Artificial Intelligence Project--RLE and MIT Computation Center
Memo 44--

A Simple Direct Proof of Post's Normal Form Theorem

by Marvin Minsky

The theorem proved in this note is the Normal Form Theorem proved in Post's 1943 paper, "Formal Reductions of the General Combinatorial Decision Problem". We have long felt that this result is one of the most beautiful in mathematics. The fact that any formal systems can be reduced to Post canonical systems with a single axiom and productions of the restricted form

$$\sigma\alpha \rightarrow \alpha\tau$$

is in itself a remarkable discovery, and even more so when we learn that this was found in 1921, long before the formalization of metamathematics became so popular. Post's proof of the theorem is quite readable. Our proof is somewhat simpler, and quite a bit shorter. Its brevity is a result of doing the necessary operations with less concern about the order in which they are done. Some clarity is gained because this technique requires fewer auxiliary symbols. Some clarity is lost because of the superposition of operations and, accordingly, we resort to illustrating the process by a detailed example (which brings the writing back to roughly its original length). Still, the techniques used here are somewhat different and give, perhaps, a better feeling for why the theorem is true.

The paper concludes with a modest contribution--a very simple "Universal Canonical Extension" based on the same techniques. We end up with a "Universal Canonical Extension" with three letters (minimal in a sense) and four productions (minimal ?).

A Simple Direct Proof of Post's Normal Form Theorem

We first prove the theorem for systems having single-antecedent productions. Suppose that we have a Post Canonical System P for which a typical production has the form

$$\underbrace{\sigma_0\alpha_1\sigma_1\alpha_2 \cdots \alpha_n\sigma_n}_{\text{antecedent}} \rightarrow \underbrace{\tau_0\alpha_{i_1}\tau_1\alpha_{i_2} \cdots \alpha_{i_m}\tau_m}_{\text{consequent}} \qquad (\pi)$$

where each variable string α_{i_k} is one of the α_i and σ_0, ..., σ_n and τ_0, ..., τ_m are given fixed strings. This production will apply to any string which has the form of the antecedent--that is, any string which begins with σ_0, contains non-overlapping occurences of σ_1, σ_2, ... σ_{n-1} (in that order) and ends with σ_n. We will construct a new Normal system, an extension P* of P, by replacing each production π by a set of new productions using some new symbols. The task for the system P* is (i) to verify that a string S can be so analysed as an antecedent of π and then (ii) to assemble a consequent string composed of the τ-strings with the proper strings for α_{i_1}, ..., α_{i_m} inserted in the correct positions. The problem is to do all this using only normal productions--productions of the form

$$\sigma\alpha \rightarrow \alpha\tau$$

The difficulty is that normal productions can "look" only at the beginning of a string; they cannot examine a string's interior to see, e. g., if it contains an occurrence of σ_1.

We can evade this limitation by "rotating" the string so that successive symbols are brought up to the beginning. The following productions will thus test whether, in an initial string $T\alpha$, the unknown string α has the required antecedent form. Suppose the letters of the original alphabet are a_1, a_2, ..., a_r. We introduce new letters T_1, T_2, ..., T_n and two more, Z and Y. (We use upper-case letters for all new symbols used in the extension P*.)

$$T\sigma_0\alpha \rightarrow \alpha Z T_1$$

$$a_j\alpha \rightarrow \alpha a_j \qquad (j = 1,\ldots, r)$$

$$T_i a_j\alpha \rightarrow \alpha a_j T_i \qquad (j = 1,\ldots,r \quad i = 1,\ldots,n)$$

$$T_i\sigma_i\alpha \rightarrow \alpha T_{i+1} \qquad (i = 1,\ldots,n-1)$$

$$Z\alpha \rightarrow \alpha Z$$

$$T_n \sigma_n Z\alpha \rightarrow \alpha Y$$

To see how this works, we study an example. Suppose that π is

$$a_3\alpha_1 a_2 a_5 \alpha_2 a_1 a_3 \alpha_3 a_4 \rightarrow a_2 a_1 \alpha_3 a_2 a_2 \alpha_3 a_1 a_5,$$

that is,

$\sigma_0 = a_3$ $\quad \tau_0 = a_2 a_1$ $\quad \alpha_{i_1} = \alpha_3$ $\quad n = 3$

$\sigma_1 = a_2 a_5$ $\quad \tau_1 = a_2 a_2$ $\quad \alpha_{i_2} = \alpha_3$ $\quad m = 2$

