A SIMPLE PROOF OF THE CORRESPONDENCE THEOREM*

by

Peter J. Denning
Jack B. Dennis

MASSACHUSETTS INSTITUTE OF TECHNOLOGY
Cambridge, Massachusetts

*Work reported herein was supported in part by Project MAC, an M.I.T. research project sponsored by the Advanced Project Research Agency, Department of Defense, under Office of Naval Research Contract Number Nonr-4102(01).

Introduction

Post's correspondence theorem[1] is a powerful result in computability theory; with it many interesting and important problems can be shown undecidable by simple, elegant proofs. Unfortunately the only published proof for the correspondence theorem is long, involved, and difficult to understand. As a result, many workers in the field do not fully grasp the elegance of Post's result. In this paper we present a simple proof of the theorem. Our aim is to set forth a development that makes pedagogical sense and gives the student an intuitive appreciation for the meaning of the theorem.

We start discussing how unsolvability of the Instantaneous Description Problem for Turing machines follows from the well-known unsolvability of the halting problem. On this basis we prove the main result using a domino concept. After reviewing some important properties of context-free grammars, we present simple proofs for:

1. Context-free language intersection problems.
2. Context-free grammar and language ambiguity problems.

Even though these theorems are well-known, they are included here to demonstrate how powerful a tool Post's result is. An important ambiguity problem--"does there exist an unambiguous grammar that generates a given language?" --has been an open question until recently. Ginsburg and Ullian[2], for example, have a proof which unfortunately is lost amid a flurry of other results.

Turing Machines

A Turing machine has a deterministic finite-state control unit which controls the action of a read/write head on its tape. We imagine that the tape is oriented horizontally, extending to infinity in either direction. The tape is divided into squares, each of which contains a single symbol.

Definition. A Turing machine is a set

$$M = \{Q, T, I, q_I\}$$

where

- Q is a finite set of internal states for M's control unit;
- T is a finite set of tape symbols;
- I is a finite set of instructions specifying the behavior or M's control unit;
- $q_I \in Q$ is the initial state.

There is a special blank tape symbol called sharp ($\# \notin T$).

An instruction in I is of the form

<label>] <action>

A <label> is a pair (q, t) where $q \in Q$ and $t \in T$. An <action> is one of {left, right, print, halt}. The instructions have the following meanings.

(q, t)] <u>left</u>, p	Move head left one square, observe the symbol u ∈ (T ∪ {#}). The next instruction is labelled (p, u).
(q, t)] <u>right</u>, p	Move head right one square, observe the symbol u ∈ (T ∪ {#}). The next instruction is labelled (p, u).
(q, t)] <u>print</u> u, p	Print symbol u ∈ T in current square, without moving the head. The next instruction is labelled (p, u).
(q, t)] <u>halt</u>, p	Halt, entering state p.

We imagine that the tape is initially filled with a special symbol, sharp (# ∉ T), everywhere except in a finite region which contains a string of symbols called the <u>initial tape</u>. After a single action either a symbol has been altered and the head has not moved, or else the head has moved and no symbol has been altered. Using this idea we can describe the action of a Turing machine quite compactly.

<u>Definition</u>. An instantaneous description, or configuration, of a Turing machine $M = \{Q, T, I, q_I\}$ is a string

$$\omega\ q\ t\ \varphi$$

where ω, φ are (possibly empty) strings in T^*, $q \in Q$, and $t \in T$.

An instantaneous description has the following meaning. M is a state q scanning symbol t. To the left of the head a string ω appears on the tape; and to the right a string φ appears. It is clear that when M executes

an instruction, there is a simple change in the instantaneous description, so that the effect of an instruction can also be represented as follows.

