Deciding the NTS Property of Context-Free Grammars

Joost Engelfriet *

Department of Computer Science, Leiden University
P.O.Box 9512, 2300 RA Leiden, The Netherlands

Abstract. An algorithm is presented that is a variation of the one of Senizergues in [4]. It decides the NonTerminal Separation property of context-free grammars in polynomial time. A straightforward generalization of the algorithm decides the NTS property of extended context-free grammars (but not in polynomial time).

It was shown in [4] that it is decidable whether an arbitrary context-free grammar is NTS. The algorithm in [4] takes exponential time in the worst case. It was recently shown in [3] that the NTS property can in fact be decided in polynomial time. This is not stated as such in [3], but immediately follows from the more general result, shown in Proposition 3.8 of [3], that it is decidable in polynomial time whether a monadic semi-Thue system is weakly confluent: the reversed productions of a (λ-free and chain-free) context-free grammar form a monadic semi-Thue system that is weakly confluent if and only if the context-free grammar is NTS.

Here we present an independent proof of the polynomial time decidability of the NTS property that is a variation of the decidability algorithm in [4]. We also show that the NTS property is decidable for extended context-free grammars, i.e., context-free grammars of which the productions have regular expressions as right-hand sides. We first recall some definitions and facts (see [1, 4]).

We consider context-free grammars $G = (X, V, P, Z)$ where X is the set of terminals, V the set of nonterminals, P the set of productions, and $Z \subseteq V$ the set of initial nonterminals. We assume that G is λ-free and chain-free (i.e., for every production $A \to \alpha$, $|\alpha| \geq 1$ and if $|\alpha| = 1$ then $\alpha \in X$). Such a grammar is NTS (*nonterminally separated*) if the following holds: for every $A, B \in V$ and $\alpha, \beta, \gamma \in (X \cup V)^*$, if $A \Rightarrow^* \alpha\beta\gamma$ and $B \to \beta$ is in P, then $A \Rightarrow^* \alpha B \gamma$. Note that this does not depend on Z.

Let $G = (X, V, P, Z)$ be a context-free grammar. A pair $\langle \alpha, \beta \rangle$ with $\alpha, \beta \in (X \cup V)^*$ is called a shift-reduce configuration. For two shift-reduce configurations we define the *shift-reduce step relation* $\vdash$ as follows (where $\alpha, \beta, \gamma \in (X \cup V)^*$, $b \in X$, and $A \in V$):

1. $\langle \alpha, b\beta \rangle \vdash \langle \alpha b, \beta \rangle$
   if no suffix of α is the right-hand side of a production in P, and

* Supported by ESPRIT BR Working Group No.6317 ASMICS

2. $\langle \alpha\gamma, \beta\rangle \vdash \langle \alpha A, \beta\rangle$
   if $A \to \gamma$ is in P, A is the "first" nonterminal that has a production with right-hand side γ (where we assume some fixed, but arbitrary order on V), and γ is the shortest suffix of $\alpha\gamma$ that is the right-hand side of a production in P.

A step of type 1 is a shift, and a step of type 2 is a reduction. We note that, in a step of type 2, it is irrelevant that γ is taken to be the shortest suffix of $\alpha\gamma$ that is the right-hand side of a production; we might as well take the longest suffix (or any other choice that makes $\vdash$ deterministic).

For $A \in V$, we define $\mathrm{LM}(G, A) = \{\alpha \in (X \cup V)^* \mid \langle \lambda, \alpha\rangle \vdash^* \langle A, \lambda\rangle\}$. Intuitively, $\mathrm{LM}(G, A)$ is the set of strings that are left-most reducible to A, where a reduction takes place as soon as the right-hand side of a production is detected. Obviously, if $\alpha \in \mathrm{LM}(G, A)$ then $A \Rightarrow^* \alpha$, but in general this is not true in the other direction. If G is an NTS grammar then $\alpha \in \mathrm{LM}(G, A)$ if and only if $A \Rightarrow^* \alpha$ (also because G is λ-free and chain-free). It should be clear from the definition of $\vdash$ that $\mathrm{LM}(G, A)$ is a deterministic context-free language over $X \cup V$.

