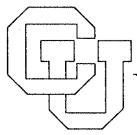
Computations in Coordinated Pair Systems

A. Ehrenfeucht, H. J. Hoogeboon G. Rozenberg*

CU-CS-260-83 September 1984



University of Colorado at Boulder
DEPARTMENT OF COMPUTER SCIENCE

*This research was supported in part by NSF Grant number MCS 83-05245.

COMPUTATIONS IN COORDINATED PAIR SYSTEMS

A. EHRENFEUCHT,

University of Colorado, Department of Computer Science, Boulder, CO 80309, USA

H.J. HOOGEBOOM and G. ROZENBERG

University of Leiden, Institute of Applied Mathematics and Computer Science, P.O. Box 9512, 2300 RA LEIDEN, The Netherlands.

Selective substitution grammars provide a rather general framework for the *grammatically* oriented formal language theory (see, e.g., [R1], [K], and [KR]). They were generalized in [R2] to *coordinated table selective substitution systems (cts systems)* which provide a convenient unifying framework for both *grammars and machines (automata).* The present paper investigates *coordinated pair (cp) systems* which form a subclass of cts systems corresponding in a very natural way to push-down automata: thus cp systems generate context-free languages (all and only). This paper investigates the structure of computations in cp systems.

INTRODUCTION

Selective substitution grammars provide a general framework for rewriting systems (see, e.g., [R1], [K], and [KR]). Through the notion of *selective substitution* they formalize the notion of "rewriting of selected occurrences in a string". The theory of selective substitution grammars developed until now has turned

out to be successful in the sense that, within this theory, several central notions of formal language theory were captured in a natural way and a number of interesting new notions (together with results concerning them) have emerged.

The theory of selective substitution grammars is "grammatically oriented" in the sense that it directly formalizes the notion of a rewriting system (grammar) and so it concentrates on grammars as the way of defining formal languages.

In [R2] the notion of a selective substitution grammar was extended so as to provide a unifying framework for both grammars and machines; the new model is called an *(extended) coordinated table selective substitution system,* abbreviated ects *system.* This extension has turned out to be rather simple and natural: rather than to rewrite strings one rewrites n-tuples of strings, where the way strings on various coordinates are rewritten is "coordinated" by n-tuples of productions. In this way one can easily formalize the notion of "the way the input is read", "access to the memory" and "processing of the input and of the memory" which are certainly very central notions in the theory of machines (automata). It is demonstrated in [R2] how quite a considerable number of known machine (and grammar) models can be expressed as ects systems.

The present paper continues research on the theory of ects systems by initiating the investigation of "concrete" submodels of ects systems. We consider the notion of a *coordinated pair (cp) system* which seems to be a very natural

submodel of the ects model and which at the same time captures in a very natural way the push-down automaton model. A cp system is essentially a pair of grammars, the first one right-linear and the second right-boundary (roughly speaking a right-boundary grammar is a right linear grammar in which no distinction is made between terminal and nonterminal symbols - productions are still applied to the last occurrence in a word only) which work in a coordinated (in the sense of ects systems) fashion. The main aim of this paper is to provide a rather detailed analysis of computations in cp systems - in this way we "try out" the cp systems approach to the investigation of computations in push-down automata. Since push-down automata occupy a very central role in (machine oriented) formal language theory, we feel that such an analysis is worthwhile - we believe that it sheds a somewhat different light on the nature of context-free languages. Although some of the results we present are known (either "directly" or in the "folkore of push-down automata"), this analysis provides useful technical tools and is absolutely necessary for further formal investigation of cp systems. As a matter of fact the usefulness of this analysis and of the formalism we present in our paper was demonstrated already in [EHR2] and [EHR3] where a new approach to the investigation of pumping properties of context-free languages is presented.

The paper is organized as follows.

In Section 1 the basic notion of this paper - a *coordinated pair system* (cp *system* for short) - is introduced. In Section 2 several ways of describing compu-

tations in cp systems are considered - such descriptions assign to each "successful computation" (i.e., a computation leading to a word of the language of a cp system) in a cp system a word. It is demonstrated that the set of all such "description words" for a cp system can be computed by a cp system - one might say that the class of cp systems is "descriptionally closed". In Section 3 we investigate how "to recover" the language computed by a cp system from (one kind of - the so called "weak") descriptions of its successful computations - this analysis yields an alternative proof of the Chomsky-Schutzenberger theorem for context-free languages. In Section 5 the "exchange theorem" is proved which says how one can "sweep" subcomputations between two (not necessarily) different computations in a cp system.

0. PRELIMINARIES

We assume that the reader is familiar with basic formal language theory and in particular with the theory of context-free grammars and the theory of push-down automata (see, e.g., [H] and [S]).

In order to fix the notation and terminology for this paper we will recall now some more specific points.

Given an alphabet Σ, $\hat{\Sigma}$ will always mean the alphabet $\{\hat{a} \mid a \in \Sigma\}$ and it is assumed that $\Sigma \cap \hat{\Sigma} = \emptyset$.

For a word x, $|x|$ denotes its length and, if $1 \leq k \leq |x|$, then $x(k)$ denotes the $k-th$ letter of x; Λ denotes the empty word.

D_Σ denotes the set of (semi-)Dyck words over $\Sigma \bigcup \hat{\Sigma}$, where, for each letter a in Σ, its matching "right parenthesis" is $\hat{a}$ in $\hat{\Sigma}$. More formally, D_Σ is the minimal language L over $\Sigma \bigcup \hat{\Sigma}$ that satisfies

(i) $\Lambda \in L$,

(ii) if $w \in L$, then $a w \hat{a} \in L$ for every $a \in \Sigma$, and

(iii) if w_1 , $w_2 \in L$, then $w_1 w_2 \in L$.