Instruction of M	Change in Configuration
(q, t)] left, p	$\omega\ u\ q\ t\ \varphi \rightarrow \omega\ p\ u\ t\ \varphi$
(q, t)] right, p	$\omega\ q\ t\ \varphi \rightarrow \omega\ t\ p\ \varphi$
(q, t)] print u, p	$\omega\ q\ t\ \varphi \rightarrow \omega\ p\ u\ \varphi$
(q, t)] halt p	$\omega\ q\ t\ \varphi \rightarrow \omega\ p\ t\ \varphi$

It should be clear that these changes in configuration can be regarded as a set of context-sensitive rewriting rules, and that a Turing machine computation can be represented by a sequence of applications of such rules*. We shall denote the local change in configuration by the rules $X \rightarrow X'$, which are:

Instructions of M	Local change $X \rightarrow X'$
(q, t)] left, p	$u\ q\ t \rightarrow p\ u\ t$
(q, t)] right, p	$q\ t \rightarrow t\ p$
(q, t)] print u, p	$q\ t \rightarrow p\ u$
(q, t)] halt, p	$q\ t \rightarrow p\ t$

*To clarify our notation: If C is a configuration and C' is the configuration resulting from the execution of an instruction, we can describe the change by the ordered pair (C, C') or equivalently by the "rule" $C \rightarrow C'$.

There is one such X-rule for each instruction of M. The total change in configuration is denoted by the infinite set of rules $Y \rightarrow Y'$, where $Y = \omega X \varphi$, $Y' = \omega X' \varphi$, ω and φ are arbitrary strings in T^*, and $X \rightarrow X'$ is any local change rule.

It may be the case that ω is empty and M's action is to move left, scanning a sharp (#) on the previously unused portion of tape; or that φ is empty and M's action is to move right, scanning a sharp in that unused portion of tape. So two extra Y-rules are needed to "lengthen the tape":

$$X \varphi \rightarrow \# X \varphi$$

$$\omega X \rightarrow \omega X \#$$

This is equivalent to adding the X-rule

$$q\ t \rightarrow t\ p\ \#$$

to the existing X-rules for each right instruction of M, and the X-rule

$$q\ t \rightarrow p\ \#\ t$$

to the existing X-rules for each left instruction of M. It should be clear that the number of X-rules [ordered pairs (X,X')] is finite since M's instruction set is finite.

M's computation can be represented by a sequence of configurations of the form:

$$\alpha, Y_1, Y_2, \cdots, Y_n, \cdots$$

where

$$\alpha = \omega_0 q_1 t_0 \varphi_0$$

is the starting configuration, and each pair (Y_n, Y_{n+1}) is related by a Y-rule

$$Y_n \rightarrow Y_{n+1}$$

The Instantaneous Description Problem

Next we show that it is undecidable whether a given Turing machine M, starting from a given configuration α, ever enters a specified configuration β. This is known as the Instantaneous Description Problem. Let $P(M, \alpha, \beta)$ be the proposition:

$$P(M, \alpha, \beta) = \begin{cases} 1 & \text{if M ever reaches configuration } \beta \text{ when started in } \alpha \\ 0 & \text{otherwise} \end{cases}$$

Theorem 1. It is undecidable whether $P(M, \alpha, \beta) = 1$.

Proof: We show that a decision procedure for $P(M, \alpha, \beta)$ can be used to solve the Halting Problem.

Let H be any Turing machine and C an initial configuration for H. The Halting Problem is: Does H ever halt when started in C? This question is known to be undecidable[3].

Suppose a decision procedure for $P(M, \alpha, \beta)$ exists. The following specialized universal Turing machine U can be constructed. U is presented initially with an encoded description D(H) of the instructions of H, together with an encoded description D(C) of H's initial configuration. If, while simulating H, U finds that H halts, U erases D(H), D(C), and any other records from its tape, and enters a special halting state q_h. That is, U enters configuration $\beta' = q_h$ iff H halts. Letting α' be the known initial configuration of U (e.g., $\alpha' = q_I \# D(H) \# D(C)$), we have $P(H, \alpha', \beta') = 1$ iff H halts.

It follows that $P(M, \alpha, \beta)$ is in general undecidable.

(QED)

Because this result is widely known, we have omitted the details of the encodings D(H) and D(C) and consequently how we may be assured that U can erase its tape whenever H halts. [see, for example, reference 4].

