Nov 21 1977

John Backus

Class Notes. Red Languages

Programs for Transpose

I

The first "transpose" operator we define is neither "iterative" or "recursive" but is very wasteful of "storage" [i.e. creates a large intermediate operand].

Shorthand notations

(1) We shall write $\alpha\underline{c}$ as a shorthand for (AA, c). Thus $\alpha 1$ is short for $(AA, 1)$ and $\langle \alpha 1, (x_1, \ldots, x_n)\rangle \longrightarrow (\langle 1, x_1\rangle, \ldots, \langle 1, x_n\rangle)$. And $\alpha(C, A)$ is short for $(AA, (C, A))$ and $\alpha\alpha$ TAIL is short for $(AA, (AA, TAIL))$

(2) We shall write $x:y$ for $\mu\langle x, y\rangle$ and $x^n:y$ for $\mu\langle (x, \ldots x), y\rangle$ (n) and $x:y:z$ for $\mu\langle x\langle y z\rangle\rangle$.

(3) As noted before, $[f_1, \ldots, f_n]$ is short for $(NF, f_1, \ldots, f_n)$. Thus $[\emptyset]:x = (x)$ and $[f, g]:x = (f:x, g:x)$. [Except both yield $\S$ when $x = \S$.]

965 - IBM - 03

$$TRANS_1 \overset{df}{\equiv} (\alpha\alpha 1, f_1, APPLY, f_2)$$

$$f_2 \overset{df}{\equiv} [(\alpha(C, g), 1), [\phi]]$$

$$g \equiv (UN, [1, \alpha\alpha TAIL])$$

$$f_1 \equiv ((CN, p, TAIL, \S), ROTR)$$

$$p \equiv ((INSERT, AND), \alpha NULL, 1)$$

The reader can check the following:

(1) $[\phi]:x \equiv x = \S \rightarrow \S\ ;\ (x)$

(2) $(\alpha(C, g), 1):x \equiv x = (x_1, \ldots, x_n) \rightarrow$
$x_1 = \phi \rightarrow \phi$
$x_1 = (x_{11}, \ldots, x_{1m}) \rightarrow (\overbrace{g, \ldots, g}^{m})$
$T \rightarrow \S$

$T \rightarrow \S$

(3) $f_2:x \equiv ((\overbrace{g, \ldots, g}^{m}), (x))$ when x is a seq whose first element is a seq of length m [$\equiv (\phi\ (x))$ when $x_1 = \phi$]

$\equiv (\S, (x))$ when x is not as above ~~but not~~ and $x \neq \S$

$\equiv \S$ when $x = \S$

(4) $(APPLY, f_2):x \equiv g^{m}:(x)$ when x is as in first case above

$\equiv (x)$ when $m = 0$.

$\equiv \S$ otherwise

(5) $g:(x) \equiv UN:(x, (\alpha TL:x)) \equiv (x, \alpha TL:x)$

$g^2:(x) \equiv UN:(x, (\alpha TL:x, \alpha TL^2:x))$

$\equiv (x, \alpha TL:x, \alpha TL^2:x)$

$g^m:(x) \equiv (x, \alpha TL:x, \ldots, \alpha TL^m:x)$

(6) $(APPLY, f_2):x \equiv g^m:(x)$

$\equiv (x, \alpha TL:x, \ldots, \alpha TL^m:x)$

when $x = (\underbrace{(x_{11}, \ldots, x_{1m})}_{x_1}, \ldots, x_n)$

$\equiv (x, (TL:x_1, \ldots, TL:x_n), \ldots$

$(TL^m:x_1, \ldots TL^m:x_n))$

$\equiv \S$ otherwise

(7) $(f_1, APPLY, f_2):x \equiv (x, \alpha TL:x, \ldots, \underbrace{\alpha TL^{m-1}:x}_{n})$