A letter to letter homomorphism is called a *coding* and a homomorphism that maps each letter either into a letter or into the empty word is called a *weak coding*. If Σ and Δ are alphabets such that $\Delta \subseteq \Sigma$, then $pres_{\Sigma,\Delta}$ is the weak coding of Σ^* defined by: $pres_{\Sigma,\Delta}(a) = a$ if $a \in \Delta$ and $pres_{\Sigma,\Delta}(a) = \Lambda$ if $a \in \Sigma \backslash \Delta$. Whenever the alphabet Σ is understood from the context, then we write $pres_\Delta$ rather than $pres_{\Sigma,\Delta}$.

A *context-free grammar,* abbreviated cf grammar, is specified in the form $G = (\Sigma, P, S, \Delta)$ where Σ is its alphabet, P its set of productions, S its axiom and $\Delta \subseteq \Sigma$ its terminal alphabet. For $x, y \in \Sigma^*$ and $\pi \in P$ we write $x \underset{G}{\overset{\pi}{\Rightarrow}} y$ if x directly derives y using π. Productions of the form $X \rightarrow Y$, where X and Y are nonterminal symbols (i.e., the elements of $\Sigma \backslash \Delta$) are called *chain productions.*

We use $\mathbf{L}(CF)$ to denote the class of context-free languages.

A *right-linear grammar,* abbreviated *rl grammar,* is a context-free grammar $G = (\Sigma, P, S, \Delta)$, which has its productions in the set $(\Sigma\backslash\Delta)\times\Delta^{*}\cdot((\Sigma\backslash\Delta) \cup\{\Lambda\})$.

A *derivation (in G)* is a sequence $\sigma = \sigma(0)$, $\sigma(1)$, ..., $\sigma(n)$, $n \geq 0$, of words from Σ^{*} such that, for every $1 \leq i \leq n$, $\sigma(i-1) \underset{G}{\overset{\pi_i}{\Longrightarrow}} \sigma(i)$, where π_i is a production from P. The sequence π_1 , ..., π_n is called the *control sequence of* σ and denoted by $cont(\sigma)$; if $n = 0$, then $cont(\sigma)$ is the empty sequence. If $\sigma(n) \in \Delta^{*}$, then both σ and $cont(\sigma)$ are called *successful.*

A *right-boundary grammar,* abbreviated *rb grammar,* differs from a right-linear grammar in the fact that it does not distinguish between terminal and nonterminal symbols. A rb grammar is specified in the form of a 3-tuple $G = (\Sigma, P, S)$, where Σ is its alphabet, $P \subseteq \Sigma \times \Sigma^{*}$ is its set of productions and $S \in \Sigma$ its axiom. As in the case of a rl grammar productions are applied to the last occurrence in a word only. Thus, for $x, y \in \Sigma^{*}$ and $\pi = a \longrightarrow w \in P$, x directly derives y (in G using π), written $x \underset{G}{\overset{\pi}{\Longrightarrow}} y$, if $x = za$ and $y = zw$ for some $z \in \Sigma^{*}$.

1. BASIC NOTIONS

In this section the basic notion of this paper - a *coordinated pair system* - is introduced and several key notions pertinent to it are discussed.

Definition 1.1 A *coordinated pair system,* abbreviated *cp system,* is a triple $G = (G_1, G_2, R)$ such that

(1) $G_1 = (\Sigma_1, P_1, S_1, \Delta)$ is a rl grammar,

(2) $G_2 = (\Sigma_2, P_2, S_2)$ is a rb grammar, and

(3) $R \subseteq P_1 \times P_2$. □

G_1 and G_2 are referred to as the *first* and *second* component of G respectively. Elements of R will be referred to as *rewrites of* G.

Remark. It is easily seen that a cp system is a special case of a *sequential uniform ects system* (see [R2]) with two coordinates only. We have simplified the notation and terminology in a way that seems more convenient to deal with a simple subcase of a rather general definition. □

Definition 1.2. Let $G = (G_1, G_2, R)$ be a cp system, where $G_1 = (\Sigma_1, P_1, S_1, \Delta)$ and $G_2 = (\Sigma_2, P_2, S_2)$.

(1) Let $x = (x_1, x_2)$, $y = (y_1, y_2) \in \Sigma_1^* \times \Sigma_2^*$.

x directly computes y (in G), denoted $x \underset{G}{\Rightarrow} y$, if there exists a rewrite $\pi = (\pi_1, \pi_2) \in R$ such that $x_1 \overset{\pi_1}{\underset{G_1}{\Rightarrow}} y_1$ and $x_2 \overset{\pi_2}{\underset{G_2}{\Rightarrow}} y_2$; we write then $x \overset{\pi}{\underset{G}{\Rightarrow}} y$ and

we say that x *directly computes* y *(in* G*) using* π.

$\overset{*}{\underset{G}{\Rightarrow}}$ denotes the reflexive and transitive closure of $\underset{G}{\Rightarrow}$. If $x \overset{*}{\underset{G}{\Rightarrow}} y$, then we say that x *computes* y *(in* G*).*

(2) A *computation (in* G*)* is a sequence $\rho = \rho(0), \ldots, \rho(n)$ of elements from $\Sigma_1^* \times \Sigma_2^*$, $n \geq 0$, such that $\rho(0) = (S_1, S_2)$ and, for $1 \leq i \leq n$, $\rho(i-1) \underset{G}{\Rightarrow} \rho(i)$.

We say that ρ is *successful* if $\rho(n) = (u, \Lambda)$ for some $u \in \Delta^*$.

(3) Let $\rho = \rho(0), \ldots, \rho(n)$, $n \geq 1$, be a computation in G. The sequence $\pi_1, \pi_2, \ldots, \pi_n$ of rewrites from R such that, for $1 \leq i \leq n$, $\rho(i-1) \overset{\pi_i}{\underset{G}{\Rightarrow}} \rho(i)$ is called the *control sequence of* ρ and denoted by $cont(\rho)$. If $\rho = \rho(0)$, then we define $cont(\rho)$ to be the empty sequence. If ρ is a successful computation, then we say that $cont(\rho)$ is a *successful control sequence.*

(4) Let $\rho = \rho(0), \ldots, \rho(n)$ be a successful computation in G. Then the *result of* ρ, denoted by $res(\rho)$, is defined by $res(\rho) = u$, where $\rho(n) = (u, \Lambda)$.