The Correspondence Problem

Let V be a finite alphabet. Let A and B be ordered, finite sets of non-empty strings on V, each having the same number of elements:

$$A = \{\alpha_1, \alpha_2, \ldots, \alpha_n\}$$

$$B = \{\beta_1, \beta_2, \ldots, \beta_n\}$$

The Correspondence Problem is this: Does there exist a sequence of subscripts $i_1, i_2, \ldots, i_m$ $1 \leq i_k \leq n$, $k = 1,2,\ldots, m$, such that

$$\alpha_{i_1} \alpha_{i_2} \ldots \alpha_{i_m} = \beta_{i_1} \beta_{i_2} \ldots \beta_{i_m}$$

An equivalent statement, which we shall use extensively, is this: Let

$$D = \left\{\binom{\alpha_1}{\beta_1}, \binom{\alpha_2}{\beta_2}, \cdots, \binom{\alpha_n}{\beta_n}\right\} \qquad \begin{matrix}\alpha_i \in A \\ \beta_i \in B\end{matrix}$$

Is there an ordering of elements of D (chosen with repetition allowed) such that the upper row is identical to the lower?

$$\binom{\alpha_{i_1}}{B_{i_1}} \binom{\alpha_{i_2}}{\beta_{i_2}} \cdots \binom{\alpha_{i_n}}{\beta_{i_n}} \qquad \begin{matrix}\leftarrow \text{ upper row} \\ \leftarrow \text{ lower row}\end{matrix}$$

We regard the elements of D as <u>domino-types</u>, each with a string α_i inscribed on the top and the corresponding string β_i on its bottom. Posing the Correspondence Problem is the same as asking whether there is a linear arrangement of dominoes with the top row identical to the bottom. Of course there must be an infinite supply of dominoes, but only a finite number of the different types specified by D.

Whenever there is an ordering of D-elements satisfying the Correspondence Problem, we shall say that D possesses a <u>C-ordering</u>. As we will see below, if D possesses one C-ordering it possesses an infinite number of C-orderings.

<u>Example</u>.

Before proceeding, an example is instructive. Let

$$D = \left\{ \begin{pmatrix} ba \\ a \end{pmatrix}, \begin{pmatrix} ab \\ abbb \end{pmatrix}, \begin{pmatrix} bb \\ aa \end{pmatrix}, \begin{pmatrix} aabb \\ bbb \end{pmatrix} \right\} = \{d_1, d_2, d_3, d_4\}$$

The ordering $d_1\ d_2\ d_2\ d_4$ yields these upper and lower strings:

upper: baababaabb

lower: aabbbabbbbbb

which are not the same. However the ordering $d_2\ d_3\ d_4\ d_1$ yields:

upper: abbbaabbba

lower: abbbaabbba

which are identical. Thus $d_2\ d_3\ d_4\ d_1$ is a C-ordering. Note also that

$$\{(d_2\ d_3\ d_4\ d_1)^k \mid k \geq 1\}$$

is an infinite set of C-orderings.

The Correspondence Theorem

Theorem 2. Given a finite set D of domino types, it is undecidable whether a C-ordering exists on D.

Proof: We show how a decision procedure for determining the existence of a C-ordering can be used to decide the instantaneous description problem, P(M, α, β). Suppose that M, α, β are given. Let

$$(1)\quad D_\infty(M, \alpha, \beta) = \left\{ \begin{pmatrix} * \\ * \alpha \end{pmatrix}, \begin{pmatrix} \beta * \\ * \end{pmatrix} \right\} \cup \left\{ \begin{pmatrix} Y * \\ * Y' \end{pmatrix} \right\}$$

where * is a special symbol not in $(Q \cup T)$, and $Y \rightarrow Y'$ are the Y-rules of M. $D_\infty(M, \alpha, \beta)$ has an infinite number of elements because there are an infinite number of Y-rules each having the form

$$\omega X \varphi \rightarrow \omega X' \varphi \qquad \omega, \varphi \in T^*$$

That is, even though there are finitely many X - rules, one for each instruction of M, there are infinitely many possible strings ω and φ, giving rise to infinitely many distinct elements in D_∞ (M, α, β). We will show first that a C-ordering on $D_\infty(M, \alpha, \beta)$ exists iff P(M, α, β) = 1. Then we show how each element of $D_\infty(M, \alpha, \beta)$ can be generated by a unique sequence of elements chosen from a finite set D(M, α, β), which will complete the proof.

