INTERPRETIVE LOGIC

Wil Baden
339 Princeton Drive
Costa Mesa CA 92626

A programmer's best friend is his source-code editor. For Forth programmers the relationship can be particularly intimate. A Forth editor will often do very kinky things for the programmer.

The next best friend that a programmer can have is the ability to do conditional interpretation and conditional compilation, both from immediate console entry and from a prepared script. In Unix (tm) parlance this is the Shell.

The Forth-83 Uncontrolled Reference Set has three words, IFTRUE , OTHERWISE , and IFEND , inherited from the Forth-79 Reference Set, whose meaning is specified by:

```
IFTRUE            flag   --                     interpret  only
    Begin an
        IFTRUE ... OTHERWISE ... IFEND
    conditional sequence.  These conditional words operate like
        IF ... ELSE ... THEN
    except  that they cannot be nested, and are to be used only
    during interpretation.  In conjunction with the words [ and
    ] they may be used within a  colon  definition  to  control
    compilation, although they are not to be compiled.
```

There is no reason why these could not and should not be nested. If they are STATE-smart they do not have to bear new names, which will make them easier to learn.

One method of implementation is to scan the source input for the relevant key words and perform an appropriate action for each. The string processing routines needed to do this already exist in Forth -- WORD , FIND and EXECUTE . An easy way to restrict the words that are recognized is to use a sealed vocabulary containing just those words and no others.

After IF , ELSE and THEN are defined it is a small step to BEGIN , WHILE , REPEAT and UNTIL . Another small step will lead to DO and LOOP . This extends Forth to become a very powerful job control language.

With a few extensions to the line editing words such as found in Laxen-Perry's F83 Model or Leo Brodie's Starting Forth we get a powerful programmable macro editor.

Some examples of use are found on the load block.

```
CR .( INTERPRETIVE VERSION OF LOGIC STRUCTURES.)   CAPS ON

3 7 THRU  ( IF ELSE THEN   HAVE   BEGIN WHILE REPEAT UNTIL )


STRETCH FORTH    MS-DOS    WWB 851019
```

```
CR .( Do you want interpretive DO ? )
KEY ASCII _ AND   ASCII Y =
if CR .( And do you want interpretive FOR ? )
   KEY ASCII _ AND   ASCII Y =
   if    8 10 THRU        ( DO FOR LOOP I )
   else  9 10 THRU then   ( DO LOOP I )
then

have PRINT not if 11 LOAD then
have CONSULT if 2 2 CONSULT then
```

First we load IF ELSE THEN BEGIN WHILE REPEAT UNTIL from blocks 3 through 7. We then ask whether interpretive DO (and LOOP) are wanted. KEY is used to accept your answer, and if it was uppercase or lowercase Y we ask if you also want interpretive FOR . The appropriate blocks are loaded.

IF and ELSE are very useful with conditional compilation. We can compile or execute different things depending on the value of a variable or computed expression. We can compile different systems from just one source program. The word HAVE can be used to test whether we already have a word, i.e., whether it has already been defined.

The code is compatible with the Laxen-Perry F83 Model (and of course my F83X). A simpler, more powerful, and easier to use definition of SEAL is given.

SEAL
   Usage: SEAL will change the search order such that only the first vocabulary in the search order will be searched. The other vocabularies in the search order are present and unchanged but do not participate in the search.

UNSEAL
   Allow all vocabularies in the search order to be searched.

PASSOVER is used by the system to pass over all words but those in the vocabulary CONDITIONAL . Those words and only those words will be executed.

The word FOR has been introduced for interpretive counted loops. In most applications it is more convenient to express a range as <first>,<last> rather than <last+1>,<first> . When a word is compiled it is trivial to say "1+ SWAP" before "DO" but it becomes a nuisance and a potential source of confusion when interpreting. FOR also checks that there are at least two arguments on the stack. A compilation version of FOR has been given for completeness.

