STEPHAN HEILBRUNNER

An algorithm for the solution of fixed-point equations for infinite words

RAIRO. Informatique théorique, tome 14, nº 2 (1980), p. 131-141
<http://www.numdam.org/item?id=ITA_1980__14_2_131_0>

R.A.I.R.O. Informatique théorique/Theoretical Informatics
(vol. 14, n° 2, 1980, p. 131 à 141)

AN ALGORITHM FOR THE SOLUTION OF FIXED-POINT EQUATIONS FOR INFINITE WORDS (*)

by Stephan HEILBRUNNER (¹)

Communicated by M. NIVAT

Abstract. — *The solutions of fixed-point equations for generalized (infinite) words have previously been obtained as frontiers of infinite trees. An algorithm is given which computes an explicit solution using a shuffle operation on words. This solution can be proven equal to the previous one so that the structure of the frontiers of infinite trees is completely analyzed for those context-free grammars which contain exactly one production rule for each syntactic variable.*

Résumé. — *On sait que les solutions d'équations au point fixe pour les mots généralisés (infinis) peuvent être obtenues comme frontières d'arbres infinis. Dans cet article, on donne un algorithme qui calcule une solution explicite grâce à une opération de mélange (shuffle) sur les mots. On peut montrer que cette solution est égale à la précédente, de sorte que la structure des frontières d'arbres infinis est complètement élucidée dans le cas des grammaires algébriques ayant exactement une règle de réécriture pour chaque variable syntactique.*

1. INTRODUCTION

Infinite words, i. e. sequences of elements of some alphabet indexed by the set of natural numbers have been considered in a number of papers (in particular [7, 8, 9]). A more general type of infinite words, i. e. "sequences" of elements of some alphabet indexed by an arbitrary, linearly ordered set were obtained in an very natural way by considering the frontiers of infinite derivation trees for context-free grammars. These generalized words will be called (infinite) *strings* in this paper.

Strings and systems of fixed-point equations for strings were introduced and solved in [6] and [1]. The solutions were given in terms of frontiers of infinite trees which were defined as the limit of a fairly complicated substitution process. The problem of finding simple operations on strings such that the solutions can be computed and represented in an finitary way was posed and partially solved in

(*) Received December 1978, revised July 1979.
(¹) Hochschule der Bundeswehr München, Neubiberg.

[1] by introducing "regular" operations on strings in connection with "quasi-rational" systems of equations. In this paper we shall add a *shuffle* operation and present an algorithm which computes an explicit solution for all systems of fixed-point equations for strings.

The theory of strings turns out to be a generalization of the theory of order types which has a long tradition in set theory [3, 4, 10]. The definition of order types and strings poses a foundational problem in that it involves generalization on all sets. There are two ways of circumventing this difficulty. One way is to augment set theory by axioms characterizing order types and strings. The other way is to chose a canonical system of representatives for order types and strings within the system of sets generated by the usual axioms of set theory. Details are given in [2]. We would like to point out that an intensive search in the order type literature did not unfold a theory of solving fixed-point equations for order types.

2. INFINITE STRINGS

Let W be a nonempty set. Given two ordered (²) sets D and E and two functions $R: D \to W$ and $S: E \to W$ we say that R and S are equivalent iff there is a bijective, order preserving mapping $H: D \to E$ such that $R = S \circ H$. Consider all the functions which map ordered sets into W. Note that the ordered sets may be different for different functions. The equivalence classes of such functions are called *strings* with respect to W. Strings are called *order types* if W consists of one element. For foundational problems connected with this definition recall the remarks in the introduction. By string (R) we denote the equivalence class of the function R. For each $w \in W$ we define the function R_w: $\{1\} \to W$ by $R_w(1) = w$ and call $V = \{\text{string}(R_w): w \in W\}$ the vocabulary. The empty string is denoted by ε. It is generated by the function $R: \varnothing \to W$, where $\varnothing$ is the empty set.