Suppose M traces the sequence of configurations

$$(2)\quad \alpha, Y_1, Y_2, \ldots, Y_n, \beta$$

Then $D_\infty(M, \alpha, \beta)$ has the C-ordering

$$(3)\quad \binom{*}{*\ \alpha}\binom{Y_1\ *}{*\ Y_1'}\binom{Y_2\ *}{*\ Y_2'}\cdots\binom{Y_n\ *}{*\ Y_n'}\binom{\beta\ *}{*}$$

To see that (3) is a valid C-ordering, note that the special symbol * appears exactly (n + 2) times in both rows. Therefore the substrings caught between successive pairs of * must be identical:

$$\begin{aligned} Y_1 &= \alpha \\ Y_2 &= Y_1' \\ Y_3 &= Y_2' \\ &\vdots \\ Y_n &= Y_{n-1}' \\ \beta &= Y_n \end{aligned} \qquad (4)$$

Next, note that every C-ordering must begin with the α-element and end with the β-element, for otherwise the symbols * cannot match. Each pair (Y_i, Y_i') is related by the a single action of M, via the rule $Y_i \rightarrow Y_i'$. Thus if (3) is a C-ordering, M traces sequence (2). Therefore the upper row in (3) is identical to the lower row iff M traces sequence (2). With $D_\infty(M, \alpha, \beta)$, a procedure that decides the existence of a C-ordering also decides $P(M, \alpha, \beta)$.

The final step in the proof is to show how to replace the infinite set $D_\infty(M, \alpha, \beta)$ with a finite set $D(M, \alpha, \beta)$. $D_\infty(M, \alpha, \beta)$ is infinite because there are infinitely many Y-rules . However there are finitely many (say r) X-rules. If we think of D's elements as dominoes, the solution becomes apparent: add new domino types to generate the ω and φ parts of $Y = \omega X \varphi$.

Let

$$D(\alpha, \beta) = \left\{ \begin{pmatrix} * \\ * & \alpha \end{pmatrix}, \begin{pmatrix} \beta & * \\ * \end{pmatrix} \right\}$$

$$D(t) = \left\{ \begin{pmatrix} t & \Delta \\ * & t \end{pmatrix}, \begin{pmatrix} t & \Delta \\ \Delta & t \end{pmatrix}, \begin{pmatrix} t & \square \\ \square & t \end{pmatrix}, \begin{pmatrix} t & * \\ \square & t \end{pmatrix} \right\} \quad \text{where } t \in T$$

$$D(\lambda) = \left\{ \begin{pmatrix} \Delta \\ * \end{pmatrix}, \begin{pmatrix} * \\ \square \end{pmatrix} \right\}$$

$$D(M) = \left\{ \begin{pmatrix} X_1 & \square \\ \Delta & X_1' \end{pmatrix}, \cdots, \begin{pmatrix} X_r & \square \\ \Delta & X_r' \end{pmatrix} \right\}$$

and $$D(M, \alpha, \beta) = D(\alpha, \beta) \cup \left(\bigcup_{t \in T} D(t) \right) \cup D(\lambda) \cup D(M)$$

Again, *, Δ, □ are special symbols not in (Q ∪ T). $D(\alpha, \beta)$ contains elements for M's initial configuration α, and M's final configuration β. For each $t \in T$, $D(t)$ contains building blocks for the ω and φ portions of Y. The Δ - types are used to construct ω, □- types to construct φ. $D(M)$ contains building blocks for each of the r X-rules of M. $D(\lambda)$ are elements for empty tape regions; that is, in case ω or φ is empty. The symbols Δ and □ insure that no configuration can be constructed that does not contain a local change.

