American Journal of Computational Linguistics *Microfiche 55 : 84*

S N O P A R : A Grammar Testing System

T. P. Kehler

Department of Mathematics
Texas Woman's University
Denton, Texas 76204

A grammar testing program has been developed which permits modeling augmented transition network grammars as a series of SNOBOL4 functions. SNOPAR is designed for linguistics teaching and research. Emphasis is placed on the development of small to medium grammars in a variety of languages. The system has been used so far to develop a grammar of English for use in a transformational grammar course and to develop small grammars of a Nigerian and an American Indian language. Intended applications of SNOPAR are in field linguistics and grammar model testing.

The main part of the program is the routine PARSER. When PARSER is called with a lexicon and grammar, input strings are parsed according to the model grammar. The PARSER functions available for grammar development are CAT, PARSE, SETR, GETR, RESET, TESTR, GETF, GETCL, TO, BACK, FINDWRD, and BUILDS. The function operations and descriptions of their arguments are given in Table 1. After a parsing, PARSER returns control to the user permitting examination of stacks and registers at all

TABLE 1

PARSER FUNCTIONS

CAT	looks up the word class of the current first word in the input string. If the word is not in the lexicon an add routine is called which permits additions. If CAT succeeds by matching the current word class with its argument, the word is removed from the input string and pushed onto a stack (SAVEW). If it fails an alternate class is tested, provided that the alternate flag is on. Fail return leaves the surface string unaltered.
PARSE	calls the function given by its argument and if successful pushes the structure returned by the function onto a stack (SAVEQ) and assigns the structure to the Q register.
SETR	sets the values of registers. It has three arguments level, register name, and value. Each call of SETR causes the register name specified to be placed on a list for the specified level. SETR entries are treated as stacks, providing automatic saves for recursive calls.
GETR	returns the contents of the register name specified by its argument, and pops it off the stack saving the last value.
TESTR	looks at the value of the register name specified by its argument without popping it off the stack.
RESET	changes the value of a register without changing stack levels.
GETF	looks up the feature value for a feature specified by its argument of the current value of the word register. Any word can be specified by giving a second argument. If GETF fails for the word it looks at the root form of the word for certain features
GETCL	looks up the word class of the word specified by its argument.
TO	has as its argument, the new state label. It pushes the label onto a stack (PATH); outputs the state, outputs the contents of the Qregister, and transfers control to the new state.
BACK	backs to the state specified by its argument.
FINDWRD	tests for the word specified by its argument.
BUILDS	builds a structure from the register name list.

levels. In the examination stage, traces may be turned on lexical entries may be examined or minor changes to the grammar may be made. Functions available for the examination of stacks, registers and lexicon are POP, OUT, GETR, LOOKLEX, and TRACE. A function GETENG is also available for dictionary lookup in other languages. PARSER requires approximately 150 lines of SNOBOL code and is currently operating on a DEC 10. A batch version has been tested on an IBM 360

In order to use PARSER, a grammar and lexicon must be developed as disc files. Since the grammar is developed as a separate file different components of the grammar can be tested and put together in a variety of configurations. If a lexicon is not developed as a disc file prior to a parse, it may be entered from the terminal A simple grammar which produces surface structure trees is shown in Example 1 along with a sample parsing. A portion of the lexicon is shown at the bottom of the page. Example 2 shows the use of the GETF function to handle agreement between plural adjectives and a plural marker in Angas, a Nigerian language. Example 3 shows a grammar which handles sentence embedding in English. Some sample parsings are shown. The model used for the Example 3 grammar is basically the one developed in English Transformational Grammar by Jacobs and Rosenbaum. A basic case grammar for English as well as a semantic oriented grammar for Choctaw (an American Indian language) are in development.

The complete SNOPAR system has in addition to PARSER a routine for generating grammars from a state transition graph and a register action table. This routine called NEW guides the user through a state transition graph and register actions to produce a grammar compatible with PARSER. The SNOPAR NEW routine is still in development. The current routine allows development of small grammars. The new developments will provide diagnostics of grammar errors. SNOPAR also has a line editor (FIXUP) and disc I/O commands. The complete system allows repetitive testing of model grammars, permits editing; and has trace capabilities for grammar debugging.