$\sigma_2 = a_1 a_3$ $\quad \tau_2 = a_1 a_5$

$\sigma_3 = a_4$

Suppose that the actual string S is, in fact,

$$a_3 \boxed{a_1 a_2} a_2 a_5 \boxed{a_4 a_2} a_1 a_3 \boxed{a_2 a_1} a_4,$$

which is in the form of the antecedent of π. Then the following strings are produced:

$T\ a_3 a_1 a_2 a_2 a_5 a_4 a_2 a_1 a_3 a_2 a_1 a_4$

$a_1 a_2 a_2 a_5 a_4 a_2 a_1 a_3 a_2 a_1 a_4 Z\ T_1$

$a_2 a_2 a_5 a_4 \ldots a_1 a_4 Z\ T_1 a_1$

$a_2 a_5 a_4 \ldots a_1 a_4 Z\ T_1 a_1 a_2$

$\ldots\ldots\ldots\ldots\ldots\ldots$

$Z\ T_1 a_1 a_2 a_2 a_5 a_4 a_2 a_1 a_3 a_2 a_1 a_4$

$T_1 a_1 a_2 a_2 \ldots\ldots a_4 Z$

$a_2 a_2 \ldots\ldots a_4 Z\ a_1 T_1$

$\ldots\ldots\ldots\ldots\ldots\ldots$

$T_1 a_2 a_2 a_5 a_4 \ldots\ldots a_4 Z\ a_1$

$a_2 a_5 a_4 \ldots\ldots a_4 Z\ a_1 a_2 T_1$

$\ldots\ldots\ldots\ldots\ldots$

$T_1 a_2 a_5 a_4 \ldots\ldots a_4 Z\ a_1 a_2 \cdot$

Now there is a choice, for the system can apply either

$$T_1 a_2 \alpha \rightarrow \alpha a_2 T_1$$

or

$$T_1 a_2 a_5 \alpha \rightarrow \alpha T_2$$

Thus two strings are produced;

$$a_5 a_4 \ldots\ldots a_4 Z\ a_1 a_2 a_2 T_1$$

and

$$a_4 \ldots\ldots a_4 Z\ a_1 a_2 T_2 \cdot$$

The first string is ultimately doomed, because there will never again be an opportunity to eliminate the symbol T_1 from it--so it can never yield a string in the original alphabet of P. But the second string can continue on to

$$T_2a_4a_2a_1a_3a_2a_1a_4Z\ a_1a_2$$
$$a_2a_1a_3a_2a_1a_4Z\ a_1a_2a_4T_2$$
$$\cdots\cdots\cdots\cdots$$
$$T_2a_2a_1a_3a_2a_1a_4Z\ a_1a_2a_4$$
$$a_1a_3a_2a_1a_4Z\ a_1a_2a_4a_2T_2$$
$$\cdots\cdots\cdots\cdots$$
$$T_2\underline{a_1a_3}a_2a_1a_4Z\ a_1a_2a_4a_2.$$

Again there is a choice, producing

$$a_3a_2a_1a_4Z\ a_1a_2a_4a_2a_1T_2$$

and

$$a_2a_1a_4Z\ a_1a_2a_4a_2T_3$$

Again the first string is doomed, since T_2 will never be eliminated, but the second string leads to

$$T_3a_2a_1a_4Z\ a_1a_2a_4a_2$$
$$a_1a_4Z\ a_1a_2a_4a_2a_2T_3$$
$$\cdots\cdots\cdots\cdots$$
$$T_3a_1a_4Z\ a_1a_2a_4a_2a_2$$
$$a_4Z\ a_1a_2a_4a_2a_2a_1T_3$$
$$\cdots\cdots\cdots\cdots$$
$$T_3a_4Z\ a_1a_2a_4a_2a_2a_1$$
$$a_1a_2a_4a_2a_2a_1Y$$

A string containing a Y can be produced <u>only</u> if the original string had the required antecedent form! This shows how we can compound normal productions to analyse into the interior of a string.

We have shown how normal productions can be used to determine whether or not a string has the <u>antecedent</u> form. But we cannot use our result to construct the <u>consequent</u>, because the resulting string (containing 'Y') has the α-strings "run together". That is, we have destroyed information needed to show how the string

$$\underline{a_1a_2}\underline{a_2a_4}\underline{a_2a_1}$$

is composed of three strings α_1, α_2, and α_3. To correct this we present

a revised (and final) version of the productions above.