The loop index I presents a problem when the editor words include an I also. We use conditional compilation to check whether the editor does contain I and when it does we define ^ (caret or up-arrow) as a synonym of the editor I for use with interpretive logic. Conditionally selected system dependent code is used to check the depth of the return stack so that the editor I can still be used outside of interpretive logic.

The use of interpretive logic is simplified when entering commands from the keyboard by not requiring THEN , REPEAT or LOOP at the end of a line. They are required on a source block, or when you want to do something after the interpretion of the control structure, or when compiling. With interpretive logic the Laxen-Perry F83 MANY and TIMES are redundant.

To prevent an infinite loop from happening the interpretive BEGIN and DO will be halted by pressing any key, and terminated if the next key pressed is <return>.

The implementation of +LOOP presents no problem and has been left as an exercise. It just requires more bookkeeping to maintain and update the loop index. I do not feel that this extra overhead would be justified by its frequency of use.

The code and examples are case insensitive. The use of upper and lower case is intended only for clarity

The method of printing this exhibit demonstrates several applications of interpretive logic. The word DOC is used to display the lines of a source block other than the first. In addition, multiple blank lines are suppressed. If DOC has not already been defined then the definition will be made.

This word is then used in interpretive loops to display text. If the listing goes to the terminal rather than a line printer, i.e., the value of variable PRINTING is zero, then you will be asked if you now want to forget the word DOC , which you presumably have just defined, used, and no longer need.

CONCLUSION

Interpretive logic, for conditional execution, conditional compilation, and text editing, extends Forth to be responsive to modern system requirements.

```
have DOC not if
: DOC   ( block# -- )  1 ?ENOUGH    L/SCR 1
   DO I C/L *    OVER BLOCK +    C/L -TRAILING    ?DUP
      IF CR >TYPE
      ELSE DROP #OUT @
         IF CR THEN
      THEN
   LOOP DROP ;
then
13 16 for i doc loop    1 doc    17 22 for i doc loop    0 fh doc
printing @ not if
   .( FORGET DOC ? )    key ascii _ and ascii Y =
   if FORGET DOC then
then
```

```
    0
( Interpretive logic structures.                  WWB 19OCT85WWB)

            INTERPRETIVE LOGIC

The logical structure words are extended for interpretation.

                        Wil Baden
                   339 Princeton Drive
                   Costa Mesa CA 92626
                       714-546-9894
```

```
    1
( INTERPRETIVE LOGIC                              WWB/20OCT85WWB)
CR .( INTERPRETIVE VERSION OF LOGIC STRUCTURES.)   CAPS ON
0 VIEW# !   0 VIEWS IF.BLK
3 7 THRU  ( IF ELSE THEN   HAVE   BEGIN WHILE REPEAT UNTIL )
CR .( Do you want interpretive DO ? )
key dup emit   ascii _ and   ascii Y =
if CR .( And do you want interpretive FOR ? )
   key dup emit   ascii _ and   ascii Y =
   if    8 10 THRU        ( DO FOR LOOP I )
   else  9 10 THRU then  ( DO LOOP I )
then
have PRINT not if 11 LOAD then
have CONSULT if 2 2 CONSULT then
```

```
    2
( DIRECTORY OF LOGICAL INTERPRETATION WORDS.   WWB  20OCT85WWB)
SEAL 4 UNSEAL 4 PASSOVER 4 IF 5 ELSE 5 THEN 5 HAVE 5 HAVE 5
THENIF 5 NUF? 6 BEGIN 7 REPEAT 7 UNTIL 7 WHILE 7 INCL 8 FOR 8
DO 9 LOOP 9 FOR 9 I 10 ^ 10 I 10
```

```
    12
( SAY "CAPACITY 2/ LOAD" TO PRINT EXHIBIT.      WWB 19OCT85WWB)
have DOC not if
: DOC  ( block# -- ) 1 ?ENOUGH   L/SCR 1
   DO I C/L *   OVER BLOCK +   C/L -TRAILING   ?DUP
      IF CR >TYPE
      ELSE DROP #OUT @
         IF CR THEN
      THEN
   LOOP DROP ;
then
13 16 for i doc loop   1 doc   17 22 for i doc loop   0 fh doc
printing @ not if
   .( FORGET DOC ? )   key ascii _ and ascii Y =
   if FORGET DOC then
then
```

```
    13
( Interpretive logic structures.                  WWB 19OCT85WWB)
                    INTERPRETIVE LOGIC

                         Wil Baden
                    339 Princeton Drive
                    Costa Mesa CA 92626

    A programmer's best friend is his source-code editor.  For
Forth programmers the relationship can be particularly
intimate.  A Forth editor will often do very kinky things for
the programmer.

    The next best friend that a programmer can have is the
ability to do conditional interpretation and conditional
compilation, both from immediate console entry and from a
prepared script.  In Unix (tm) parlance this is the Shell.
```

```
    14
( Interpretive logic structures.                  WWB 19OCT85WWB)

    The Forth-83 Uncontrolled Reference Set has three words,
IFTRUE , OTHERWISE , and IFEND , inherited from the Forth-79
Reference Set, whose meaning is specified by:

IFTRUE          flag  --                         interpret only
   Begin an
      IFTRUE ... OTHERWISE ... IFEND
   conditional sequence.  These conditional words operate like
      IF ... ELSE ... THEN
   except that they cannot be nested, and are to be used only
   during interpretation.  In conjunction with the words [ and
   ] they may be used within a colon definition to control
   compilation, although they are not to be compiled.
```