(5) The *language of* G, denoted $L(G)$, is defined by

$L(G) = \{res(\rho) \mid \rho \text{ is a successful computation in } G\}$.

$L(G)$ is also referred to as a *coordinated pair language* or *cp language* for short. □

If G is a cp system, then we say that G *computes* the language $L(G)$.

The class of all cp languages is denoted by $\mathbf{L}(CP)$.

Remark. (1) We realize that we somewhat abuse the notation by writing sequences in the form $\rho = \rho(0), \ldots, \rho(n)$ rather than $\rho = <\rho(0), \ldots, \rho(n)>$ (this leads to somewhat ambiguous expressions like, e.g., $\rho = \rho(0)$). However, using brackets to delimit sequences would lead to an additional burden on the already involved notation. We hope that abuses of notation of this type will not lead to misunderstandings.

(2) It is instructive to note that (in the notation as above) if $\rho = \rho(0), \ldots, \rho(n)$ is a computation in G, then for each $1 \leq i \leq n$ there is a unique $\pi \in R$ such that $\rho(i-1) \underset{G}{\overset{\pi}{\Rightarrow}} \rho(i)$. Hence the notion of control sequence is well defined. □

Example 1.1. Let $G = (G_1, G_2, R)$ be the cp system, where

(a) $G_1 = (\{X, Y, a, b, c\}, P_1, X, \{a, b, c\})$ with

$P_1 = \{X \rightarrow aX, X \rightarrow bY, Y \rightarrow bX, Y \rightarrow cY, Y \rightarrow c\}$,

(b) $G_2 = (\{A, B\}, P_2, A)$ with $P_2 = \{A \rightarrow BA, A \rightarrow \Lambda, B \rightarrow A, B \rightarrow \Lambda\}$,

(c) R consists of the following rewrites:

$\psi_1 = (X \rightarrow aX, A \rightarrow BA)$,

$\psi_2 = (X \rightarrow bY, A \rightarrow \Lambda)$,

$\psi_3 = (Y \rightarrow bX, B \rightarrow A)$,

$\psi_4 = (Y \rightarrow cY, B \rightarrow \Lambda)$ and

$\psi_0 = (Y \rightarrow c, B \rightarrow \Lambda)$.

Consider in G the successful computation $\rho = \rho(0), \rho(1), \ldots, \rho(6)$, where

$\rho(0) = ($ $X,\ A$ $),$

$\rho(1) = ($ $a\ X,\ BA$ $),$

$\rho(2) = ($ $a\ a\ X,\ BBA$ $),$

$\rho(3) = ($ $a\ a\ b\ Y,\ BB$ $),$

$\rho(4) = ($ $a\ a\ b\ b\ X,\ BA$ $),$

$\rho(5) = ($ $a\ a\ b\ b\ b\ Y,\ B$ $),$

$\rho(6) = ($ $a\ a\ b\ b\ b\ c\ ,\ \Lambda$ $).$

For this computation we have $cont(\rho) = \psi_1, \psi_1, \psi_2, \psi_3, \psi_2, \psi_0$. □

It is easily seen (see [R2]) that a push-down automaton may be interpreted as a cp system and that a cp system may be interpreted as a push-down automaton. These observations yield the following result.

Theorem 1.1. $\mathbf{L}(CP) = \mathbf{L}(CF)$. □

Remark. In order to stress the fact that in this paper we deal with (push-down automata via) the formalism of cp systems only, we will use the phrase "cp languages" rather than "cf languages" - according to above theorem the two phrases are equivalent. □

2. DESCRIPTIONS OF COMPUTATIONS

In this section we will consider several ways of describing computations in a cp system. We will consider descriptions which assign to each successful computation in a cp system a word and then we demonstrate that the set of all such descriptions can be computed by a cp system.

Let $G = (G_1, G_2, R)$ be a cp system, where $G_1 = (\Sigma_1, P_1, S_1, \Delta)$ and $G_2 = (\Sigma_2, P_2, S_2)$ and assume moreover that $\Sigma_1 \cap \Sigma_2 = \emptyset$.

In order to simplify our notation we will assume that: *G is an arbitrary cp system, but it is fixed for the considerations of this section.*

Definition 2.1. Let

$\Gamma = \{[S_1; S_2]\} \cup \{[\pi, i] \mid \pi = (\pi_1, A \rightarrow w) \in R,\ i \in \mathbf{N} \text{ and } 0 \leq i \leq |w|\}$.

(1) Let $\pi = (\pi_1, A \rightarrow w) \in R$.

The *trail of* π, denoted by $trl(\pi)$, is the word over Γ defined by

$trl(\pi) = [\pi, 0]\ [\pi, 1] \cdots [\pi, |w|]$.

(2) Let ρ be a computation in G with control sequence $\pi = \pi_1, \ldots, \pi_n$, for some $n \geq 0$, $\pi_1, \ldots, \pi_n \in R$.

The *trail of* ρ, denoted $trl(\rho)$, is the word over Γ defined by

$trl(\rho) = [S_1;S_2]\ trl(\pi_1) \cdots\ trl(\pi_n)$. □

Remark. Note that if $\rho = \rho(0) = (S_1, S_2)$, then $trl(\rho) = [S_1; S_2]$. □

Definition 2.2

(1) *ctb* is the homomorphism from Γ^* into Δ^* defined as follows.

For $\tau \in \Gamma$, $ctb(\tau)$ equals

- u, if $\tau = [\pi, 0]$ for some $\pi \in R$ where either $\pi = (X \rightarrow uY, \pi_2)$ or $\pi = (X \rightarrow u, \pi_2)$ with $X, Y \in \Sigma_1 \backslash \Delta$, $u \in \Delta^*$,
- Λ, otherwise.

