File created: 16-Apr-86 13:56:19 **{PHYLUM}<STANSBURY>PARSER>RELEASE.1>PARSERG.;5**

changes to: (VARS PARSERGCOMS ARITHLEXG)
(FNS MAKE.ARITH)

previous date: 3-Apr-86 16:57:21 {PHYLUM}<STANSBURY>PARSER>RELEASE.1>PARSERG.;3

Read Table: OLD-INTERLISP-FILE

Package: INTERLISP

Format: XCCS


```
(RPAQQ PARSERGCOMS
       ((* * TOY1 stuff %. This grammar is "LALR(0)" and the language is "ABC")
        (FNS MAKE.TOY1 READCHAR TEST.TOY1)
        (VARS TOY1G)
        (GLOBALVARS TOY1G TOY1)
        (* * TOY2 stuff %. This grammar is "LALR(1)" and the language is "(aa*)|(aa*+aa*)" %. This is G1 from
           the Brosgol paper.)
        (FNS MAKE.TOY2 TEST.TOY2)
        (GLOBALVARS TOY2 TOY2G)
        (VARS TOY2G)
        (* * TOY3 stuff %. This grammar is "LALR(1)" and the language is "b|(bb)" %. This is G2 from the Brosgol
           paper.)
        (FNS MAKE.TOY3 TEST.TOY3)
        (GLOBALVARS TOY3 TOY3G)
        (VARS TOY3G)
        (* * TOY4 stuff %. This grammar is "LALR(2)" and the language is "a|(aaa)|(aab)" %.)
        (FNS MAKE.TOY4 TEST.TOY4)
        (GLOBALVARS TOY4 TOY4G)
        (VARS TOY4G)
        (* * TOY5 stuff %. The language is "(afc)|(afd)|(bfd)|(bfc)" , but the grammar is not "LALR(k)" for any
           k, so the parser-generator will loop forever, indicating its progress.)
        (FNS MAKE.TOY5 TEST.TOY5)
        (GLOBALVARS TOY5 TOY5G)
        (VARS TOY5G)
        (* * ARITH stuff %. This translates a conventional arithmetic expression language into something
           evaluable by Interlisp EVAL. The language is "LALR(1)" %. Special features: It has a lexical
           analyzer, ARITHLEX, generated by the same mechanism. The lexical analyzer uses semantic actions
           cleverly to remove whitespace and construct number values. The structure parser uses semantic actions
           to generate the Lisp function calls from the parse tree. Since the language is "LALR(1)" , the
           lookahead queue need only be one token deep, and so is specially implemented that way to avoid
           consing. In the lexical language (ARITHLEX)
           , because of the order of reduction implied by the grammar rules, the stack can never be very deep,
           and it is implemented with a small array to minimize consing.)
        (FNS MAKE.ARITH TEST.ARITH MAKE.ARITHLEX TEST.ARITHLEX)
        (GLOBALVARS ARITH ARITHG ARITHLEX ARITHLEXG)
        (VARS ARITHG ARITHLEXG)
        (MACROS ARITHDQ FUNNYCAR FUNNYCDR ARITHTOP ARITHPUSH ARITHPOP ARITHSTACK)
        (RECORDS ARITHSTACK)))
```

(* * TOY1 stuff %. This grammar is "LALR(0)" and the language is "ABC")

```
(DEFINEQ
```

(MAKE.TOY1

```
  [LAMBDA NIL                                                    (* hts: " 3-Apr-86 16:18")
```

(* * Makes the parser generator specification for the TOY1 language)

```
    (SETQ TOY1 (create PARSERSPEC
                       PARSERNAME _ (QUOTE TOY1)
                       GRAMMAR _ (create GRAMMAR
                                         StartSymbol _ (QUOTE TOP)
                                         PRODUCTIONS _ TOY1G)
                       READFN _ (FUNCTION READCHAR])
```

(READCHAR

```
  [LAMBDA (EXPECTED STATE)                                       (* hts: "28-Feb-86 22:02")
    (if (EOFP (CAR STATE))
        then (QUOTE EOF)
      else (READC (CAR STATE])
```

(TEST.TOY1

```
  [LAMBDA NIL                                                    (* hts: "28-Feb-86 22:20")
```

(* * Run the parser on the file {core}foo)

```
    (CLOSEF? (QUOTE {CORE}FOO))
    (LET [(S (OPENSTREAM (QUOTE {CORE}FOO)
                         (QUOTE INPUT]
```

```
         (PROG1 (TOY1 NIL (LIST (CONS)
                                S))
                (CLOSEF S])

)

(RPAQQ TOY1G [(TOP ((A B C)
                    (CONS LHS RHS])

(DECLARE: DOEVAL@COMPILE DONTCOPY

(GLOBALVARS TOY1G TOY1)
)
```

