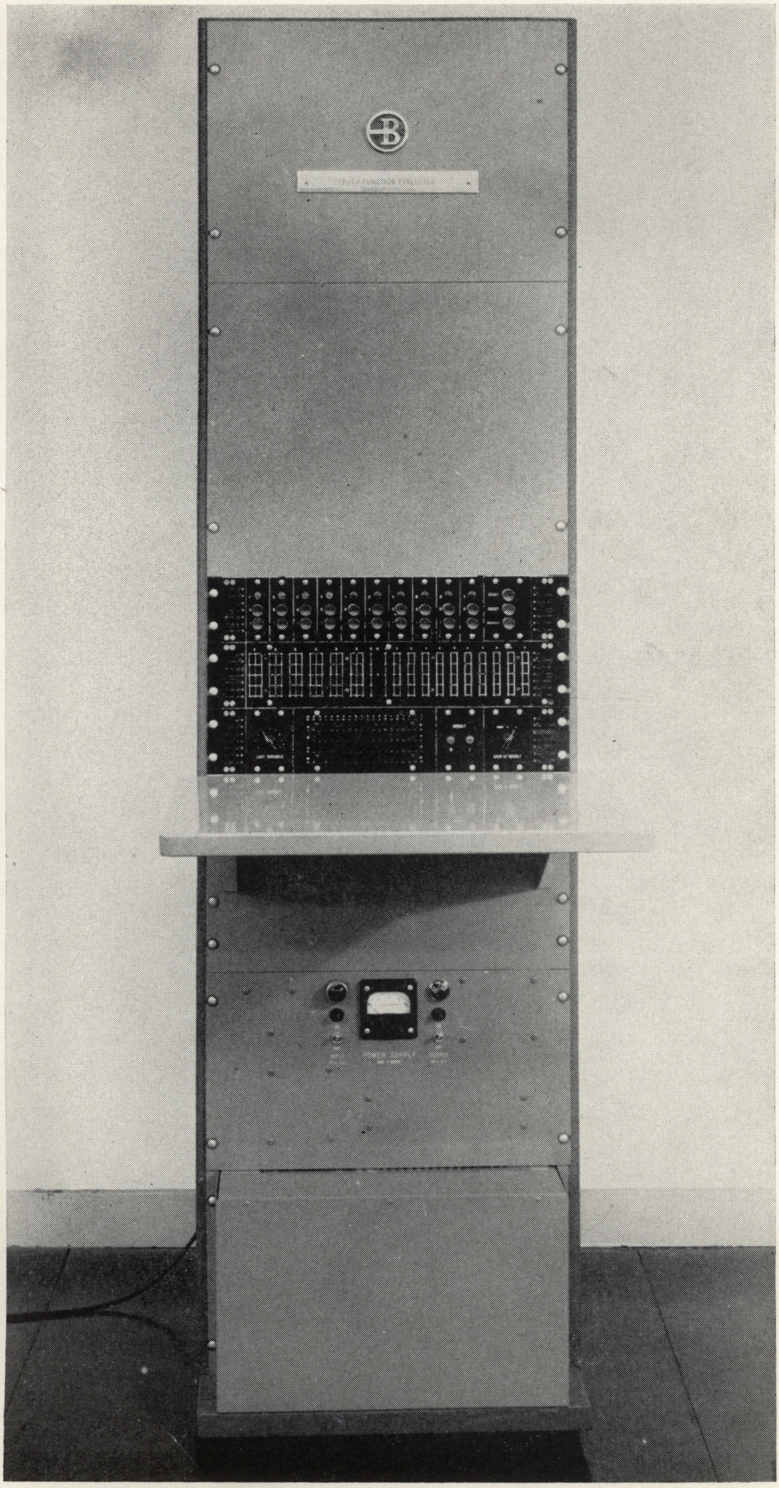

BURROUGHS TRUTH FUNCTION EVALUATOR

An Analysis of a Logical Machine Using Parenthesis-Free Notation

1. Introduction. ŁUKASIEWICZ' parenthesis-free notation[1] permits a simple and easily mechanizable process of truth-table computation. We shall describe this process and prove some relevant theorems. A 10-variable relay machine employing this method for a two-valued logic has been constructed and used by the Burroughs Corporation (see frontispiece) and it is feasible in the present state of the computer art to construct a 25-variable machine which would reduce all evaluants of a 250-character formula in a two-valued logic with monadic and dyadic operators in about an hour. The operation of such a machine will be described in terms of the manipulations which it performs upon a formal propositional language.