For a word $\alpha \in \Gamma^*$, $ctb(\alpha)$ is referred to as the *contribution of* α.

(2) Let $\Theta = \Sigma_1 \backslash \Delta$ and let $\Xi = \Sigma_1 \cup \hat{\Theta} \cup \Sigma_2 \cup \hat{\Sigma}_2$.

des is the homomorphism from Γ^* into Ξ^* defined as follows.

For $\tau \in \Gamma$, $des(\tau)$ equals

- $S_1 S_2$, if $\tau = [S_1 ; S_2]$,
- $\hat{X} u \hat{A}$, if $\tau = [\pi, 0]$, where $\pi = (X \rightarrow u, A \rightarrow w) \in R$,
- $w(k)$, if $\tau = [\pi, k]$, where $\pi = (X \rightarrow u, A \rightarrow w) \in R$ and $1 \leq k \leq |w|$.

For a word $\alpha \in \Gamma^*$, $des(\alpha)$ is referred to as the *description of* α.

(3) *wdes* is the homomorphism from Γ^* into $(\Sigma_2 \cup \hat{\Sigma}_2)^*$ defined as follows.

For $\tau \in \Gamma$, $wdes(\tau)$ equals

- S_2 , if $\tau = [S_1 ; S_2]$,
- $\hat{A}$, if $\tau = [\pi , 0]$, where $\pi = (\pi_1 , A \rightarrow w) \in R$,
- $w(k)$, if $\tau = [\pi , k]$, where $\pi = (\pi_1 , A \rightarrow w) \in R$ and $1 \leq k \leq |w|$.

For a word $\alpha \in \Gamma^*$, $wdes(\alpha)$ is referred to as the *weak description of* α. □

If ρ is a computation in G, then $des(trl(\rho))$ ($wdes(trl(\rho))$ respectively) is simply called the *(weak) description of* ρ.

The set of (weak) descriptions of all successful computations in G is denoted by $dsc(G)$ ($wdsc(G)$ respectively).

Example 1.1. (continued)

$\alpha =$

$trl(\rho) = [X;A]\ [\psi_1,0]\ [\psi_1,1]\ [\psi_1,2]\ [\psi_1,0]\ [\psi_1,1]\ [\psi_1,2]\ [\psi_2,0]\ [\psi_3,0]\ [\psi_3,1]\ [\psi_2,0]\ [\psi_0,0]$

$ctb(\alpha) = \Lambda \quad a \quad \Lambda \quad \Lambda \quad a \quad \Lambda \quad \Lambda \quad b \quad b \quad \Lambda \quad b \quad c$

$des(\alpha) = XA \quad \hat{X}aX\hat{A} \quad B \quad A \quad \hat{X}aX\hat{A} \quad B \quad A \quad \hat{X}bY\hat{A} \quad \hat{Y}bX\hat{B} \quad A \quad \hat{X}bY\hat{A} \quad \hat{Y}c\hat{B}$

$wdes(\alpha) = A \quad \hat{A} \quad B \quad A \quad \hat{A} \quad B \quad A \quad \hat{A} \quad \hat{B} \quad A \quad \hat{A} \quad \hat{B}$

For a successful computation ρ, both $trl(\rho)$ and $des(trl(\rho))$ carry "all" information about ρ and in particular they carry the information about $res(\rho)$. As a matter of fact it is obvious that

$res(\rho) = ctb(trl(\rho)) = pres_{\searrow}(des(trl(\rho)))$.

Thus we have the following result.

Theorem 2.1. $pres_{\searrow}(dsc(G)) = L(G)$. □

Furthermore we can prove the following "self-descriptional" property of cp systems as far as the set of descriptions of successful computations is concerned.

Theorem 2.2. $dsc(G)$ is a cp language.

Proof.

Let Z be a symbol not in $\Sigma_1 \cup \Sigma_2$ and let $\overline{\Theta} = \{\overline{a} \mid a \in \Sigma_1 \backslash \Delta\}$.

Let P be the set of rewrites defined as follows.

(1) $\pi_Z = (Z \longrightarrow S_1\, S_2 \overline{S}_1\, ,\, S_2 \longrightarrow S_2) \in P$.

(2) If $\pi = (\pi_1, \pi_2) \in R$, $\pi_1 = X \longrightarrow uY$, where $u \in \Delta^*$ and $Y \in \Sigma_1 \backslash \Delta$, then $(\overline{X} \longrightarrow des(trl(\pi))\overline{Y}\ ,\ \pi_2) \in P$.

(3) If $\pi = (\pi_1, \pi_2) \in R$, $\pi_1 = X \longrightarrow u$, where $u \in \Delta^*$, then $(\overline{X} \longrightarrow des(trl(\pi))\ ,\ \pi_2) \in P$.

(4) Only productions obtained as in (1) through (3) above are in P.

Then let $H = (H_1\ ,\ H_2\ ,\ P)$ be the cp system such that $H_2 = G_2$ and $H_1 = (\Xi \cup \overline{\Theta} \cup \{Z\}, P_1, Z, \Xi)$, where P_1 consists of all productions which occur as first components of rewrites in P.

Now we observe the following, easy to prove, properties of H.

(i) The cases (2) and (3) of the above definition of P define a bijection ϕ from R to $P - \{\pi_Z\}$.

(ii) A sequence $\pi_1\ ,\ \pi_2\ ,\ ...,\ \pi_n\ ,\ n \geq 0$, in R is the control sequence of a (successful) computation ρ in G if and only if $\pi_Z, \phi(\pi_1)\ ,\ ...,\ \phi(\pi_n)$ is the control

sequence of a (successful) computation σ in H.

(iii) Moreover if ρ and σ are successful computations related as above, then $res(\sigma) = des(trl(\rho))$.

From the above observations it follows immediately that $L(H) = dsc(G)$. □

Unlike $des(trl(\rho))$, $wdes(trl(\rho))$ carries only some information about the computation ρ. In particular, if ρ is successful, then it does not carry any information about $res(\rho)$. But, again, the set of weak descriptions of all successful derivations in G can be computed by a cp system.