We turn to the concatenation of strings and define the *addition* of ordered sets first. Suppose that $\{D_a: a \in A\}$ is a set of disjoint ordered sets where A is ordered, too. The sum

$$\sum_{a \in A} D_a \quad \text{is the set} \quad \bigcup_{a \in A} D_a,$$

ordered such that the order in each D_a is preserved and such that $D_a < D_{\hat{a}}$ whenever $a < \hat{a}$. Given strings r_a with representatives R_a: $D_a \to W$ we define the function

$$R: \sum_{a \in A} D_a \to W \quad \text{by } R(d) = R_a(d),$$

(²) Order in this paper means linear (total) order.

for all $a \in A$ and $d \in D_a$. *Concatenation* is then defined by

$$\sum_{a \in A} r_a = \text{string } (R).$$

The set A is called the *argument* of the concatenation. If the argument ist $\{1, 2, \ldots, n\}$ we also write $r_1 r_2 \ldots r_n$ to denote concatenation.

The following rules can be derived from the definitions: (i) Concatenation is well defined for all ordered sets of strings. (ii) Concatenation is associative. (iii) If B is an ordered set and s_b a string for each $b \in B$ then the existence of a bijective, order preserving mapping $H: A \to B$ such that $r_a = s_{H(a)}$ for all $a \in A$ implies

$$\sum_{a \in A} r_a = \sum_{b \in B} s_b.$$

Moreover, (iv) the reserve implication is true, if $r_a, s_b \in V$ for all a and b. (v) Any string r can be written as $\sum_{a \in A} r_a$ with $r_a \in V$ for suitably chosen A. Because of (v) we suppress W and talk of *strings over* V.

Simple examples of concatenations with infinite arguments are the operations of *repetition* defined now. Let $\mathbb{Z}$ be the set of integers with the natural order. Let r be some string and define $r_i = r$ for all $i \in \mathbb{Z}$ and

$$r^{\omega} = \sum_{i \geq 0} r_i \quad \text{and} \quad r^{\omega^*} = \sum_{i \leq 0} r_i.$$

Obviously, $r^{\omega} = \sum_{i>0} r_i$ and $r^{\omega^*} = \sum_{i<0} r_i$ so that we obtain the following table.

Rule	$r^{\omega} = rr^{\omega}$	$r^{\omega^*} = r^{\omega^*} r$	$r^{\omega} s^{\omega^*} = rr^{\omega} s^{\omega^*} s$
Equation	$x = rx$	$x = xr$	$x = rxs$
Solution	r^{ω}	r^{ω^*}	$r^{\omega} s^{\omega^*}$

There are other obvious rules for repetition

$$s(ts)^{\omega} = (st)^{\omega}, \qquad (st)^{\omega^*} s = (ts)^{\omega^*},$$
$$(tt)^{\omega} = t^{\omega}, \qquad (tt)^{\omega^*} = t^{\omega^*}.$$

3. SHUFFLING

Let r and t be some strings and consider the equation $x = xrxtx$. For any solution of this equation we derive successively

$$x = xrxtx;$$

$$x = xrxtxrxrxtxtxrxtx;$$

$$x = xrxtxrxrxtxtxrxtxrxrxtxrxrxtxtxrxtx\ txrxtxrxrxtxtxrxtx$$

$$\ldots$$

The trick which solves these equations is to assign fixed places to the r's and t's as soon as they appear in some equation and insert the r's and t's emerging in the following equations in proper places in between. This can be done if the r's and t's are placed on the real line at rational points because there are sufficiently many numbers between any two rational numbers. In this way we obtain a dense and perfectly shuffled string of r's and t's.

The following theorem guarantees the unique existence of strings of this kind. Recall that an ordered set is *open* if it has neither a first nor a last element.