(* * TOY2 stuff %. This grammar is "LALR(1)" and the language is "(aa*)|(aa*+aa*)" %.
This is G1 from the Brosgol paper.)

```
(DEFINEQ
```

(MAKE.TOY2

```
  [LAMBDA NIL                                                    (* hts: " 3-Apr-86 16:25")
```

(* * Makes the parser generator specification for the TOY2 language)

```
    (SETQ TOY2 (create PARSERSPEC
                       PARSERNAME _ (QUOTE TOY2)
                       GRAMMAR _ (create GRAMMAR
                                         StartSymbol _ (QUOTE S)
                                         PRODUCTIONS _ TOY2G)
                       READFN _ (FUNCTION READCHAR])
```

(TEST.TOY2

```
  [LAMBDA NIL                                                    (* hts: "28-Feb-86 22:23")
```

(* * Run the parser on the file {core}foo)

```
    (CLOSEF? (QUOTE {CORE}FOO))
    (LET [(S (OPENSTREAM (QUOTE {CORE}FOO)
                     (QUOTE INPUT]
         (PROG1 (TOY2 NIL (LIST (CONS)
                                S))
                (CLOSEF S])

)

(DECLARE: DOEVAL@COMPILE DONTCOPY

(GLOBALVARS TOY2 TOY2G)
)

(RPAQQ TOY2G
       [(S ((S + A)
            (CONS LHS RHS))
           ((A)
            (CONS LHS RHS)))
        (A ((a A)
            (CONS LHS RHS))
           ((a)
            (CONS LHS RHS])
```

(* * TOY3 stuff %. This grammar is "LALR(1)" and the language is "b|(bb)" %.
This is G2 from the Brosgol paper.)

```
(DEFINEQ
```

(MAKE.TOY3

```
  [LAMBDA NIL                                                    (* hts: " 3-Apr-86 16:27")
```

(* * Makes the parser generator specification for the TOY3 language)

```
    (SETQ TOY3 (create PARSERSPEC
                       PARSERNAME _ (QUOTE TOY3)
                       GRAMMAR _ (create GRAMMAR
                                         StartSymbol _ (QUOTE S)
                                         PRODUCTIONS _ TOY3G)
                       READFN _ (FUNCTION READCHAR])
```

(TEST.TOY3

```
  [LAMBDA NIL                                                    (* hts: "28-Feb-86 22:28")
```

(* * Run the parser on the file {core}foo)

```
    (CLOSEF? (QUOTE {CORE}FOO))
```

```
    (LET [(S (OPENSTREAM (QUOTE {CORE}FOO)
                         (QUOTE INPUT]
          (PROG1 (TOY3 NIL (LIST (CONS)
                                 S))
                 (CLOSEF S])

)

(DECLARE: DOEVAL@COMPILE DONTCOPY

(GLOBALVARS TOY3 TOY3G)
)

(RPAQQ TOY3G
       [(S ((A A)
            (CONS LHS RHS))
           ((b)
            (CONS LHS RHS)))
        (A ((B)
            (CONS LHS RHS)))
        (B ((b)
            (CONS LHS RHS])
```

(* * TOY4 stuff %. This grammar is "LALR(2)" and the language is "a|(aaa)|(aab)" %.)

```
(DEFINEQ
```

(MAKE.TOY4

```
  [LAMBDA NIL                                                    (* hts: " 3-Apr-86 16:30")
```

(* * Makes the parser generator specification for the TOY4 language)

```
    (SETQ TOY4 (create PARSERSPEC
                       PARSERNAME _ (QUOTE TOY4)
                       GRAMMAR _ (create GRAMMAR
                                         StartSymbol _ (QUOTE S)
                                         PRODUCTIONS _ TOY4G)
                       READFN _ (FUNCTION READCHAR])
```

(TEST.TOY4

```
  [LAMBDA NIL                                                    (* hts: " 1-Mar-86 21:08")
```

(* * Run the parser on the file {core}foo)

```
    (CLOSEF? (QUOTE {CORE}FOO))
    (LET [(S (OPENSTREAM (QUOTE {CORE}FOO)
                         (QUOTE INPUT]
          (PROG1 (TOY4 NIL (LIST (CONS)
                                 S))
                 (CLOSEF S])

)

(DECLARE: DOEVAL@COMPILE DONTCOPY

(GLOBALVARS TOY4 TOY4G)
)

(RPAQQ TOY4G
       [(S ((A a a)
            (CONS LHS RHS))
           ((B a b)
            (CONS LHS RHS))
           ((C)
            (CONS LHS RHS)))
        (A ((a)
            (CONS LHS RHS)))
        (B ((a)
            (CONS LHS RHS)))
        (C ((a)
            (CONS LHS RHS])
```