Example 1

```
S       PARSE(NP())          :S(TO(.SNP))
        CAT('AUX')           :S(TO(.QUES))F(FRETURN)
SNP     SETR(.S,'TYPE','DCL')
        SETR(.S,'SUBJ',Q)
TRYVP   PARSE(VP())          :S(TO(.POPS))F(FRETURN)
QUES    SETR(.S,'TYPE','QUESTION')
        SETR(.S,'AUX',Q)
        SETR(.S,'TENSE',GETF('TNS'))
        PARSE(NP())          :S(TO(.QNP))F(FRETURN)
QNP     SETR(.S,'SUBJ',Q)            :(TO(.TRYVP))
POPS    SETR(.S,'PRED',Q)
        $ = BUILDS(S)        :(RETURN)
NP      CAT('DET')           :S(TO(.DET))
        CAT('PRO')           :S(TO(.PRO))
        CAT('NPR')           :S(TO(.NPR))F(FRETURN)
NPR     SETR(.NP,'PROP',Q)           :(TO(.POPNP))
PRO     SETR(.NP,'PRO',Q)            :(TO(.POPNP))
DET     SETR(.NP,'DET',Q)
ADJ     CAT('ADJ')           :F(TO(.TRYN))
        SETR(.NP,'ADJ',Q)            :(TO(.ADJ))
TRYN    CAT('N')             :F(FRETURN)
        SETR(.NP,'N',Q)
TRYPP   PARSE(PP())          :F(TO(.POPNP))
        SETR(.NP,'PP',Q)             :(TO(.TRYPP))
POPNP   NP = BUILDS(NP)      :(RETURN)
PP      CAT('PREP')          :F(FRETURN)
        SETR(.PP,'PREP',Q)
        PARSE(NP())          :F(FRETURN)
        SETR(.PP,'PREPNP',Q)
        PP = BUILDS(PP)      :(RETURN)
VP      CAT('V')             :F(FRETURN)
        SETR(.VP,'V',Q)
        PARSE(NP())          :S(TO(.VNP))
TRYVPP  PARSE(PP())          :F(TO(.POPVP))
        SETR(.VP,'PP',Q)             :(TO(.TRYVPP))
VNP     SETR(.VP,'NP',Q)             :(TO(.POPVP))
POPVP   VP = BUILDS(VP)      (RETURN)

 TY LEXENG.1
        DID= (AUX)(TNS PAST).
        CAN= (AUX)(TNS PRES).
        COULD= (FORM 'CAN).
        WILL= (AUX)(TNS FUT).
        THE= (DET).
        A= (DET).
        AN= (DET).
        THAT= (CLIND).
        BOY= (N)(NBR SING).
        BOYS= (N)(NBR PL).
        GIRL= (N)(NBR SING).
        GIRLS= (FORM GIRL)(NBR PL).
        MAN= (N)(NBR SING).
        MEN= (N)(NBR PL).
        WOMAN= (N)(NBR SING).
        WOMEN= (N)(NBR PL).
        TABLE= (N)(NBR SING).
```

```
DID YOU WALK TO THE VILLAGE
DID YOU WALK TO THE VILLAGE
STATE QUES
COMPLEMENT STRING: YOU WALK TO THE VILLAGE
BUILD STRUCTURE DID
STATE PRO
COMPLEMENT STRING: WALK TO THE VILLAGE
BUILD STRUCTURE YOU
STATE POPNP
COMPLEMENT STRING: WALK TO THE VILLAGE
BUILD STRUCTURE YOU
STATE QNP
COMPLEMENT STRING: WALK TO THE VILLAGE
BUILD STRUCTURE (NP(PRO YOU))
STATE TRYVP
COMPLEMENT STRING: WALK TO THE VILLAGE
BUILD STRUCTURE (NP(PRO YOU))
STATE DET
COMPLEMENT STRING: VILLAGE
BUILD STRUCTURE THE
STATE TRYN
COMPLEMENT STRING: VILLAGE
BUILD STRUCTURE
STATE POPNP
COMPLEMENT STRING:
BUILD STRUCTURE VILLAGE
STATE TRYVPP
COMPLEMENT STRING:
BUILD STRUCTURE (PP(PREP TO)(PREPNP (NP(DET THE)(N VILLAGE))))
STATE POPVP
COMPLEMENT STRING:
BUILD STRUCTURE (PP(PREP TO)(PREPNP (NP(DET THE)(N VILLAGE))))
STATE POPS
COMPLEMENT STRING:
BUILD STRUCTURE:
(VP(V WALK)(PP (PP(PREP TO)(PREPNP (NP(DET THE)(N VILLAGE))))))

STATE S
COMPLEMENT STRING:
BUILD STRUCTURE:
(S(TYPE QUESTION)(AUX DID)(TENSE PAST)(SUBJ (NP(PRO YOU)))
(PRED (VP(V WALK)(PP (PP(PREP TO)(PREPNP (NP(DET THE)(N VILLAGE)))))))
)


DO YOU WANT TO EXAMINE THE REGISTERS ?
YES
*
OT      OUTPUT = POP(PATH)      ;S(OT)F(EXAS\S\MIN)
EP\P\OF
POPS
POPVP
TRYVPP
POPNP
TRYN
DET
TRYVP
QNP
POPNP
PRO
QUES
DO YOU WANT TO EXAMINE THE REGISTERS ?
^C
```