2. The Language. We consider any language L based on a quadruple $\langle C, W, P, F\rangle$ whose terms will be characterized in this section[2]. C is a finite non-empty set of *characters*.

(*Example 1:* The theory will be illustrated from time to time by the use of a specific sample L whose characters (members of C) are the following: the functors N (negation), K (dyadic conjunction), A (dyadic alternation); the propositional variables, p, q, r; the truth-constants 1 (truth), 0 (falsity).)

Definition 1: Any finite sequence of characters (including the null sequence) is a *formula.*

'Λ' will designate the null formula. Except for 'Λ' and 'π' (to be introduced later), all lower case and upper case Greek letters (with and without natural number subscripts) will range over all characters and all formulas respectively. Juxtaposition in the syntax language will be used to denote juxtaposition in L. We will use the following terminology pertaining to formulas.

Definition 2:

A (length): $\boldsymbol{L}(\Delta)$ is the number of character tokens in Δ.

Let $\Delta = \Phi\Psi$, then

B (tail): $\boldsymbol{T}_i(\Delta) = \Psi$ where $i \leqq \boldsymbol{L}(\Delta)$ and $\boldsymbol{L}(\Psi) = i$, or $i > \boldsymbol{L}(\Delta)$ and $\Psi = \Delta$;

C (head): $\boldsymbol{H}_i(\Delta) = \Phi$ where $i \leqq \boldsymbol{L}(\Delta)$ and $\boldsymbol{L}(\Phi) = i$, or $i > \boldsymbol{L}(\Delta)$ and $\Phi = \Delta$.

$\boldsymbol{W}$ is a function assigning to each character (element of C) an integral *weight* $\leqq 1$.

(*Example 2:* N has weight 0, K and A have weight -1, the variables and truth-constants have weight 1.)

The following concepts are defined in terms of weight.

Definition 3:

A (character degree): $\boldsymbol{D}(\delta) = 1 - \boldsymbol{W}(\delta)$.

B (formula weight): $\boldsymbol{W}(\Delta)$ is the sum of the weights of the character tokens of Δ; $\boldsymbol{W}(\Lambda) = 0$.

C (max. weight, tail): $\boldsymbol{W}_{\mathrm{Max}}(\Delta) = \mathrm{Max}(\boldsymbol{W}[\boldsymbol{T}_i(\Delta)])$ for $i > 0$.

D (min. weight, tail): $W_{\text{Min}}(\Delta) = \text{Min}(W[T_i(\Delta)])$ for $i > 0$.

E (positive formula): Δ is a positive formula if and only if $W_{\text{Min}}(\Delta) > 0$.

F (well-formed formula): Δ is a well-formed formula if and only if Δ is a positive formula and $W(\Delta) = 1$.

(*Example 3:* $pKqN$ is not a positive formula; $ApqNr$ is positive but not well-formed; $KpNq$ is well-formed.)

Note that the non-recursive definition 3F shows that a single scan of a formula from right to left is sufficient to determine whether it is well-formed.

P is a non-empty subset of C whose elements are *truth-constants*. 'π' with or without natural number subscripts will range over this set. P must satisfy the condition

$$W(\pi) = 1. \tag{1}$$

F is a function which assigns to each character of degree > 0 a "truth function"; i.e., which satisfies the conditions

$$\text{If } D(\delta) > 0, \quad \text{then } F(\delta\pi_{D(\delta)} \cdots \pi_1) \epsilon P; \tag{2a}$$

$$F(\pi) = \pi. \tag{2b}$$

(*Example 4:* In the sample language P consists of 0 and 1; the function F applied to K gives $F(K00) = 0$, $F(K01) = 0$, $F(K10) = 0$, $F(K11) = 1$.)

