THE BEHAVIOUR OF PARSING TIME UNDER GRAMMAR MORPHISMS (*)

by Hermann K.-G. WALTER (1) Joannis KEKLIKOGLOU (1)
and Werner KERN (1)

Communicated by J. BERTEL

Abstract. — We show that expanding transformations applied to context-free grammars preserve parsing time (and space) in order of magnitude.

0. INTRODUCTION

Many problems related to grammars, languages, syntax analysis, etc. are solved with the help of certain transformations of grammars (for example: normal forms).

A great part of these transformations can be interpreted in such a way, that they give rise to grammar morphisms with certain properties, especially the property of preserving the generated language (Hotz [7, 8], Benson [2]).

With respect to context-free grammars grammar morphisms are one-state-tree-transductions.

The aim of this paper is to discuss in which way the parsing time is carried over if a grammar morphism is applied.

E. Bertsch [3] has shown, that parsing time is preserved applying strictly length-preserving morphisms to context-free grammars. We generalize this result to a class of grammar morphisms which is much more greater.

As a consequence we’ll get the result that related context-free grammars (in the sense of Hotz [7, 8]) have (asymptotically) the same parsing time.

1. GRAMMAR MORPHISMS

We use syntactical categories (X-categories) as a framework for our definitional apparatus (G. Hotz [6], D. Benson [1]). If $G = (\Sigma, I, P, \sigma)$ is a grammar

(*) Reçu décembre 1977.

(1) Institut für Theoretische Informatik, Fachbereich Informatik, Technische Hochschule Darmstadt, D-6100 Darmstadt
with terminal alphabet $\Sigma$, intermediate alphabet $I$, productionsystem $P$ and startsymbol $\sigma$, we denote by $S(G)$ the associated syntactical category. A rough description of $S(G)$ is the following:

Objects of $S(G)$ are words over $\Sigma \cup I$, morphisms are the classes of inessentially different derivations. For convenience, we write $f \in S(G)$ to denote that $f$ is a derivation (class). If $f \in S(G)$, the functions $d$ (domain) and $c$ (codomain) assign to $f$ the word $w (= d(f))$ to which $f$ is applied and the word $w' (= c(f))$ which results by applying $f$.

Each $f \in S(G)$ has a definite length $\| f \|$. The derivations $f$ with $\| f \| = 0$ are the identities of $S(G)$, which we identify with the corresponding objects.

$S(G)$ is structured by two operations "•" and "×", where "•" denotes the concatenation and "×" the parallel composition of derivations. It is well-known that in the context-free case classes of derivations can be identified with so called derivation trees.

The most interesting set of derivation is

$$D(G) = \{ f \in S(G) | d(f) = \sigma \text{ and } c(f) \in \Sigma^* \};$$

then the generated language is given by

$$\Omega(G) = c(D(G)).$$

All details about syntactical categories can be found in Hotz [6], D. Benson [1].

In this paper we only consider context-free grammars, though this restriction is not necessary in any case.

**DEFINITION 1.1:** Consider two grammars

$$G_1 = (\Sigma_1, I_1, P_1, \sigma_1), \quad G_2 = (\Sigma_2, I_2, P_2, \sigma_2).$$

A (grammar) morphism $\varphi$ from $G_1$ to $G_2$ ($\varphi : G_1 \to G_2$) is a pair $\varphi = (\varphi_A, \varphi_P)$, where:

$$\varphi_A : (\Sigma_1 \cup I_1)^* \to (\Sigma_2 \cup I_2)^*$$

is a monoidhomomorphism and $\varphi_P : P_1 \to S(G_2)$ is a mapping, such that the following conditions hold:

1. For all $r (= p \to q) \in P_1$:

   $$\varphi_A(p) = d(\varphi_P(r)) \quad \text{and} \quad \varphi_A(q) = c(\varphi_P(r)),$$

2. $\varphi_A(\sigma_1) = \sigma_2$,

R.A.I.R.O. Informatique théorique/Theoretical Computer Science
\( \varphi_A(I_1) \subseteq I_2 \),
\( \varphi_A(\Sigma_1) \subseteq \Sigma_2^* \).

**Remark:** Since it is not necessary to distinguish \( \varphi_A \) and \( \varphi_P \) by subscripts we shall omit these subscripts from now on. It can be shown, that we can extend \( \varphi \) to a syntactical functor \( \varphi : S(G_1) \to S(G_2) \) in a unique way. Using this extension we get \( \varphi(D(G_1)) \subseteq D(G_2) \) and therefore \( \varphi(\Upsilon(G_1)) \subseteq \Upsilon(G_2) \).

One can single out various classes of morphisms. An overview of all these classes is given in Walter [10]. We repeat those, which are necessary to derive our results. Again, some of our results are true for more general classes of morphisms.