ANGAS NOUN PHRASE

```
NP      CAT('NOUN')          :S(TO(.POS))F(FRETURN)
POS     SETR(.NP,'NOUN',Q)
        CAT('POSPRO')        :F(TO(.KI))
        SETR(.NP,'POSPRO',Q)      :(TO(.ADJ))
KI      CAT('KI')            :F(TO(.ADJ))
        SETR(.NP,'POSS',Q)
        CAT('NPR')           :F(FRETURN)
        SETR(.NP,'NPR',Q)
ADJ     CAT('ADJ')           :F(TO(.KOM))
        SETR(.NP,'ADJ',Q)          :(TO(.DET))
KOM     FINDWRD('KOMEYE')          :F(TO(.DET))
        SETR(.NP,'REL',Q)           :(TO(.ADJ))
DET     CAT('DET')           :F(TO(.PL))
        SETR(.NP,'DET',Q)
PL      CAT('PL')            :S(TO(.PLT))
        IDENT(GETF('PL',GETR('ADJ')),'PL')        :S(FRETURN)F(TO(.NUM))
PLT     SETR(.NP,'PL',Q)
        IDENT(GETF('PL',GETR('ADJ')),'-PL')       :S(FRETURN)
NUM     CAT('NUM')           :F(TO(.POPNP))
        SETR(.NP,'NUM',Q)
        IDENT(WORD,'BAP')          :S(TO(.TMWA))F(TO(.POPNP))
TMWA    IDENT(GETR('PL'),'MWA') :F(FRETURN)
POPNP   NP = BUILDS(NP) :(RETURN)
EOG
END
```