THEOREM 3.1 : *Let $T \neq \emptyset$ be a finite or countable set of nonempty strings. Then, there is an ordered set A and there are strings $r_a \in T$ for each $a \in A$ such that:*

(i) *A is open and countable;*

(ii) *$a_1 \in A \wedge a_2 \in A \wedge a_1 < a_2$ implies $\forall\, t \in T$, $\exists\, a \in A$, $r_a = t \wedge a_1 < a < a_2$.*

Moreover, if there is another set B with strings $s_b \in T$ for each $b \in B$ satisfying (i) *and* (ii) *with A replaced by B then*

$$\sum_{a \in A} r_a = \sum_{b \in B} s_b.$$

Proof: We define A as a subset of the real open interval (0, 1). Let p_t be a different prime number for each $t \in T$ and define

$$A = \left\{ \frac{a}{p_t^i} \,\middle|\, i > 0 \wedge t \in T \wedge a < p_t^i \right\} \qquad \text{and} \qquad r_q = t \quad \text{for } q = \frac{a}{p_t^i}.$$

It is not hard to verify that A satisfies (i) and (ii).

The proof of the second claim is due to [11] if the notions used there are interpreted appropriately. Details are given in [5]. For the sake of completeness we outline the proof.

Let A and B be given accordingly. Both sets are countable so that we may assume

$$A = \{a_1, a_2, a_3, \ldots\} \qquad \text{and} \qquad B = \{b_1, b_2, b_3, \ldots\}.$$

We define infinite sequences of sets A_0, A_1, ... and B_0, B_1, ... Let $A_0 = B_0 = \emptyset$. For $i = 1, 2, \ldots$ do the following. If i is even take the element, say b, with the smallest index which is in $B \setminus B_i$. Determine an element $a \in A \setminus A_i$ such that $r_a = s_b$ and

$$\left| \{ b' \mid b' < b \wedge b' \in B_i \} \right| = \left| \{ a' \mid a' < a \wedge a \in A_i \} \right|.$$

Associate a with b, and define $A_{i+1} = A_i \setminus \{a\}$ and $B_{i+1} = B_i \setminus \{b\}$.

If i is odd start with an element in $A \setminus A_i$ and proceed in the analagous way.

The set of pairs obtained by this procedure is an order preserving bijection h from A onto B with $r_a = s_{h(a)}$.

DEFINITION : For any finite set T of strings we define the *shuffle* of T by

$$T^{\eta} = \sum_{a \in A} r_a,$$

where A and r_a are chosen according to theorem 3.1 if $T \neq \emptyset$ and by $T^{\eta} = \varepsilon$ if $T = \emptyset$.

Our theorem implies that T^{η} is well defined and implies the following rules·
(i) $T^{\eta} T^{\eta} = T^{\eta}$;
(ii) $\forall t \in T,\ T^{\eta} t T^{\eta} = T^{\eta}$;
(iii) $(R \cup S)^{\eta} = T^{\eta}$ whenever

$$R \subseteqq T \wedge \emptyset \neq S \subseteqq (T \cup \{\varepsilon\})\ T^{\eta} (T \cup \{\varepsilon\}).$$

Special cases of (iii) are the equations

$$\{T^{\eta}\}^{\eta} = \{t T^{\eta}\}^{\eta} = \{T^{\eta} t\}^{\eta} = T^{\eta} \quad \text{for each } t \in T.$$

Returning to the equation $x = xrxtx$ we see that $\{r,\ t\}^{\eta}$ is a solution because rule (ii) allows the computation

$$\{r,\ t\}^{\eta} r \{r,\ t\}^{\eta} t \{r,\ t\}^{\eta} = \{r,\ t\}^{\eta} t \{r,\ t\}^{\eta} = \{r,\ t\}^{\eta}.$$

As another example for an application of rule (ii) consider the equation $x = uxrxv$. We compute

$$u(u^{\omega} \{v^{\omega^*} r u^{\omega}\}^{\eta} v^{\omega^*})\ r (u^{\omega} \{v^{\omega^*} r u^{\omega}\}^{\eta} v^{\omega^*})\ v$$
$$= (u u^{\omega})\ \{v^{\omega^*} r u^{\omega}\}^{\eta} (v^{\omega^*} r u^{\omega})\ \{v^{\omega^*} r u^{\omega}\}^{\eta} (v^{\omega^*} v)$$
$$= u^{\omega} \{v^{\omega^*} r u^{\omega}\}^{\eta} v^{\omega^*},$$

so that $u^{\omega}\ \{v^{\omega^*}\ r u^{\omega}\}^{\eta}\ v^{\omega^*}$ solves the equation $x = uxrxv$.