3

```
( Interpretive conditional                        WWB/WWB 850725)
VOCABULARY CONDITIONAL   CONDITIONAL DEFINITIONS
( PASSOVER will ignore all words but these.)
( You should extend this set if you define other structures.)
: else   2DUP = + ;
: if     1+ ;
: begin  1+ ;
: then   1- ;
: repeat 1- ;
: until  1- ;
: ."     [ ASCII " ] LITERAL PARSE 2DROP ;
: .(     [COMPILE] ( ;
: (      [COMPILE] ( ;
: \      [COMPILE] \ ;
FORTH DEFINITIONS
```

4

```
( Interpretive bypass                             WWB/WWB 850725)
: SEAL    ( -- ) 1 IS #VOCS ;                  ( Redefined.)
: UNSEAL  ( -- ) [ #VOCS ] LITERAL IS #VOCS ;

: PASSOVER  ( n -- ) 1 ( initial depth)
  BEGIN BL WORD   DUP COUNT   ?DUP 0=
     IF 2DROP 2DROP   BLK @ ABORT" Unexpected end." EXIT THEN
     UPPER   CONTEXT @ >R   SEAL CONDITIONAL   FIND   UNSEAL
     R> CONTEXT !   IF EXECUTE ?DUP THEN   0=
  UNTIL DROP ;
```

5

```
( Interpretive if-else-then                       WWB/WWB 850722)
: if      ( n -- )
   STATE @ if [compile] IF exit then   1 ?ENOUGH
   0= IF 1 PASSOVER THEN ; IMMEDIATE
: else    ( -- )
   STATE @ if [compile] ELSE exit then
   0 PASSOVER ; IMMEDIATE
: then    ( -- )
   STATE @ if [compile] THEN then  ; IMMEDIATE
\ : have  ( -- flag ) BL WORD FIND   SWAP DROP   0= NOT ;
: have  ( -- flag ) DEFINED NIP 0<> ;

have THENIF not if EXIT then
: thenif  ( n -- )
   STATE @ if [compile] THENIF exit then
   [COMPILE] if ; IMMEDIATE
```

15

(Interpretive logic structures.   WWB 19OCT85WWB)

There is no reason why these could not and should not be nested. If they are STATE-smart they do not have to bear new names, which will make them easier to learn.

One method of implementation is to scan the source input for the relevant key words and perform an appropriate action for each. The string processing routines needed to do this already exist in Forth -- WORD , FIND and EXECUTE . An easy way to restrict the words that are recognized is to use a sealed vocabulary containing just those words and no others.