Each element $\begin{pmatrix} Y & * \\ * & Y' \end{pmatrix} \in D_\infty(M, \alpha, \beta)$

is generated now by a sequence of elements chosen from $D(M, \alpha, \beta)$:

$$\begin{pmatrix} Y & * \\ * & Y' \end{pmatrix} \sim \underbrace{\begin{pmatrix} t_1 & \Delta \\ * & t_1 \end{pmatrix} \begin{pmatrix} t_2 & \Delta \\ \Delta & t_2 \end{pmatrix} \cdots \begin{pmatrix} t_k & \Delta \\ \Delta & t_k \end{pmatrix}}_{Y \qquad \omega = t_1\, t_2 \cdots t_k} \underbrace{\begin{pmatrix} X & \square \\ \Delta & X' \end{pmatrix}}_{X} \underbrace{\begin{pmatrix} u_1 & \square \\ \square & u_1 \end{pmatrix} \begin{pmatrix} u_2 & \square \\ \square & u_2 \end{pmatrix} \cdots \begin{pmatrix} u_\ell & * \\ \square & u_\ell \end{pmatrix}}_{\varphi = u_1\, u_2 \cdots u_\ell}$$

It should be clear that each element in $D_\infty(M, \alpha, \beta)$ is generated by a unique sequence of elements from $D(M, \alpha, \beta)$. Therefore a C-ordering on $D_\infty(M, \alpha, \beta)$ exists iff there is a C-ordering on $D(M, \alpha, \beta)$. Hence a C-ordering on $D(M, \alpha, \beta)$ exists iff $P(M, \alpha, \beta) = 1$.

The undecidability of a C-ordering on arbitrary D follows at once from the undecidability of $P(M, \alpha, \beta)$.

(QED)

The proof just given makes use of domino alphabets of arbitrary finite size. Since any n-symbol alphabet can be coded appropriately into a 2-symbol alphabet, the theorem is true even in the case that V contains just two symbols.

Undecidable Questions Concerning Context-free Languages

Using the Correspondence Theorem we can prove several important results about context-free languages, simply and elegantly. The problems we consider are:

1. Intersection Problems

   A. Is $L_1 \cap L_2 = \phi$?

   B. Is $L_1 \cap L_2$ infinite?

   C. Is $L_1 \cap L_2$ regular?

   D. Is $L_1 \cap L_2$ context-free?

   E. Given that $L_1 \cap L_2$ is context-free, can a grammar be constructed to generate $L_1 \cap L_2$?

2. Ambiguity Problems

   A. Is an arbitrary grammar ambiguous?

   B. Is an arbitrary language inherently ambiguous?

      [That is, does there exist an unambiguous grammar for the language?]

Before proving these questions undecidable, we define our terms.

Definition. A context-free grammar is a set $G = [N, T, P, \Sigma]$

where

N is a finite, non-terminal alphabet
T is a finite, terminal alphabet
} $N \cap T = \phi$

P is a finite set of ordered pairs (A, ω) where $A \in (N \cup [\Sigma])$, $\omega \in (N \cup T)^*$. If $(A, \omega) \in P$ we write $A \rightarrow \omega$, and say that $A \rightarrow \omega$ is a production of G.

Σ is the sentence symbol, from which each sentence is derived.

The language L(G) generated by G is the set of all strings derivable from Σ in G. With each string $\omega \in L(G)$ is associated one or more derivation trees, whose nodes correspond to applications of productions used to derive ω from Σ in G. If each $\omega \in L(G)$ has a unique derivation tree, G is unambiguous. If a language has no unambiguous grammar, it is inherently ambiguous.

The union of context-free languages is context-free. Let L_1 and L_2 be two context-free languages, generated by G_1 and G_2. Relabel non-terminals so that $N_1 \cap N_2 = \phi$. Then

$$G_1 \cup G_2 = \{N_1 \cup N_2, T_1 \cup T_2, P_1 \cup P_2, \Sigma\}$$

generates $L_1 \cup L_2$. Furthermore, suppose G_1 and G_2 are unambiguous. Then, if $L(G_1) \cap L(G_2) = \phi$, $G_1 \cup G_2$is unambiguous; if $L(G_1) \cap L(G_2) \neq \phi$, $G_1 \cup G_2$ is ambiguous since each string in the intersection has two derivations.

From now on the terms "language" and "grammar" always refer to context-free languages and grammars, unless otherwise stated.