(* * TOY5 stuff %. The language is "(afc)|(afd)|(bfd)|(bfc)" , but the grammar is not "LALR(k)" for any k, so the parser-generator will loop forever, indicating its progress.)

```
(DEFINEQ
```

(MAKE.TOY5

```
  [LAMBDA NIL                                                    (* hts: " 3-Apr-86 16:37")
```

(* * Makes the parser generator specification for the TOY5 language)

```
    (SETQ TOY5 (create PARSERSPEC
                       PARSERNAME _ (QUOTE TOY5)
```

```
                    GRAMMAR _ (create GRAMMAR
                                      StartSymbol _ (QUOTE S)
                                      PRODUCTIONS _ TOY5G)
                    READFN _ (FUNCTION READCHAR])
```

(TEST.TOY5

```
  [LAMBDA NIL                                                    (* hts: " 1-Mar-86 21:47")

          (* * Run the parser on the file {core}foo)

    (CLOSEF? (QUOTE {CORE}FOO))
    (LET [(S (OPENSTREAM (QUOTE {CORE}FOO)
                     (QUOTE INPUT]
         (PROG1 (TOY5 NIL (LIST (CONS)
                                S))
                (CLOSEF S])

)

(DECLARE: DOEVAL@COMPILE DONTCOPY

(GLOBALVARS TOY5 TOY5G)
)

(RPAQQ TOY5G
       [(S ((a A1 c)
            (CONS LHS RHS))
           ((a A2 d)
            (CONS LHS RHS))
           ((b A1 d)
            (CONS LHS RHS))
           ((b A2 c)
            (CONS LHS RHS)))
        (A1 ((f)
             (CONS LHS RHS)))
        (A2 ((f)
             (CONS LHS RHS])
```

(* * ARITH stuff %. This translates a conventional arithmetic expression language into something evaluable by Interlisp EVAL. The language is "LALR(1)" %. Special features: It has a lexical analyzer, ARITHLEX, generated by the same mechanism. The lexical analyzer uses semantic actions cleverly to remove whitespace and construct number values. The structure parser uses semantic actions to generate the Lisp function calls from the parse tree.
Since the language is "LALR(1)" , the lookahead queue need only be one token deep, and so is specially implemented that way to avoid consing. In the lexical language (ARITHLEX), because of the order of reduction implied by the grammar rules, the stack can never be very deep, and it is implemented with a small array to minimize consing.)

```
(DEFINEQ
```

(MAKE.ARITH

```
  [LAMBDA NIL                                                    (* hts: "16-Apr-86 13:42")

          (* * Makes the parser generator specification for the ARITH language)

    (SETQ ARITH (create PARSERSPEC
                        PARSERNAME _ (QUOTE ARITH)
                        GRAMMAR _ (create GRAMMAR
                                          StartSymbol _ (QUOTE EXP)
                                          PRODUCTIONS _ ARITHG)
                        READFN _ (QUOTE ARITHLEX)
                        CLASSFN _ (QUOTE FUNNYCAR)
                        INSTANCEFN _ (QUOTE FUNNYCDR)
                        QUEUEINITFN _ (QUOTE NILL)
                        ENQUEUEFN _ (QUOTE SETQ)
                        DEQUEUEFN _ (QUOTE ARITHDQ)
                        QUEUENOTEMPTYFN _ (QUOTE SELF])
```

(TEST.ARITH

```
  [LAMBDA NIL                                                    (* hts: " 1-Mar-86 19:14")

          (* * Run the parser on the file {core}foo)

    (CLOSEF? (QUOTE {CORE}FOO))
    (LET [(S (OPENSTREAM (QUOTE {CORE}FOO)
                     (QUOTE INPUT]
         (PROG1 (ARITH NIL (LIST NIL NIL S))
                (CLOSEF S])
```

(MAKE.ARITHLEX

```
  [LAMBDA NIL                                                    (* hts: " 3-Apr-86 16:53")

          (* * Makes the parser generator specification for the ARITHLEX language)

    (SETQ ARITHLEX (create PARSERSPEC
```

```
                        PARSERNAME _ (QUOTE ARITHLEX)
                        GRAMMAR _ (create GRAMMAR
                                          StartSymbol _ (QUOTE TOKEN)
                                          PRODUCTIONS _ ARITHLEXG)
                        READFN _ (QUOTE READCHAR)
                        EOFFN _ (QUOTE TRUE)
                        STACKINITFN _ (QUOTE ARITHSTACK)
                        PUSHFN _ (QUOTE ARITHPUSH)
                        POPFN _ (QUOTE ARITHPOP)
                        TOPFN _ (QUOTE ARITHTOP)
                        QUEUEINITFN _ (QUOTE NILL)
                        ENQUEUEFN _ (QUOTE SETQ)
                        DEQUEUEFN _ (QUOTE ARITHDQ)
                        QUEUENOTEMPTYFN _ (QUOTE SELF])
```