ANGAS LEXICON

```
L       L<'AS'> = '(NOUN)(ENG DOG)'
        L<'MAT'> = '(NOUN)(ENG WOMAN)'
        L<'FANA'> = '(POSPRO)(ENG,MY)'
        L<'RIIT'> = '(ADJ)(PL -PL)(ENG GOOD)'
        L<'RIIT-RIIT'> = '(ADJ)(PL PL)(ENG GOOD)'
        L<'BIJIM'> = (ADJ)(PL -PL)(ENG BIG)'
        L<'NAN-NAN'> = '(ADJ)(PL PL)(ENG BIG)'
        L<'GAK'> = '(NUM)(ENG ONE)'
        L<'BAP'> = '(NUM)(ENG TWO)'
        L<'NYII'> = '(DET)(ENG THIS)'
        L<'DA'> = '(DET)(ENG THE)'
        L<'CE'> = '(DET)(ENG A)'
        L<'MWA'> = '(PL)(ENG PLUR)'
        L<'BULUS'> = '(NPR)(ENG NAME)
        L<'KI'> = '(KI)(ENG POSSESSIVE)
*EX$$
```

```
STATE POPNP
COMPLEMENT STRING:
BUILD STRUCTURE MWA
STATE NP
COMPLEMENT STRING:
BUILD STRUCTURE (NP(NOUN AS)(POSPRO FANA)(ADJ NAN-NAN)(DET CE)(PL MWA)
)
ENGLISH: DOG MY BIG A PLUR
DO YOU WANT TO EXAMINE THE REGISTERS ?
NO
INPUT STRUCTURE TO BE PARSED
AS FANA BIJIM CE MWA
AS FANA BIJIM CE MWA
STATE POS
COMPLEMENT STRING: FANA BIJIM CE MWA
BUILD STRUCTURE AS
STATE ADJ
COMPLEMENT STRING: BIJIM CE MWA
BUILD STRUCTURE FANA
STATE DET
COMPLEMENT STRING: CE MWA
BUILD STRUCTURE BIJIM
STATE PLT
COMPLEMENT STRING:
BUILD STRUCTURE MWA
STATE NP
COMPLEMENT STRING: DID NOT PARSE
BUILD STRUCTURE MWA
DO YOU WANT TO EXAMINE THE REGISTERS ?
NO
INPUT STRUCTURE TO BE PARSED
AS MWA
AS MWA
STATE POS
COMPLEMENT STRING: MWA
BUILD STRUCTURE AS
STATE KI
COMPLEMENT STRING: MWA
BUILD STRUCTURE
STATE ADJ
COMPLEMENT STRING: MWA
BUILD STRUCTURE
STATE KOM
COMPLEMENT STRING: MWA
BUILD STRUCTURE
STATE DET
COMPLEMENT STRING: MWA
BUILD STRUCTURE
STATE PL
COMPLEMENT STRING: MWA
BUILD STRUCTURE
STATE PLT
COMPLEMENT STRING:
BUILD STRUCTURE MWA
STATE POPNP
COMPLEMENT STRING:
BUILD STRUCTURE MWA
STATE NP
COMPLEMENT STRING:
BUILD STRUCTURE (NP(NOUN AS)(PL MWA))
ENGLISH: DOG PLUR
DO YOU WANT TO EXAMINE THE REGISTERS ?
NO
```

```
        FUNCTION DEFINITIONS
GRAM     DEFINE('S()N')
         DEFINE('ES()')
         DEFINE('NP()M,N')
         DEFINE('PP()')
         DEFINE('VP()N')
         DEFINE('IO()')
*        S PARSER
         PARSE(S())
         OUT('S',STR,Q)   :(NXT,COM)
S        PARSE(NP())              :S(TO(,SNP))
         CAT(,AUX)              :S(TO(,Q))
         PARSE(VP())      :S(TO(,IMP))F(FRETURN)
SNP      SETR(,S,'SUBJ',Q)
         SETR(,S,'TYPE','DCL')
         PARSE(VP())      :S(TO(,POPS))
         CAT(,AUX)        :S(TO(,AX))F(FRETURN)
IMP      SETR(,S,'TYPE','IMP')
         SETR(,S,'SUBJ','(PRO YOU)')      :(TO(,POPS))
Q        SETR(,S,'AUX',Q)
         SETR(,S,'TNS',GETF('TNS'))
         SETR(,S,'TYPE','Q')
         PARSE(NP())                          :S(TO(,QNP))F(FRETURN)
AX       SETR(,S,'AUX',Q)
         SETR(,S,'TNS',GETF('TNS'))
         FINDWRD('HAVE') SETR(,S,'HA','HAVE')
         PARSE(VP())                           :S(TO(,POPS))F(FRETURN)
QNP      SETR(,S,'SUBJ',Q)
         FINDWRD('HAVE') SETR(,S,'HA','HAVE')
         PARSE(VP())                      :S(TO(,POPS))F(FRETURN)
POPS     SETR(,S,'PRED',Q)
         # = BUILDS('/S/TYPE/SUBJ/PRED/')                    :(RETURN)
* NP PARSER
NP       CAT('DET')                            :S(TO(,DET))
         CAT('PRO')                    :S(TO(,PRO))
         CAT('NPR')                           :S(TO(,NPR))
         PARSE(ES())      :S(TO(,NPES))
                                                  :(TO(,PLNP))
DET      SETR(,NP,'DET',Q)
ADJ      CAT('ADJ')                           :F(TO(,N))