Let $A \in V$ and $\alpha, \beta, \gamma \in (X \cup V)^*$. Let t be a derivation tree with root label A and $\mathrm{yield}(t) = \alpha\beta\gamma$, and let M be the set of leaves of t that correspond to the indicated occurrence of β in $\mathrm{yield}(t)$ (the "marked" leaves). Then t is said to be *nearly essential* for the derivation $A \Rightarrow^* \alpha\beta\gamma$ if

1. the root is the only internal node of t that is an ancestor of all leaves in M, and
2. every internal node is ancestor of some leaf in M.

In [4] the test for the NTS property is based on the set of "essential" derivation trees, which is a finite subset of the (generally infinite) set of nearly essential derivation trees. For each such essential tree a right- (or left-) linear grammar is constructed. Since, for a given grammar G, there may exist exponentially many essential trees (in the size of the grammar), the algorithm in [4] takes exponential time. Here we will show that it suffices to construct only polynomially many right- (or left-) linear grammars. For the readers familiar with [4] we give the following exponential example.

Consider, for $n \in \mathbf{N}$, the context-free grammar G with productions $A \to aAc$, $A \to bAc$, $A \to d$ and $A \to c^n$. Then all the 2^n derivation trees for derivations $A \Rightarrow^* \alpha\beta\gamma$ with $\alpha \in \{a, b\}^n d$, $\beta = c^n$, and $\gamma = \lambda$, are essential. Note that this grammar is not NTS; adding the constant amount of productions $A \to aA \mid bA \mid cA \mid dA \mid AA \mid a \mid b \mid c$ makes it NTS.

We now turn to our polynomial time variation of the algorithm of [4]. It is based on the following two facts; the first fact is part of the proofs of Propositions 1 and 2 of [4].

Fact 1. A context-free grammar $G = (X, V, P, Z)$ is NTS iff the following holds, for all $A, B \in V$ and $\alpha, \beta, \gamma \in (X \cup V)^*$:
if $A \Rightarrow^* \alpha\beta\gamma$, $B \to \beta$ is in P, and there is a nearly essential derivation tree for $A \Rightarrow^* \alpha\beta\gamma$, then $\alpha B\gamma \in \mathrm{LM}(G, A)$.

Fact 2. Let $G = (X, V, P, Z)$ be a context-free grammar. For every $A \in V$ and every production $p = (B \to \beta)$, the language $E_{A,p} = \{\alpha B\gamma \mid A \Rightarrow^* \alpha\beta\gamma$ and there is a nearly essential derivation tree for $A \Rightarrow^* \alpha\beta\gamma\}$ is regular. Moreover, a right-linear grammar for $E_{A,p}$ can be constructed in polynomial time.

Before proving these facts we show that they imply the polynomial time decidability of the NTS property. It follows from Fact 1 that a grammar G is NTS iff $E_{A,p} \subseteq \mathrm{LM}(G, A)$ for every $A \in V$ and $p \in P$. Consider a fixed A and p. It suffices to show that $E_{A,p} \cap \mathrm{LM}(G, A)^{\mathrm{c}} = \emptyset$ can be decided in polynomial time. Since $E_{A,p}$ is regular by Fact 2, and $\mathrm{LM}(G, A)$ is a deterministic context-free language, the property can in fact be decided in polynomial time, provided grammars (or automata) for both $E_{A,p}$ and $\mathrm{LM}(G, A)^{\mathrm{c}}$ can be constructed in polynomial time. For $E_{A,p}$ this is shown in Fact 2. As observed in [1], a deterministic pushdown automaton D can be constructed that directly simulates the shift-reduce algorithm recognizing $\mathrm{LM}(G, A)$. To decide between shifting or reducing, D should keep the top-most part of the pushdown α (the first element of the shift-reduce configuration) in its finite state. Clearly, it suffices that the state contains the longest suffix of α that is a prefix of a right-hand side of a production of G. Hence D needs polynomially many states only. From this it easily follows that D can be constructed in polynomial time. Since G is λ-free and chain-free, D always reads all of its input. Hence, an automaton recognizing $\mathrm{LM}(G, A)^{\mathrm{c}}$ is obtained by interchanging the final and nonfinal states of D. This shows the polynomial time decidability of the NTS property. We now prove Facts 1 and 2.

