Recognizers

Customize the Interpreter

Bernd Paysan

EuroForth 2012, Oxford

Overview

Motivation

Gforth's Recognizers

Examples

The Problem

- Forth is extensible, provided all your extensions are simple, space–delimited words
- Literals are already part of the non–extensible, unchangeable part of the standard interpreter
- Many systems have mechanisms like `notfound`, where you can plug in something in a system–dependent way. . .

Recent Development

- During the number prefix RfD discussion, ANTON ERTL [1] suggested a system called “Recognizer,” which was roughly sketched, but would allow to dynamically reconfigure the interpreter
- MATTHIAS TRUTE had several discussions on IRC and implemented a recognizer system in amForth[1]
- Win32Forth got recognizers in the current development snapshot, as well as Gforth
- All these recognizers look slightly different, as they are still experimental stuff

Gforth's Recognizers

x–RECOGNIZER (addr u | token r:x / addr u r:fail)

A recognizer takes a string, and converts it to a token, which consist of some data on the stack and a method table. The method table have three "virtual" methods (which are only concept):

INT (x*i token — y*j)

Invokes the interpretation semantics of a token (similar to EXECUTE)

COMP (token —)

Invokes the compilation semantics of a token

LIT (token —)

Add the token to the currently defined word, so that tokens can be postponed

Gforth's Recognizers

RECOGNIZER: (xt-int xt-comp xt-lit „name“ —)
Creates a recognizer table

Recognizers are organized as a stack (similar to wordlists), therefore you can

GET–RECOGNIZERS (rec–addr — rec_n .. rec_1 n)
get the all the recognizers out of a stack

SET–RECOGNIZER (rec_n .. rec_1 n rec–addr —)
set the recognizers of a stack

Gforth's Recognizers

DO–RECOGNIZER (addr u rec–addr — token r:table | addr u r:fail)
walks through all the recognizers in a stack until one matches, and either return its result or the input string and r:fail

R:FAIL (– r:fail)
recognizer table, where all three methods fail with `-13 throw`

Predefined Recognizers: Forth words

```
: lit, ( n -- ) postpone Literal ;
: nt, ( nt -- ) name>comp execute ;
: nt-ex ( nt -- ) name>int execute ;
' nt-ex ' nt, ' lit, recognizer: r:word
: word-recognizer ( addr u -- nt r:word | addr u r:fail
  2dup find-name
  [ [IFDEF] prelude-mask ] run-prelude [ [THEN] ] dup
  IF  nip nip r:word  ELSE  drop r:fail  THEN ;
```

Predefined Recognizers: Literals

```
: 2lit, postpone 2Literal ;
' noop ' lit, dup recognizer: r:num
' noop ' 2lit, dup recognizer: r:2num
: num-recognizer ( addr u -- n/d table | addr u r:fail )
  2dup 2>r snumber?  dup
  IF  2rdrop 0> IF r:2num ELSE r:num THEN EXIT  THEN
  drop 2r> r:fail ;
```

Advanced Recognizers: Strings

```
: slit,  postpone sliteral ;
’ noop ’ slit, dup recognizer: r:string
: string-recognizer
    ( addr u -- addr’ u’ r:string | addr u r:fail )
    2dup s\" \"" string-prefix?
    IF    drop source drop - 1+ >in !
          \"-parse save-mem r:string
    ELSE  r:fail  THEN ;
’ string-recognizer
forth-recognizer get-recognizers
1+ forth-recognizer set-recognizers
```

For Further Reading

Anton Ertl

Usenet Posting *number parsing hooks*

`https://groups.google.com/forum/?fromgroups#!msg/comp.lang.forth/r7Vp3w1xNus/Wre1BaKeCvcJ`

Matthias Trute

Recognizer — Interpreter dynamisch verändern

VD 2011/02

Bernd Paysan

Recognizer

VD 2012/02