         SETR(,NP,'ADJ' M,Q)
         BUMP('M')                             :(TO(,ADJ))
N        CAT('N')                      :F(FRETURN)
         SETR(,NP,'N',Q)                       :(TO(,NPP))
POSPRO   SETR(,NP,'PRO',Q)             :(TO(,ADJ))
NPR      SETR(,NP,'NPR',Q)
         IS(GETF('CASE'),'POS') CHGNAM('NP','NPR','POSNPR')  :F(TO(,NPP))
         PARSE(ES())      :F(TO(,ADJ))
         SETR(,NP,'POSS',Q)
         NP = BUILDS('/NP/NPR/POSS/')      :(RETURN)
POPNP    PARSE(ES())      :S(TO(,NPES))
         NP = BUILDS(NP)  :(RETURN)
NPP      PARSE(PP())                   :F(TO(,POPNP))
         SETR(,NP,'NPP' N,Q)
         BUMP('N')                        :(TO(,NPP))
PRO      GETF('CASE') 'POS'            :S(TO(,POSPRO))
         SETR(,NP,'PRO',Q)             :(TO(,POPNP))
PLNP     CAT('ADJ')          :F(TO(,NPL))
         SETR(,NP,'ADJ' ,Q)
         :(TO(,PLNP))
```

```
NPL     CAT('N')        IS(GETF('NBR'),'PL')          :F(FRETURN)
        SETR(,NP,'N',Q)                        :(TO(,POPNP))
NPES    SETR(,NP,'COMP',Q)
        NP = BUILDS(NP) :(RETURN)
*               PP PARSER
PP      CAT('PREP')     :S(TO(,PREP))F(FRETURN)
PREP    P<'PREP'> = Q
        PP = '(PREP ' P<'PREP'>   NP()
        ')'   :S(RETURN)F(FRETURN)
*       VP PARSER
VP      CAT('V')              :F(TO(,AUXBE))
        SETR(,VP,'TNS',GETF('TNS'))
        HASNAM('S','AUX') GETR('TNS')
        IS(GETF('VTYP'),'TRANS')
        :S(TO(,TRANS))F(TO(,ITRAN))
TRANS   SETR(,VP,'VT',Q)                       :(TO(,VNP))
ITRAN   SETR(,VP,'V',Q)              :(TO(,NTPP))
TADJ    CAT('ADJ')                   :S(TO(,VADJ))
        PARSE(NP())                  :S(TO(,NTNP))
NTPP    PARSE(PP())                  :F(TO(,POPVP))
        SETR(,VP,'VPP' N,Q)
        BUMP('N')           :(TO(,NTPP))
VADJ    SETR(,VP,'ADJ',Q)
        PARSE(ES())         :S(TO(,VADJES))
        VP = BUILDS(VP)
VDJPP   PARSE(PP())         :F(RETURN)
        VP = VP Q           :(TO(,VDJPP))
VADJES  SETR(,VP,'ADJES',Q)        :(TO(,POPVP))
NTNP    SETR(,VP,'NTNP',Q)                 :(TO(,POPVP))
VNP     PARSE(IO())         :S(TO(,IOI))
        PARSE(NP())         :F(FRETURN)
        SETR(,VP,'OBJ',Q)
        PARSE(IO())                  :S(TO(,IOL))F(TO(,POPVP))
IOI             SETR(,VP,'IO',Q)
        PARSE(NP())                  :S(TO(,VIONP))F(FRETURN)
VIONP   SETR(,VP,'OBJ',Q)                      :(TO(,POPVP))
IOL     SETR(,VP,'IO',Q)                       :(TO(,POPVP))
AUXBE   CAT('V','ALT')  :S(TO(,BE))
        IS(TESTR('TYPE'),'Q') IS(TESTR('AUX'),'BE')      :F(TO(,TRYES))
        SETR(,VP,'V',GETR('AUX'))              :(TO(,PAS))
TRYES   IS(TESTR('IF')) PARSE(ES())            :F(FRETURN)
        NP = Q  :(RETURN)
BE      SETR(,VP,'V',Q)
        SETR(,VP,'TNS',GETF('TNS'))
PAS     CAT('V')                     :F(TO(,TADJ))
        WORD 'ING'                   :S(TO(,ING))
        IS(GETF('VTYP'),'TRANS')               :F(FRETURN)
        GETF('TNS') 'PPRT'           :S(TRPAS)F(FRETURN)
ING     VP =
        SETR(,VP,'AUX','BE')
        SETR(,VP,'TNS','PPRG')
        SETR(,VP,'V',Q)
        PARSE(NP())         :S(TO(,PRNP))
        VP = BUILDS(VP)
PRPP    PARSE(PP())         :F(RETURN)
        VP = VP Q           :(TO(,PRPP))
PRNP    SETR(,VP,'PRNP',Q)             :(TO(,POPVP))
TRPAS   VP =
        SETR(,VP,'AUX','BE')
        DIFFER(TESTR('TYPE'),'Q') SETR(,VP,'TNS','PPRT')
```