After IF , ELSE and THEN are defined it is a small step to BEGIN , WHILE , REPEAT and UNTIL . Another small step will lead to DO and LOOP . This extends Forth to become a very powerful job control language.

16

(Interpretive logic structures.   WWB 19OCT85WWB)

With a few extensions to the line editing words such as found in Laxen-Perry's F83 Model or Leo Brodie's Starting Forth we get a powerful programmable macro editor.

Some examples of use are found on the load block.

17

(Interpretive logic structures.   WWB 19OCT85WWB)

First we load

IF   ELSE   THEN   BEGIN   WHILE   REPEAT   UNTIL

from blocks 3 through 7. We then ask whether interpretive DO (and LOOP) are wanted. KEY is used to accept your answer, and if it was uppercase or lowercase Y we ask if you also want interpretive FOR . The appropriate blocks are loaded.

IF and ELSE are very useful with conditional compilation. We can compile or execute different things depending on the value of a variable or computed expression. We can compile different systems from just one source program. The word HAVE can be used to test whether we already have a word, i.e., whether it has already been defined.

6

```
( NUF?                                          MJT/19OCT85WWB)
have NUF? not if
: NUF?  ( -- flag )
  KEY? DUP IF KEY 2DROP   KEY 13 ( return) = THEN ;
then

( NUF? halts after one key-press, and then returns TRUE if the )
( next key-press is <return>.  Suggested by Martin Tracy.      )

have FH not if
: FH  ( u -- block#) BLK @ ?DUP 0= IF SCR @ THEN + ;
then

( Relative block number.  Suggested by Leo Brodie.            )
```

7

```
( Interpretive begin-while-repeat-until          WWB/WWB 850722)
: begin   ( -- )
   STATE @ if [compile] BEGIN exit then   >IN @ >R
   BEGIN R@ >IN !   INTERPRET   NUF? UNTIL R> DROP ; IMMEDIATE
: repeat  ( -- )
   STATE @ if [compile] REPEAT exit then
   R> DROP ; IMMEDIATE
: until   ( n -- )
   STATE @ if [compile] UNTIL exit then   1 ?ENOUGH
   IF R> R> 2DROP THEN R> DROP ; IMMEDIATE
: while   ( n -- )
   STATE @ if [compile] WHILE exit then   1 ?ENOUGH
   0=
   IF R> R> 2DROP   R> DROP   0 PASSOVER THEN ; IMMEDIATE
```

8

```
( Compiled for-loop                                 WWB/09AUG85WWB)
: INCL  ( n1,n2 -- n2+1,n1 ) 2 ?ENOUGH   OVER MAX   1+ SWAP ;
: FOR  ( first,last -- )
   COMPILE INCL   [COMPILE] DO ; IMMEDIATE

( In most applications it is more convenient to express a range)
( as <first>,<last> rather than <last+1>,<first> -- especially )
( when interpreting.  This definition of FOR is in anticipation)
( of an interpretive version;  otherwise it is not worth doing.)
```

18

(Interpretive logic structures.   WWB 19OCT85WWB)

The code is compatible with the Laxen-Perry F83 Model (and of course my F83X). A simpler, more powerful, and easier to use definition of SEAL is given.

SEAL

Usage: SEAL will change the search order such that only the first vocabulary in the search order will be searched. The other vocabularies in the search order are present and unchanged but do not participate in the search.

UNSEAL

Allow all vocabularies in the search order to be searched.

PASSOVER is used by the system to pass over all words but those in the vocabulary CONDITIONAL . Those words and only

19

(Interpretive logic structures.   WWB 19OCT85WWB)

those words will be executed.

The word FOR has been introduced for interpretive counted loops. In most applications it is more convenient to express a range as <first>,<last> rather than <last+1>,<first> . When a word is compiled it is trivial to say "1+ SWAP" before "DO" but it becomes a nuisance and a potential source of confusion when interpreting. FOR also checks that there are at least two arguments on the stack. A compilation version of FOR has been given for completeness.