Consider \( \varphi : G_1 \to G_2 \). \( \varphi \) is called *internal* if \( \Sigma_1 = \Sigma_2 = \Sigma \) and \( \varphi(t) \equiv t \) for \( t \in \Sigma \). \( \varphi \) is called *closed* if \( \varphi(D(G_1)) = D(G_2) \). A closed internal morphism is called a *transformation*. If \( \varphi \) is a transformation, then \( \Upsilon(G_1) = \Upsilon(G_2) \) holds, i. e. the language is preserved. A morphism \( \varphi \) is *expanding* if \( \| \varphi(r) \| \geq 1 \) for all \( r \in P_1 \); if \( \| \varphi(r) \| = 1 \) for all \( r \in P_1 \), we call \( \varphi \) a *fine* morphism. A fine transformation is called a *reduction*. Reynolds covers (Benson [2], Gray-Harrison [5]) are reductions. Furthermore we get reductions by embedding the theory of grammars as the generalisation of reductions of finite automata (G. Hotz [8]). A second class of transformations is given by well-known normal-form theorems like the binary form of a context-free grammar. Roughly, such normal-form-theorems include constructions in which any production is simulated by a certain derivation of the normal-form.

We want to formalize this property.

If \( G = (\Sigma, I, P, \sigma) \) is a grammar, then \( G' = (\Sigma', I', P', \sigma') \) is a subgrammar of \( G (G' \subseteq G) \) if \( \Sigma' \subseteq \Sigma, I' \subseteq I, P' \subseteq P, \sigma' = \sigma \).

Set-theoretic operations transfer to subgrammars in a natural way.

Let \( G \) be a grammar and \( M \subseteq S(G) \). We denote by \( \langle M \rangle \) the smallest subgrammar of \( G \) with \( M \subseteq S(\langle M \rangle) \). If \( M = \{ f \} \) we write \( \langle M \rangle = \langle f \rangle \).

Consider an expanding transformation \( \varphi : G_1 \to G_2 \). We call \( \varphi \) a *simulation* if \( \varphi \) operates identically on \( I_1 \) and bijective between \( D(G_1) \) and \( D(G_2) \) and if the following holds:

(i) \( \langle \varphi(r) \rangle \cap \langle \varphi(r') \rangle \subseteq (\Sigma, I_1, \emptyset, \sigma_1) \) for all \( r, r' \in P_1 \) with \( r \neq r' \);

(ii) for any \( r \in P_1 \) there exists exactly one

\[ r_a \in P(\langle \varphi(r) \rangle) \quad \text{with} \quad d(r_a) \in I_1. \]

We want to show, that we can restrict ourselves to simulations and reductions if we are discussing transformations.

vol. 12, n° 2, 1978
THEOREM 1: If $\varphi : G_1 \to G_2$ is an expanding transformation, then there is a factorisation $\varphi = \varphi_2 \circ \varphi_1$ such that $\varphi_1$ is a simulation and $\varphi_2$ is a reduction.

Proof: Part 1: "Construction of $G_3$, $\varphi_1 : G_1 \to G_3$ and $\varphi_2 : G_3 \to G_2$".

Consider $r \in P_1$ and a so called sequential representation of $\varphi (r)$ (G. Hotz [6]):

$$\varphi (r) = (u_s \times r_s \times v_s) \circ \ldots \circ (u_1 \times r_1 \times v_1) \quad (s \geq 1).$$

We want to construct a set $P (r)$ of rules "simulating" $r$. Consider for any $1 \leq i \leq s$:

$$f_i = (u_i \times r_i \times v_i) \circ \ldots \circ (u_1 \times r_1 \times v_1)$$

and

$$\overline{f}_i = (u_s \times r_s \times v_s) \circ \ldots \circ (u_{i+1} \times r_{i+1} \times v_{i+1}).$$

We determine inductively $f_i^*$, $P_i (r)$, $I_i (r)$ and $\varphi_1$, $\varphi_2$ with $f_s^* = \varphi_1 (r)$, $P_s (r) = P (r)$ and $f_s = \varphi_2 (f_s^*)$.

If $u \in \Sigma \cup I_2$, $f \in S (G_2)$ with $d (f) = xuy$, we say: $u$ is unchanged under $f$ [relative to $(x, y)$] if $f = g_1 \times u \times g_2$ with $d (g_1) = x$ and $d (g_2) = y$; otherwise $u$ is changed under $f$ [relative to $(x, y)$].

Furthermore, if $w \in (\Sigma \cup I_2)^*$, then

$$w = y_0 \xi_1 y_1 \ldots \xi_m y_m$$

with $y_0, \ldots, y_m \in \Sigma^*$, $\xi_1, \ldots, \xi_m \in I_2$.