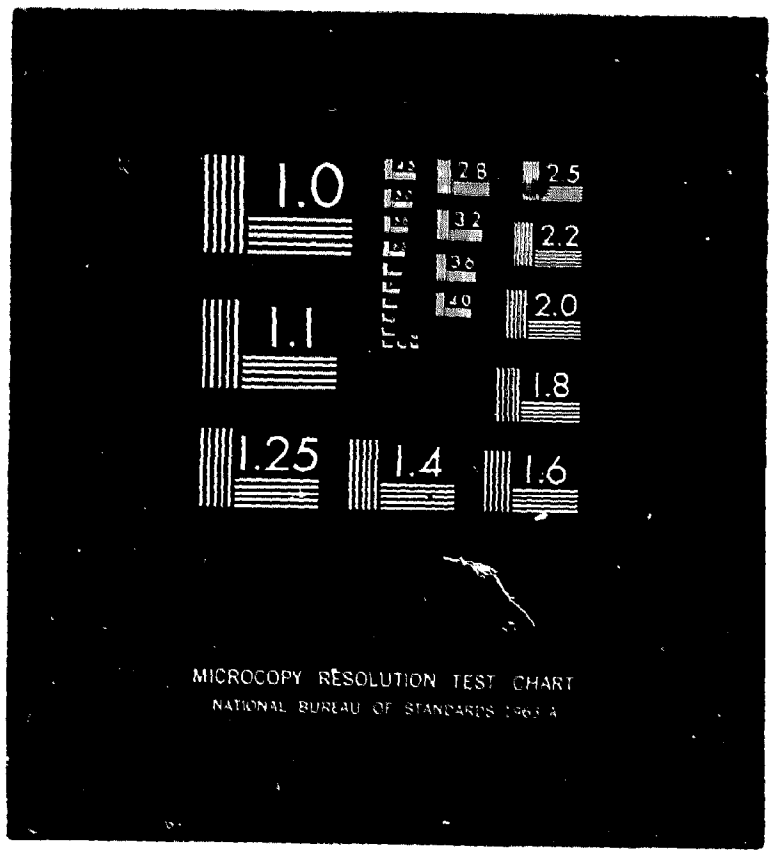
```
        GETR('V')
        SETR(,VP,'V',Q)
        PARSE(IO())..     :S(TO(,PIO))
        FINDWRD('BY')              :S(TO(,PNPTST))
        FINDWRD('FROM')            :S(TO(,PNPTST))
        PARSE(ES())       :S(TO(,VPES))F(TO(,CHGSBJ))
CHGSBJ  SETR(,VP,'OBJ',R<'SUBJ'>)
        IS(TESTR('TYPE'),'DCL') RESET('TYPE','TRPAS')
        IS(TESTR('TYPE'),'Q') RESET('TYPE','QPAS')
        RESET('SUBJ','SOMEONE')           :(TO(,POPVP))
VPES    SETR(,VP,'OBJES',Q)
        VP = BUILDS(VP)
TRPP    PARSE(PP())       :F(RETURN)
        VP = VP Q         :(TO(,TRPP))
PIO     SETR(,VP,'IO',Q)
        FINDWRD('BY')     :S(TO(,PNPTST))
        FINDWRD('FROM')            :S(TO(,PNPTST))F(FRETURN)
PNPTST  PARSE(NP())       :S(TO(,PNP))F(FRETURN)
PNP     SETR(,VP,'OBJ',GETR('SUBJ'))
        RESET('SUBJ',Q)
        IS(TESTR('TYPE'),'DCL') RESET('TYPE','TRPAS')
        IS(TESTR('TYPE'),'Q') RESET('TYPE','QPAS')
        IS(R<'IO'>)                :F(TO(,POPVP))
        PARSE(IO())                :S(TO(,PIOL))F(TO(,POPVP))
PIOL    SETR(,VP,'IO',Q)           :S(TO(,POPVP))
POPVP   VP = BUILDS(VP)            :(RETURN)
*       INDIRECT OBJECT
IO      FINDWRD('TO')              :S(TO(,IOTO))
        FINDWRD('FOR')             :S(TO(,IOFOR))F(FRETURN)
IOTO    SETR(,IO,'PREP','TO')
        PARSE(NP())                :S(TO(,IONP))
ADD,TO  STR = 'IO ' STR  :(FRETURN)
IOFOR   SETR(,IO,'PREP',Q)
        PARSE(NP())       :S(TO(,IONP))
        STR = 'FOR ' STR           :(FRETURN)
IONP    SETR(,IO,'IONP',Q)
        IO = BUILDS(IO)  :(RETURN)
ES      CAT('CLIND')      :S(TO(,TES))
        FINDWRD('TO')     :S(TO(,THV))
        FINDWRD('HAVING')          :S(TO(,ESVP))
        IS(GETF('TNS'),'PPRG')     :F(FRETURN)
        PARSE(VP())       :F(FRETURN)
        ES = Q   :(RETURN
TES     SETR(,ES,'CL,IND ',Q)
        PARSE(S())        :S(TO(,POPES))F(FRETURN)
THV     SETR(,ES,'INF', TO')
        FINDWRD('HAVE')   :S(TO(,ESVP))
        PARSE('    '))    :F(ADD,TO)
        SETR(,ES,'ESVP',Q)
        ES = BUILDS('/S/TYPE/SUBJ/ESVP/')          :(RETURN)
ESVP    SETR(,ES,'AUX','HAVE')
        PARSE(VP())       :F(FRETURN)
        SETR(,ES,'ESVP',Q)
        ES = BUILDS('/S/TYPE/SUBJ/AUX/ESVP/')      :(RETURN)
POPES   ES = Q   :(RETURN)
END
```

```
JOHN MAY BELIEVED TO HAVE BEEN DELAYED
JOHN MAY BELIEVED TO HAVE BEEN DELAYED
STATE 2
COMPLEMENT STRING: DELAYED
BUILD STRUCTURE:
