Context Congruence Closure and Context Rewriting for Commutative Context-Free Grammars

Siva Anantharaman

LIFO, Université d'Orléans
F-45067 Orléans Cedex 02, France
siva@lifo.univ-orleans.fr

1 Introduction

Standard (or usual) context-free grammars are well-known for programmers. The concept of a *commutative* grammar arises when analyzing concurrent processes via Petri net models. In particular, for processes without synchronization the Petri net model can be assumed "communication-free" and with simple pre-conditions (cf. [1]); the transition rules are then of the form $X \to {Y_1}^{m_1} \dots {Y_p}^{m_p}$ where the X, Y_j are places in the net and the m_j are positive integers; these transition rules define the productions of a context-free commutative grammar. Over such grammars, the uniform word problem - whether or not any given commutative word w is accepted - turns out to be NP-complete (cf. [1,2]). Indeed, in the commutative case, if $\alpha_1\alpha_2$ derives into β, we cannot always expect to cut β into a *left*-part β_1 and a *right*-part β_2 such that α_1 (resp. α_2) derives into β_1 (resp. β_2); so a CYK-like approach needs often to backtrack. We propose here to circumvent this partially, via a rewrite approach on commutative strings with context, mixing top-down reductions with bottom-up completion steps, all with a goal-oriented strategy.

As in the standard case, we first transform our context-free commutative grammar G_0 into a Chomsky normal form G with some additional restrictions. This will allow us to view any given (commutative) word w over the terminal alphabet of G as a word over a fixed set $\mathbf{T}(G)$ of non-terminals of G, in a unique manner. We derive from G a set $\mathcal{R}_1$ of commutative reduction rules of the form $Y[\mathcal{C}]\, Z[\mathcal{D}] \Rightarrow X[\mathcal{C}\mathcal{D}]$ where X, Y, Z are non-terminals in G, and $\mathcal{C}, \mathcal{D}$ are context-variables to be instantiated to sub-words of w for the $\mathcal{R}_1$-rewrite steps. Most of the rules of $\mathcal{R}_1$ are essentially reversed productions of G, but some more are needed to take into account recursion. We also derive from G a set of context-congruence pairs, which store information on the basic divergence points of derivations. Under a (well-founded) rewrite ordering on commutative strings defined over a total precedence relation on the non-terminals of G, and compatible with the system $\mathcal{R}_1$, the pairs of $\mathcal{R}_2$ can be turned into congruence rewrite rules. In our analysis of any given commutative word w, the rules of $\mathcal{R}_2$ compensate for the absence of confluence of the rewrite system $\mathcal{R}_1$; to help them do that well enough, we will in general need to "saturate" $\mathcal{R}_2$, relative to w.

Given the word w to analyze, we first perform $\mathcal{R}_1$-reductions on w; this will be part of the *top-down* component of our approach. If this is inconclusive, we proceed with inferences of congruence closure or saturation on $\mathcal{R}_2$ *à la KB*, so as to render further top-down inferences possible; such saturation steps constitute the *bottom-up* component of the approach; saturation can be "lazy". The approach in its entirety can be formulated in terms a non-deterministic automaton $\mathcal{A} = \mathcal{A}(G, w)$. Let w be seen as the commutative word $w = {X_1}^{n_1} \ldots {X_r}^{n_r}$ over $\mathbf{T}(G)$. The states of the automaton are (as a first approximation) commutative monomials over the non-terminals of G each labeled with a sub-word of w (in short: **labeled words** over w). The initial state of $\mathcal{A}$ will be the monomial $X_1[-]^{n_1} \ldots X_r[-]^{n_r}$, i.e., the word w itself whose non-terminals are labeled with the empty context; the acceptance state is the monomial $A[{X_1}^{n_1} \ldots {X_r}^{n_r}]$, where A is the axiom of G. The transitions of $\mathcal{A}$ are 'top-down' if they correspond to $\mathcal{R}_1$-reductions or to $\mathcal{R}_2$-rewrite steps; or 'bottom-up' if they are saturation steps on $\mathcal{R}_2$. In more formal terms, the states of our automaton will be couples formed by labeled words over w, and "current" sets of rules $\mathcal{R}_2'$ obtained via lazy saturation of $\mathcal{R}_2$. The saturation steps do part of what a *CYK*-like approach would have tried with backtrackings; but they will do that in a very limited manner and more intelligently, with a rewrite-based and goal-oriented strategy.

2 The Approach: Basic Definitions, Examples

Notational convention: since we will also be manipulating rewrite rules, we will employ the symbol "$\longrightarrow\!\!\!\!\!\!+\;$" for grammatical productions, reserve the symbols "$\Rightarrow$" and "$\rightarrow$" for rewrite rules, and the corresponding long symbols "$\Longrightarrow$, $\longrightarrow$" for the respective rewrite steps using these rules. Composite reductions via a "combination" of these two sets of rules will be denoted "$\succ\!\!\longrightarrow$". **cfcg** will be short for "context-free commutative grammar".