This decomposition is called $I$-decomposition of $w$.

Initial step: Consider the $I$-decomposition of $c (r_1) = y_0 \xi_1 \ldots \xi_m y_m$.

Let $d (r) = \overline{x}$ with $\varphi (\overline{x}) = d (r_1)$. Now create to any $\xi_\lambda$ which is changed under $f_1$ relative to $(u_1, y_0 \xi_1 \ldots y_{i-1}, y_i \xi_{i+1} \ldots y_m v_1)$ a new letter $\xi_\lambda (1, r)$ $(1 \leq \lambda \leq m)$. If a $\xi_\lambda$ is unchanged it corresponds to an unique $\hat{\xi}_\lambda$ in $c (r)$.

Define

$$\hat{\xi}_\lambda := \begin{cases} \xi_\lambda (1, r), & \text{if } \xi_\lambda \text{ is changed}, \\
\xi_\lambda^*, & \text{otherwise.} \end{cases}$$

Construct:

$$P_1 (r) = \{ \overline{\xi} \rightarrow y_0 \hat{\xi}_1 \ldots \hat{\xi}_m y_m \},$$

$$I_1 (r) = \{ \xi_\lambda (1, r) \mid \xi_\lambda \text{ is changed under } f_1 \},$$

$$f_1^* = \overline{u}_1 \times (\overline{\xi} \rightarrow y_0 \hat{\xi}_1 \ldots \hat{\xi}_m y_m) \times \overline{v}_1,$$

$$\varphi_2 (\xi_\lambda (1, r)) = \xi_\lambda; \quad \varphi_2 (\xi_\lambda^*) = \varphi (\xi_\lambda^*).$$

Induction step: Suppose $f_i^* \cdot I_i \cdot P_{i-1} (r)$ are constructed for $i > 1$. Consider $u_i d (r_i) v_i$ and $u_i c (r_i) v_i$. Then $c (f_{i-1}^*) = \overline{u} \hat{\xi} \overline{v}$ with $\xi \in I_{i-1} (r)$, $\varphi_2 (u) = u_i$, $\varphi_2 (v) = v_i$ and $\varphi_2 (\xi) = d (r_i)$.
Let \( c (r) = y_0 \eta_1 \ldots \eta_n y_n \) be the \( I \)-decomposition. Again, create to those \( \eta_h \) which are changed under \( f_i \) relative \( (u_i y_0 \eta_1 \ldots \eta_{h-1}, \eta_i \eta_{h+1} \ldots \eta_n y_n v_i) \) a new letter \( \eta_h (i, r) \). An unchanged \( \eta_h \) corresponds to an unique \( \eta_h^* \) in \( c (r) \). Denote by
\[
\hat{\eta}_h := \begin{cases} 
\eta_h (i, r), & \text{if } \eta_h \text{ is changed under } f_i, \\
\eta_h^*, & \text{otherwise}
\end{cases}
\]
and \( \hat{r}_i = \xi \to y_0 \hat{\eta}_1 \ldots \hat{\eta}_n y_n \).

Construct:
\[
P_i (r) = P_{i-1} (r) \cup \{ \hat{r}_i \},
\]
\[
I_i (r) = I_{i-1} (r) \cup \{ \eta_h (i, r) \mid \eta_h \text{ is changed under } f_i \},
\]
\[
f_i^* = (\bar{u} \times \bar{r}_i \times \bar{v}) \circ f_{i-1},
\]
\[
\varphi_2 (\eta_h (i, r)) = \eta_h, \quad \varphi_2 (\eta_h^*) = \varphi (\eta_h^*).
\]

By this construction we get for each \( r \in P_1 \) a production set \( P (r) : = P_s (r) \), an alphabet \( I (r) : = I_s (r) \) and a derivation \( f^{(r)} : = f_s^* \).

Now define \( G_3 \) and \( \varphi_1 \) by:

(1) \( G_3 : = (\Sigma, I_1 \cup \bigcup_{r \in P_1} I (r), \bigcup_{r \in P_1} P (r), \sigma_1) \).

(2) \( \varphi_1 (r) : = f^{(r)} \).

Part 2: "\( \varphi_1 \) is a simulation and \( \varphi_2 \) is a reduction".

Obviously, for all \( r, r' \in P_1 \):
\[
P (r) \cap P (r') \neq \emptyset \Rightarrow f^{(r)} = f^{(r')}
\]
holds. Using this fact it is easy to see that \( \varphi_1 \) is a simulation.

On the other hand \( \varphi = \varphi_2 \circ \varphi_1 \). Since \( \varphi \) is surjective on \( D (G_1) \), \( \varphi_2 \) must be surjective too. But this implies that \( \varphi_2 \) is a reduction.