For simplicity in the following, take the elements of $D(M, \alpha, \beta)$ of equation (5) and recode them using the alphabet {0, 1}. Let these coded elements comprise the set E:

$$(6) \qquad E = \left\{ \binom{\rho_1}{\sigma_1}, \binom{\rho_2}{\sigma_2}, \cdots, \binom{\rho_n}{\sigma_n} \right\}$$

Each ρ_i is a coded representation on $\{0, 1\}$ for each upper element of $D(M, \alpha, \beta)$, and each σ_i is the coded representation for the corresponding lower element. A C-ordering on E exists iff $P(M, \alpha, \beta) = 1$. Furthermore, suppose

$$(7) \qquad \xi = \rho_{i_1} \rho_{i_2} \cdots \rho_{i_m}$$

is a C-ordering on E; then ρ_{i_1} is the coded representation of M's initial configuration α, and ρ_{i_m} is the coded representation of M's final configuration β. Since M is deterministic, the C-ordering ξ is unique; that is, there is one and only one way for M to reach β from α. It follows that

$$(8) \qquad \{\xi^k \mid k \geq 1\}$$

are all the C-orderings on E.

Three of the following theorems (Theorems 5,6,and 7) depend on the fact that ξ is the unique shortest C-ordering on E. The others do not depend on this, so in their proofs E may be regarded as an arbitrary set of domino types set up for any correspondence problem.

From now on we will use the following notation for the proposition that a set E has a C-ordering:

$$CP(E) = \begin{cases} 1 & \text{if E has a C ordering} \\ 0 & \text{otherwise} \end{cases}$$

Theorem 2 tells CP(E) is undecidable for arbitrary E.

Let R and S be the languages generated by G(R) and G(S) respectively:

G(R):

$$\Sigma \rightarrow R$$
$$R \rightarrow (1)\ R\ \rho_1$$
$$\vdots$$
$$R \rightarrow (n)\ R\ \rho_n$$
$$R \rightarrow (1)\ *\ \rho_1$$
$$\vdots$$
$$R \rightarrow (n)\ *\ \rho_n$$

G(S):

$$\Sigma \rightarrow S$$
$$S \rightarrow (1)\ S\ \sigma_1$$
$$\vdots$$
$$S \rightarrow (n)\ S\ \sigma_n$$
$$S \rightarrow (1)\ *\ \sigma_1$$
$$\vdots$$
$$S \rightarrow (n)\ *\ \sigma_n$$

Strings in R and S are of the form

$$(m) \ldots (2)\ (1)\ *\ \varphi_1 \varphi_2 \cdots \varphi_m$$

where φ_i are upper (lower) elements of E. These strings consist of a sequence of upper (lower) E-elements preceded by the corresponding subscript sequence. The symbol * is a special center-marking symbol.

Comparing with equations (7) and (8) we have

$$(9) \qquad R \cap S = \begin{cases} \Phi \text{ if there is no C-ordering on E} & [CP(E) = 0] \\ \{\eta^k * \xi^k \mid k \geq \} \text{ otherwise} & [CP(E) = 1] \end{cases}$$

where η is the subscript sequence corresponding to ξ. Note that the form of $R \cap S$ depends on the fact that E has a unique shortest C-ordering, ξ.

Theorem 3. For arbitrary languages L_1, L_2 it is undecidable whether $L_1 \cap L_2 = \Phi$.

Proof. Let $L_1 = R$ and $L_2 = S$. From equation (9), $R \cap S = \Phi$ iff $CP(E) = 0$. Any decision procedure for determining whether $L_1 \cap L_2$ is empty would, if applied to $R \cap S$, also decide CP(E), which is impossible.

(QED)

Theorem 4. For arbitrary languages L_1, L_2 it is undecidable whether $L_1 \cap L_2$ is infinite.

Proof: Let $L_1 = R$ and $L_2 = S$. From equation (9), $R \cap S$ contains infinitely many strings iff CP(E) = 1. Any decision procedure for determining whether $L_1 \cap L_2$ is infinite would, if applied to $R \cap S$, also decide CP(E) which is impossible.

(QED)

Theorem 5. For arbitrary languages L_1, L_2 it is undecidable whether $L_1 \cap L_2$ is a regular set.

Proof: From equation (9) we see that $R \cap S$ is regular ($= \phi$) iff CP(E) = 0, and is non-regular iff CP(E) = 1. Any decision procedure for determining whether $R \cap S$ is regular would, if applied to $R \cap S$, also decide CP(E), which is impossible.