We look at a **cfcg** G essentially as a set of string rewrite rules on labeled words (the set of reversed productions of G, plus a few more), and a set of "congruence rules": (oriented) pairs of labeled words defining the basic bridges between different derivations. We do not want too many congruences, nor the contexts to be too big; and we want to express any (commutative) word over the terminal alphabet of G uniquely over a fixed set of non-terminals of G.

Definition 1 *A* **cfcg** $G = (\Sigma, N, A, \mathcal{P})$ *is said to be in* restricted Chomsky normal form *(RCN-*form *for short) iff the following conditions are satisfied:*

i) All productions in G are either standard, *i.e. of the form $X \longrightarrow\!\!\!\!\!\!+\; YZ$ for non-terminals X, Y, Z; or* unit, *i.e. of the form $X \longrightarrow\!\!\!\!\!\!+\!\!+\; V$, for non-terminals X, V; or* scalar, *i.e. of the form $V \longrightarrow\!\!\!\!\!\!+\; v$, with $v \in \Sigma$.* (We'll say in these last two cases: X produces a unit, resp. V produces a scalar.)

ii) if $V \longrightarrow\!\!\!\!\!\!+\; v$ is a scalar production in G with $v \in \Sigma$, then there is no **other** *scalar production in G with v on the right-hand side.*

iii) the unit productions of G appear only 'coupled' with recursive productions, that is to say under the form $X \twoheadrightarrow UX \mid V \mid \ldots$, where the non-terminal V must produce a scalar. []

The terminal words on such grammars are generated symbol by symbol: productions of the form $V \twoheadrightarrow ab$ are not allowed in our RCN-forms.

Example 1 If G_0 is given (in usual Chomsky normal form) as $A \twoheadrightarrow PA \mid QB \mid c$, $B \twoheadrightarrow AD \mid QB \mid c$, $D \twoheadrightarrow PB \mid c$, $P \twoheadrightarrow a$, $Q \twoheadrightarrow a \mid b$, then an equivalent RCN-form is the **cfcg** G: $P \twoheadrightarrow a$, $Q \twoheadrightarrow b$, $R \twoheadrightarrow c$, $A \twoheadrightarrow PA \mid QB \mid R$, $B \twoheadrightarrow AD \mid AR \mid QB \mid PB \mid R$, $D \twoheadrightarrow PB$ □

Contexts are commutative words over the non-terminals in G, inside square brackets. A *labeled non-terminal* is a non-terminal with an attached context; notation: $N[\mathcal{C}]$; we say that the context is *"empty"*, when it is the non-terminal itself; we write then more simply N, or $N[-]$. Monomials over the labeled non-terminals will be referred to as *labeled words*.

Definition 2 *Let G be a given* **cfcg** *in RCN-form.*

i) We associate with G a commutative string rewrite system, denoted $\mathcal{R}_1(G)$, whose elements are obtained as follows:

- if $X \twoheadrightarrow YZ$ (resp. $X \twoheadrightarrow Z$) is a standard (resp. unit) production in G, then $Y[\mathcal{C}]\, Z[\mathcal{D}] \Rightarrow X[\mathcal{C}\mathcal{D}]$ (resp. $Z[\mathcal{C}] \Rightarrow X[\mathcal{C}]$), where $\mathcal{C}, \mathcal{D}$ are context-variables, is a rewrite rule in $\mathcal{R}_1(G)$;
- if $Y \twoheadrightarrow YV$ is a recursive production in G, then for any non-terminals X, Z of G, context-variables $\mathcal{C}, \mathcal{D}$, the rewrite rules $X[\mathcal{C}\, Y]\, Z[YV] \Rightarrow X[\mathcal{C}\, Y^2 V]$, as well as $X[\mathcal{C}\, Y]\, V[\mathcal{D}] \Rightarrow X[\mathcal{C}\, Y \mathcal{D}]$ are elements of $\mathcal{R}_1(G)$;
- if $X \twoheadrightarrow UX \mid Z \mid \ldots$, are productions in G with $Z \in \mathbf{T}(G)$, i.e. produces a scalar, then for any context variable $\mathcal{C}$, $U[\mathcal{C}]\, Z \Rightarrow X[\mathcal{C}Z]$ is in $\mathcal{R}_1(G)$.

ii) Also associated with G is a set of context-congruence pairs, denoted $\mathcal{R}_2(G)$, whose elements are obtained as follows:

- if $X \twoheadrightarrow PQ$ and $Y \twoheadrightarrow PQ$ are two standard productions of G, then $X[PQ] \equiv Y[PQ]$ is an element of $\mathcal{R}_2(G)$;
- if $X \twoheadrightarrow PQ$ and $Y \twoheadrightarrow PR$ are two standard productions of G, then $X[PQ]\; R \equiv Y[PR]\; Q$ is an element of $\mathcal{R}_2(G)$;
- if $X \twoheadrightarrow Z$, $Y \twoheadrightarrow Z$ are unit productions in G with $Z \in \mathbf{T}(G)$, i.e. Z produces a scalar, then $X[Z] \equiv Y[Z]$ is in $\mathcal{R}_2(G)$;
- if $X \twoheadrightarrow UX \mid Z \mid ..$, and $Y \twoheadrightarrow VY \mid Z \mid ..$, are productions in G with Z producing a scalar, then $X[UZ] \equiv Y[UZ]$ is in $\mathcal{R}_2(G)$. []

We turn the congruence pairs into rewrite rules under a rewrite ordering $\prec$ (compatible with $\mathcal{R}_1$), via a suitable total precedence $<$ over the non-terminals of G, by setting: $w_1 \prec w_2$ iff **either** $|w_1| < |w_2|$, **or** $|w_1| = |w_2|$ and $w_1 \ll w_2$ as a multiset. A *labeled word* is *over* w if its contexts are sub-words of w. The set of labeled words over w gets ordered under $\prec$, by ignoring the contexts attached to the non-terminals in the words.

A labeled word is *w-relevant* if its *non-empty* contexts concatenated together give a sub-word of w up to commutativity. A congruence pair (rule) of $\mathcal{R}_2(G)$ is said to be *w-relevant* iff its left-hand side (right-hand side) is w-relevant.

Example 2 Let G be the following **cfcg** in RCN-form:

$$A \twoheadrightarrow PY \mid QB, \quad B \twoheadrightarrow XR \mid WR, \quad X \twoheadrightarrow PQ, \quad Y \twoheadrightarrow QR, \quad W \twoheadrightarrow WW$$

$$P \twoheadrightarrow a, \quad Q \twoheadrightarrow b, \quad R \twoheadrightarrow c, \quad W \twoheadrightarrow d$$

and suppose we are given three words to analyze: $w_1 = abc$, $w_2 = bd^3c$, $w_3 = cd^2$. Assume that our rewrite ordering $\prec$ is defined over the total precedence:

$$A < B < X < Y < P < Q < R < W$$

The word w_1 is transformed uniquely into PQR, w_2 into QW^3R, and w_3 into RW^2; the following congruence rules in $\mathcal{R}_2$:

$$X[PQ]\,R \longrightarrow Y[QR]\,P, \quad W[WW]\,R \longrightarrow B[WR]\,W, \quad Y[QR]\,W \longrightarrow B[WR]\,Q$$

will be relevant to their analysis.

For the analysis of $w_1 = PQR$: from the starting state PQR on $\mathcal{A}$, suppose we have gone along the following transitions via $\mathcal{R}_1$-reductions:

$$PQR \implies X[PQ]\,R \implies B[PQR]$$

The last state is not of acceptance. However, the first of the congruence rules mentioned above is right simplifiable by $\mathcal{R}_1$ into the rule $X[PQ]\,R \longrightarrow A[PQR]$, then left simplifiable into the rule $B[PQR] \longrightarrow A[PQR]$. (This is a *saturation* step, cf. below.) This last congruence rule defines an $\mathcal{R}_2$-reduction from the state reached above, to the acceptance state $A[PQR]$; so w_1 is accepted by G.

For the analysis of $w_2 = QW^3R$: consider first the situation where we have gone along the chain of $\mathcal{R}_1$-transitions

$$QW^3R \implies QW^2\,B[WR] \implies A[QWR]W[WW];$$

A rewrite rule in $\mathcal{R}_1$, defined via recursion on W, leads us subsequently to the state $A[QW^3R]$ of acceptance. Consider next the situation where we apply the following $\mathcal{R}_1$-transitions: $QW^3R \implies W^3\,Y[QR]$. Here an $\mathcal{R}_2$-rewrite step via the congruence rule $Y[QR]\,W \longrightarrow B[WR]\,Q$ can be applied, to lead us to the state $W^2Q\,B[WR]$; we are back to case considered first.