Finally we note that rule (iii) allows the solution of rather complicated equations. As an example consider $x = u \{u,\ x\}^{\eta} v$ which contains a dense string of infinitely many occurences of the unknown x. It is easy to verify that the string $u \{u,\ v\}^{\eta} v$ solves this equation.

4. THE ALGORITHM

In this section we present the algorithm to compute an explicit and finite representation of a solution of a system of equations of the form

$$x_1 = u_1, \qquad x_2 = u_2, \qquad \ldots, \qquad x_n = u_n,$$

where $X = \{x_1, \ldots, x_n\}$ is an alphabet of unknowns and where the strings u_i are strings over $V \bigcup X$. An equation is called *finite* if it contains only a finite number of occurences of unknowns.

A *regular expression* is an expression consisting of concatenations, repetitions, shuffles, and denotations for strings. Note that our regular expressions are extensions of the ones used in [1]. Every regular expression denotes a string in the natural way. Examples of regular expressions are the expressions used at the end of section 3.

THEOREM 4.1 : *There is an algorithm (described informally below) which computes for any system of finite equations a system of regular expressions solving the equations.*

The complete and formal proof of this theorem is a rather technical matter (*see* [5]). Hence, we shall restrict ourselves to an outline.

The first step in the algorithm is to draw the dependency graph for the given system of equations. The nodes of this graph are the unknowns of the system. There is a directed arc from node x_i to node x_j iff x_j appears in the right side u_i of the equation $x_i = u_i$.

Example: Consider the system Σ and its dependency graph.

$$\begin{aligned}
\Sigma:\ x_1 &= ux_3x_2x_6x_2u,\\
x_2 &= ux_1v,\\
x_3 &= x_4x_5x_3,\\
x_4 &= x_1x_4x_1,\\
x_5 &= vvv,\\
x_6 &= v,\\
x_7 &= vux_2uvx_7.
\end{aligned}$$

$$\begin{tikzcd}
x_7 \arrow[loop above] \arrow[r] & x_2 \arrow[d, shift left] & \\
x_6 & x_1 \arrow[l] \arrow[u, shift left] \arrow[d] & x_4 \arrow[loop above] \arrow[l] \\
 & x_3 \arrow[loop left] \arrow[ur] \arrow[r] & x_5
\end{tikzcd}$$

Figure 1. — Dependency graph for Σ.

This dependency graph may contain sinks, i. e. nodes of out-degree zero. The sinks correspond to trivial equations $x_i = u_i$ whose right side contains no unknowns. The next step in the algorithm is to eliminate the sinks from the other equations by straightforward substitution. This transformation yields a system whose dependency graph has no nodes of out-degree zero.

Example (continued): Elimination of sinks for Σ yields Σ_1.

$$\begin{aligned}\Sigma_1:\ x_1&=ux_3x_2vx_2u,\\ x_2&=ux_1v,\\ x_3&=x_4vvvx_3,\\ x_4&=x_1x_4x_1,\\ x_7&=vux_2uvx_7.\end{aligned}$$

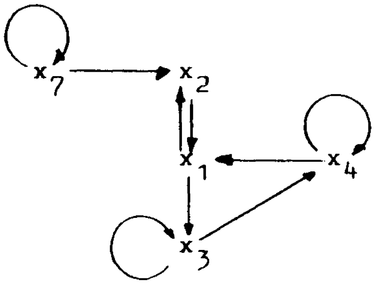

Figure 2. – Dependency graph for Σ_1.

Let us call a set X of nodes in the dependency graph a closed component iff (i) X is a component, and (ii) no node in X has an outgoing arc leaving X. Recall that X is a component iff it is a maximal subset of nodes such that there is a path connecting any two nodes of X. In the example the set $\{x_1, x_2, x_3, x_4\}$ is a closed component for Σ_1. The set $\{x_7\}$ is a component but not a closed component for Σ_1.

