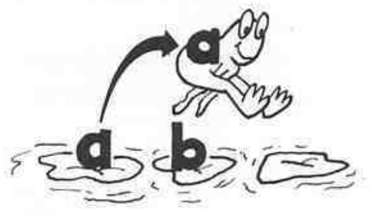
A Recognizer Influenced Handler Based Outer Interpreter Structure

Ulrich Hoffmann

uho@

.de



over view

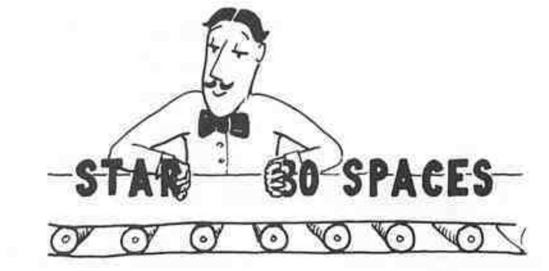
- recognizers
- outer interpreter: what needs to be done?
- handlers
  - idea
  - code
  - design options
    - possible stack effects
    - haeh?
    - token scanning
    - search order
- summary
- disussion

recognizers

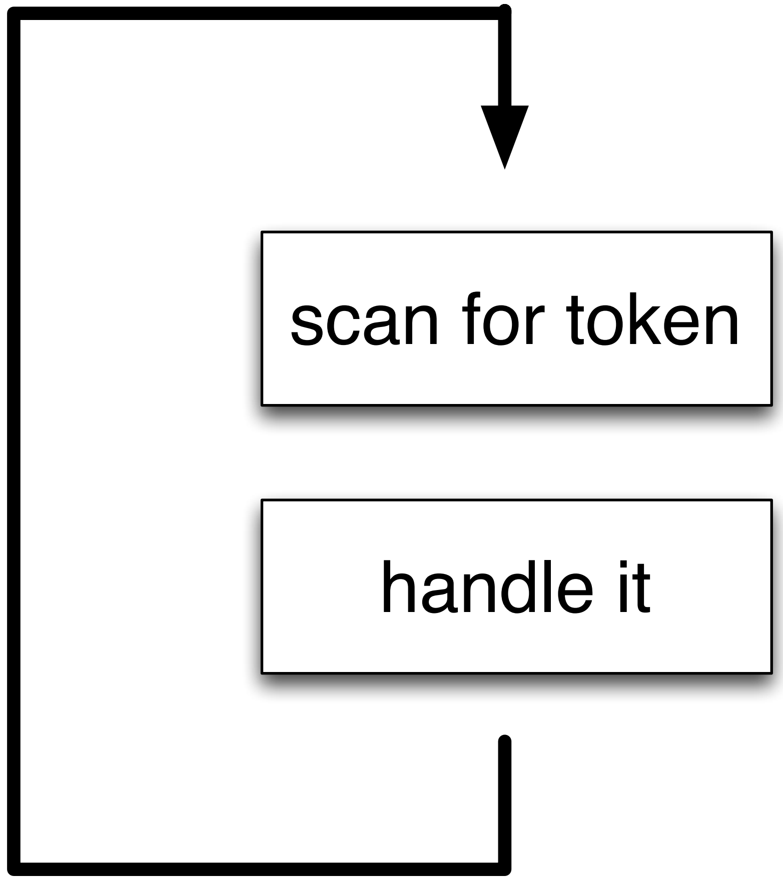
- new extensible outer interpreter[1] structure proposed by mathias trute
- on its way to become a standard's committee supported proposal
- interpret/compile/postpone structure for syntactic classes that describes their treatment in the outer interpreter
- stack structure for combining recognizers

[1] http://amforth.sourceforge.net/pr/Recognizer-rfc-D.html

outer interpreter: what needs to be done?

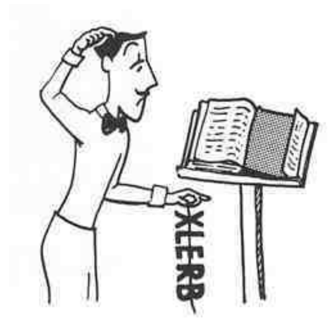


"the text interpreter scans the input stream, looking of strings of characters separated by spaces."



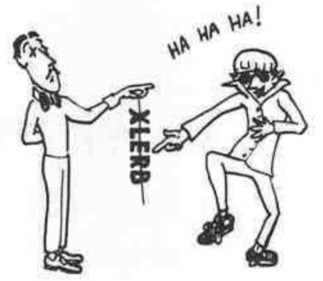
outer interpreter: what needs to be done?

token →



is it a word?

run it



is it a number?

push it

haeh?

consult user

outer compiler: what needs to be done?

is it a word?

is it a immediate?

run it

compile it

is it a number?

compile push it

haeh?