Finally, the word $w_3 = RW^2$: we have the following two possible chains of $\mathcal{R}_1$-transitions, both leading to the same state:

$$RW^2 \implies B[RW]W[-] \implies B[RW^2], \quad \text{or}$$
$$RW^2 \implies W[WW]\,R \implies B[RW^2]$$

From the last state reached no more transitions are possible under $\mathcal{R}_1$, or under $\mathcal{R}_2$ even after saturation; so the word w_3 is rejected. $\square$

If N is a non-terminal in w, then an N*-translate* is defined for a w-relevant congruence rule $\mathcal{W}_1 \longrightarrow \mathcal{W}_2$ iff : i) at least one of the two labeled words $N\mathcal{W}_i, i = 1, 2$, is *reducible under* $\mathcal{R}_1$, and ii) if $\mathcal{U}_i$ are $\mathcal{R}_1$-normal forms of $N\mathcal{W}_i, i = 1, 2$, then the congruence pair $\mathcal{U}_1 \equiv \mathcal{U}_2$ obtained after $\mathcal{R}_1$-normalization is also w-relevant. By definition, a *saturation*, or a *congruence closure* step on $\mathcal{R}_2(G)$ w.r.t. w, is the process of augmenting the set $\mathcal{R}_2(G)$ with such w*-relevant translates*, or via right-simplification by rules in $\mathcal{R}_1$, or via superposition with rules of $\mathcal{R}_1$; such a step is denoted by: $\mathcal{R}_2(G) \vdash \mathcal{R}_2'(G)$.

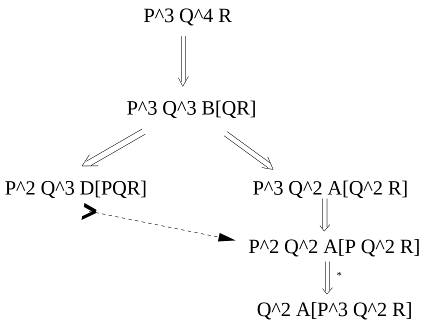
Let $\mathcal{C}$ be a labeled word, and $\mathcal{D}$ labeled word over w. Then $\mathcal{D}$ is a (w, G)*-instance* of $\mathcal{C}$ iff $\mathcal{D}$ can be reduced to $\mathcal{C}$ by using the productions of G in the reverse direction.

Example 3 *(Example* 1 - *contd.)* Let G be the following **cfcg**:

$$A \twoheadrightarrow PA \mid R \mid QB, \quad B \twoheadrightarrow AD \mid AR \mid QB \mid PB \mid R,$$

$$D \twoheadrightarrow PB, \quad P \twoheadrightarrow a, \quad Q \twoheadrightarrow b, \quad R \twoheadrightarrow c$$

Suppose $\prec$ is defined over the total precedence $A < B < D < P < Q < R$, and $w = a^3b^4c$ is the word to analyze; w is represented over $T(w)$ by P^3Q^4R, which admits more than one reduction path w.r.t. $\mathcal{R}_1$; for instance, we have:



Both branches seem to lead to dead-ends. Let us first look at the final state reached (via $\mathcal{R}_1$) along the right branch. We have by definition, a congruence rule $B[QR] \longrightarrow A[QR]$ in $\mathcal{R}_2(w)$; from which we derive, via P-translations, the rules $B[PQR] \longrightarrow A[PQR]$, $B[P^2QR] \longrightarrow A[P^2QR]$, .. etc. And then, via Q-translations, the rules $QA[PQR] \longrightarrow A[PQ^2R]$, $Q^2A[PQR] \longrightarrow A[PQ^3R]$, .. etc.; these w-relevant translations suffice to lead from the state mentioned, to the acceptance state $A[P^3Q^4R]$ (the word w is therefore accepted by G).

From the final state reached (via $\mathcal{R}_1$) along the left branch no such translations seem available. However, we have the following congruence rule in $\mathcal{R}_2(G)$: $D[PB]Q \longrightarrow A[QB]P$. It suffices then to see that PQR is a (w, G)-instance of PB in the sense defined above; so we derive the congruence rule $D[PQR]Q \longrightarrow A[Q^2R]P$, which defines a "composite" reduction step: $P^2Q^3D[PQR] \rightarrowtail P^2Q^2A[PQ^2R]$, from this 'final' state on the left, to a state on the right at a lower level, i.e. where a bigger sub-word of w is covered. □

Remark: Guessing the appropriate (w, G)-instances is easy if we "remember" the sub-words of w covered along the top-down inferences; e.g. the sub-word QR covered by B gave us the clue in the above example.

Theorem 1 *The word w of size m is accepted by the* **cfcg** *G if and only if* every *chain of length $p \leq m$ of top-down transitions $t_1 \dots t_p$ from the starting state on $\mathcal{A}(G, w)$, can be completed by a sequence of transitions $u_{p+1}t_{p+1} \dots u_m t_m$ towards an acceptance state, where each t_j is top-down, and u_j a (possibly empty) chain of bottom-up transitions.*

References

1. J. Esparza, "*Petri Nets, Commutative Context-Free Grammars, and Basic Parallel Processes*", Fundamenta Informaticae, Vol. 30, 1997, pp. 23-41.
2. D.T. Huynh, "*Commutative Grammars: The Complexity of the Uniform Word Problems*", Information and Control, Vol. 57, 1983, pp. 21-39.