(QED)

Let $\mathcal{R}$ and $\mathcal{S}$ be the languages generated by $G(\mathcal{R})$ and $G(\mathcal{S})$ respectively:

$G(\mathcal{R})$:

$$\begin{aligned}
\Sigma &\rightarrow R * Q \\
R &\rightarrow \rho_1 R (1) \\
&\vdots \\
R &\rightarrow \rho_n R (n) \\
R &\rightarrow \rho_1 * (1) \\
&\vdots \\
R &\rightarrow \rho_n * (n) \\
Q &\rightarrow \rho_1 Q \\
&\vdots \\
Q &\rightarrow \rho_n Q \\
Q &\rightarrow \rho_1 \\
&\vdots \\
Q &\rightarrow \rho_n
\end{aligned}$$

$G(\mathcal{S})$:

$$\begin{aligned}
\Sigma &\rightarrow T * S \\
S &\rightarrow (1) S \sigma_1 \\
&\vdots \\
S &\rightarrow (n) S \sigma_n \\
S &\rightarrow (1) * \sigma_1 \\
&\vdots \\
S &\rightarrow (n) * \sigma_n \\
T &\rightarrow \sigma_1 T \\
&\vdots \\
T &\rightarrow \sigma_n T \\
T &\rightarrow \sigma_1 \\
&\vdots \\
T &\rightarrow \sigma_n
\end{aligned}$$

$G(\mathcal{R})$ generates strings of the form

$$\varphi_1 \varphi_2 \cdots \varphi_m * (m) \cdots (2)(1) * \psi_1 \psi_2 \cdots \psi_p$$

where φ_i, ψ_i are upper elements of E. Similarly $G(\mathcal{S})$ generates strings

$$\psi_1 \psi_2 \cdots \psi_p * (m) \cdots (2)(1) * \varphi_1 \varphi_2 \cdots \varphi_m$$

where φ_i, ψ_i are lower elements of E. Comparing with equations (7) and (8) we have

$$(10) \qquad \mathcal{R} \cap \mathcal{S} = \begin{cases} \phi & \text{if } CP(E) = 0 \\ \{\xi^k * \eta^k * \xi^k \mid k \geq 1\} & \text{if } CP(E) = 1 \end{cases}$$

where η is the subscript sequence corresponding to ξ.

Theorem 6. For arbitrary languages L_1, L_2 it is undecidable whether $L_1 \cap L_2$ is a context-free language.

Proof: Let $L_1 = \mathcal{R}$ and $L_2 = \mathcal{S}$. $\mathcal{R} \cap \mathcal{S}$ is regular (a subset of context-free) iff there is no C-ordering on E. $\mathcal{R} \cap \mathcal{S}$ is non-context free iff there is a C-ordering on E. Any decision procedure for determining whether $L_1 \cap L_2$ is context-free would, if applied to $\mathcal{R} \cap \mathcal{S}$, also decide CP(E), which is impossible.

(QED)

Although we cannot decide if $L_1 \cap L_2$ is context-free, we might ask this question: Suppose we already know that $L_1 \cap L_2$ is context-free; can we construct a grammar G which generates $L_1 \cap L_2$? The next theorem allays any suspicions we might have.

Theorem 7. Let G_1 and G_2 be grammars and suppose it is known that $L(G_1) \cap L(G_2)$ is a language. There is no procedure for constructing a grammar that generates $L(G_1) \cap L(G_2)$.

Proof. Suppose there is a procedure for constructing a grammar G that generates $L(G_1) \cap L(G_2)$, provided that it is known that $L(G_1) \cap L(G_2)$ is a language. Let $G_1 = G(R)$ and $G_2 = G(S)$; equation (9) tells us that $R \cap S$ is always a language, so construct G that generates $R \cap S$. Now it is known that there is a decision procedure for determining whether an arbitrary grammar generates any strings at all, that is, whether $L(G) = \Phi$ (See, for example, Ginsburg[5]). Applying this procedure tells whether or not $R \cap S = \Phi$ and so decides CP(E), which is impossible.

(QED)

We turn attention to ambiguity questions.

Theorem 8. It is undecidable whether an arbitrary grammar is ambiguous.

