THE JOURNAL OF SYMBOLIC LOGIC
Volume 41, Number 1, March 1976

AN ABSTRACTION ALGORITHM FOR COMBINATORY LOGIC

S. KAMAL ABDALI[1]

This note presents a practical algorithm for carrying out abstraction on combinatory terms. The well-known abstraction algorithms [1, pp. 188ff.] defining abstracts in terms of the combinator sets $\{S, K\}$, $\{B, C, K, W\}$, etc. operate on one variable at a time, and result in rather long abstracts when several variables are involved. These algorithms are not practical for the applications of the combinatory logic to the theory of computing which make much use of multi-variable abstraction (e.g., [2], [3]). The present algorithm performs the abstraction with respect to all specified variables in a single step, and yields abstracts in a concise form (with sizes proportional to those of given combinatory terms).[2]

§1. Requisite combinators. Our algorithm employs the combinators K_n, I_n^m, B_n^m having the following reduction properties:

$$\text{(1)}\quad \begin{aligned} &K_n a b_1 \cdots b_n \to a, && n \geq 1,\\ &I_n^m a_1 \cdots a_n \to a_m, && n \geq m \geq 1,\\ &B_n^m a b_1 \cdots b_m c_1 \cdots c_n \to a(b_1 c_1 \cdots c_n) \cdots (b_m c_1 \cdots c_n), && m, n \geq 1, \end{aligned}$$

where the a's, b's, c's denote arbitrary combinatory terms. It is desirable to generate the combinator families K_n, I_n^m, B_n^m from the suitably chosen combinators $\mathscr{K}$, $\mathscr{I}$, $\mathscr{B}$ and the combinator sequence $\{\hat{0}, \hat{1}, \hat{2}, \cdots\}$ representing natural numbers, by defining

$$\text{(2)}\quad \begin{aligned} K_n &\equiv \mathscr{K}\hat{n}, && n \geq 1,\\ I_n^m &\equiv \mathscr{I}\hat{m}\hat{n}, && n \geq m \geq 1,\\ B_n^m &\equiv \mathscr{B}\hat{m}\hat{n}, && m, n \geq 1. \end{aligned}$$

Based on [1], a simple way of defining $\mathscr{K}$, $\mathscr{I}$, $\mathscr{B}$ and $\{\hat{n}\}$ in terms of the familiar combinators S, K, B, C, I is the following: Let a, b, c be combinatory terms. Write $a \cdot b$ for Bab. (In the absence of parentheses, the "dot" operation is considered to have lower precedence than combination; thus $a \cdot bc$ is to be regarded as $a \cdot (bc)$, not $(a \cdot b)c$.) Also write $\langle a, b\rangle$ for $C(Ta)b$, where $T \equiv CI$, and $\langle a, b, c\rangle$ for $C\langle a, b\rangle c$.

Received August 5, 1974.

[1] The material in this paper is derived from the author's Ph.D. dissertation [3], written under the supervision of Professor George W. Petznick. The author is also thankful to Professor Haskell B. Curry for several suggestions, including the present definition of $\mathscr{B}$ (in formulae (3)). The preparation of this paper was supported by NSF grant GJ 25393.

[2] Professor Curry has kindly pointed out that an abstraction algorithm, his own, dealing with many variables at once has appeared much earlier in the literature [4]. The present algorithm is, however, simpler.

Define:

$$
\begin{aligned}
\Phi &\equiv BS\cdot B,\\
\hat{0} &\equiv KI,\\
\widehat{n+1} &\equiv SB\hat{n}, \quad \text{for } n \text{ a natural number} \geq 0,\\
\pi &\equiv \langle S(BCT)(SB)T\hat{0}, \langle \hat{0}, \hat{0}\rangle, K\rangle,\\
\mathscr{K} &\equiv TK,\\
\mathscr{I} &\equiv \Phi B(C\pi K)(CCK\cdot T\pi),\\
\mathscr{B} &\equiv T\cdot C(B\cdot C(T(C(\hat{2}B)S))I)K.
\end{aligned} \tag{3}
$$

The reduction relations (1) can now be verified using the definitions (2) and (3). This verification is straightforward, though laborious, and will be omitted.

Note. The combinator π represents the predecessor function on natural numbers, and possesses the reduction property

$$\pi\hat{n} \to \begin{cases} \hat{0}, & \text{if } n = 0,\\ \widehat{n-1}, & \text{if } n > 0. \end{cases}$$

§2. **The abstraction algorithm.** Before stating our abstraction algorithm, we first need some terminology: Let e be a combinatory term. If $e \equiv (ab)$ then a and b are, respectively, the *left* and *right immediate components* of e. A *component* of e is (recursively) either e itself or a component of an immediate component of e. An *initial component* of e is either e itself or an initial component of a left immediate component of e. A *primal component* of e is either a right immediate component of an initial component of e, or an initial component of e which is an atom.