Remark: The construction given above can be used to decide the property "closed" for expanding internal \( \varphi : G_1 \to G_2 \). The algorithm works as follows:

Stage 1: Perform the factorisation \( \varphi = \varphi_2 \circ \varphi_1 \), where \( \varphi_1 \) is a simulation and \( \varphi_2 \) is length-preserving, i. e. \( \varphi_2 (P_1) \subseteq P_2 \).

Stage 2: Decide with Schnorr's algorithm [9], whether or not \( \varphi_2 \) is a reduction. If the answer is "yes" then \( \varphi \) is closed, otherwise \( \varphi \) is not closed.

2. PARSING TIME AND INVERSE TRANSFORMATIONS

In this section we want to derive the main result. Consider a grammar \( G_1 = (\Sigma, I, P, \sigma) \). As analysers we use Turing machines which—faced with
$w \in \Sigma^*$—produce a derivation whenever it is possible and a failure-message if not, that means if $w \notin \mathcal{L}(G)$.

We indicate in which form the output is performed. We assign to any derivation $f$ a representation $\bar{f}$ which is in its essence the preorder representation of the corresponding derivation tree, more formally:

Consider to each $\xi \in I$ a pair of brackets $[\xi, \xi]$.

Let $f \in S(G)$:

(i) $\|f\| = 0 \Rightarrow f = u : \bar{f} = u$;

(ii) $\|f\| = 1 \Rightarrow f = u \times (\xi \rightarrow w) \times v$:

$$\bar{f} = u[\xi, \xi]v;$$

(iii) $\|f\| > 1 \Rightarrow f = (u \times r \times v) \circ f_1$; with

$$\bar{f}_1 = u' \cdot d(r) \cdot v'.$$

Define

$$\bar{f} = u' \cdot \bar{r} \cdot v'.$$

**Remark 1**: It is easy to see that $\bar{f}$ is well-defined.

**Remark 2**: As usual we can define the bracketing depth $bd(\bar{f})$.

Now, our analyser—faced with $w$—should produce $\bar{f}$ with $d(f) = \sigma$ and $c(f) = w$ if such an $f$ exists, otherwise the relation $w \notin \mathcal{L}(G)$ should be indicated by producing a special signal.

Given such an analyser $\mathcal{A}_G$, we can define the time function $T_{\mathcal{A}_G}(w)$ as usual. Note that always

$$\|f\| \leq T_{\mathcal{A}_G}(w),$$

if $f$ is the output to the input $w$.

**Theorem 2**: If $\varphi : G_1 \rightarrow G_2$ is an expanding transformation and $\mathcal{A}_{G_2}$ is an analyser such that

$$T_{\mathcal{A}_{G_2}}(w) \leq F(\|w\|) \quad (w \in \Sigma^*),$$

where $F : \mathbb{Z}_+ \rightarrow \mathbb{Z}_+$ is a function, then there is a constant $c$ and an analyser $\mathcal{A}_{G_1}$ such that

$$T_{\mathcal{A}_{G_1}}(w) \leq c \cdot F(\|w\|) \quad (w \in \Sigma^*).$$

**Proof**: By theorem 1 we can factorize $\varphi = \varphi_2 \circ \varphi_1$ with $\varphi_2$ a reduction and $\varphi_1$ a simulation. E. Bertsch has shown that the result is true for reductions [3]. Thus the theorem follows if we can show the result under the additional assumption that $\varphi$ is a simulation.
To prove this we first show:

Consider an expanding morphism $\varphi : G_1 \rightarrow G_2$, which operates identically on $I_1$. Suppose, for every $r \in P_1$ there exists exactly one $r_a \in P (\langle \varphi (r) \rangle )$ with $d (r_a) \in I_1$. Define the homomorphism $h$ by

$$h(x) = \begin{cases} \square, & \text{if } x \in \{[\xi], [\xi_1, \xi_2, \ldots, \xi_{k-1}] | \xi_1, \xi_2, \ldots, \xi_{k-1} \in I_2 - I_1 \}, \\ x, & \text{otherwise.} \end{cases}$$

Then for any $f \in S (G_1)$ with $d (f) \in I_1$:

$$h (\varphi (f)) = \bar{f} \quad \text{holds.} \quad \quad (\star)$$

Proof by induction on $|f|$

"$|f| = 1" \quad \text{then } f = r \in P_1.\)

We show the assertion by induction on $bd (\varphi (r))$:

"$bd (\varphi (r)) = 1" \quad \text{then } \varphi (r) \in P_1\)

and therefore $\varphi (r) = r$ ($\varphi$ operates identically on $\Sigma \cup I_1$), which proves the assertion.

Consider the case "$bd (\varphi (r)) = t > 1$".