Productions for the Normal Extension

The complete method requires a new set of symbols

$$A_1^1,\ A_2^1,\ \ldots\ A_r^1$$
$$A_1^2,\ A_2^2,\ \ldots\ A_r^2$$
$$\ldots\ \ldots$$
$$A_1^n,\ A_2^n,\ \ldots\ A_r^n.$$

These are used to keep track of the contents of the strings $\alpha_1, \alpha_2, \ldots, \alpha_n$. The new productions are

$$T\sigma_0\alpha \rightarrow \alpha\ Z\ T_1$$
$$Z\ \alpha \rightarrow \alpha\ Z$$
$$Y\ \alpha \rightarrow \alpha\ Y$$
$$a_j\alpha \rightarrow \alpha a_j \quad ----- (j = 1, \ldots, r)$$
$$T_i\sigma_i\alpha \rightarrow \alpha T_{i+1} \quad ----- (i = 1, \ldots, n-1)$$
$$T_i a_j \alpha \rightarrow \alpha A_j^i T_i \quad ----- (i = 1, \ldots, n-1;\ j = 1, \ldots, r)$$
$$A_j^i\alpha \rightarrow \alpha A_j^i \quad ------ (\text{all } i, j)$$
$$T_n\sigma_n Z\alpha \rightarrow \alpha Q.$$

The only change concerns the symbols A_j^i. If the reader retraces the example above he will find that the final result will be

$$Q\ \overbrace{A_1^1A_2^1}\ \overbrace{A_4^2A_2^2}\ \overbrace{A_2^3A_1^3}$$

so that now, by inspecting the superscripts, one can tell that $\alpha_1 = a_1a_2$, $\alpha_2 = a_4a_2$, and $\alpha_3 = a_2a_1$. All the required information has been preserved, so we can go on to construct the consequent.

Before going on, observe that if the original string can be resolved into antecedent form in several ways we will get a distinct Q-string for each. Thus if we start with the string

$$Ta_3a_1a_2a_5a_2a_2a_5a_1a_3a_2a_4$$

in our previous example, we can analyse it either as

$$\sigma_0\boxed{a_1}\sigma_1\boxed{a_2a_2a_5}\sigma_2\boxed{a_2}\sigma_3$$

or as

$$\sigma_0\boxed{a_1a_2a_5a_2}\sigma_1\boxed{}\sigma_2\boxed{a_2}\sigma_3.$$

Therefore the system yields the two Y-strings

$$Q\ A_1^1A_2^2A_2^2A_4^2A_2^3$$

and

$$Q\ A_1^1A_2^1A_5^1A_2^1A_2^3.$$

In the latter, α_2 is a null-string, which is perfectly all right.

Now we construct the consequent. We will use the letter Q to initiate the process by setting-up the consequent form into which the -strings are to be copied. We will use new letters $V^1,\ldots,V^n$ to mark the places corresponding to the variables $\alpha_1,\ldots,\alpha_n$ in the consequent. The consequent form is introduced by the production

$$\boxed{Q\alpha \rightarrow \alpha\tau_0V^{i1}\tau_1V^{i2}\tau_2\ .\ .\ .\ V^{im}\tau_mZ\ Y,}$$

where $\tau_0,\ldots,\tau_m$ are the constant strings of the consequent of π. A set of productions

$$\boxed{A_j^ia_k\alpha \rightarrow \alpha a_kA_j^i}$$

allow the A's to move freely to the right, passing over letters in the original alphabet. We do not allow the A's to pass one another, hence their order is always preserved. We also allow the A's to pass over some of the V's:

$$\boxed{A_j^iV^k\alpha \rightarrow \alpha V^kA_j^i}\text{------(if } i\neq k).$$

Hence the A's ignore V's with different superscripts. But when the superscripts agree, we use

$$\boxed{A_j^iV^i\alpha \rightarrow \alpha V^ia_jA_j^i.}$$

Hence, whenever an A_j^i passes over a V with the same superscript--corresponding to the same variable string α_i--it leaves there a copy of that letter a_i which caused that A-symbol to appear. The result is that when all A's have passed over all V's, each V^i will be followed by a copy of the corresponding α_i-string. To see this in our example,