We can now define the ordinary concept of (propositional) variable.

Definition 4: δ is a *variable* if and only if $W(\delta) = 1$ and δ is not in P.

An important relation between positive formulas and well-formed formulas which we will need is given by:

THEOREM I: (A) *Δ is a positive formula if and only if Δ can be partitioned (i.e., divided into a sequence of disjoint segments which exhaust it) into exactly $W(\Delta) \geqq 1$ well-formed formulas.* (B) *There is at most one partition of a positive formula Δ into well-formed formulas.*[3]

Proof: (A) The "if" part is obvious. The "only if" part is proved as follows. Let Γ be the shortest head of positive weight of a positive formula Δ. Hence, for any $i > 0$, $W[T_i(\Gamma)] > 0$; and since Γ is not null it is a positive formula. Again, since Γ is the shortest head of Δ of positive weight, $W[H_a(\Gamma)] \leqq 0$ (where $a = L(\Gamma) - 1$); and since Γ is a positive formula, $W[T_1(\Gamma)] = 1$; hence $W(\Gamma) \leqq 1$. But then $W(\Gamma) = 1$ and Γ is a well-formed formula. It follows that $W[T_b(\Delta)] = W(\Delta) - 1$, for $b = L(\Delta) - L(\Gamma)$; then if $T_b(\Delta)$ is not null it is a positive formula and this process may be repeated until Δ is partitioned into $W(\Delta)$ well-formed formulas.

(B) The proof is by induction on $L(\Delta)$. If $L(\Delta) = 1$, Δ is a single character and "B" obviously holds. Suppose "B" holds for all positive formulas of length $< n$; consider a positive formula Δ with $L(\Delta) = n$. If there are two partitions of Δ, one will have a first (leftmost) well-formed formula Φ of length $\leqq$ the length of the first well-formed formula $\Phi\Psi$ in the other partition. But $W(\Phi\Psi) - W(\Phi) = W(\Psi) = 0$ since $W(\Phi\Psi) = W(\Phi) = 1$. Therefore $\Psi = \Lambda$, else $W_{\text{Min}}(\Phi\Psi) = 0$ implying $\Phi\Psi$ is not well-formed. If $\Phi = \Phi\Psi = \Delta$, then "B" has been established for Δ. Otherwise $\Delta = \Phi\Gamma = \Phi\Psi\Gamma$ where Γ is a positive formula and $L(\Gamma) < L(\Delta)$. In this case the inductive hypothesis shows that "B" holds for Γ and hence for $\Phi\Gamma = \Delta$.

Theorem I makes possible the following definition.

Definition 5 (partition function): If $\Delta = \Delta_J \cdots \Delta_j \cdots \Delta_1$ and each Δ_j is well-formed, then $\boldsymbol{P}_j(\Delta) = \Delta_j$.

(*Example 5:* For $\Delta = pKpqNr$, $\boldsymbol{P}_1(\Delta) = p$, $\boldsymbol{P}_2(\Delta) = Kpq$, $\boldsymbol{P}_3(\Delta) = Nr$.)

With the aid of Theorem I and Definition 3A it can be shown that our formulation of the language L is easily reducible to a more conventional formulation. For example, Δ is a well-formed formula if and only if it is of the form $\delta\Delta_{D(\delta)} \cdots \Delta_1$, where each Δ_d is well-formed. Note that $\boldsymbol{D}(\delta)$ is the number of well-formed formulas following δ in the above decomposition; if $\boldsymbol{D}(\delta) > 0$, δ is an operator of the propositional calculus and $\boldsymbol{D}(\delta)$ is its degree in the ordinary sense, while if $\boldsymbol{D}(\delta) = 0$, δ is well-formed by itself and hence is a truth-constant or propositional variable.