First observe $\bar{r} = [\begin{array}{c} c (r) \\ d (r) \end{array}].$

Since $d (\varphi (r)) = d (r)$ and $c (\varphi (r)) = c (r)$ we can decompose $\varphi (r) = \bar{g}$ in the following way

$$\bar{g} = v_0 [u_1] v_1 \ldots [u_k] v_k,$$

where $u_1, \ldots, u_k \in (I_1 \cup \Sigma)^*$ and

$$v_0, \ldots, v_k \in (I_1 \cup \Sigma \cup \{[\xi], [\xi_1, \xi_2, \ldots, \xi_{k-1}] | \xi_1, \xi_2, \ldots, \xi_{k-1} \in I_2 \})^*$$

and $v_j$ contains no word of the form $[u]$, $u \in (I_1 \cup \Sigma)^*$ for $j = 0, \ldots, k$.

Now, define $G_2$ and $\varphi'$ as follows:

Eliminate the rules $\xi \rightarrow u_i$ by substituting $\xi_i$ by $u_i$ in all predecessor rules of $P (\langle \varphi (r) \rangle )$. We obtain $G_2', g$ changes into a derivation $g' \in S (G_2')$ with $\bar{g}' = v_0 u_1 v_1 \ldots u_k v_k$ and $bd (\bar{g}') \leq t - 1$.

Define $\varphi' : G_1 \rightarrow G_2'$ by

$$\varphi' (r') = \begin{cases} g' & \text{if } r' = r, \\ \varphi (r') & \text{otherwise,} \end{cases}$$

again $\varphi'$ fulfills the presumptions, as before define $h'$ for $G_2'$.
It is seen immediately that:

(i) \( \bar{g}' = v_0 \ u_1 \ v_1 \ u_2 \ldots \ v_{k-1} \ u_k \ v_k; \)
(ii) \( bd(\bar{g}') < bd(\bar{g}); \)
(iii) \( h'(\bar{g}') = h(\bar{g}) \) holds.

By the induction hypothesis we get \( h(\bar{g}) = h'(\bar{g}') = \bar{r}. \)

Induction step:

\[ \| f \| = s - 1 \Rightarrow \| f \| = s. \]

Observe that

\[ f = (u \times r \times v) \circ f_0 \quad \text{with} \quad r \in P_1, \quad f_0 \in S(G_1). \]

Then \( \bar{f} \) is obtained from \( \bar{f}_0 \) by substituting \( d(r) \) by \( \bar{r} \) using the decomposition \( \bar{f}_0 = w_1 \ d(r) \ w_2 \) with appropriate \( w_1, w_2. \)

Applying \( \varphi \) to \( f \) we get

\[ \varphi(f) = (u \times \varphi(r) \times v) \circ \varphi(f_0). \]

By our assumption we get

\[ \bar{\varphi(f)} = w_1' \ varphi(r) \ w_2' \]

and \( \bar{\varphi(f_0)} = w_1' \ d(r) \ w_2' \) with appropriate \( w_1', w_2'. \) By this \( \bar{\varphi(f)} \) is obtained from \( \varphi(f_0) \) by substituting \( d(r) \) by \( \varphi(r) \).

Application of \( h \) yields:

\[ h(w_1' \ d(r) \ w_2') = \tilde{w}_1 \ d(\tilde{r}) \ \tilde{w}_2 = f_0 \quad \text{(induction hypothesis)} \]

and

\[ h(\varphi(r)) = \bar{r}. \]

But then

\[ \tilde{w}_1 = w_1 = h(w_1') \quad \text{and} \quad \tilde{w}_2 = w_2 = h(w_2'), \]

we get

\[ h(\bar{\varphi(f)}) = w_1, \quad h(\varphi(r)) \ w_2 = w_1 \ \bar{r} \ w_2 = \bar{f} \]

and the proof of (\( \star \)) is complete.

Now, we are able to design the analyser \( \mathcal{A}_{G_1}. \)

Consider an input \( w \in \Sigma^*. \)

**Stage 1:** Using \( \mathcal{A}_{G_2} \) produce \( \bar{f} \) with \( d(f) = \sigma_2 \) and \( c(f) = w \) if \( w \in \mathcal{L}(G_2) = \mathcal{L}(G_1). \) Otherwise \( \mathcal{A}_{G_2} \) indicates that \( w \notin \mathcal{L}(G_2), \) and \( \mathcal{A}_{G_1} \) gives a message that \( w \notin \mathcal{L}(G_1). \)

**Stage 2:** Compute \( h(\bar{f}). \)
By the above assertion, we get

\[ h(\overline{f}) = h(\varphi(\varphi^{-1}(f))) = \varphi^{-1}(f). \]

[\(\varphi^{-1}(f)\) exists and is a derivation of \(w\) in \(D(G_1)\).]