The next step in the algorithm is to solve the subsystems of equations corresponding to the closed components in the dependency graph. Note that there must be closed components if there are no sinks. Let

$$x_1=u_1x_{i_1}w_1x_{j_1}v_1, \qquad \ldots, \qquad x_n=u_nx_{i_n}w_nx_{j_n}v_n,$$

be a subsystem belonging to a closed component. We assume that the right sides are written such that the strings $u_1, \ldots, u_n$ and $v_1, \ldots, v_n$ do not contain unknowns. Note that every closed component allows such a representation. From this system we derive two new systems:

the initial system:

$$y_1=u_1y_{i_1}, \qquad \ldots, \qquad y_n=u_ny_{i_n}$$

and the final system:

$$z_1=z_{j_1}v_1, \qquad \ldots, \qquad z_n=z_{j_n}v_n,$$

where the $y_1, \ldots, y_n, z_1, \ldots, z_n$ are new unknowns.

Example (continued): We consider the subsystem $\{x_1, x_2, x_3, x_4\}$ and derive the following systems:

initial system:

$$y_1=uy_3, \qquad y_2=uy_1, \qquad y_3=y_4, \qquad y_4=y_1,$$

final system:

$$z_1=z_2u, \qquad z_2=z_1v, \qquad z_3=z_3, \qquad z_4=z_1.$$

The initial and final systems are quasi-rational in the sense of [1] and admit simple solutions. They are obtained as follows. Draw the dependency graph for the initial system and label the arc belonging to the equation $y_j = u_j y_k$ by u_j. Proceed in an analogous fashion for the final system.

Example (*continued*):

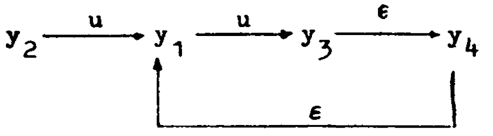

Figure 3. – Dependency graph for the initial system.

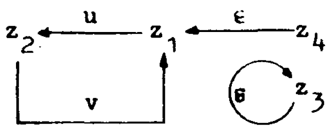

Figure 4. – Dependency graph for the final system.

For each node in these graphs there is exactly one path of infinite length because each node has out-degree one. The next step is to describe the infinite strings associated with these paths in terms of concatenation and repetition. The paths for the final system have to be reversed. This process yields solutions $y_1 = r_1, \ldots, y_n = r_n$ and $z_1 = s_1, \ldots, z_n = s_n$ for the initial and final systems.

Example (*continued*); Let $\ldots{}^*$ denote reversal

$$y_1 = (u\varepsilon\varepsilon)^\omega = u^\omega =: r_1, \qquad y_2 = uy_1 = uu^\omega = u^\omega =: r_2,$$
$$y_3 = (\varepsilon\varepsilon u)^\omega = u^\omega =: r_3, \qquad y_4 = (\varepsilon u \varepsilon)^\omega = u^\omega =: r_4,$$
$$z_1 = ((uv)^\omega)^* = (vu)^{\omega^*} =: s_1, \qquad z_2 = ((vu)^\omega)^* = (uv)^{\omega^*} =: s_2,$$
$$z_3 = (\varepsilon^\omega)^* = \varepsilon =: s_3, \qquad z_4 = (\varepsilon(uv)^\omega)^* = (vu)^{\omega^*} =: s_4.$$

At this point we arrive at the crucial step of the algorithm. Let T be the set of all strings of the form $s_i t r_j$ such that there is an equation $x_k = u_k$ of the closed subsystem whose right side u_k contains the substring $x_i t x_j$, where t is free of unknowns. Then

$$x_1 = r_1 T^\eta s_1, \qquad x_2 = r_2 T^\eta s_2, \qquad \ldots, \qquad x_n = r_n T^\eta s_n,$$

solves subsystem.