An interesting property peculiar to the parenthesis-free notation is given by:

THEOREM II: *Every formula of a language L is a segment of some well-formed formula of L if and only if L contains at least one character of negative weight.*

Proof: The proof of the "if" part is as follows. Let ω be a character of negative weight. For Δ an arbitrary formula of the language L, $\Psi\Delta\Phi$ is well-formed where Φ consists of $1 - \boldsymbol{W}_{\mathrm{Min}}(\Delta)$ occurrences of π and Ψ consists of a formula of weight -1 (e.g., ω followed by $-1 - \boldsymbol{W}(\omega)$ occurrences of π) repeated $\boldsymbol{W}(\Delta\Phi) - 1$ times. The proof of the "only if" part is as follows. $\boldsymbol{W}(\pi\pi) = 2$, and, since $\pi\pi$ is a segment of some well-formed formula Δ, L must contain a character of negative weight in order that $\boldsymbol{W}(\Delta) = 1$.

The need for a machine to do truth-table computation occurs only for languages containing at least one symbol of negative weight and a multiplicity of symbols of weight 1.

3. The Machine. With the aid of the two following definitions we can describe, for any given language L, the construction of a machine to evaluate formulas of L.

Definition 6 (the set of specification functions): $\boldsymbol{G}$ is a specification function if and only if (1) if δ is a variable, $\boldsymbol{G}(\delta) \epsilon P$ and (2) if δ is not a variable, $\boldsymbol{G}(\delta) = \delta$.

(Hereafter '$\boldsymbol{S}$' will represent an arbitrary specification function.)

Definition 7 (evaluation function): $\boldsymbol{E}_S(\Lambda) = \Lambda$; if $\delta\Delta$ is a positive formula, $\boldsymbol{E}_S(\delta\Delta) = \boldsymbol{F}(\boldsymbol{S}(\delta)\boldsymbol{H}_{D(\delta)}[\boldsymbol{E}_S(\Delta)])\boldsymbol{T}_a[\boldsymbol{E}_S(\Delta)]$ where $a = \boldsymbol{L}[\boldsymbol{E}_S(\Delta)] - \boldsymbol{D}(\delta)$.

(*Example 6:* For the formula of Example 5, $\Delta = pKpqNr$, the recursive evaluation is as follows: let the specification, $\boldsymbol{S}$, of the variables be $p = 1$, $q = 0$, $r = 0$, then $\boldsymbol{E}_S[\boldsymbol{T}_1(\Delta)] = \boldsymbol{F}[\boldsymbol{S}(r)] = 0$, $\boldsymbol{E}_S[\boldsymbol{T}_2(\Delta)] = \boldsymbol{F}(N0) = 1$, $\cdots$, $\boldsymbol{E}_S[\boldsymbol{T}_5(\Delta)] = \boldsymbol{F}(K10)1 = 01$, $\boldsymbol{E}_S(\Delta) = \boldsymbol{F}(p)01 = 101$.)

The following lemma shows that $\boldsymbol{L}(\boldsymbol{H}_{D(\delta)}[\boldsymbol{E}_S(\Delta)]) = \boldsymbol{D}(\delta)$ and hence that $\boldsymbol{E}_S$ is defined for all positive formulas.

LEMMA: If Δ is a positive formula, then $\boldsymbol{E}_S(\Delta)$ is of the form $\pi_n\pi_{n-1} \cdots \pi_1$ where $n = \boldsymbol{W}(\Delta)$.

Proof: That all characters are π's follows directly from the conditions (2a and 2b) which characterize $\boldsymbol{F}$. That $n = \boldsymbol{W}(\Delta)$ is established by an induction on $\boldsymbol{L}(\Delta)$.

Note that if Δ is well-formed, $\boldsymbol{L}[\boldsymbol{E}_S(\Delta)] = 1$; it is easy to show that when Δ is well-formed, $\boldsymbol{E}_S(\Delta)$ is its truth-value relative to $\boldsymbol{S}$ in the ordinary sense.

A number of well-formed formulas may be evaluated concurrently by juxtaposing them to form (by Theorem I) a positive formula Δ. Let $\Delta = \delta_{L(\Delta)} \cdots \delta_i \cdots \delta_1$, let m be the number of truth-constants in the language L (i.e., L is an m-valued logic), and let v be the number of distinct variables of Δ. The machine makes m^v scans of Δ, one for each distinct specification of the variables, producing each time $E_S(\Delta)$; a single complete scan for a given S with the accompanying computation determines the "Sth" *major cycle.* Each major cycle is divided into $L(\Delta)$ *minor cycles*, the ith minor cycle encompassing the processing of the ith character δ_i.