$Q\ A_1^1A_2^1A_4^2A_2^2A_2^3A_1^3 \rightarrow$

$$A_1^1A_2^1A_4^2A_2^2A_2^3A_1^3a_2a_1V^3a_2a_2V^3a_1a_5Z\ Y$$
$$\ldots.$$
$$A_1^1A_2^1A_4^2A_2^2A_2^3a_2a_1V^3a_1a_2a_2V^3a_1a_1a_5A_1^3Z\ Y$$
$$\ldots.$$
$$a_2a_1V^3a_2a_1a_2a_2V^3a_2a_1a_1a_5A_1^1A_2^1A_4^2A_2^2A_2^3A_1^3Z\ Y$$

and we see that a copy of $\alpha_3 \equiv a_1a_2$ has been inserted to the right of

each occurrence of $V^3 = V^{i1} = V^{i2}$.

When the string reaches this form, all the work is done and it remains only to eliminate the auxiliary letters of P*. The productions

$$\boxed{A^i_j Z\alpha \rightarrow \alpha Z}$$

remove the A's, while the productions

$$\boxed{\begin{array}{l} Ya_j\alpha \rightarrow \alpha a_j Y \\ Y\ V^i\alpha \rightarrow \alpha Y \end{array}}$$

erase all the V's. Now all these events can occur in many different orders. However the net effect is always the same, once Y arrives at the immediate left of Z. For then all A's have been erased and also all V's. When this happens we apply, finally,

$$\boxed{YZ\alpha \rightarrow \alpha T}$$

yielding the ultimate desired result

$$T\tau_0\alpha_{i_1}\tau_1\alpha_{i_2}\ldots\alpha_{i_m}\tau_m.$$

Observe that Y cannot pass over an A^i_j. Therefore, if Y is to move over to Z it must "push" the A's ahead of it, and hence all A's must eventually pass over all V's. Y erases V's when it comes to them, but clearly it cannot erase a V until it gets to it, and to get there it must have pushed all A's past that point. Thus no V is erased until it has thus done its work of copying.

Problem: Prove the theorem using productions $\sigma\alpha \rightarrow \alpha\tau$ in which neither σ nor τ have more than two, and not both have two, letters. Do not hesitate to use great quantities of letters.

Completing the proof

There remain a few loose ends. Observe first that we have not really produced a legitimate extension P* of P, because P* does not produce any strings which have only letters of P. Proof of this statement: The axioms of P* all contain the upper-case letter T. Every production which involves an upper-case letter in its antecedent constant string produces at least one such letter in its consequent string. Therefore, (by induction) each produced string contains an upper-case letter.

Obviously, the strings we wish to "detach" are those which start with 'T' In fact any string $T\alpha$ can be interpreted as an assertion

that α is a Theorem of the system P. It would be tempting simply to introduce a production

$$T\alpha \rightarrow \alpha$$

to remove an initial T. The trouble is that this would lead to spurious strings in the P-alphabet, because of the productions

$$a_j\alpha \rightarrow \alpha a_j$$

which are already part of the system. This shows, in fact, that there is no way to obtain the desired result in our system, because we will always get "rotated" strings with legitimate strings. The cure requires a new alphabet. Suppose that the alphabet of P is really $b_1,\ldots,b_r$ and that $a_1,\ldots,a_r$ are really new letters of P*. Then we can "detach" 'T' by the productions

$$\boxed{\begin{array}{l} Ta_j\alpha \rightarrow \alpha Rb_j \\ a_j\alpha \rightarrow \alpha b_j \\ R\,\alpha \rightarrow \alpha \end{array}}$$

The trick depends on the fact that we do not allow the b's to rotate (there are no productions $b_j\alpha \rightarrow \alpha b_j$.)

Of course these productions make it possible for an a_j to be converted to a b_j at any step of the process, but when this happens prematurely it only yields "doomed" strings which are not pure strings in P and cannot yield any of same, once the 'b' letter rotates to the front of the string. Only in the case that the conversion begins with 'Ta_j', so that the resulting string comes to begin with 'R', can the (otherwise) always-present upper-case letter be removed, for

$$R\alpha \rightarrow \alpha$$

is the only production of the entire system that can eliminate the auxiliary upper-case letter.

Another loose end is involved in the case of systems with more than one production. Here we simply carry out the whole above construction for each production, using entirely new letters for all upper-case letters except T. Then the productions can operate independently, only linked by T so that each system can operate on the <u>final</u> results of the operation of other systems.