consult user

outer interpreter: extensions

is it a word?	run it
is it a number?	push it
is it a float?	fpush it
haeh?	consult user

outer interpreter: extensions

is it a word?	run it
is it a number?	push it
is it a char?	push it
haeh?	consult user

outer interpreter: extensions

is it a word?	run it
is it a number?	push it
is it a hex ?	push it
haeh?	consult user

outer interpreter: extensions

is it a word?	run it
is it a number?	push it
is it a char?	push it
is it a hex ?	push it
is it a float?	fpush it
haeh?	consult user

ok

handlers
idea

- give the token to a list of handlers one handler at a time until one can cope with it
- if a handler can cope with it, it does it and reports
- if it cannot, it reports

handlers code

```
Variable handlers

: interpret ( -- )
    BEGIN  parse-name dup
    WHILE
      handlers @ length handle
      0= IF -13 throw THEN
    REPEAT 2drop ;
```

handlers code

```
Variable handlers

: interpret ( -- )
    BEGIN  parse-name dup
    WHILE
      handlers @ length handle
      0= IF -13 throw THEN
    REPEAT 2drop ;
```

and ***state?***

handlers code
interpret words

```
\ interpret words in forth wordlist

:noname ( c-addr u1 -- i*x true | c-addr2 u2 false )
   2dup  forth-wordlist  search-wordlist
    IF  nip nip execute true EXIT THEN false ;
```

difference to recognizers?

- 1 task vs. 3 in 1
- immediate copeing vs. later execution

handlers code
compile words

```
\ compile words in forth wordlist

:noname ( c-addr u1 -- i*x true | c-addr2 u2 false )
   2dup forth-wordlist search-wordlist
   dup 0< IF ( not immediate )
              drop compile,
              2drop true EXIT THEN
          IF ( immediate )
              nip nip execute
              true EXIT THEN
   false ;
```

handlers code
interpret character literals

```
\ interpret character literals

: charlit ( c-addr u1 -- i*x true | c-addr2 u2 false )
   dup 3 = IF over c@ [char] ' = 2 pick c@ [char] ' = and
   IF  drop char+ c@ true EXIT THEN THEN false ;
' charlit
```

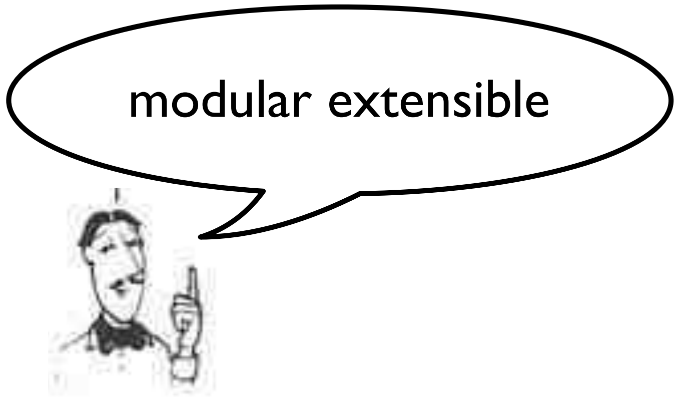
handlers code compile character literals

```
\ compile character literals

[: ( c-addr u1 -- i*x true | c-addr2 u2 false )
   charlit IF postpone literal true EXIT THEN  false ;]
```

possible handlers

- words
- base numbers (single cell)
- base prefix numbers (hex decimal bin)
- character literals
- string literals
- s"
- double precision numbers
- floating point numbers
- namespace scoped identifiers
- object systems
- date&time
- ...



handlers properties

- modular **extensible** (1. dimension)
  - interpreter (extensible) →
  - compiler (extensible) →
  - postponer (extensible) →

  - more **extensions** (2. dimension) ↓
    - target compiler
    - remote compiler
    - DSL compiler

handlers properties

- handlers are **simply** colon definitions

- composing handlers give new handlers
- handler lists
  - layed out in memory with **`create`** and **`,`**
  - **`n@ n!`** operate on cell counted lists
  - handler lists can be in **`allocate`**d memory
  - handler chained in :-definitions

handlers design options

- possible stack effects
- haeh?
- token scanning
- search order

- prototypes for each options on git branches

handlers design options possible stack effect

- what stack effect shall a handler have?
  - (c-addr u1 -- i*x true | c-addr2 u2 false)
  - (c-addr u -- i*x true | false)
  - (c-addr u -- i*x c-addr u true | c-addr u false)

handlers design options haeh?

- if no handler can cope with the token, what should be done?
  - signal error (`-13 throw`)
  - ignore

handlers design options token scanning

- shall handlers work on pre scanned tokens?
- of shall they inspect the input stream on their own?

handler design options search order

- shall a handler search the search order
- or look into a single word list?
  - the search order will be a sublist of handlers

summary

- **simple**
    - handlers are ordinary :-definitions
    - handler lists are easy to build and manage

- **extensible**
  in 2 dimensions:
    1. extending handler lists with new handlers
    2. different compilers/interpreters (postponers)

may the **`swap`** be with you!

discussion

handle code

```
:  handle  ( c-addr1 u1 addr u -- i*x true | c-addr2 u2 false )
   cells bounds
   ?DO ( c-addr1 u1 )
       I @ execute ?dup IF ( i*x ) UNLOOP EXIT THEN
   cell +LOOP
   false ;
```