Example (*continued*): For the considered closed subsystem we find

$$T = \{ s_3 r_2, s_2 v r_2, s_4 vvv r_3, s_1 r_4, s_4 r_1 \}$$
$$= \{ u^\omega, (uv)^{\omega^*} vu^\omega, (vu)^{\omega^*} vvvu^\omega, (vu)^{\omega^*} u^\omega, (vu)^{\omega^*} u^\omega \}$$
$$= \{ u^\omega, (uv)^{\omega^*} vu^\omega, (uv)^{\omega^*} vvu^\omega, (vu)^{\omega^*} u^\omega \}$$

and

$$x_1 = u^\omega T^\eta (vu)^{\omega^*}, \qquad x_2 = u^\omega T^\eta (uv)^{\omega^*},$$
$$x_3 = u^\omega T^\eta, \qquad x_4 = u^\omega T^\eta (vu)^{\omega^*}.$$

We verify the first equation in the example

$$\begin{aligned} ux_3 x_2 vx_2 u &= u(u^\omega T^\eta)(u^\omega T^\eta (uv)^{\omega^*}) v (u^\omega T^\eta (uv)^{\omega^*}) u \\ &= (uu^\omega) T^\eta (u^\omega) T^\eta ((uv)^{\omega^*} vu^\omega) T^\eta ((uv)^{\omega^*} u) \\ &= u^\omega T^\eta ((uv)^{\omega^*} u) = u^\omega T^\eta (vu)^{\omega^*} = x_1. \end{aligned}$$

In order to prove that we obtain in this way a solution of the closed subsystem we consider an arbitrary equation:

$$x_k = u_k x_{k_1} t_2 x_{k_2} t_3 x_{k_3} \dots t_m x_{k_m} v_k,$$

of the subsystem and compute

$$u_k x_{k_1} t_2 \dots t_m x_{k_m} v_k = u_k r_{k_1} T^\eta s_{k_1} t_2 r_{k_2} T^\eta \dots t_m r_{k_m} T^\eta s_{k_m} v_k.$$

By definition of T we have

$$s_{k_1} t_2 r_{k_2}, \quad \dots, \quad s_{k_{m-1}} t_m r_{k_m} \in T,$$

so that

$$T^\eta s_{k_1} t_2 r_{k_2} T^\eta \dots T^\eta s_{k_{m-1}} t_m r_{k_m} T^\eta = T^\eta.$$

Moreover, recall that

$$r_k, r_{k_1} \quad \text{solve } y_k = u_k y_{k_1} \quad \text{so that } r_k = u_k r_{k_1}$$

and

$$s_k, s_{k_m} \quad \text{solve } z_k = z_{k_m} v_k \quad \text{so that } s_k = s_{k_m} v_k.$$

We arrive at

$$u_k x_{k_1} t_2 \dots t_m x_{k_m} v_k = r_k T^\eta s_k = x_k,$$

which had to be shown.

The remaining steps of the algorithm are quite simple. We remove the equations of the closed subsystems and eliminate their unknowns in the remaining equations by simple substitution. At this point the algorithm either finds sinks or closed components and continues as described above until all the equations are solved.

Example (*continued*): Elimination in Σ_1 of the solved subsystem yields the equation

$$x_7 = vuu^\omega T^\eta (uv)^{\omega^*} uvx_7 = vu^\omega T^\eta (uv)^{\omega^*} x_7,$$

which is solved by the string

$$x_7 = (vu^\omega T^\eta (uv)^{\omega^*})^\omega.$$

This completes the solution of the original system Σ.

5. OTHER SOLUTIONS

Unfortunately, theorem 4.1. does not say how to test for the equality of solutions of different systems. The problem is equivalent to finding a normal form for regular expressions and a set of rules whose application transforms them into normal form. We conjecture that our set of rules for the string operations defined in this paper is complete in the sense that it is powerful enough to transform different regular expressions which denote the same string into each other. If this conjecture could be proven it should not be too hard to find a normal form.