Theorem 2.3. $wdsc(G)$ is a cp language.

Proof.

One notes that $wdsc(G) = pres_{\Sigma_2 \cup \Sigma_2}(dsc(G))$. Since it is easily seen that the class of cp languages is closed under weak codings, the theorem follows from Theorem 2.2. □

Theorem 2.4. $wdsc(G) \subseteq D_{\Sigma_2}$.

Proof. (sketch)

We sketch here the main idea only leaving the formal (perhaps tedious) proof to the reader.

(i) Observing the behavior of second components of a successful computation $\rho = \rho(0), \ldots, \rho(n)$ one notices that the symbol to be rewritten at a given step is the last symbol previously "written into" the second component that has not been rewritten yet. (If in a step a word $a_1 \ldots a_n$ is written "at the end of" the second component, then we will assume that the symbols $a_1, \ldots a_n$ are written in in this order.)

(ii) For a computation ρ in G, $wdes(trl(\rho))$ closely represents this sort of behavior, where

- a symbol from Σ_2 corresponds to the addition of a nonterminal on the second component,

- a symbol from $\hat{\Sigma}_2$ corresponds to rewriting of a nonterminal on the second component.

(iii) Moreover, in a successful computation every symbol that was introduced on the second component will eventually be rewritten.

From these observations our theorem easily follows. □

Example 1.1 (continued)

$wdes(\sigma)$ is a Dyck word over $\{A, B\} \cup \{\hat{A}, \hat{B}\}$. Its balanced structure is as follows:

$A \quad \hat{A} \quad B \quad A \quad \hat{A} \quad B \quad A \quad \hat{A} \quad \hat{B} \quad A \quad \hat{A} \quad \hat{B}$

FIGURE 1.

□

We would like to conclude this section with the following remark.

Remark. As a matter of fact descriptions of computations translate "directly" trails of computations into words more explicitly connected to the working alphabet (the alphabet of both coordinates) of the cp system considered.

One could also work directly with trails of successful computations. Then Theorem 2.1. would be reformulated as: "$ctb(tsc(G)) = L(G)$"

and Theorem 2.2 would be reformulated as "$tsc(G)$ is a cp language",

where $tsc(G)$ denotes the set of trails of successful computations in G. □

3. THE CHOMSKY-SCHUTZENBERGER THEOREM

As we have observed already, in general the language $wdsc(G)$ does not carry any information about $L(G)$.

In this section we will consider the so-called smooth cp systems for which it is possible to recover the language from the set of weak descriptions (of its

successful computations). Recovering this information and analyzing the form of words in $wdsc(G)$ yields the well-known Chomsky-Schutzenberger theorem for context-free languages (see, e.g., [S]).

We begin by establishing a normal form result for cp systems.

The following result can be obtained by either considering the 2-Greibach normal form for cf grammars and then translating this into the theory of cp systems or by proving a "real time normal form" directly within the theory of cp systems - this turns out to be a nontrivial task (see [EHR1]).

Theorem 3.1. For each cp language K there exists a cp system $G = (G_1, G_2, R)$ such that $K = L(G)$, where moreover

(1) $G_1 = (\{S_1\} \cup \Delta, P_1, S_1, \Delta)$, where

$P_1 \subseteq \{S_1 \rightarrow aS_1 \mid a \in \Delta\} \cup \{S_1 \rightarrow a \mid a \in \Delta\}$,

(2) $G_2 = (\Sigma_2, P_2, S_2)$, where all productions in P_2 are in one of the following three forms:

$A \rightarrow \Lambda$, $A \rightarrow B$ or $A \rightarrow BC$, with $A, B, C \in \Sigma_2$, and

(3) for all $a \in \Delta, \pi_2 \in P_2$ we have

$(S_1 \rightarrow a_1 S_1 , \pi_2) \in R$ if and only if $(S_1 \rightarrow a , \pi_2) \in R$. □

An important property of a cp system G satisfying Theorem 3.1 is that the first component cannot "block" a computation, since

(1) there is only one nonterminal to which every rewrite of G can be applied, and moreover

(2) if $(S_1,S_2) \overset{*}{\underset{G}{\Rightarrow}} (u_1S_1,\Lambda)$ for some $u \in \Delta^*$, then $(S_1,S_2) \overset{*}{\underset{G}{\Rightarrow}} (u,\Lambda)$.

We now define the basic notion of this section.

Definition 3.1. A cp system $G = (G_1,G_2,R)$, where $G_1 = (\Sigma_1,P_1,S_1,\Delta)$ and $G_2 = (\Sigma_2,P_2,S_2)$, is *smooth* if

(1) all productions for S_1 and S_2 are chain productions, S_1 and S_2 do not occur at the right-hand side of any production,

(2) $\Sigma_1 \backslash \Delta$ contains only one symbol besides S_1, say Z,

(3) there exists a weak coding $h : \Sigma_2^* \rightarrow \Delta^*$ such that, for each $(X \rightarrow u\ ,\ A \rightarrow w) \in R$, $pres_{\Delta}(u) = h(A)$, where $h(A) = \Lambda$ if and only if $A = S_2$, and

(4) for every $u \in \Delta \bigcup \{\Lambda\}$ and $\pi_2 \in P_2$, $(Z \rightarrow uZ\ ,\ \pi_2) \in R$ if and only if $(Z \rightarrow u\ ,\ \pi_2) \in R$. □

Smooth cp systems are "universal" in the following sense.

Theorem 3.2. For each cp language K there exists a smooth cp system G such that $K = L(G)$.

Proof.

Let K be a cp language and let $G' = (G_1',G_2',R')$, with $G_1' = (\{S_1\} \bigcup \Delta,P_1',S_1,\Delta)$ and $G_2' = (\Sigma_2',P_2',S_2)$, be a cp system such that $L(G') = K$; we assume that G' satisfies the normal form requirements of

Theorem 3.1.