The machine consists of two basic components, a Memory and an Evaluator. The *Memory*[4] (e.g., a magnetic drum, an acoustic delay line) stores Δ and during each major cycle sends it characters $\delta_{L(\Delta)} \cdots \delta_1$ to the Evaluator in order of ascending subscripts. During the Sth major cycle the *Evaluator* realizes the recursive function $E_S(\Delta)$ by producing successively the $E_S[T_i(\Delta)]$. It does this by means of three parts: a Specifier, a Function Switch, and a Register.

The *Specifier* (e.g., an electronic counter with switching gates) effects the sequence of S functions required by Δ; during the ith minor cycle it receives δ_i and produces $S(\delta_i)$, and at the end of the Sth major cycle it introduces a new S. During the ith minor cycle of the Sth major cycle the *Function Switch* (e.g., an array of electronic switching gates) receives $S(\delta_i)$ from the Specifier, $H_{D(\delta_i)}(E_S[T_{i-1}(\Delta)])$ from the Register, and produces the new character $F[S(\delta_i)H_{D(\delta_i)}(E_S[T_{i-1}(\Delta)])]$. This new character is sent to the *Register* (e.g., an electronic shifting register) which by shifting an amount $W(\delta_i)$ (a positive value indicates a right shift, a negative value a left shift) substitutes it for $H_{D(\delta_i)}(E_S[T_{i-1}(\Delta)])$.

4. Some Theorems. The following theorem justifies the use of a positive formula Δ to compute the truth-values of its well-formed components $P_j(\Delta)$.

THEOREM III: *If Δ is a positive formula then* $E_S(\Delta) = E_S[P_{W(\Delta)}(\Delta)] \cdots E_S[P_1(\Delta)]$.

(*Example 7:* In the formula of Examples 5 and 6, $E_S(\Delta) = E_S(p)E_S(Kpq)E_S(Np)$ which, for the S of Example 6, evaluates to 101.)

Proof: The theorem follows directly from the fact that, if Φ and Ψ are positive formulas or null, $E_S(\Phi\Psi) = E_S(\Phi)E_S(\Psi)$. This may be proved by an induction on $L(\Phi)$. It is obviously true for $\Phi = \Lambda$. We assume it to be true for some Γ and show that it holds for $\gamma\Gamma$. By Definition 7 $E_S(\gamma\Gamma\Psi) = F(S(\gamma)H_{D(\gamma)}[E_S(\Gamma\Psi)])(T_a[E_S(\Gamma\Psi)])$, where $a = L[E_S(\Gamma\Psi)] - D(\gamma)$, and by the inductive hypothesis $E_S(\gamma\Gamma\Psi) = F(S(\gamma)H_{D(\gamma)}[E_S(\Gamma)E_S(\Psi)])(T_b[E_S(\Gamma)E_S(\Psi)])$ where $b = L[E_S(\Gamma)E_S(\Psi)] - D(\gamma)$. Since $\gamma\Gamma$ is a positive formula, $L[E_S(\Gamma)] \geqq D(\gamma)$ and we have $E_S(\gamma\Gamma\Psi) = F(S(\gamma)H_{D(\gamma)}[E_S(\Gamma)])H_c[E_S(\Gamma)]E_S(\Psi)$, where $c = L[E_S(\Gamma)] - D(\gamma)$; hence $E_S(\gamma\Gamma\Psi) = E_S(\gamma\Gamma)E_S(\Psi)$.

For a given positive formula Δ the Register must be able to store a (positive) formula of length $\operatorname{Max}_{i=1}^{L(\Delta)} [L(E_S[T_i(\Delta)])]$. The following two theorems show how this quantity depends upon the structure of Δ.

THEOREM IV: *If Δ is a positive formula then* $\operatorname{Max}_{i=1}^{L(\Delta)} [L(E_S[T_i(\Delta)])] = W_{\text{Max}}(\Delta)$.

