}(G'')$ is the regular approximation obtained by eliminating l-colored edges from $CP(G)$. Since $\mathcal{L}(G') = T^+$ independently of n whereas $\mathcal{L}(G'')$ is given as in Lemma 1, the difference between the former approximation and the latter can be made as large as we please.

7 Using a 1–Lookahead

In the MN-approximation, there is a certain memory in the rules carried by symbols such as A'_i which allow us to continue parsing from where we left off. We can remember more of the context of the branching by using a type of lookahead. This removes some of the ambiguity and therefore result in a smaller regular approximation, but it is at the cost of increasing the size of the new grammar. Mohri and Nederhof's grammar has size $O(|G|)$. The approximating grammar we obtain by eliminating all l-colored or all r-colored edges in $CP(G)$ in cycle-breaking is also $O(|G|)$. The lookahead considered here has $O(|G|^2)$ productions, as the approximating grammar construction in [6] (see also [4]). A k–lookahead approximation will cost $O(|G|^k)$ nonterminals, probably an unrealistically large bound of theoretical interest only.

For simplicity, in this section we consider the grammar to be in Greibach Normal Form (GNF). In GNF, all productions are of the form $A \to aA_1A_2 \ldots A_n$ or $A \to b$. In *1–lookahead*, we introduce new nonterminals for *pairs* of consecutive nonterminals that appear on the right hand side of a production with nonterminals. For a generic GNF production with a nonterminal right hand side, these would be $A_{12}, A_{34}, \ldots$. The idea is to preserve the memory of the production from where a branch occurred so that the derivation can continue if the next nonterminal is also present. This memory is at the odd indices only since we do not remember A_2A_3, for example. We will demonstrate the 1–lookahead idea with the help of an example.

Example 5. Let $T = \{a, b\}$ and start with the following grammar G:

$$\begin{aligned} A_1 &\to aA_2A_2 \mid a \\ A_2 &\to bA_2A_1 \mid b \end{aligned}$$

Straightforward MN-transformation results in the right-linear grammar

$$\begin{array}{ll} A'_1 \to \varepsilon & A'_2 \to \varepsilon \\ A_1 \to aA'_1 & A_2 \to bA'_2 \\ A_1 \to aA_2 & A_2 \to bA_2 \\ A'_2 \to A_2 & A'_2 \to A_1 \\ A'_2 \to A'_1 & A'_1 \to A'_2 \end{array} \tag{10}$$

In a leftmost derivation from A_1, after the first A_2 is processed, the end marker A'_2 allows for the derivation to continue with A_2, but also with A_1. In the approximation with 1–lookahead we remember that the current A'_2 should be followed by

processing A_2 and not A_1. This can be achieved by using the MN-transformation in the following way. Start with the first production above and introduce new nonterminals A_{22} and B_2 to indicate that the continuation is by A_2. We change the original production $A_1 \to aA_2A_2$ by using A_{22} for the first A_2 and using B_2 for the second A_2 as $A_1 \to aA_{22}B_2$, and then apply the MN-transformation. The first column in (10) is replaced by the rules

$$A_1' \to \varepsilon,\ A_1 \to aA_1',\ A_1 \to aA_{22},\ A_{22}' \to B_2,\ B_2' \to A_1' \ . \tag{11}$$

The second column of rules in (10) becomes

$$A_{22}' \to \varepsilon,\ A_{22} \to bA_{22}',\ A_{22} \to bA_{22},\ A_{22}' \to A_1,\ A_1' \to A_{22}' \ . \tag{12}$$

The new set of rules in (11) and (12) are the 1–lookahead transformation of the production $A_1 \to aA_2A_2$ of the original grammar.