EXAMPLE. The combinatory term $e \equiv SK(x(KK)yz)(S(wz)(SSy))(xyz)$ has five initial components, four of which are S, SK, $SK(x(KK)yz)$ and e itself; the primal components of e are S, K, $x(KK)yz$, $S(wz)(SSy)$, and xyz.

Let e be a combinatory term and x a variable. Then x oc e iff x is a component of e; x o$\not c$ e otherwise.

DEFINITION. Given a combinatory term e and variables $x_1, \cdots, x_n$, for some $n \geq 1$, an *abstract* of e with respect to $x_1, \cdots, x_n$ is a combinatory term f such that

(a) x_i o$\not c$ f, $1 \leq i \leq n$,

(b) $fx_1 \cdots x_n \to e$.

The abstraction algorithm is now given by the following:

DEFINITION. Let e be a combinatory term and $x_1, \cdots, x_n$ ($n \geq 1$) be variables. Then $[x_1, \cdots, x_n]e$ is the first of the following combinatory terms, selected in the given order, according as the condition is satisfied:

(i) $K_n e$, if x_i o$\not c$ e for all $1 \leq i \leq n$,

(ii) I, if $e \equiv x_1 \cdots x_n$,

(iii) I_n^i, if $e \equiv x_i$ for some $1 \leq i \leq n$,

(iv) g, if $e \equiv gx_1 \cdots x_n$ and x_i o$\not c$ g for all $1 \leq i \leq n$,

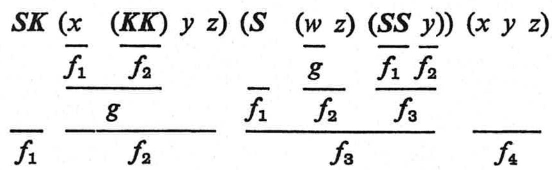
(v) $[x_1, \cdots, x_{m-1}]g$, if $e \equiv gx_m \cdots x_n$ for some $1 < m \leq n$, and x_i o$\not c$ g for all $m \leq i \leq n$,

(vi) $B_n^m \Pi_n^i([x_1, \cdots, x_n]f_2) \cdots ([x_1, \cdots, x_n]f_m)$, if $e \equiv f_1 f_2 \cdots f_m$, $f_1, \cdots, f_m$ are primal components of e, and $f_1 \equiv x_i$ for some $1 \leq i \leq n$,

(vii) $B_n^{m-1} f_1([x_1, \cdots, x_n] f_2) \cdots ([x_1, \cdots, x_n] f_m)$, if $e \equiv f_1 f_2 \cdots f_m$, f_1 is the longest initial component of e such that x_i o$\not\subset f_1$ for all $1 \leq i \leq n$, and $f_2, \cdots, f_m$ are primal components of e.

Note that, for the cases (vi) and (vii), we decompose e into an initial component and a number of primal components, with the initial component chosen to be either a single variable among $x_1, \cdots, x_n$, if possible, or else the longest possible combinatory term not containing any of $x_1, \cdots, x_n$.

EXAMPLE. To find $[x, y, z]e$, where e is as defined in the previous example. Diagrammed below is the decomposition of e and its components according to the conditions prescribed above.



Hence, $[x, y, z]e \equiv B_3^3(SK)(B_1^2 I I_1^1(K_1(KK))(B_3^2 S(K_2 w)(B_3^1(SS)I_3^2))I$.

The correctness of our algorithm is stated in the following theorem, which can be easily proved from the above definitions by using induction on n and the construction of e.

THEOREM. *$[x_1, \cdots, x_n]e$ is an abstract of e with respect to $x_1, \cdots, x_n$.*

REFERENCES

[1] H. B. CURRY and R. FEYS, *Combinatory logic*, vol. 1, North-Holland, Amsterdam, 1958.

[2] R. J. ORGASS and F. B. FITCH, *A theory of programming languages*, *Studium Generale*, vol. 22 (1969), pp. 113–136.

[3] S. K. ABDALI, *A combinatory logic model of programming languages*, Ph.D. Dissertation, University of Wisconsin, 1974.

[4] H. B. CURRY, *Apparent variables from the standpoint of combinatory logic*, *Annals of Mathematics (2)*, vol. 34 (1933), pp. 381–404.

RENSSELAER POLYTECHNIC INSTITUTE
TROY, NEW YORK 12181