Let $\Sigma_2 = \{A_a \mid A \in \Sigma_2^{'}, a \in \Delta\}$ and let μ be the coding from Σ_2^{*} into $(\Sigma_2^{'})^{*}$ defined by

$\mu(A_a) = A$, for every $A \in \Sigma_2^{'}$, $a \in \Delta$.

We define a new set R of rewrites as follows (Z is a new symbol).

(1) $(S_1 \rightarrow Z\ ,\ S_2 \rightarrow (S_2)_a) \in R$, for every $a \in \Delta$.

(2) $(Z \rightarrow aZ\ ,\ A_a \rightarrow w) \in R$, if $w \in \Sigma_2^{*}$, $a \in \Delta$ and $(S_1 \rightarrow aS_1\ ,\ A \rightarrow w^{'}) \in R^{'}$ for some $w^{'} \in \mu(w)$.

(3) $(Z \rightarrow a\ ,\ A_a \rightarrow w) \in R$, if $w \in \Sigma_2^{*}$, $a \in \Delta$ and $(S_1 \rightarrow a\ ,\ A \rightarrow w^{'}) \in R^{'}$ for some $w^{'} \in \mu(w)$.

(4) R contains no rewrites other than those defined under (1), (2) and (3) above.

Let $G = (G_1, G_2, R)$ with $G_1 = (\{S_1, Z\} \cup \Delta, P_1, S_1, \Delta)$ and $G_2 = (\Sigma_2 \cup \{S_2\}, P_2, S_2)$ be the cp system with above set of rewrites, where P_1 and P_2 are the sets of productions occurring on the first, respectively second, coordinate of rewrites from R.

It is easily seen that $L(G^{'}) = L(G)$. Moreover, G obviously satisfies the requirements (1), (2) and (4) of the definition of a smooth cp system.

Finally, let h be the weak coding from $(\Sigma_2 \cup \{S_2\})^{*}$ into Δ^{*} defined by $h(S_2) = \Lambda$, and

$h(A_a) = a$, for every $A \in \Sigma_2'$, $a \in \Delta$.

Now, obviously, h is the required weak coding (see Definition 3.1.(3)) and consequently G is smooth. Thus the theorem holds. □

Theorem 3.3. Each cp language K is the image through a weak coding of the intersection of a regular language and a Dyck language.

Proof.

Let K be a cp language and let $G = (G_1, G_2, R)$, where $G_1 = (\Sigma_1, P_1, S_1, \Delta)$ and $G_2 = (\Sigma_2, P_2, S_2)$, be a cp system computing K. By Theorem 3.2. we can assume that G is smooth; let h be the weak coding satisfying the definition of a smooth cp system.

Let H be a cp system generating $wdsc(G)$.

We define the weak coding $g: (\Sigma_2 \cup \hat{\Sigma}_2)^* \rightarrow \Delta^*$ in the following way:

for $a \in \Sigma_2$, $g(a) = \Lambda$ and $g(\hat{a}) = h(a)$.

Then it follows directly from the definition of a smooth cp system that

$g(L(H)) = L(G)$.....(1)

Let Q be the following regular language:

$$Q = S_2 \cdot \{wdes(trl(\pi)) \mid \pi = (S_1 \rightarrow Z\ ,\ \pi_2) \in R\} \cdot$$
$$\cdot \{wdes(trl(\pi)) \mid \pi = (Z \rightarrow u\ ,\ \pi_2) \in R\}^*,$$

where Z is the unique nonterminal in $\Sigma_1 - \{S_1\}$.

To conclude the proof of the theorem it suffices now to prove the following result.

Lemma 3.1. $L(H) = D_{\Sigma_2} \cap Q$.

Proof of Lemma 3.1

By Theorem 2.4. $L(H) = wdsc(G) \subseteq D_{\Sigma_2}$.

It is easily seen that $L(H) \subseteq Q$ (see Definition 2.1. and Definition 2.2.(3)). Hence $L(H) \subseteq D_{\Sigma_2} \cap Q$.

On the other hand let $w \in D_{\Sigma_2} \cap Q$.

By the definition of Q there exists a sequence $\kappa = \pi_1 , ..., \pi_n , n \geq 1$, of rewrites $\pi_i \in R$ such that $w = wdes([S_1 ; S_2]trl(\pi_1) \cdots trl(\pi_n))$.

By property (4) from Definition 3.1. (of a smooth cp system) we may assume that $\pi_n = (Z \rightarrow u_n , \pi_{n,2})$ and for $2 \leq i < n$, $\pi_i = (Z \rightarrow u_i Z , \pi_{i,2})$, where $u_2 , ..., u_n \in \Delta$. Moreover, by property (1) from Definition 3.1, $\pi_1 = (S_1 \rightarrow Z , \pi_{1,2})$.

In order to prove our lemma we will show that $\pi_1 , \pi_2 , ..., \pi_n$ is a successful control sequence in G.

We will consider the effect of κ on each of the two components separately. Let $\pi_i = (\pi_{i,1} , \pi_{i,2})$ for every $1 \leq i \leq n$ and let $\kappa_1 = \pi_{1,1} , ..., \pi_{n,1}$ and $\kappa_2 = \pi_{1,2} , ..., \pi_{n,2}$.

First we observe that κ_1 induces a well defined derivation in G_1 of a word in Δ^* - this follows from the above assumption.

Secondly, we observe that κ_2 yields a derivation in G_2 of the empty word (because $w \in D_{\Sigma_2}$). Hence κ is a successful control sequence in G, w is the weak description of its trail; consequently it is in $L(H)$.

Hence $L(H) \supseteq D_{\Sigma_2} \cap Q$ and the lemma holds. □

Now Theorem 3.3. follows from (1) and the above lemma. □

4. THE EXCHANGE THEOREM

In this section we consider the possibilities of "swapping" pieces of (not necessarily different) computations in cp systems.

