This newsletter presents, in revised SETL, the algorithm to produce the tables for the McKeeman parse. The APL program of newsletter No. 4 was produced from this routine, and so there is a close correspondence between these programs.

Input to PRECEDENCE is assumed to be the productions of a grammar, represented by a set $G$ of ordered $k$-tuples. Each $k$-tuple represents the characters of a single production, e.g., $A \leftarrow BCD$ is represented as $\langle A B C D \rangle$.

The procedure UNORDER, which converts an ordered $k$-tuple into an unordered set is used in forming the set $C$, the collection of the unique characters of the grammar.

$B\{X\}$ gives all syntactic types which begin with the character $X$. The set $B$ is initialized by entering for each $k$-tuple of $G$, an ordered pair consisting of the $k$-tuple's first two elements in reverse order. The subroutine COMPLETE then fills out $B$ by adding the elements $(X,Y)$ to $B$ if there exists a $Z$ such that $\langle X,Z \rangle \in B$ and $\langle Z,Y \rangle \in B$. Similarly, $E$ becomes the set of endings.

The desired table, $T$, for the McKeeman parse contains for each pair of characters in the grammar:

$$T(\langle I,J \rangle) = \begin{cases} 
1 & \text{if } I = J \\
2 & \text{if } I < J \\
3 & \text{if } I > J \\
5 & \text{if ambiguous} \\
0 & \text{if } I,J \text{ illegal}
\end{cases}$$

**Coding observation:** In SETL, the equivalent of the LISP MAPCAR
or MAPLIST can be coded in-line and without recursion. See, for example, the fourth line, which sets y to the last component of x, and the NEXTTO procedure, which searches the components of the k-tuple Y for equality with a given P. (Reference: Cocke and Schwartz, "Programming Languages and Their Compilers", pp. 152-171.)

Errata: Declarations of the following external variables are required:

<table>
<thead>
<tr>
<th>Routine</th>
<th>Variables</th>
</tr>
</thead>
<tbody>
<tr>
<td>PRECEDENCE</td>
<td>G, T</td>
</tr>
<tr>
<td>NEXTTO</td>
<td>G</td>
</tr>
<tr>
<td>SMALL</td>
<td>B</td>
</tr>
<tr>
<td>LARGE</td>
<td>E, B</td>
</tr>
</tbody>
</table>

DEFINE PRECEDENCE;

C=NL.; (∀X∈G) C=UNORDER. X U. C ;
B=NL.; (∀X∈G) ⟨*-X, *X⟩ IN. B ; COMPLETE B ;
E=NL.; (∀X∈G) Y = -X ; (WHILE PAIR. Y) ⟨-, Y⟩ Y ;
⟨Y, *X⟩ IN. E ; COMPLETE E ;
T=NL.; (∀X∈C, Y∈C)
T⟨X, Y⟩ = X NEXTTO. Y ; Z = X LARGE. Y + (X SMALL. Y);
IF Z ≠ T ⟨X, Y⟩ NE. 0 THEN (T⟨X, Y⟩) = 5 ;
ELSE T⟨X, Y⟩ = Z ;
DEFINE UNORDER. X ; P = X ; Q = NL.; (WHILE PAIR. P) ⟨*, P⟩ P IN. Q ;
RETURN Q WITH. P ; END UNORDER ;
DEFINE COMPLETE. M ; A = 0 ; (WHILE # M GT. A) A = # M ;
(∀Y ∈ M, X ∈ {-Y}) ⟨Y, X⟩ IN. M ; ; END COMPLETE ;
DEFINE F NEXTTO. Q; (∀x∈G) y = −x; (WHILE PAIR. Y)

IF P EQ. <*, X>Y \ AND Q EQ. *Y THEN RETURN 1;;

RETURN 0; END NEXTTO;

DEFINE F SMALL Q; (∀y∈B\{0\})

IF P NEXTTO. Y EQ. 1 THEN RETURN 2;; RETURN 0; END SMALL;

DEFINE F LARGE. Q; (∀y∈B\{0\}, y∈\mathcal{O}\{0\}); \ (\forall)

IF X NEXTTO. Y EQ. 1 THEN RETURN 3;; RETURN 0; END LARGE;

END PRECEDENCE;