The loop index I presents a problem when the editor words include an I also. We use conditional compilation to check whether the editor does contain I and when it does we define ^ (caret or up-arrow) as a synonym of the editor I for use with

20

(Interpretive logic structures.   WWB 19OCT85WWB)

interpretive logic. Conditionally selected system dependent code is used to check the depth of the return stack so that the editor I can still be used outside of interpretive logic.

The use of interpretive logic is simplified when entering commands from the keyboard by not requiring THEN , REPEAT or LOOP at the end of a line. They are required on a source block, or when you want to do something after the interpretion of the control structure, or when compiling. With interpretive logic the Laxen-Perry F83 MANY and TIMES are redundant.

To prevent an infinite loop from happening the interpretive BEGIN and DO will be halted by pressing any key, and terminated if the next key pressed is <return>.

9

```
( Interpretive do & for                           WWB/09AUG85WWB)
: do  ( limit,start -- )
   STATE @ if [compile] DO exit then   2 ?ENOUGH
   >IN @ -ROT
   DO DUP >IN ! >R   INTERPRET   R> NUF? ?LEAVE LOOP
   DROP ; IMMEDIATE
: loop  ( -- )
   STATE @ if [compile] LOOP exit then
   R> DROP ; IMMEDIATE

have FOR not if EXIT then
: for  ( first,last -- )
   STATE @ if [compile] FOR exit then
   >IN @ -ROT
   FOR DUP >IN ! >R   INTERPRET   R> NUF? ?LEAVE LOOP
   DROP ; IMMEDIATE
```

10

```
( Interpretive loop-index                         WWB/30JUL85WWB)
: i   ( -- loop-index )
   STATE @ if [ forth ] COMPILE I exit then
   R> R> R> [ forth ] I SWAP >R SWAP >R SWAP >R ; IMMEDIATE

' I EDITOR ' I = NOT if DEFINITIONS
   : ^   ( -- ) [ editor ] I ; ( In interpreted DO-loops.)
   : I   ( -- loop-index )
      STATE @ if [ forth ] [compile] I exit then
      ( *** RP0 and RP@ are implementation dependent. *** )
      [ have RP0 if ] RP0 @ [ else ] R0 @ [ then ]
      RP@ 12 ( 6 CELLS) + U< IF [ editor ] I EXIT THEN
      R> [ forth ] [compile] i SWAP >R ; IMMEDIATE
then FORTH DEFINITIONS
```

11

```
( SINGLE LINESPACE                                WWB 20OCT85WWB)
5 CONSTANT PRINT-OFFSET
: LINESPACE ( -- ) PRINTING @
   IF 13 ( CR) (PRINT)
      [ .( Do you want LF after CR ? )
      Key dup emit   ascii _ and ascii Y =
      if ] 10 ( LF) (PRINT) [ then ]
      PRINT-OFFSET SPACES
   THEN
   13 ( CR) CONSOLE   10 ( LF) CONSOLE
   #OUT OFF  1 #LINE +! ;
```

21

(Interpretive logic structures.   WWB 19OCT85WWB)

The implementation of +LOOP presents no problem and has been left as an exercise. It just requires more bookkeeping to maintain and update the loop index. I do not feel that this extra overhead would be justified by its frequency of use.

The code and examples are case insensitive. The use of upper and lower case is intended only for clarity

The method of printing this exhibit demonstrates several applications of interpretive logic. The word DOC is used to display the lines of a source block other than the first. In addition, multiple blank lines are suppressed. If this word or another of the same name has not already been defined then the definition will be made.

22

(Interpretive logic structures.   WWB 20OCT85WWB)

This word is then used in interpretive loops to display text. If the listing goes to the terminal rather than a line printer, i.e., the value of the variable PRINTING is zero, then you will be asked if you now want to forget the word DOC which you presumably have just defined, used, and no longer need.

CONCLUSION

Interpretive logic, for conditional execution, conditional compilation, and text editing, extend Forth to be responsive with modern system requirements.

23

(SINGLE LINESPACE   WWB 20OCT85WWB)

LINESPACE   (--)

Configure vertical spacing for your printer with interactive conditional compilation.