Proof of Fact 1. (Only if) Let G be NTS. If $A \Rightarrow^* \alpha\beta\gamma$ and $B \to \beta$ is in P, then (since G is NTS) $A \Rightarrow^* \alpha B\gamma$ and hence (since G is NTS) $\alpha B\gamma \in \mathrm{LM}(G, A)$.
(If) Let the stated condition be true. To show that G is NTS, assume that $A \Rightarrow^* \alpha\beta\gamma$ and $B \to \beta \in P$. Let t be a derivation tree for $A \Rightarrow^* \alpha\beta\gamma$ and let M be the set of leaves of t that correspond to the occurrence of β in $\mathrm{yield}(t)$. Let x be the least internal node of t that is a common ancestor of all leaves in M. Considering the subtree t' of t rooted at x we obtain derivations $A \Rightarrow^* \alpha_1 A'\gamma_1$ and $A' \Rightarrow^* \alpha_2\beta\gamma_2$ where x has label A' , $\alpha = \alpha_1\alpha_2$, and $\gamma = \gamma_2\gamma_1$. Now consider the nearly essential derivation tree t'' obtained from t' by pruning all subtrees that are rooted at internal nodes of t' that are not ancestor of any leaf in M. Clearly, t'' is nearly essential for a derivation $A' \Rightarrow^* \alpha_2'\beta\gamma_2'$ with $\alpha_2' \Rightarrow^* \alpha_2$ and $\gamma_2' \Rightarrow^* \gamma_2$. The stated condition now implies that $\alpha_2' B\gamma_2' \in \mathrm{LM}(G, A')$ and so $A' \Rightarrow^* \alpha_2' B\gamma_2'$. Hence $A \Rightarrow^* \alpha_1 A'\gamma_1 \Rightarrow^* \alpha_1\alpha_2' B\gamma_2'\gamma_1 \Rightarrow^* \alpha_1\alpha_2 B\gamma_2\gamma_1 = \alpha B\gamma$. $\square$

Proof of Fact 2. Let $G = (X, V, P, Z)$, $A \in V$, and $p = (B \to \beta) \in P$. In what follows we will construct a (λ-free, but not necessarily chain-free) context-free grammar $G' = (X \cup V, V', P', Z')$ for the language $E_{A,p}$ which is itself not right- or left-linear but from which such a grammar can easily be constructed. In fact, $V' = \{S\} \cup V_L \cup V_R$ with $Z' = \{S\}$, $V_L \cap V_R = \emptyset$, $S \notin V_L \cup V_R$, and the productions of P' are also partitioned in three parts: productions with left-hand side S, right-linear productions that contain nonterminals from V_L only, and left-linear productions that contain nonterminals from V_R only. Clearly, any grammar of this form can be turned into an equivalent right-linear grammar (or finite automaton) in polynomial time.

Intuitively, a production of P' with left-hand side S simulates the production of P with left-hand side A that is applied at the root of a nearly essential derivation tree for some $A \Rightarrow^* \alpha\beta\gamma$. Such a production generates the B that replaces β, and it generates a prefix of α and a suffix of γ. The right-linear productions with nonterminals from V_L then generate the remainder of α (from left to right), and the left-linear productions with nonterminals in V_R generate the remainder of γ (from right to left). The generation of β is simulated in the nonterminals. To this aim every nonterminal from V_L contains a prefix of β, which is the part of β of which the generation still has to be simulated by this nonterminal. Similarly every nonterminal from V_R contains a suffix of β.