We need one more theorem to complete our proof that the Canonical

systems of Post, in their full generality, can be replaced by Normal Extensions. We have to account for the case of productions which require several, rather than just one, strings in the antecedent. The proof of this, below, shows that this most general system can be reduced (in the form of a Canonical Extension) to the case of the single-antecedent production. Having shown this, the result of the present section completes the proof of the general theorem. We feel that the method of the following proof is instructive in itself, if only in showing how it may be useful to employ, for theoretical use only, a kind of reckless extravagance.

Reduction of multiple-antecedent productions to single-antecedent productions

We consider now the most general form of Post's Canonical systems, in which a production may require more than one existing string in order to produce a new string. A most classical example of this occurs in elementary logic where from A and $A \supset B$ one produces B. In the most general system P a production has the form

$$\begin{array}{l} \sigma_{10}\alpha_{11}\sigma_{11}\alpha_{12}\cdots\sigma_{1n_1}, \\ \sigma_{20}\alpha_{21}\sigma_{21}\alpha_{22}\cdots\sigma_{2n_2}, \\ \sigma_{S0}\alpha_{S1}\sigma_{S1}\alpha_{S2}\cdots\sigma_{Sn_S} \\ \rightarrow \tau_0\alpha_1\tau_1\alpha_2\cdots\tau_r \end{array} \qquad (\pi)$$

where the σ's and τ's are given constant strings and each α_k is one of the α_{ij}'s.

Below we show how to make a canonical extension P* of P which has only single-antecedent productions. Let $a_1,\ldots,a_r$ be the alphabet of P. Let $\Phi_1,\ldots,\Phi_p$ be the strings which are the Axioms of P. Our new system will have but one axiom: let 'A' be a new symbol.

$$\underline{AA\Phi_1AA\Phi_2AA\ \ldots\ AA\Phi_pAA}$$

(There would be little point to having several axioms now, since with single-antecedent productions there could be no combining of their separate consequences.) Corresponding to the old production π we introduce one new symbol X and the following monstruous production π^*:

$$
\begin{array}{l}
A\delta_0 \backslash\sigma_{10}\alpha_{11}\sigma_{11}\alpha_{12} \cdots \sigma_{1n_1} A\delta_1 A\sigma_{20}\alpha_{21}\sigma_{21}\alpha_{22} \cdots A\delta_2 A \cdots \\
\cdots A\sigma_{S0}\alpha_{S1}\sigma_{S1}\alpha_{S2} \cdots \sigma_{Sn_S} A\delta_S A \\
\qquad\qquad\qquad\qquad \longrightarrow \\
A\delta_0 A\sigma_{10} \cdots \qquad\qquad \cdots \qquad\qquad \cdots \\
\cdots \qquad \cdots \qquad\qquad \sigma_{Sn_S} A\delta_S AA\tau_0\alpha_1\tau_1\alpha_2 \qquad \cdots \\
\cdots\alpha_\gamma\tau_r X\alpha_{11}\alpha_{12} \cdots \alpha_{1n_1}\alpha_{21} \cdots \alpha_{2n_2} \cdots \alpha_{S1} \cdots \alpha_{Sn_S} X
\end{array}
\qquad (\pi^*)
$$

This has the effect of analysing a string to see if it contains substrings which have the forms required of the antecedents. If it finds such strings, sandwiched between A's, and if they occur in the given order--an undesirable requirement to be lifted shortly--it adjoins the resulting consequent, and some other material.

Why cannot we simply adjoin the consequent? The reason is that the analysis may be spurious, in that one of the α_{ij}'s can contain too much--parts from more than one proper P-string. This will be the case if any α_{ij} contains one or more 'A's in its interior. This is why our production π^* finally appends all the discovered variable strings in the form

$$X\alpha_{11} \cdots \alpha_{Sn_S} X.$$

Note that the form does not end now with an A, so that π^* can not apply. We now check to see if this appendage contains no A's:

$$
\begin{array}{|l|}
\hline
\alpha_1 X a_j \alpha_2 X \rightarrow \alpha_1 X \alpha_2 X \\
\alpha_1 XX \rightarrow \alpha_1 AA \\
\hline
\end{array}
\qquad (j = 1,\ldots,r)
$$

Then the string will return to standard form only if there were no A's in any of the selected α_{ij}'s.