This proves that the algorithm \(\mathfrak{A}_{G_1}\) is correct.

To perform stage 1 we need time

\[ T_{\mathfrak{A}_{G_1}}(w) \leq F(|w|). \]

To perform stage 2 we need time

\[ T_h \leq c'.|\overline{f}| \]

with a constant \(c'\).

Since \(\mathfrak{A}_{G_2}\) has to produce the output \(\overline{f}\) we get

\[ |\overline{f}| \leq T_{\mathfrak{A}_{G_2}}(w). \]

Combining both we get

\[ T_{\mathfrak{A}_{G_1}}(w) \leq T_{\mathfrak{A}_{G_2}}(w) + c'. T_{\mathfrak{A}_{G_2}}(w) \leq (c' + 1).F(|w|). \]

But this proves our result.

3. PARSE TIME AND TRANSFORMATIONS

Now we will show a converse result:

If \(\varphi : G_1 \rightarrow G_2\) is a expanding transformation, then from the analyzability of \(\mathfrak{L}(G_1)\) in time \(\leq f(|w|)\) it results that \(\mathfrak{L}(G_2)\) is analyzable in time \(\leq c.f(|w|)\). First we show this for reductions and then for simulations. Then by theorem 1 the result also holds for expanding transformations.

**Proposition:** Let \(\varphi : G_1 \rightarrow G_2\) be a reduction, \(\mathfrak{A}_{G_1}\) an analyser for \(\mathfrak{L}(G_1)\) with

\[ T_{\mathfrak{A}_{G_1}}(w) \leq F(|w|) \quad (w \in \Sigma^*), \]

where \(F : Z_+ \rightarrow Z_+\) is a function, then there is a constant \(c\) and an analyzer \(\mathfrak{A}_{G_2}\) for \(\mathfrak{L}(G_2)\) such that

\[ T_{\mathfrak{A}_{G_2}}(w) \leq c.F(|w|). \]

**Proof:** We remark that \(w \in \mathfrak{L}(G_1) \Rightarrow \varphi(w) = w \in \mathfrak{L}(G_2)\). Let be \(f \in D(G_1)\) with \(d(f) = \sigma_1\) and \(c(f) = w\) and \(\overline{f}\) defined as in 2.

vol. 12, n° 2, 1978
Consider the homomorphism $g$ defined by
\[
g(x) = \begin{cases} 
\varphi(x) & \text{if } x \in \Sigma \cup I_1, \\
\varphi(\xi) & \text{if } x = [r], \quad \xi \in I_1,
\end{cases}
\]
Then it is easy to see that
\[
(★★) \quad g(\bar{f}) = \bar{\varphi}(f).
\]
Now we construct the analyser $\mathcal{A}_{\alpha_2}$ in the same way as in theorem 2 with the homomorphism $g$ instead of $h$. Using (★★) instead of (★) the assertion follows by the same argument.

To prove a similar result for $\varphi$ being a simulation, we require that $\mathcal{A}_{\alpha_1}$ analysing $w \in \mathcal{L}(G_1)$ gives an output $\bar{f}$, which is again a parenthesis-representation of a derivation $f$ but contains some more information about the used rules:

Consider a grammar $G$ and to each $\xi \in I$ and each $r \in P$ a pair of brackets $[r]_{\xi}$.  

Let $f \in S(G)$ then:

(i) $||f|| = 0$:

\[f = 1_u, \quad \bar{f} = u;\]

(ii) $||f|| = 1$:

\[f = u \times r \times v, \quad \bar{f} = u \left[ r \left( c(r) \right) \right] \bar{v};\]

(iii) $||f|| > 1 \Rightarrow f = (u \times r \times v) \circ f_1$; with

\[\bar{f}_1 = u' \bar{d}(r) v';\]

define

\[\bar{f} = u' \bar{r} \bar{v}'.\]

For abbreviation we set

\[\left[ \begin{array}{l} r \\ \xi \end{array} \right] = \left\{ \left[ \begin{array}{l} r \\ \xi \end{array} \right] \left| \xi \in I, r \in P \right. \right\}\]

and

\[\left[ \begin{array}{l} r \\ \xi \end{array} \right] = \left\{ \left[ \begin{array}{l} r \\ \xi \end{array} \right] \left| \xi \in I, r \in P \right. \right\}.\]
Now we assume, that an analyzer $\mathcal{U}_{g_i}$ produces this parenthesis-representation of a derivation if possible.

The role of the homomorphisms $h$ respective $g$ in the proofs of theorem 2 and 3 is played by a pushdown-transducer which transduces $f$ into $\varphi(f)$ for $f \in D(G_1)$. We use the conception of a pdt as given in [4].