Proof: The grammars G(R) and G(S) are unambiguous. Form the grammar $G(R) \cup G(S)$. If $R \cap S \neq \Phi$, any string in $R \cap S$ has two derivations, one in G(R), the other in G(S); such a string has two derivations in $G(R) \cup G(S)$. $G(R) \cup G(S)$ is unambiguous if $R \cap S = \Phi$. Any decision procedure for determining whether an arbitrary grammar is ambiguous would, if applied to $G(R) \cup G(S)$, also decide CP(E), which is impossible.

(QED)

Before proving the last theorem, we need a lemma.

Lemma. Let ω be any string not on the alphabet $\{a, b, c\}$. Then the language

$$L(\omega) = \{ \omega\, a^k\, b^k\, c^* \mid k \geq 1\} \cup \{ \omega\, a^*\, b^k\, c^k \mid k \geq 1\}$$

is inherently ambiguous.

Proof: Suppose $L(\omega)$ is unambiguous. Since $L(\omega) = \omega L'$, where

$$L' = \{a^k\, b^k\, c^* \mid k \geq 1\} \cup \{a^*\, b^k\, c^k \mid k \geq 1\}$$

it follows that L' is unambiguous. But L' is known to be inherently ambiguous [see Ginsburg[2]]. This establishes a contradiction.

(QED)

Theorem 9. For an arbitrary language L, it is undecidable whether L is inherently ambiguous. Equivalently, for an arbitrary grammar G, it is undecidable whether there exists an unambiguous grammar G' such that $L(G') = L(G)$.

Proof: From equation (9), $R \cap S \neq \Phi$ iff $CP(E) = 1$. R and S are on an alphabet V, so let a, b, c, be symbols not in V. Form the set

$$L(R, S) = \{R\, a^k\, b^k\, c^* \mid k \geq 1\} \cup \{S\, a^*\, b^k\, c^k \mid k \geq 1\}.$$

It is clear that L(R, S) is a language, and if $R \cap S = \Phi$, L(R, S) is unambiguous. We claim that, if $R \cap S \neq \Phi$, L(R, S) is inherently ambiguous. So suppose $R \cap S \neq \Phi$.

For any language X and regular set Y, $X \cap Y$ is a language and there is an effective procedure for constructing a grammar for $X \cap Y$. Further, if X is unambiguous, so is $X \cap Y$. [see Ginsburg and Ullian[6]].

Suppose L(R, S) is unambiguous. Therefore, for any $\omega \in (R \cap S)$

$$L(R, S) \cap \{\omega\, a^* b^* c^*\}$$

$$= \left(\{R\, a^k b^k c^* \mid k \geq 1\} \cup \{S\, a^* b^k c^k \mid k \geq 1\} \right) \cap \{\omega\, a^* b^* c^*\}$$

$$= \{\omega\, a^k b^k c^* \mid k \geq 1\} \cup \{\omega\, a^* b^k c^k \mid k \geq 1\}$$

$$= L(\omega)$$

where $L(\omega)$ is defined in the Lemma. Since L(R, S) is assumed unambiguous, $L(R, S) \cap \{\omega\, a^* b^* c^*\} = L(\omega)$ is unambiguous. But the lemma shows $L(\omega)$ is inherently ambiguous. Therefore, for each $\omega \in (R \cap S)$, L(R, S) is inherently ambiguous. Any decision procedure for determining if L is inherently ambiguous would, if applied to L(R, S), also decide CP(E), which is impossible.

(QED)

References

[1] E. L. Post. "A Variant of a Recursively Unsolvable Problem." Bulletin of Am. Math. Soc. Vol. 52, 1946, pp 264-268.

[2] S. Ginsburg and J. Ullian. "Ambiguity in Context-Free Languages." JACM, Vol. 13, No. 1, Jan. 1966.

[3] B. A. Trakhtenbrot. Algorithms and Automatic Computing Machines. D. C. Heath and Co., 1963, pp 86-88.

[4] Ibid. pp 80-85

[5] S. Ginsburg. The Mathematical Theory of Context-Free Languages. McGraw-Hill, 1966. p 211

[6] Ibid. p 117