Thus, V_L consists of all $\langle Y, L, \phi\rangle$ where $Y \in V$, L stands for "Left", and ϕ is a non-empty prefix of β. Symmetrically, V_R consists of all $\langle \psi, R, Y\rangle$ where ψ is a non-empty suffix of β, R stands for "Right", and $Y \in V$. The productions of P' are defined as follows (where $\Rightarrow$ always refers to G).

(1) Productions with left-hand side S. For every production $A \to Y_1 \cdots Y_k$ in P, with $Y_i \in X \cup V$ for $1 \leq i \leq k$, P' contains the following productions.
(1.1) All productions $S \to Y_1 \cdots Y_{i-1}\langle Y_i, L, \phi\rangle B\langle \psi, R, Y_j\rangle Y_{j+1} \cdots Y_k$ where $1 \leq i < j \leq k$, $Y_i, Y_j \in V$, and there exists $\pi \in (X \cup V)^*$ such that $\beta = \phi\pi\psi$ and $Y_{i+1} \cdots Y_{j-1} \Rightarrow^* \pi$. Note that in the case that $i+1 = j$ the last condition means that $\beta = \phi\psi$.
(1.2) All productions $S \to Y_1 \cdots Y_{i-1} B\langle \psi, R, Y_j\rangle Y_{j+1} \cdots Y_k$ where $1 \leq i < j \leq k$, $Y_j \in V$, and there exists $\pi \in (X \cup V)^*$ such that $\beta = \pi\psi$ and $Y_i \cdots Y_{j-1} \Rightarrow^* \pi$.
(1.3) All productions $S \to Y_1 \cdots Y_{i-1}\langle Y_i, L, \phi\rangle B Y_{j+1} \cdots Y_k$ where $1 \leq i < j \leq k$, $Y_i \in V$, and there exists $\pi \in (X \cup V)^*$ such that $\beta = \phi\pi$ and $Y_{i+1} \cdots Y_j \Rightarrow^* \pi$.
(1.4) All productions $S \to Y_1 \cdots Y_{i-1} B Y_{j+1} \cdots Y_k$ where $1 \leq i < j \leq k$ and $Y_i \cdots Y_j \Rightarrow^* \beta$.

To explain the intuition behind the above productions, consider a nearly essential derivation tree for $A \Rightarrow^* \alpha\beta\gamma$ with production $A \to \delta$ applied at the root, and let $\delta = Y_1 \cdots Y_k$. Let M be the set of leaves corresponding to β. Then $Y_1, \ldots, Y_{i-1}$ are the symbols of δ that label leaves to the left of M, and $Y_{j+1}, \ldots, Y_k$ those that label leaves to the right of M. Furthermore, $\langle Y_i, L, \phi\rangle$ occurs in the production of P' if and only if Y_i is a nonterminal that generates both leaves in M and leaves not in M, and ϕ is the sequence of labels of the generated leaves in M. And a similar statement holds for $\langle \psi, R, Y_j\rangle$. This same intuition also explains the remaining productions.

(2) Productions with left-hand side in V_L. For every production $Y \to Y_1 \cdots Y_k$ in P, with $Y_i \in X \cup V$ for $1 \leq i \leq k$, P' contains the following productions.
(2.1) All productions $\langle Y, L, \phi\rangle \to Y_1 \cdots Y_{i-1}\langle Y_i, L, \phi_1\rangle$ where $1 \leq i \leq k$, $Y_i \in V$, and there exists $\phi_2 \in (X \cup V)^*$ such that $\phi = \phi_1\phi_2$ and $Y_{i+1} \cdots Y_k \Rightarrow^* \phi_2$. In the case that $i = k$ the last condition means that $\phi = \phi_1$.
(2.2) All productions $\langle Y, L, \phi\rangle \to Y_1 \cdots Y_{i-1}$ where $2 \leq i \leq k$ and $Y_i \cdots Y_k \Rightarrow^* \phi$.