(1 (TYPE TRANS) (SUBJ SOMEONE) (PRED (VP (AUX BE) (TNS PERF PPRT)
(V BELIEVE) (OBJ1 (1 (TYPE TRANS) (SUBJ SOMEONE) (AUX HAVE)
(E1VP (VP (AUX BE) (TNS PPRT) (V DELAY)) (OBJ (NP (NPP JOHN)))))))))

DO YOU WANT TO EXAMINE THE REGISTERS ?
NO
INPUT STRUCTURE TO BE PARSED
I WANT TO GO
I WANT TO GO
STATE 1
COMPLEMENT STRING: GO
BUILD STRUCTURE:
(1 (TYPE DCL) (SUBJ (NP (PRO I))) (PRED (VP (TNS PRES) (VT WANT)
(OBJ (NP (COMP (1 (TYPE DCL) (SUBJ (NP (PRO I))) (E1VP (VP (V GO)
(TNS PRES)))))))))))
DO YOU WANT TO EXAMINE THE REGISTERS ?
NO
INPUT STRUCTURE TO BE PARSED
I THINK THAT I SAW YOU WITH HER
I THINK THAT I SAW YOU WITH HER
SAW NOT IN LEXICON
LEXICON ADD. TO ABORT- PARSE TYPE STOP. ELSE TYPE YES
YES
WORD?
SAW
FEATURE STRING
(VA (VTYP TRANS) (TNS PAST))
WORD?

STATE 3
COMPLEMENT STRING: HER
BUILD STRUCTURE:
(1 (TYPE DCL) (SUBJ (NP (PRO I))) (PRED (VP (TNS PRES) (VT THINK)
(OBJ (NP (COMP (1 (TYPE DCL) (SUBJ (NP (PRO I))) (PRED (VP (TNS PAST)
(VT SAW) (OBJ (NP (PRO YOU)) (PREP WITH (NP (PRO HER)))))))))))))
```

```
INPUT STRUCTURE TO BE PARSED
JOHN'S BELIEVING THAT MARY IS GOING TO THE VILLAGE IS MYSTERIOUS
JOHN'S BELIEVING THAT MARY IS GOING TO THE VILLAGE IS MYSTERIOUS
STATE S
COMPLEMENT STRING: MYSTERIOUS
BUILD STRUCTURE:
(S(TYPE DCL)(SUBJ (NP(NPR JOHN'S)(POSS (VP(TNS PRES)(VT BELIEVE)
(OBJ (NP(COMP (S(TYPE DCL)(SUBJ (NP(NPR MARY)))(PRED (VP(AUX BE)
(TNS PRNG)(V GO))(PREP TO(NP(DET THE)(N VILLAGE)))))))))))
(PRED (VP(V BE)(TNS PRES)(ADJ MYSTERIOUS))))
DO YOU WANT TO EXAMINE THE REGISTERS ?
NO
INPUT STRUCTURE TO BE PARSED
THAT HE,BROKE HER DISH IS SERIOUS
THAT HE BROKE HER DISH IS SERIOUS
STATE S
COMPLEMENT STRING: SERIOUS
BUILD STRUCTURE:
(S(TYPE DCL)(SUBJ (NP(COMP (S(TYPE DCL)(SUBJ (NP(PRO HE)))
(PRED (VP(TNS PAST)(VT BREAK)(OBJ (NP(PRO HER)(N DISH))))))))
(PRED (VP(V BE)(TNS PRES)(ADJ SERIOUS))))
DO YOU WANT TO EXAMINE THE REGISTERS ?
NO
INPUT STRUCTURE TO BE PARSED
THE BOY BREAKING THE GLASS IS MULLIGAN
THE BOY BREAKING THE GLASS IS MULLIGAN
STATE S
COMPLEMENT STRING: MULLIGAN
BUILD STRUCTURE:
(S(TYPE DCL)(SUBJ (NP(DET THE)(N BOY)(EMB (VP(TNS PPRG)(VT BREAK)
(OBJ (NP(DET THE)(N GLASS))))))(PRED (VP(V BE)(TNS PRES)
(NTNP (NP(NPR MULLIGAN)))))
DO YOU WANT TO EXAMINE THE REGISTERS ?
NO
INPUT STRUCTURE TO BE PARSED
JOHN'S BEING THIN IS NICE
JOHN'S BEING THIN IS NICE
STATE S
COMPLEMENT STRING: NICE
BUILD STRUCTURE:
(S(TYPE DCL)(SUBJ (NP(NPR JOHN'S)(POSS (VP(V BE)(TNS PPRG)
(ADJ THIN))))(PRED (VP(V BE)(TNS PRES)(ADJ NICE))))