Let G be the cp system specified at the beginning of Section 2. Again *we will assume that G is arbitrary, but fixed for the considerations of this section.*

Before we state the theorem of this section we need a lemma and some additional terminology.

Lemma 4.1.

Let $Q = \{trl(\pi_1) \cdots trl(\pi_n) \mid n \geq 1\ ,\ \pi_i = (\pi_{i,1}\ ,\ \pi_{i,2}) \in R$ for every $1 \leq i \leq n$, and $\pi_{1,1}\ ,\ \pi_{2,1}\ ,\ \ldots,\ \pi_{n,1}$ forms the control sequence of a successful derivation in $G_1\}$.

(1) Q is a regular language.

(2) Let $\kappa = \pi_1, ..., \pi_n$, $n \geq 1$, be a sequence of rewrites in R and let $\alpha = trl(\pi_1) \cdots trl(\pi_n)$.

Then κ is a successful control sequence in G if and only if

(i) $\alpha \in Q$, and

(ii) $S_2\ wdes\ (\alpha) \in D_{\Sigma_2}$. □

Proof.

(1) Using the fact that G_1 is a rl grammar, one easily sees that Q is a regular language.

(2) Let $\kappa = \pi_1 , ..., \pi_n$, $n \geq 1$, be a sequence of rewrites from R, where $\pi_i = (\pi_{i,1} , \pi_{i,2})$, $1 \leq i \leq n$. Let $\kappa_1 = \pi_{1,1} , ..., \pi_{n,1}$, $\kappa_2 = \pi_{1,2} , ..., \pi_{n,2}$ and let $\alpha = trl(\pi_1) \cdots trl(\pi_n)$.

(a) If κ is a successful control sequence in G, then κ_1 is a successful control sequence in G_1. Hence $\alpha \in Q$. Moreover, by Theorem 2.4.,

$S_2\ wdes(\alpha) = wdes([S_1 ; S_2]\ trl(\pi_1) \cdots trl(\pi_n)) = wdes(trl(\kappa)) \in D_{\Sigma_2}$.

This proves the "only if" part of the statement.

(b) In the proof of the reverse implication (the "if" part of the statement) we will consider separately the effect of κ on each of the two components (like we have done in the proof of Lemma 3.1.).

$\alpha \in Q$ implies that κ_1 is a successful control sequence in G_1.

If $S_2\ wdes(\alpha) \in D_{\Sigma_2}$, then κ_2 is the control sequence of a derivation in G_2

of the empty word.

Consequently, if both $\alpha \in Q$ and $S_2\ wdes(\alpha) \in D_{\Sigma_2}$, then κ is a successful control sequence in G. Hence the lemma holds. □

Definition 4.1.

(1) A word $\alpha \in \Gamma^*$ is *balanced,* if $wdes(\alpha) \in D_{\Sigma_2}$.

(2) Two non-empty balanced words α and β from Γ^* are *equivalent,* denoted $\alpha \sim \beta$, if $\alpha(1) = \beta(1)$ and $\alpha(|\alpha|) = \beta(|\beta|)$. □

Theorem 4.1. *(Exchange Theorem).* Let ρ_1,ρ_2 be (not necessarily different) successful computations in G, where $trl(\rho_1) = \alpha_1\beta_1\gamma_1$, $trl(\rho_2) = \alpha_2\beta_2\gamma_2$ and $\beta_1 \sim \beta_2$. Let $\omega_{12} = \alpha_1\beta_2\gamma_1$ and $\omega_{21} = \alpha_2\beta_1\gamma_2$. Then there exist unique successful computations ρ_{12} and ρ_{21} in G such that $trl(\rho_{12}) = \omega_{12}$ and $trl(\rho_{21}) = \omega_{21}$.

Proof.

We will only sketch the proof: it closely follows the arguments from the proof of Lemma 3.1..

(a) The equivalence of the exchanged parts β_1 and β_2 assures us that each of ω_{12} and ω_{21} represents a unique sequence of rewrites in G (except for the initial symbol $[S_1\ ;\ S_2]$) as a sequence of their trails; let κ_{12} and κ_{21} be these sequences of rewrites.

(b) Moreover, due to the same fact, the first components of κ_{12} and κ_{21} "behave well" (i.e., like a successful control sequence in G_1) on the first coordinate.

(c) Finally one observes that κ_{12} and κ_{21} yield in G_2 derivations of the empty word: this follows from the fact that the exchange of two balanced subwords does not "unbalance" the computations involved.

Consequently κ_{12} and κ_{21} are successful control sequences in G - observations (b) and (c) above correspond to requirements (i) and (ii) from the statement of Lemma 4.1. Hence our theorem holds. □

Remark. The Exchange Theorem, combined with the observation that $res(\rho) = ctb(trl(\rho))$ allows us not only to exchange equivalent pieces of computations, but also the corresponding pieces of the *results* of the computations.

With the notation as in the statement of Theorem 4.2 we have

$res(\rho_1) = ctb(\alpha_1\beta_1\gamma_1) = ctb(\alpha_1)ctb(\beta_1)ctb(\gamma_1)$,

$res(\rho_2) = ctb(\alpha_2\beta_2\gamma_2) = ctb(\alpha_2)ctb(\beta_2)ctb(\gamma_2)$,

$res(\rho_{12}) = ctb(\omega_{12}) = ctb(\alpha_1\beta_2\gamma_1) = ctb(\alpha_1)ctb(\beta_2)ctb(\gamma_1)$, and

$res(\rho_{21}) = ctb(\alpha_2)ctb(\beta_1)ctb(\gamma_2)$. □

Example 1.1. (continued). Let ρ_1,ρ_2 be successful computations in G with

$cont(\rho_1) = \psi_1,\psi_1,\psi_1,\psi_2,\psi_4,\psi_4,\psi_0$ and

$cont(\rho_2) = \psi_1,\psi_2,\psi_3,\psi_1,\psi_1,\psi_2,\psi_4,\psi_0$ respectively.