(3) Productions with left-hand side in V_R. For every production $Y \to Y_1 \cdots Y_k$ in P, P' contains the following productions.

(3.1) All productions $\langle \psi, R, Y\rangle \rightarrow \langle \psi_1, R, Y_j\rangle Y_{j+1} \cdots Y_k$ where $1 \leq j \leq k$, $Y_j \in V$, and there exists $\psi_2 \in (X \cup V)^*$ such that $\psi = \psi_2\psi_1$ and $Y_1 \cdots Y_{j-1} \Rightarrow^* \psi_2$. In the case that $j = 1$ the last condition means that $\psi = \psi_1$.
(3.2) All productions $\langle \psi, R, Y\rangle \rightarrow Y_{j+1} \cdots Y_k$ where $1 \leq j \leq k-1$ and $Y_1 \cdots Y_j \Rightarrow^* \psi$.

This concludes the construction of the grammar G' generating $E_{A,p}$. It remains to show that G' can be constructed in polynomial time from G. Let n be the size of G. Since the number of prefixes and suffixes of β is $\mathrm{O}(n)$, G' has $\mathrm{O}(n^2)$ nonterminals. Now consider the productions $S \rightarrow Y_1 \cdots Y_{i-1}\langle Y_i, L, \phi\rangle B\langle \psi, R, Y_j\rangle Y_{j+1} \cdots Y_k$ of P', corresponding to the production $A \rightarrow Y_1 \cdots Y_k$ of P, as defined in (1.1) above. There are $\mathrm{O}(n^4)$ such productions in P', one for each choice of i, j, ϕ, and ψ. Note that the condition $Y_{i+1} \cdots Y_{j-1} \Rightarrow^* \pi$ can be verified in polynomial time. Similar statements hold for the productions of all other types. From these remarks it should be clear that G' can be constructed in polynomial time. $\square$

We now turn to the decidability of the NTS property for extended context-free grammars. An extended context-free grammar (or extended BNF) has productions of which the right-hand sides are regular expressions over $X \cup V$. An alternative way of viewing this is as follows. An *extended context-free grammar* is a context-free grammar $G = (X, V, P, Z)$ such that P is infinite and for each nonterminal B the language $R_B = \{\beta \in (X \cup V)^* \mid B \rightarrow \beta \in P\}$ is regular. The regular languages R_B should be given effectively as regular expressions, finite automata, or right-linear grammars. In what follows we will assume that a deterministic finite automaton A is given, with state set Q, initial state q_0, and state transition function $\delta : Q \times (X \cup V) \rightarrow Q$, and that for each nonterminal B a set $F_B \subseteq Q$ of final states is given, such that, with this set of final states, A recognizes the language R_B. All the usual definitions for context-free grammars also apply to extended context-free grammars, including the definition of NTS.

The algorithm that decides the NTS property for extended context-free grammars is a variation of the one above. First of all, it should be clear that Fact 1 is still true in the extended case (with the same proof). Instead of Fact 2 we will show the following, closely related, fact.

Fact 3. Let $G = (X, V, P, Z)$ be an extended context-free grammar. For all $A, B \in V$, the language $E_{A,B} = \{\alpha B\gamma \mid A \Rightarrow^* \alpha\beta\gamma$ and there is a nearly essential derivation tree for $A \Rightarrow^* \alpha\beta\gamma$, for some $\beta \in R_B\}$ is regular. Moreover, a right-linear grammar for $E_{A,B}$ can be obtained effectively.