DO YOU WANT TO EXAMINE THE REGISTERS ?
NO
INPUT STRUCTURE TO BE PARSED
TO BE A MAN WAS HIS DREAM
TO BE A MAN WAS HIS DREAM
STATE S
COMPLEMENT STRING: DREAM
BUILD STRUCTURE:
(S(TYPE DCL)(SUBJ (NP(COMP (S(TYPE )(SUBJ )(ESVP (VP(V BE)
(TNS PRES)(NTNP (NP(DET A)(N MAN))))))))(PRED (VP(V BE)
(TNS PAST))))
DO YOU WANT TO EXAMINE THE REGISTERS ?
NO
INPUT STRUCTURE TO BE PARSED
BREAKING DISHES IS RECKLESS
BREAKING DISHES IS RECKLESS
STATE S
COMPLEMENT STRING: RECKLESS
BUILD STRUCTURE:
(S(TYPE DCL)(SUBJ (NP(COMP (VP(TNS PPRG)(VT BREAK)(OBJ (NP(N DISHES))
```

```
THE BOY RUNNIJ\J\NG TO THE HOUSE IS JOHN
THE BOY RUNNING TO THE HOUSE IS JOHN
STATE S
COMPLEMENT STRING: JOHN
BUILD STRUCTURE:
(S(TYPE DCL)(SUBJ (NP(DET THE)(N BOY)(EMB (VP(V RUN)(TNS PPRG)
(VPP (PREP TO(NP(DET THE)(N HOUSE))))))))(PRED (VP(V BE)
(TNS PRES)(NTNP (NP(NPR JOHN))))))
DO YOU WANT TO EXAMINE THE REGISTERS ?
NO
INPUT STRUCTURE TO BE PARSED
BREAKING DISHES IS RECKLESS
BREAKING DISHES IS RECKLESS
STATE S
COMPLEMENT STRING: RECKLESS
BUILD STRUCTURE:
(S(TYPE DCL)(SUBJ (NP(COMP (VP(TNS PPRG)(VT BREAK)(OBJ (NP(N DISHES)))
))))
(PRED (VP(V BE)(TNS PRES)(ADJ RECKLESS))))
DO YOU WANT TO EXAMINE THE REGISTERS ?
NO
INPUT STRUCTURE TO BE PARSED
I WAS THINKING THAT YOU B\B\WERE CONSERVATIVE
I WAS THINKING THAT YOU WERE CONSERVATIVE
STATE S
COMPLEMENT STRING: CONSERVATIVE
BUILD STRUCTURE:
(S(TYPE DCL)(SUBJ (NP(PRO I)))(PRED (VP(AUX BE)(TNS PPRG)
(V THINK)(PRNP (NP(COMP (S(TYPE DCL)(SUBJ (NP(PRO YOU)))
(PRED (VP(V BE)(TNS PAST)(ADJ CONSERVATIVE)))))))))))
DO YOU WANT TO EXAMINE THE REGISTERS ?
NO
INPUT STRUCTURE TO BE PARSED
JOHN WAS BELIEVED TO BE DELAYED
JOHN WAS BELIEVED TO BE DELAYED
STATE S
COMPLEMENT STRING: DELAYED
BUILD STRUCTURE:
(S(TYPE TRPAS)(SUBJ SOMEONE)(PRED (VP(AUX BE)(TNS PRES)(V BELIEVE)
(OBJES (S(TYPE TRPAS)(SUBJ SOMEONE)(ESVP (VP(AUX BE)(TNS PPRT)
(V DELAY)(OBJ (NP(NPR JOHN)))))))))
DO YOU WANT TO EXAMINE THE REGISTERS ?
NO
INPUT STRUCTURE TO BE PARSED
THAT HE BROKE HER DISH WAS SERIOUS
THAT HE BROKE HER DISH WAS SERIOUS
STATE S
COMPLEMENT STRING: SERIOUS
BUILD STRUCTURE:
(S(TYPE DCL)(SUBJ (NP(COMP (S(TYPE DCL)(SUBJ (NP(PRO HE)))
(PRED (VP(TNS PAST)(VT BREAK)(OBJ (NP(PRO HER)(N DISH)))))))))
(PRED (VP(V BE)(TNS PAST)(ADJ SERIOUS))))
DO YOU WANT TO EXAMINE THE REGISTERS ?
YES
```

END


1.0
1.1
1.25
1.4
1.6
1.8
2.0
2.2
2.5
2.8
3.2
3.6
4.0
MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS 1963 A