Is Forth Code Compact?

A Case Study

M. Anton Ertl

anton@mips.complang.tuwien.ac.at

http://www.complang.tuwien.ac.at/anton/

Institut für Computersprachen

Technische Universität Wien

Comparing Programming Languages

- Solve the same problem with equally competent teams
- Compare solutions for a frequently-solved problem: parser generator

Parser Generator Language and Size

Program	Implementation Languages	Output	Source Lines	All Lines
bnfparse	Forth	-	14	16
DCG	Prolog	Prolog	68	226
mop	Perl	-	156	291
Gray	Forth	bin. Forth	473	754
kwParsing	Python	bin. Python	1691	2883
Coco/R	Modula-2, Coco/R	Modula-2	4005	5106
mlyacc	ML, mlyacc	ML	4395	6352
rdp	C, rdp	C	4947	6519
bison	C	C	7806	11258
ell	Modula-2, Cocktail tools	Modula-2, C	8712	11384
ANTLR	C, ANTLR	C	18577	23318

Parser Generator Features

Program	algorithm	scanner	error recovery	Source Lines
bnfparse	top-down backtracking	-	-	14
DCG	top-down backtracking	-	-	68
mop	top-down backtracking	-	-	156
Gray	LL(1) recursive descent	-	-	473
kwParsing	SLR(1) table-driven	√	-	1691
Coco/R	LL(1) recursive descent	√	√	4005
mlyacc	LALR(1) table-driven	-	√	4395
rdp	LL(1) recursive descent	√	√	4947
bison	LALR(1) table-driven	-	√	7806
ell	LL(1) recursive descent	-	√	8712
ANTLR	pred-LL(k) recursive descent	-	√	18577

Gray (Forth) vs. Coco/R (Modula-2): Reasons for the size difference (473 vs. 4005 source lines)

- features
- language-induced design decisions
- other design decisions
- programming style
- language requirements

Gray vs. Coco/R:
Language-induced design decisions

- run-time code generation
- parsing the grammar
- compile/run-time separation
- symbol table
- passing values between rules
- object-oriented techniques

Gray vs. Coco/R:

Piecewise comparison

	Gray	Coco/R
code generator	65	700
command line, files	-	159
first-set, check conflicts	83	327
follow-set	26	95
grammar constructors	47	78
objects	35	-
scanner	-	1300
sets	72	188
symbol table	-	274
syntax	24	258
types	83	122
other	38	504
total	473	4005

Set union in Coco/R

```
PROCEDURE Unite (VAR s1, s2: ARRAY OF BITSET); (* s1 := s1 + s2 *)
(* Unite                    s1 := s1 + s2
-------------------------------------------------------------*)
PROCEDURE Unite (VAR s1, s2: ARRAY OF BITSET);
  VAR
    i: CARDINAL;
  BEGIN
    i := 0; WHILE i <= HIGH(s1) DO s1[i] := s1[i] + s2[i]; INC(i) END
  END Unite;
```

Set union in Gray

```
: binary-set-operation ( set1 set2 [w1 w2 -- w3] -- set )
\ creates set from set1 and set2 by applying [w1 w2 -- w3] on members )
\ e.g. ' or binary-set-operation  is the union operation )
 here >r
 cells/set @ 0 do >r
  over @ over @ r@ execute ,
  cell+ swap cell+ swap
 r> loop
 drop 2drop r> ;

: union1 \ set1 set2 -- set )
 ['] or binary-set-operation ;
```

Conclusion

- Parser generators in Forth are smaller
  $> 8\times$ for LL(1) parser generators

- Less features

- Language-induced design decisions
  $> 3\times$

- Low-level code density is similar