Note that rule (iii) for the shuffle operator is needed for the simplification of the solutions of our equations. For this purpose consider the system $x_1 = x_1 u x_1$, $x_2 = u x_1$, $x_3 = x_3 x_2 x_3$ with the solution $x_1 = u^\eta$, $x_2 = u u^\eta$, $x_3 = (u u^\eta)^\eta$, where rule (iii) of section 3 is needed to find $x_3 = x_1$.

Every non-trivial system of equations has an uncountable number of countable solutions. This is a simple consequence of the following three facts. (i) Any equation satisfied by T^η is satisfied by $(T \cup \{ t \})^\eta$ for arbitrary t. (ii) There is an uncountable number of countable ordinal numbers. (iii) if t_1 and t_2 are different ordinal numbers not in T then $(T \cup \{ t_1 \})^\eta \neq (T \cup \{ t_2 \})^\eta$.

This contradicts a claim made in [1], p. 327.

There is no straightforward semi-ordering of the different solutions. For this purpose consider the equation $x = xux$. It has the two solutions u^η and $\{ u, uu \}^\eta$, among others. Obviously, u^η is the "simpler" solution because it is "contained" (in the natural way) in $\{ u, uu \}^\eta$. However, u^η is known to be a universal order type [3] which contains every countable order type. This statement is proven by a non-symmetric version of the argument used in the proof of theorem 3.1. Hence, $\{ u, uu \}^\eta$ is contained in u^η so that u^η is not minimal with respect to containment. Nevertheless, a sense of minimality has been given to u^η in [1] by using the approach to the solution of systems of equations described below.

Another method of solving fixed-point equations has been put forth [1, 5, 6]. It constructs the solution as frontiers of infinite derivation trees for the context-free grammars associated in the straightforward way with every system of fixed-point equations. A long and tedious proof contained in [5] shows that the solution obtained in this way agrees with the solution obtained by the method presented in this paper. Note that this statement gives a complete analysis of frontiers of infinite trees for those context-free grammars which contain exactly one production rule for each syntactic variable.

REFERENCES

1. B. Courcelle, *Frontiers of Infinite Trees*, R.A.I.R.O., Informatique théorique, Vol. 12, 1978, pp. 319-337.

2. A. FRAENKEL, Y. BAR-HILLEL, A. LEVY and VAN DALEN, *Foundations of Set Theory*, 2nd rev. edition, North-Holland Publ. Co., Amsterdam, 1973.
3. F. HAUSDORFF, *Grundzüge einer Theorie der geordneten Mengen*, Math. Annalen, Vol. 65, 1908, pp. 435-505.
4. F. HAUSDORFF, *Grundzüge der Mengenlehre*, Leipzig, 1914.
5. S. HEILBRUNNER, *Gleichungssysteme für Zeichenreihen*, Technische Universität München, Bericht, No. 7311, 1973.
6. S. HEILBRUNNER, *Das Problem der »unendlichen Modi« in Algol 68*, Lecture Notes in Computer Science, Vol. 26, 1975, pp. 131-139, Springer-Verlag.
7. R. MCNAUGHTON, *Testing and Generating Infinite Sequences by a Finite Automaton*, Information and Control, Vol. 9, 1966, pp. 521-530.
8. M. NIVAT, *Mots infinis engendrés par une grammaire algébrique*, R.A.I.R.O., Informatique théorique, Vol. 11, 1977, pp. 311-327.
9. M. NIVAT, *Sur les ensembles des mots infinis engendrés par une grammaire algébrique*, R.A.I.R.O., Informatique théorique, Vol. 12, 1978, pp. 259-278.
10. W. SIERPINSKI, *Cardinal and Ordinal Numbers*, Warsaw, 1965.
11. Th. SKOLEM, *Logisch-kombinatorische Untersuchungen über die Erfüllbarkeit oder Beweisbarkeit mathematischer Sätze nebst einem Theoreme über dichte Mengen*, Skrifter utgit av Videnskapselskapet i Kristiania, 1. Mathematisk-Naturvidenskabelig Klasse, 1. Bind, No. 4, 1920, p. 4-36.