**Theorem 3:** Let $\varphi : G_1 \to G_2$ be a simulation, $\mathcal{U}_{g_i}$ an analyzer with

$$T_{\mathcal{U}_{g_1}}(w) \leq F(\mid w \mid), \quad w \in \Sigma^*,$$

where $F : Z_+ \to Z_+$ is a function, then there exists an analyzer $\mathcal{U}_{g_2}$ and a constant $c$ with

$$T_{\mathcal{U}_{g_2}}(w) \leq c \cdot F(\mid w \mid).$$

**Proof:** First we construct a one-state-pdt $p$ which transduces $f$ into $\varphi(f)$ for an arbitrary $f \in D(G_1)$:

$$I_p = \Sigma \cup I_1 \cup \left[ \begin{array}{c} P_1 \cup \; \; P_1 \end{array} \right], \quad O_p = \Sigma \cup I_2 \cup \left[ \begin{array}{c} P_2 \cup \; \; P_2 \end{array} \right],$$

$$S_p = \{ s \}; \quad K_p = O_p \cup \$, \quad k_0 = \$, \quad s_0 = s \quad \text{and} \quad \delta_p$$

defined as follows:

**Initialisation of the pushdown store:**

$$\delta_p(x, s, \$) = (s, \gamma \$, x'),$$

$$x = \left[ \begin{array}{c} r \end{array} \right] \in \left[ \begin{array}{c} r \end{array} \right], \quad x' \gamma = \varphi(r);$$

**output of an symbol in $c(f)$:**

$$\delta_p(x, s, y) = (s, \square, y), \quad x = y \in \Sigma \cup I_1;$$

**output of symbols of simulation rules:**

$$\delta_p(\square, s, y) = (s, \square, y), \quad y \in \left[ \begin{array}{c} r \end{array} \right] \cup \left[ \begin{array}{c} r \end{array} \right],$$

**storing the simulated rule $\varphi(r)$ instead of $d(r)$ of the top at the pushdown store, producing the first parenthesis $x'$ of $\varphi(r)$ as output:**

$$\delta_p(x, s, y) = (s, \gamma, x'), \quad x = \left[ \begin{array}{c} r \end{array} \right] \in \left[ \begin{array}{c} r \end{array} \right], \quad y = d(r),$$

$$\varphi(r) = x' \gamma.$$

Let $F_p : I_p^* \to O_p^*$ be the realized transduction, then

$$\text{(*)} \quad f \in D(G_1) \Rightarrow F_p(f) = \varphi(f) \quad \text{holds.}$$

vol. 12, n° 2, 1978
We give a short idea how to prove this: (induction on \( s = \| f \| \)):

(i) \( \| f \| = 1 \) then \( f = r \) holds and we can verify:

\[
\left( s, \left[ \begin{array}{c} c(r) \end{array} \right]_{d(r)}, \emptyset, \square \right) \vdash_p \left( s, c(r), \left[ \begin{array}{c} r \end{array} \right]_{d(r)}, \gamma \emptyset, \left[ \begin{array}{c} 1 \end{array} \right]_{d(r)} \right) \vdash_p \ldots \vdash_p (s, \emptyset, \emptyset, \varphi(r)),
\]

analogous we get

\[
\left( s, r, d(r) \nu, \emptyset \right) \vdash_p \ldots \vdash_p (s, \emptyset, \emptyset, v, \varphi(r)),
\]

which we need in (ii).

(ii) \( \| f \| = s-1 \Rightarrow \| f \| = s \)".

Let be \( \| f \| = s > 1 \) then \( f = (u \times r \times v) \circ f_1,\| f_1 \| = s-1 \).

We can decompose \( f, \varphi(f), f_1, \varphi(f_1) \) as follows:

\[
\begin{align*}
\overline{f} &= u' \overline{r} \nu', & \overline{\varphi(f)} &= u'' \varphi(r) \nu'' \\
\overline{f_1} &= u' d(r) \nu', & \overline{\varphi(f_1)} &= u'' d(r) \nu''.
\end{align*}
\]

\( p \) transduces \( u' d(r) \nu' \) into \( u'' d(r) \nu'' \) by induction hypothesis, then one can show using the construction of \( p \):

\[
\left( s, u' d(r) \nu', \emptyset, \square \right) \vdash_p \ldots \vdash_p (s, d(r) \nu', d(r) \gamma \emptyset, u'')
\]

\([d(r)] \) is at the top of the pushdown store because it is the next output symbol]:

\[
\vdash_p (s, \nu', \gamma \emptyset, u'' d(r)) \vdash_p \ldots \vdash_p (s, \emptyset, \emptyset, u'' d(r) \nu'').
\]

Then also

\[
\left( s, u' \overline{r} \nu', \emptyset, \square \right) \vdash_p \ldots \vdash_p (s, \overline{r} \nu', d(r) \gamma \emptyset, u'') \quad \text{holds}.
\]

Now we can insert the computation on \( \overline{r} \) using part (i):

\[
\left( s, \overline{r} \nu', d(r) \gamma \emptyset, u'' \right) \vdash_p \ldots \vdash_p (s, \nu', \gamma \emptyset, u'' \varphi(r))
\]

and again using the induction hypothesis continuing the computation like that of \( f_1 \):

\[
\vdash_p \ldots \vdash_p (s, \emptyset, \emptyset, u'' \varphi(r) \nu''),
\]

which proves the assertion.