The result is that as new strings are produced, in accord with the productions of the original system P, they are appended (between 'AA's) to the axiom string. Any assertion of P is ultimately so generated, and appears preceded by its entire "production history".

An extension system is required to yield the original strings without any auxiliary letters. We do this by a similar method:

$$\boxed{\begin{array}{l} A\alpha_1 A\alpha_2 A\alpha_3 A \rightarrow Y\alpha_2 Y \\ \alpha_1 Y a_j \alpha_2 Y \rightarrow \alpha_1 a_j Y \alpha_2 Y \\ \alpha Y Y \rightarrow \alpha \end{array}} \qquad (j = 1,\ldots,r)$$

The result is that any P-string which occurs between A's in a standard form P*-string will be released without auxiliary letters.

One loose end. The production π^* requires the antecedents of π to occur in a given order. There is no meaning to such a restriction in P. To eliminate it we could introduce (rather than just π^*) a distinct production like π^* for each possible order of the antecedents--that is, we need S! forms of π^* with the antecedent parts permuted. This would complete the construction. A more elegant solution: we adjoin the single production

$$\boxed{A\alpha_1 AA\alpha_2 AA\alpha_3 A \rightarrow A\alpha_1 AA\alpha_2 AA\alpha_3 AA\alpha_2 AA.}$$

This makes it possible to take any interior P-string and append it to the end of the compound P*-string. Thus the required antecedents can be placed at the end in any order required by a π^* production.

Part II

A "universal" canonical extension

It is very easy to construct an analogue of a Universal Turing machine in the form of a canonical system. Let P be an arbitrary Normal system with axioms $\Phi_1,\ldots,\Phi_s$ --and productions $\sigma_i\alpha \rightarrow \alpha\tau_i$. Let $a_1,\ldots,a_r$ be the alphabet of P, and let A, C, S, and T be new letters. Our system $\mathcal{U}_P$ will have the single axiom

$$AC A\sigma_1 C\tau_1 A\sigma_2 C\tau_2 \ldots A\sigma_n C\tau_n AS\Phi_1 S\Phi_2 \ldots S\Phi_s STTT.$$

Observe that this axiom is a complete description of P, since from it one can reconstruct all the axioms and productions of P. Our productions are:*

$$\pi_1 \qquad \alpha_1 A\sigma C\tau A\alpha_2 S\beta\alpha S\alpha_3 TTT \rightarrow \alpha_1 A\sigma C\tau A\alpha_2 S\beta\alpha S\alpha_3 \alpha\tau ST\sigma\alpha\tau T\beta\alpha\tau T\alpha\tau$$

$$\pi_{2_j} \quad \alpha_1 Ta_j\alpha_2 Ta_j\alpha_3 T\alpha_4 \rightarrow \alpha_1 T\alpha_2 T\alpha_3 T\alpha_4 \qquad (j = 1,\ldots,r)$$

$$\pi_3 \qquad \alpha_1 TTT\alpha_2 \rightarrow \alpha_1 TTT$$

$$\pi_4 \qquad \alpha_1 TTT\alpha_2 \rightarrow \alpha_2$$

We assert that this system is a canonical extension of P! The remarkable aspect of this is that the productions of $\mathcal{U}_P$ do not depend in any way on the structure of P (except for the restriction on the alphabet, which we will dispose of shortly). Thus $\mathcal{U}_P$ is like a Universal Turing machine in that, given (as its axiom) a description of an arbitrary canonical system P, the productions of $\mathcal{U}$ will generate precisely the theorems of P.

To see that $\mathcal{U}_P$ is a canonical extension of P, let us examine the effect of π_1. π_1 "looks" for a substring $A\sigma C\tau A$ and a substring $S\beta\alpha S$ and "attempts" to adjoin the string $\alpha\tau$ in accord with some production of P

$$\sigma_i\alpha \rightarrow \alpha\tau_i.$$

This operation is valid only if (1) $\beta = \sigma$ and (2) if the strings α, β, σ, τ do not contain upper-case letters. For if a string has the form

* Here, all Greek letters represent variables.

$A\sigma C\tau A$ with σ and τ having no upper-case letters, then σ and τ must be some σ_i and τ_i. The productions π_{2_j} check both conditions, for they remove, one at a time, letters from $\sigma\alpha\tau$ and $\beta\alpha\tau$, but only if the first letter of each is at each step the same. Both strings can vanish only if $\sigma = \beta$. Then production π_3 can apply, adjoining the produced string $\alpha\tau$ as though it were a new axiom of P. Production π_4 detaches the new string from all upper-case letters, to yield a legitimate theorem of P.

