Recognizers

Why and How

M. Anton Ertl

TU Wien

How to deal with literals

Recognizers

```
123
$ff
'a'
1.2e3
```

Parsing Words

```
s" abc"
H# ff
[char] a
```

- No way to define new recognizers

- No good way to define parsing words
  non-default non-immediate compilation semantics
  `State`-smartness and the like
  Not just an implementation problem

- ⇒ user-defined recognizers

Ideal

- Recognized literal acts like a normal word

- : 123 123 ;

- Interpret
  Compile
  Postpone?   ]] a 123 b [[vs.   ]] a [[123]] literal b [[
  '?
  find
  find-name name>string ?

How to specify and implement recognizers

- Specify interpret, compile, and postpone actions
  Advantage: Optimization possible
  Disadvantage: Bugs can hide, especially for postpone

- Specify parse-time, run-time, and data-shifting actions
  interpret: parse-time run-time
  compile: parse-time shift]] run-time [[
  postpone: parse-time shift]] shift]] run-time [[[[

- Define a temporary word
  Advantages: Allows ticking etc.
  Conceptual simplicity
  Disadvantage: Optimization?

Temporary words

- Separate dictionary pointer (like ELF section)

- Should be inlined if compiled. *But how?*

- Becomes permanent if `postponed` or ticked

- Other permanent uses need explicit permanence

- Recognized string as name?
  Decompiler
  `name>string`

Coding example

```
: usingle ( c-addr u -- f )
  0. 2over >number 0= if
     drop 2drop 2drop false exit then
  drop drop rot rot ['] constant execute-parsing
  true ;
```

Inline when compiled

- Require using an intelligent `compile`,
  Quite elegant
  But `set-opt` is unlikely to be standardized

- Or specify parse-time and compile-time action
  For compilation, perform these actions
  In other cases, build the word

Performance with many recognizers

- Linear search through recognizer stack?

- Or fast pre-selection

- Pre-selection may accept invalid strings
  but must not reject valid strings

- prefix pre-selectors $\Rightarrow$ trie

- regexp pre-selectors $\Rightarrow$ NFA/DFA

Conclusion

- User-defined words are great!
  Let´s also allow user-defined recognizers

- New implementation approach:
  Define temporary words
  How to inline?

- Pre-Selectors for performance