Now we construct the analyzer \( \mathcal{A}_{\alpha_2} \) similar to that of theorems 2 and 3:

Given an input \( w \in \Sigma^* \).

R.A.I.R.O. Informatique théorique/Theoretical Computer Science
Stage 1: Using $\mathcal{U}_G$, produce $\overline{f}$, $f \in D(G_1)$, with
\[
d(f) = \sigma, \quad c(f) = w \quad \text{if} \quad w \in \mathcal{L}(G_1) = \mathcal{L}(G_2),
\]
or a failure message if $w \notin \mathcal{L}(G_2)$.

Stage 2: Compute $F_p(\overline{f})$.

Again the assertion follows with the same argument as in the proofs before, if one has in mind that $T_{F_p}(\overline{f}) \leq |\phi(\overline{f})|$ (at each step $p$ produces one output symbol!) and $|\phi(\overline{f})| \leq c', |f|$ with $c' = 2 \max \{ |\phi(r)| \mid r \in P_1 \}$.

Remark 1: If $G_1$ and $G_2$ are linear grammars we can perform the transduction $\overline{f} \rightarrow \phi(\overline{f})$ by an homomorphism.

Proof: All rules of $G_1$, $G_2$ are of the form
\[
\xi \rightarrow u \eta v, \quad u, v \in \Sigma^*,
\]
or
\[
\xi \rightarrow w, \quad w \in \Sigma^*.
\]
Consider $r = (\xi \rightarrow u \eta v), \eta \in I_1 \cup \{ \square \}$ then
\[
\phi(r) = (u_1 \ldots u_{s-1} \times r \times v_{s-1} \ldots v_1) \circ \ldots \circ (u_1 \times r_2 \times v_1) \circ (\square \times r_1 \times \square)
\]
with
\[
\xi_i = \xi_i, \quad 1 \leq i \leq s,
\]
\[
\xi_{s+1} = \eta \quad \text{and} \quad u_1 \ldots u_s = u, \quad v_s \ldots v_1 = v.
\]
It follows immediately that
\[
\overline{\phi(r)} = \left[ \begin{array}{c} u_1 \\ \xi \\ \vdots \\ \xi_s \\ \xi s+1 \\ \vdots \\ \xi_1 \end{array} \right] \quad \text{holds}.
\]
Let be
\[
u(r) = v_s \ldots v_1
\]
and define a homomorphism $f_p$ as follows:
\[
\begin{align*}
& x \quad \text{if} \quad x \in I_1, \\
& \square \quad \text{if} \quad x \in \Sigma, \\
& f_p(x) = \begin{cases} u(r) & \text{if} \quad x = \begin{array}{c} r \\ I_1 \end{array} \\
v(r) & \text{if} \quad x = \begin{array}{c} r \\ I_1 \end{array} \end{cases} 
\end{align*}
\]
vol. 12, n° 2, 1978
Then it is easy to see, that for $f \in D(G_1)$:

$$f_p(f) = \varphi(f)$$

**Remark 2:** With remark 1 we have seen, that the transduction of derivations in $G_1$ into derivations in $G_2$ can be done by a device which is less powerful than the device which is used for analyzing. That means: a deterministic $pdt$ for context-free languages, which require a non-deterministic $pda$ for analyzing, and an finite state-transducer (to perform the homomorphisms) in the case of linear grammars.

### 4. CONCLUDING REMARKS

We give some comments to our results.

**Remark 1:** As indicated in the introduction expanding transformations are induced by certain well-known normal-form theorems. The binary form of context-free grammars and the elimination of $\varepsilon$-rules in a context-free grammar are of this type.

Therefore we can conclude (with some minor addition to our proofs in the latter case) that parsing time remains unchanged under both constructions.

**Remark 2:** We can deal with parsing space too. If the space definition includes the output tape all the constructions, both Berthsc's and ours, preserve space. (For theorem 3 one should have in mind that the maximal length of the pushdown store of the $pdt$ $p$ does not exceed the output length.) Therefore parsing space remains unchanged in order of magnitude under inverse expanding transformations and expanding transformations.

### REFERENCES