First we show, as before, that Facts 1 and 3 imply the decidability of the NTS property. By Fact 1, a grammar G is NTS iff $E_{A,B} \cap \mathrm{LM}(G, A)^{\mathrm{c}} = \emptyset$ for all $A, B \in V$. Since $E_{A,B}$ is regular by Fact 3, and since the deterministic context-free languages are (effectively) closed under complement and intersection with a regular language, it suffices to show that $\mathrm{LM}(G, A)$ can be recognized by a deterministic pushdown automaton D. As before, D simulates the shift-reduce algorithm, with the first element α of the shift-reduce configuration on its pushdown. To decide between shifting or reducing, D keeps in its finite state the set S of all states $q \in Q$ (of the finite automaton A) such that $\delta(q_0, \gamma) = q$ for some suffix γ of α. When S contains a final state from some F_B, D makes a reduction, as follows. It pops symbols off its pushdown, simultaneously simulating automaton A backwards, starting with each of the final states in S (possibly for different B). To keep D deterministic, the backward simulation of A is with the usual subset construction. Thus, for each of the nonterminals B for which there is a final state in S, D keeps track of a set S_B of states of A. As soon as q_0 turns up in one (or more) of the S_B, D stops popping because it knows that it just has popped the shortest suffix that is the right-hand side of a production. D then pushes the "first" B such that S_B contains q_0. Note that in order to keep track of the set S, D should in fact store S on its pushdown, each time it pushes a symbol. It should be clear that D can be obtained effectively from G.

This shows the decidability of the NTS property. It remains to prove Fact 3.

Proof of Fact 3. A grammar $G' = (X \cup V, V', P', Z')$ for the language $E_{A,B}$ can be defined in much the same way as in the proof of Fact 2. This time V_L consists of all $\langle Y, L, p\rangle$, and V_R of all $\langle q, R, Y\rangle$, with $p, q \in Q$. Intuitively, $\delta(q_0, \phi) = p$ for some prefix ϕ of some $\beta \in R_B$, and $\delta(q, \psi) \in F_B$ for some suffix ψ of β.

Moreover, G' is an extended context-free grammar. Since the regular languages are (effectively) closed under substitution, a right-linear grammar for $E_{A,B}$ can be constructed from G', due to the form of the productions (see the proof of Fact 2).

The productions of G' are very similar to those given in the proof of Fact 2. They are defined as follows.

(1) Productions with left-hand side S. For every production $A \to Y_1 \cdots Y_k$ in P, with $Y_i \in X \cup V$ for $1 \le i \le k$, P' contains the following productions.

(1.1) All productions $S \to Y_1 \cdots Y_{i-1}\langle Y_i, L, p\rangle B\langle q, R, Y_j\rangle Y_{j+1} \cdots Y_k$ where $1 \le i < j \le k$, $Y_i, Y_j \in V$, and there exists $\pi \in (X \cup V)^*$ such that $\delta(p, \pi) = q$ and $Y_{i+1} \cdots Y_{j-1} \Rightarrow^* \pi$. In the case that $i + 1 = j$ this condition means that $p = q$.

(1.2) All productions $S \to Y_1 \cdots Y_{i-1} B\langle q, R, Y_j\rangle Y_{j+1} \cdots Y_k$ where $1 \le i < j \le k$, $Y_j \in V$, and there exists $\pi \in (X \cup V)^*$ such that $\delta(q_0, \pi) = q$ and $Y_i \cdots Y_{j-1} \Rightarrow^* \pi$.

(1.3) All productions $S \to Y_1 \cdots Y_{i-1}\langle Y_i, L, p\rangle B Y_{j+1} \cdots Y_k$ where $1 \le i < j \le k$, $Y_i \in V$, and there exists $\pi \in (X \cup V)^*$ such that $\delta(p, \pi) \in F_B$ and $Y_{i+1} \cdots Y_j \Rightarrow^* \pi$.