For the transformation of the the second production $A_2 \to bA_2A_1$ we introduce two new nonterminals A_{21} and B_1 to indicate that the continuation is by A_1. We change $A_2 \to bA_2A_1$ by using A_{21} instead of A_2 and using B_1 instead of A_1, and write $A_2 \to bA_{21}B_1$. Applying the MN-transformation to this has the effect of replacing the the second column of (10) by the rules

$$A_2' \to \varepsilon,\ A_2 \to bA_2',\ A_2 \to bA_{21},\ A_{21}' \to B_1,\ B_1' \to A_2' \tag{13}$$

and replacing the first column of (10) by

$$B_1' \to \varepsilon,\ B_1 \to aB_1',\ B_1 \to A_{21},\ A_{21}' \to A_{21},\ A_{21}' \to B_1' \ . \tag{14}$$

The rules in (13) and (14) are the 1–lookahead transformation of the production $A_2 \to bA_2A_1$ of the original grammar.

Finally, we allow B_1 and B_2 derive the same sentential forms as A_1 and A_2 in the MN-approximation (10) by making copies of these rules using Bs for the corresponding As:

$$\begin{array}{ll} B_1' \to \varepsilon & B_2' \to \varepsilon \\ B_1 \to aB_1' & B_2 \to bB_2' \\ B_1 \to aB_2 & B_2 \to bB_2 \\ B_2' \to B_2 & B_2' \to B_1 \\ B_2' \to B_1' & B_1' \to B_2' \ . \end{array} \tag{15}$$

Let G' denote the MN-transformation of the given CFG G and denote by G'' the grammar obtained from G by the 1–lookahead transformation. Since any derivation in the 1–lookahead grammar can be simulated by a derivation in the original MN-transformation of G, we have

Theorem 3. *Let G' denote the MN-transformation of the CFG G and G'' the grammar obtained from G by the 1–lookahead transformation. Then $\mathcal{L}(G) \subseteq \mathcal{L}(G'') \subseteq \mathcal{L}(G')$.*

For the grammar G in Example 5, the MN-transformation G' is given by (10). In G', A_2 derives $b(a+b)^*$ and the MN-approximation itself is given by $\mathcal{L}(G') = a(a+b)^*$. The 1–lookahead transformation gives the grammar G'' with $\mathcal{L}(G'') = a + ab(a+b)^*$, which is properly contained in $\mathcal{L}(G')$.

8 Summary and Remarks

We considered the problem of approximation of a given context-free grammar by a regular grammar while trying to preserve the structure of the original grammar as much as possible. The algorithms considered are improvements on Mohri and Nederhof's original transformation and make use of the characterization of non–self-embedding grammars as generating regular languages.

In the approximations based on cycle-breaking, we start with a grammar in Chomsky normal form as input, and provide a regular grammar as output. The language generated is a superset of the given language, and a subset of the original Mohri and Nederhof approximation. We also consider a lookahead transformation which starts with the Greibach normal form and produces a regular grammar as its output. This approximation is also a superset of the given language, and a subset of the Mohri and Nederhof approximation.

References

1. Anselmo, M., Giammarresi, D., Varricchio, S.: Non–self–embedding grammars as representation for regular languages. In: CIAA Conference Proceedings (2002)
2. Chomsky, N.: A note on phrase structure grammars. Information and Control 4(2), 386–392 (1959)
3. Hopcroft, J.E., Ullman, J.D.: Introduction to Automata Theory, Languages, and Computation. Addison-Wesley Publishing Co., Reading (1979)
4. Mohri, M., Nederhof, M.-J.: Regular approximation of context–free grammars through transformation. In: Robustness in Language and Speech Technology, vol. 9, pp. 251–261. Kluwer Academic Publishers, Dordrecht (2000)
5. Mohri, M., Pereira, F.N.: Dynamic compilation of weighted context–free grammars. In: 36th Annual Meeting of the ACL and 17th International Conference on Computational Linguistics, vol. 2, pp. 891–897 (1998)
6. Nederhof, M.-J.: Regular approximation of cfls: A grammatical view. In: International Workshop on Parsing Technologies, pp. 159–170. MIT Press, Cambridge (1997)
7. Shallit, J.: Automaticity and rationality. J. of Automata, Languages and Combinatorics 5(3), 255–268 (2000)
8. Ullian, J.S.: Partial algorithm problems for context free languages. Information and Control 11, 80–101 (1967)