Let $\alpha_1 = [A;X][\psi_1,0][\psi_1,1]$,

$\beta_1 = [\psi_1,2][\psi_1,0][\psi_1,1][\psi_1,2][\psi_1,0][\psi_1,1][\psi_1,2][\psi_2,0][\psi_4,0][\psi_4,0]$,

$\gamma_1 = [\psi_0,0]$

and let

$\alpha_2 = [A;X][\psi_1,0][\psi_1,1][\psi_1,2][\psi_2,0][\psi_3,0][\psi_3,1][\psi_1,0][\psi_1,1]$,

$\beta_2 = [\psi_1,2][\psi_1,0][\psi_1,1][\psi_1,2][\psi_2,0][\psi_4,0]$,

$\gamma_2 = [\psi_0,0]$.

Then $trl(\rho_1) = \alpha_1\beta_1\gamma_1$ and $trl(\rho_2) = \alpha_2\beta_2\gamma_2$.

Hence

$res(\rho_1) = ctb(\alpha_1)ctb(\beta_1)ctb(\gamma_1) = a\cdot aabcc\cdot c$

and

$res(\rho_2) = ctb(\gamma_2)ctb(\beta_2)ctb(\gamma_2) = abba\cdot abc\cdot c$.

β_1 and β_2 are both balanced, since

$wdes(\beta_1) = A\hat{A}BA\hat{A}BA\hat{A}\hat{B}\hat{B}$ and $wdes(\beta_2) = A\hat{A}BA\hat{A}\hat{B}$

are Dyck words over $\{A,B\} \bigcup \{\hat{A}\ ,\ \hat{B}\}$.

Moreover, β_1 and β_2 are equivalent. Hence there exist successful computations

ρ_{12} and ρ_{21} with

$trl(\rho_{12}) = \alpha_1\beta_2\gamma_1$ and $trl(\rho_{21}) = \alpha_2\beta_1\gamma_2$.

Obviously we can take

$cont(\rho_{12}) = \psi_1,\psi_1,\psi_2,\psi_4,\psi_0$ and

$cont(\rho_{21}) = \psi_1,\psi_2,\psi_3,\psi_1,\psi_1,\psi_1,\psi_2,\psi_4,\psi_4,\psi_0$.

Then $res(\rho_{12}) = ctb(\alpha_1)ctb(\beta_2)ctb(\gamma_1) = a \cdot abc \cdot c$

and $res(\rho_{21}) = ctb(\alpha_2)ctb(\beta_1)ctb(\gamma_2) = abba \cdot aabcc \cdot c$. □

DISCUSSION

In this paper we have provided an analysis of computations in cp systems. Such an analysis seems to be useful for building up the theory of ects systems - since cp systems form a very natural and rather simple submodel of the general ects model they form a natural starting point for investigating the theory of ects systems. Moreover, cp systems correspond quite closely to push-down automata and so (we believe that) the analysis of cp systems provided in this paper forms a new and convenient way of looking at context-free languages.

The exchange theorem given in Section 4 to provides a convenient basis for proving "pumping type" results for cp languages - this is demonstrated in [EHR2] and [EHR3].

The main aim of our paper was to express (formalize) in the framework of cp systems a number of basic properties of push-down computations; we hope that in doing this we have demonstrated the "naturalness" and the elegance of the cp systems approach. Once this is done one can move to the in-depth formal investigation of cp systems (as basis for the theory of push-down automata and context-free languages). The usefulness of our paper for this purpose is already demonstrated in [EHR2], [EHR3], and [EHR4], where known results about push-down automata and context-free languages are proved in a new

fashion; new results are given as well.

ACKNOWLEDGEMENTS

The authors are very obliged to the Leiden TIC group (IJ.J. Aalbersberg, J. Engelfriet and H.C.M. Kleijn) for very useful comments concerning the first draft of this paper. The first and the third author gratefully acknowledge the support of NSF grant MCS 83-05245.

REFERENCES

[EHR1] Ehrenfeucht, A., Hoogeboom, H.J. and Rozenberg, G., "Real-time coordinated pair systems," Tech. Rep. CU-CS-259-83, Dept. of Computer Science, University of Colorado at Boulder, 1983.

[EHR2] Ehrenfeucht, A., Hoogeboom, H.J. and Rozenberg, G., "Coordinated pair systems; Part I: Dyck words and classical pumping", Tech. Rep. CU-CS-275-84, Dept. of Computer Science, University of Colorado at Boulder, 1984.

[EHR3] Ehrenfeucht, A., Hoogeboom, H.J. and Rozenberg, G., "Coordinated pair systems; Part II: Sparse structure of Dyck words and Ogden's Lemma", Tech. Rep. CU-CS-276-84, Dept. of Computer Science, University of Colorado at Boulder, 1984.

[EHR4] Ehrenfeucht, A., Hoogeboom, H.J. and Rozenberg, G., "On the active and full records of the use of memory in right-boundary grammars and push-down machines", Tech. Rep. CU-CS-311-85, Dept. of Computer Science, University of Colorado at Boulder, 1985.

[H] Harrison, M., *Introduction to formal language theory,* Addison-Wesley, Reading, Massachusetts, 1978.

[K] Kleijn, H.C.M., "Selective substitution grammars based on context-free productions," Ph.D. Thesis, Department of Applied Mathematics and Computer Science, University of Leiden, The Netherlands, 1983.

[KR] Kleijn, H.C.M. and Rozenberg, G., "Context-free like restrictions on selective rewriting," *Theoretical Computer Science,* v. 16, 237-269, 1981.

[R1] Rozenberg, G., "Selective substitution grammars (towards a framework for rewriting systems), Part I: Definitions and Examples," *Elektron, Informationsverarbeit. Kybernetik,* v. 13, 455-463, 1977.

[R2] Rozenberg, G., "On coordinated selective substitutions: Towards a unified theory of grammars and machines," *Theoretical Computer Science,* v. 37, 31-50, 1985.

[S] Salomaa, A., *Formal languages,* Academic Press, London-New York, 1973.