(1.4) All productions $S \to Y_1 \cdots Y_{i-1} B Y_{j+1} \cdots Y_k$ where $1 \le i < j \le k$, and there exists $\beta \in R_B$ such that $Y_i \cdots Y_j \Rightarrow^* \beta$.

(2) Productions with left-hand side in V_L. For every production $Y \to Y_1 \cdots Y_k$ in P, with $Y_i \in X \cup V$ for $1 \le i \le k$, P' contains the following productions.

(2.1) All productions $\langle Y, L, p\rangle \to Y_1 \cdots Y_{i-1}\langle Y_i, L, p_1\rangle$ where $1 \le i \le k$, $Y_i \in V$, and there exists $\phi \in (X \cup V)^*$ such that $\delta(p_1, \phi) = p$ and $Y_{i+1} \cdots Y_k \Rightarrow^* \phi$. In the case that $i = k$ the last condition means that $p = p_1$.

(2.2) All productions $\langle Y, L, p\rangle \to Y_1 \cdots Y_{i-1}$ where $2 \le i \le k$, and there exists $\phi \in (X \cup V)^*$ such that $\delta(q_0, \phi) = p$ and $Y_i \cdots Y_k \Rightarrow^* \phi$.

(3) Productions with left-hand side in V_R. For every production $Y \to Y_1 \cdots Y_k$ in P, P' contains the following productions.

(3.1) All productions $\langle q, R, Y\rangle \to \langle q_1, R, Y_j\rangle Y_{j+1} \cdots Y_k$ where $1 \le j \le k$, $Y_j \in V$, and there exists $\psi \in (X \cup V)^*$ such that $\delta(q, \psi) = q_1$ and $Y_1 \cdots Y_{j-1} \Rightarrow^* \psi$. In the case that $j = 1$ the last condition means that $q = q_1$.
(3.2) All productions $\langle q, R, Y\rangle \to Y_{j+1} \cdots Y_k$ where $1 \le j \le k-1$, and there exists $\psi \in (X \cup V)^*$ such that $\delta(q, \psi) \in F_B$ and $Y_1 \cdots Y_j \Rightarrow^* \psi$.

This concludes the description of G'. It remains to show that G' is indeed an extended context-free grammar, and that it can be obtained effectively from G. For this we have to show that the languages $R'_C = \{\beta \in (X \cup V)^* \mid C \to \beta \in P'\}$ are regular (for each $C \in V'$), and can be obtained effectively from the regular languages R_C (for $C \in V$), i.e., from the automaton $\mathcal{A}$. This is based on the fact that for a given (extended) context-free grammar G and a regular language R, the language $\{\beta \in (X \cup V)^* \mid \beta \Rightarrow^*_G \pi \text{ for some } \pi \in R\}$ is regular, and can be obtained effectively from G and R (for an easy proof see Proposition 2.1 of [2]). From this fact, and the definition of P', it can easily be seen that a finite state transducer (even gsm mapping) τ can be constructed such that $R'_S = \tau(R_A)$. Similarly, $R'_{\langle Y,L,p\rangle}$ and $R'_{\langle q,R,Y\rangle}$ are (effectively) images of R_Y under appropriate finite state transductions. Since the class of regular languages is effectively closed under finite state transductions, this shows that G' is an extended context-free grammar that can be constructed from G. $\square$

References

1. L.Boasson, G.Senizergues; NTS languages are deterministic and congruential, J. of Comp. and Syst. Sci. 31 (1985), 332-342
2. R.V.Book; Decidable sentences of Church-Rosser congruences, Theor. Comput. Sci. 23 (1983), 301-312
3. K.Madlener, P.Narendran, F.Otto, L.Zhang; On weakly confluent monadic string-rewriting systems, Theor. Comput. Sci. 113 (1993), 119-165
4. G.Senizergues; The equivalence and inclusion problems for NTS languages, J. of Comp. and Syst. Sci. 31 (1985), 303-331