when $\alpha TL^m:x = (\emptyset, \emptyset, \ldots, \emptyset)$ and ○

$\equiv \S$ otherwise

(8) $TRANS_1:x \equiv (\alpha 1:x, \alpha 1:\{\alpha TL:x\}, \ldots, \alpha 1:\{\alpha TL^{m-1}:x\})$

when ○

$\equiv ((x_{11}\ x_{21}, \ldots\ x_{n1}), (x_{12}, x_{22}, \ldots\ x_{n2})$

$\ldots\ (x_{1m}, x_{2m}, \ldots, x_{nm}))$

Thus $TRANS_1:((12)(34)(56)) \equiv ((135)(246))$

$TRANS_1:((123)) \equiv ((1)(2)(3))$

$TRANS_1:((12)(345)) \equiv \S$

$" :\emptyset \equiv \S$

$" :(\emptyset) \equiv \emptyset$

II Our second operator for Transpose is iterative:

$$TRANS_2 \overset{df}{\equiv} (f_1, (WHILE, p, f_2), f_3)$$

$$f_1 \equiv (CN, ((INSERT, AND), \alpha NULL, 2), 1, \phi)$$

$$f_2 \equiv [(UN, [(\alpha 1, 2), 1]), (\alpha ROTR, \alpha TAIL, 2)]$$

$$f_3 \equiv [(C, \emptyset), \alpha ROTR]$$

$$p \equiv (NOT, NULL, 1, 2)$$

We assert the following effects hold:

(1) $f_3 : x \equiv x = \phi \longrightarrow \phi \; ; \; (\emptyset, \alpha ROTR : x)$

(2) $\alpha ROTR : x \equiv x = \emptyset \longrightarrow \emptyset \; ; \; x \in A \longrightarrow \phi \; ;$
$x = (x_1, \ldots, x_n) \longrightarrow (ROTR : x_1, \ldots, ROTR : x_n)$

(3) $f_2 : (y, z) \equiv y = z = \emptyset \longrightarrow ((\phi)\ \emptyset) \; ;$
$y = (y_1, \ldots, y_n) \; \& \; z = \emptyset \longrightarrow ((\emptyset, y_1, \ldots, y_n), \emptyset)$
$y = \emptyset \; \& \; z = (z_1, \ldots, z_n)$
$\longrightarrow (((1 : z_1, \ldots, 1 : z_n)), (z_1', \ldots, z_n'))$
where $z_i' = (ROTR\ TAIL) : z_i$
$y = (y_1, \ldots, y_n) \; \& \; z = (z_1, \ldots, z_n)$
$\longrightarrow (((1 : z_1, \ldots, 1 : z_n), y_1, \ldots, y_n), (z_1', \ldots, z_n'))$

(4) $p : (y, z) \equiv z = (z_1, \ldots, z_n) \longrightarrow$
$\{ z_1 = \emptyset \longrightarrow F \; ; z_1 \neq \phi \longrightarrow \phi \; ;$
$T \longrightarrow T \}$
$T \longrightarrow \phi$

965 - IBM - 03

(5) $f_1 : (y,z) \doteq z = (\emptyset, \ldots, \emptyset) \longrightarrow y ; f$

Example

TRANS$_2$: ((1234)(5678))

$\xrightarrow{f_3}$ ($\emptyset$, ((4123)(8567)))

β_1 $\xrightarrow{P_1}$ T $\xrightarrow{f_2}$ (((4,8)), ((312)(756)))

$\xrightarrow{P_1}$ T $\xrightarrow{f_2}$ (((3,7),(4,8)), ((21)(65)))

$\xrightarrow{P_1}$ T $\xrightarrow{f_2}$ $\xrightarrow{P_1}$ $\xrightarrow{f_2}$ (((15)(26)(37)(48)), ($\emptyset$ $\emptyset$))

β_1 $\xrightarrow{P_1}$ F $\xrightarrow{f_1}$ ((15)(26)(37)(48))

III The third operator for transpose is recursively defined. The semantics of Red languages with a definition function δ are discussed on pp #9-23 of "Prog. lang. semantics & CALS". The idea is simple: if a definition: $\delta \underline{a} = \underline{c}$ is given, then the transition $\langle \underline{a}, x \rangle \longrightarrow \langle \underline{c}, x \rangle$ applies whenever the defined atom $\underline{a}$ is the operator of an innermost application [and the composition rule for $\langle (\underline{a} \ldots), x \rangle$ depends on whether $\underline{c}$ is regular or meta].