(*Example 8:* In Example 6 it can be seen that $\operatorname{Max}_{i=1}^{L(\Delta)=6} [L(E_S[T_i(\Delta)])]$

$= L[E_S(\Delta)] = 3 = W_{\text{Max}}(\Delta)$. It may be checked that the evaluants of $T_3(\Delta)$ and $T_4(\Delta)$ which were not given will not change this maximum.)

Proof: By the Lemma and Definition 3C.

THEOREM V: *If* Δ *is a positive formula, then* $W_{\text{Max}}(\Delta) = \underset{j=1}{\overset{W(\Delta)}{\text{Max}}} (W_{\text{Max}}[P_j(\Delta)] + j - 1)$.

Proof: Consider all the tails of Δ and note that each complete $P_j(\Delta)$ in a tail contributes a weight of 1.

This theorem shows that, given a set of well-formed formulas to be evaluated as a positive formula or to be used as arguments for a commutative operator, a formula of smallest W_{Max} may be formed by placing these well-formed formulas in order of ascending W_{Max} from left to right.

In designing a specific machine to carry out our process some decisions must be made concerning the relative sizes of the Memory and the Register. Our concluding theorem gives information relevant to this decision.

THEOREM VI: *Let* $M_{W,D}$ *be the set of well-formed formulas* Ψ *such that* (1) *the maximum degree of any character of* Ψ *is* $D > 1$ *and* (2) $W_{\text{Max}}(\Psi) = W$. *Then* (A) *for any* $\Psi \in M_{W,D}$ *there is a* $\Phi \in M_{W,D}$ *such that* $L(\Phi) \geqq L(\Psi)$ *and* (B) $L(\Psi) \geqq W + \left\{ \dfrac{W-1}{D-1} \right\}$, *where* $\{z\}$ *is the smallest integer* $\geqq z$.

Proof: For part (A) consider for any $\Psi \in M_{W,D}$ the formula $\Psi_1 = \delta\pi_D \cdots \pi_2\Psi$, where $D(\delta) = D$. Clearly Ψ_1 satisfies (1). By Theorem V, $W_{\text{Max}}(\Psi_1) = \text{Max}(D,W)$ which equals W since, for any positive formula Δ, $W_{\text{Max}}(\Delta) \geqq$ the maximum degree of any character of Δ. Hence $\Psi_1 \in M_{W,D}$ and $L(\Psi_1) > L(\Psi)$. To establish (B) consider any $\Psi \in M_{W,D}$. By (2) Ψ must contain at least W characters of weight 1. Since $W(\Psi) = 1$ the sum of the weights of its characters of negative weight is at most $1 - W$; by (1) no character of Ψ has a weight less than $1 - D$; since $D > 1$, Ψ must contain at least $\left\{ \dfrac{W-1}{D-1} \right\}$ characters of negative weight. Hence $L(\Psi) \geqq W + \left\{ \dfrac{W-1}{D-1} \right\}$.

For a language L containing characters of all degrees between 2 and D inclusive, if $M_{W,D}$ is non-empty, it contains a well-formed formula of the minimum length which consists of $\left\{ \dfrac{W-1}{D-1} \right\}$ characters of negative weight followed by W characters of weight 1.

ARTHUR W. BURKS
DON W. WARREN
JESSE B. WRIGHT

University of Michigan
Ann Arbor

The writing of this paper and the research which it reports were done under the sponsorship of the Burroughs Corporation.

[1] JAN ŁUKASIEWICZ, *Aristotle's Syllogistic from the standpoint of Modern Formal Logic.* Oxford, 1951, p. 78.

[2] In this section we employ the results of STANISŁAW JAŚKOWSKI, KARL MENGER, KARL SCHRÖTER, and D. C. GERNETH. See PAUL ROSENBLOOM, *The Elements of Mathematical Logic.* New York, 1950, ch. 4, sec. 1, for a discussion and further references.

[3] Theorem I implies that under the operation juxtaposition the set of positive formulas together with the two-sided identity Λ is a semigroup which is generated by the set of well-formed formulas.

[4] For a description of computer components see ENGINEERING RESEARCH ASSOCIATES, *High Speed Computing Devices.* New York, 1950.