We put the initial 'AC' in π, so that the system would be sure to produce the axioms of P, in accord with the production

$$\alpha \rightarrow \alpha$$

which may not be included among the productions of P.

Now for the alphabet problem. We simply encode P's alphabet $a_1, \ldots, a_r$ into a two-letter alphabet a, b: simply let 'a_j' be represented by a 'b' followed by j 'a's. Then everything works as before, and we need only two productions π_{2a} and π_{2b} for $\mathcal{U}$. So $\mathcal{U}$ ends up with just five productions. It no longer is an extension of P, but it is (with encoded axiom of P) an extension of a trivial re-coding of P.

Problem 1. Three letters is minimal for a Universal Canonical Extension. (Why?) But we can construct a version of $\mathcal{U}$ with only four productions. Hint: A simple change in π_1 makes π_3 unnecessary. Solution on next page.

I can't see how less than four productions could suffice, for we need one to "do the work", two to check the alphabet conditions, and one to "release" the pure string. Such reasoning, however, usually turns out to be unsound!

Problem 2. Construct a version of with only one extension letter, still using only four productions. (Easy)

Problem 3. Construct a version of $\mathcal{U}$ using only normal productions. (Laborious.)

Problem 4. Using the analogy with a Universal Turing machine, construct an unsolvable decision problem for $\mathcal{U}$. Consider the prospect of developing the theory of computability on this basis, rather than on the Turing basis.

Solution to problem 1:

Don't Read Unless
You
Give Up
on
Problem 1.

π_1: $\alpha_1 A\sigma C\tau A\alpha_2 S\beta\sigma S\alpha_3 TTT\alpha_4 \rightarrow \alpha_1 A\sigma C\tau A\alpha_2 S\beta\sigma S\alpha_3 \sigma\tau ST\alpha\tau T\beta\sigma\tau T\alpha\tau$

π_{2a}: $\alpha_1 Ta\alpha_2 Ta\alpha_3 T\alpha_4 \rightarrow \alpha_1 Ta_2 Ta_3 Ta_4$

π_{2b}: $\alpha_1 Tb\alpha_2 Tb\alpha_3 T\alpha_4 \rightarrow \alpha_1 Ta_2 Ta_3 Ta_4$

π_4: $\alpha_1 TTT\alpha_2 \rightarrow \alpha_2$

CS-TR Scanning Project
Document Control Form

Date : 11/30/95

Report # AIM-44

Each of the following should be identified by a checkmark:
Originating Department:

☒ Artificial Intellegence Laboratory (AI)
☐ Laboratory for Computer Science (LCS)

Document Type:

☐ Technical Report (TR) ☒ Technical Memo (TM)
☐ Other:_____

Document Information

Number of pages: 14 (18 IMAGES)
Not to include DOD forms, printer intstructions, etc... original pages only.

Originals are:
☒ Single-sided or
☐ Double-sided

Intended to be printed as :
☒ Single-sided or
☐ Double-sided

Print type:
☐ Typewriter ☐ Offset Press ☐ Laser Print
☐ InkJet Printer ☐ Unknown ☒ Other: [redacted] COPY OF MIMEO GRAPH

Check each if included with document:

☐ DOD Form ☐ Funding Agent Form ☐ Cover Page
☐ Spine ☐ Printers Notes ☐ Photo negatives
☐ Other: _____

Page Data:

Blank Pages (by page number):_____

Photographs/Tonal Material (by page number):_____

Other (note description/page number):

Description : Page Number:
IMAGE MAP: (1-14) TITLE PAGE, 1-13
(15-18) SCANCONTROL TEST'S (3)

Scanning Agent Signoff:
Date Received: 11/30/95 Date Scanned: 12/11/95 Date Returned: 12/14/95

Scanning Agent Signature: Michael W. Cook

Scanning Agent Identification Target

Scanning of this document was supported in part by the **Corporation for National Research Initiatives,** using funds from the **Advanced Research Projects Agency** of the **United states Government** under Grant: **MDA972-92-J1029.**

The scanning agent for this project was the **Document Services** department of the **M.I.T Libraries.** Technical support for this project was also provided by the **M.I.T. Laboratory for Computer Sciences.**