Note that Red languages with definitions "understand" nothing about recursive definitions beyond the fact that if there is a definition $\delta \underline{a} = \underline{c}$ then $\langle \underline{a} x \rangle$ can be replaced

965 - IBM - 03

by $\langle \underline{\varepsilon} x \rangle$. [plus the above rule about composition which does not concern us here]. This is unlike LISP, which understands all the far-reaching and subtle implications of a recursive definition of a function which must be used in evaluating the function. Despite this lack of understanding in Red semantics, the simple replacement of $\underline{a}$ by $\underline{\varepsilon}$ in $\langle \underline{a}, x \rangle$ permits recursive definitions $\S \underline{a} = \underline{\varepsilon}$ which work wherein $\underline{a}$ is a subexpression of $\underline{\varepsilon}$. This is illustrated by the following definition of $TRANS_3$:

$$\S TRANS_3 = (CN, p, f, g)$$
$$\text{where } p \equiv ((INSERT, AND), \alpha NULL)$$
$$f \equiv (C, \emptyset)$$
$$g \equiv (UN, [\alpha 1, (TRANS_3, \alpha TAIL)])$$

Assertions

(1) $p: x \equiv x \in A \rightarrow \S\,;\ x = (x_1, \ldots, x_n)\ \&\ x_i = \emptyset \rightarrow T_{\text{all } i}$;

(2) $g: x \equiv x \in A \rightarrow \S\,;$
$x = (x_1, \ldots, x_n) \rightarrow$ ~~$(1{:}x_1, \ldots, 1{:}x_n)$~~
$UN: (\alpha 1{:}x, TRANS_3: (TAIL{:}x, \ldots, TAIL{:}x_n))$

(3) $f: x \equiv x = \S \rightarrow \S\,;\ \emptyset$

Examples

(1) $TRANS_3 : X_1$ where $X_1 = ((12)(34)(56))$

$\equiv (CN, P, f, g) : X_1$ by substitution of rt. hand side of def for $TRANS_3$

$\equiv g : X_1$ since $P : X_1 = F$

$\equiv UN : ((1,3,5), TRANS_3 : ((2)(4)(6)))$

$\equiv UN : ((1,3,5), (CN, P, f, g) : X_2)$

where $X_2 = ((2)(4)(6))$; by subst. of def. for $TRANS_3$.

$\equiv UN : ((1,3,5), g : X_2)$ since $P : X_2 = F$

$\equiv UN : ((1,3,5), UN : ((2,4,6), TRANS_3 : (\emptyset, \emptyset, \emptyset)))$

$\equiv UN : ((1,3,5), UN : ((2,4,6), (CN, P, f, g) : (\emptyset, \emptyset, \emptyset)))$ by subst.

$\equiv UN : ((1,3,5), UN : ((2,4,6), f : (\emptyset, \emptyset, \emptyset)))$ since $p : (\emptyset\emptyset\emptyset) = F$

$\equiv UN : ((1,3,5), UN : ((2,4,6), \emptyset)))$ since $f : (\emptyset\emptyset\emptyset) = \emptyset$

$\equiv UN : ((1,3,5), ((2,4,6)))$

$\equiv ((1,3,5), (2,4,6))$

(2) $TRANS_3 : ((1,2))$ $\equiv g : ((12)) \equiv UN : ((1), g : ((2)))$

$\equiv UN : ((1), UN : ((2), g : (\emptyset)))$

$\equiv UN : ((1), UN : ((2), \emptyset))$

$\equiv ((1), (2))$