(TEST.ARITHLEX

```
  [LAMBDA NIL                                                    (* hts: " 1-Mar-86 19:10")

          (* * Run the parser on the file {core}foo)

    (CLOSEF? (QUOTE {CORE}FOO))
    (LET [(S (OPENSTREAM (QUOTE {CORE}FOO)
                    (QUOTE INPUT]
          (bind S TOKEN DONE STATE first (SETQ DONE NIL)
                                         (SETQ S (OPENSTREAM (QUOTE {CORE}FOO)
                                                             (QUOTE INPUT)))
                                         (SETQ STATE (LIST NIL S))
             while (NOT DONE) collect (SETQ TOKEN (ARITHLEX NIL STATE))
                                      (OR (NEQ (QUOTE EOF)
                                               (CAR TOKEN))
                                          (SETQ DONE T))
                                      TOKEN
             finally (CLOSEF S])

)

(DECLARE: DOEVAL@COMPILE DONTCOPY

(GLOBALVARS ARITH ARITHG ARITHLEX ARITHLEXG)
)

(RPAQQ ARITHG
       [(EXP ((EXP + FACTOR)
              (LIST (QUOTE PLUS)
                    (CAR RHS)
                    (CADDR RHS)))
             ((EXP - FACTOR)
              (LIST (QUOTE DIFFERENCE)
                    (CAR RHS)
                    (CADDR RHS)))
             ((FACTOR)
              (CAR RHS)))
        (FACTOR ((FACTOR * POWER)
                 (LIST (QUOTE TIMES)
                       (CAR RHS)
                       (CADDR RHS)))
                ((FACTOR / POWER)
                 (LIST (QUOTE FQUOTIENT)
                       (CAR RHS)
                       (CADDR RHS)))
                ((POWER)
                 (CAR RHS)))
        (POWER ((MINUS ^ POWER)
                (LIST (QUOTE EXPT)
                      (CAR RHS)
                      (CADDR RHS)))
               ((MINUS)
                (CAR RHS)))
        (MINUS ((- PAREN)
                (LIST (QUOTE MINUS)
                      (CADR RHS)))
               ((PAREN)
                (CAR RHS)))
        (PAREN ((%( EXP %))
                (CADR RHS))
               ((NUMBER)
                (CAR RHS])

(RPAQQ ARITHLEXG
       ((TOKEN ((SPACES REALTOKEN)
                (CADR RHS)))
        (SPACES (NIL NIL)
                ((SPACES % )
                 NIL))
        [REALTOKEN ((+)
                    (QUOTE (+ . +)))
```

```
               ((-)
                (QUOTE (- . -)))
               ((*)
                (QUOTE (* . *)))
               ((/)
                (QUOTE (/ . /)))
               ((^)
                (QUOTE (^ . ^)))
               ((%()
                (QUOTE (%( . %())))
               ((%))
                (QUOTE (%) . %))))
               ((NUMBER)
                (CONS (QUOTE NUMBER)
                      (CAR RHS)))
               ((EOF)
                (QUOTE (EOF . EOF]
        (NUMBER ((DIGITS %. FRACTION)
                 (PLUS (CAR RHS)
                       (CADDR RHS)))
               ((%. FRACTION)
                (CADR RHS))
               ((DIGITS %.)
                (CAR RHS))
               ((DIGITS)
                (CAR RHS)))
        (DIGITS ((DIGITS DIGIT)
                 (PLUS (TIMES 10 (CAR RHS))
                       (CADR RHS)))
               ((DIGIT)
                (CAR RHS)))
        [FRACTION ((FRAC)
                   (FQUOTIENT (CAR (CAR RHS))
                              (EXPT 10 (CDR (CAR RHS]
        (FRAC [(FRAC DIGIT)
               (CONS (PLUS (TIMES 10 (CAR (CAR RHS)))
                           (CADR RHS))
                     (ADD1 (CDR (CAR RHS]
              ((DIGIT)
               (CONS (CAR RHS)
                     1)))
        (DIGIT ((0)
                0)
               ((1)
                1)
               ((2)
                2)
               ((3)
                3)
               ((4)
                4)
               ((5)
                5)
               ((6)
                6)
               ((7)
                7)
               ((8)
                8)
               ((9)
                9))))

(DECLARE: EVAL@COMPILE

(PUTPROPS ARITHDQ MACRO ((Q)
                         (PROG1 Q (SETQ Q NIL))))

(PUTPROPS FUNNYCAR MACRO ((A B)
                          (CAR A)))

(PUTPROPS FUNNYCDR MACRO ((A B)
                          (CDR A)))

(PUTPROPS ARITHTOP MACRO (OPENLAMBDA (S)
                           (ELT (fetch STACK of S)
                                (fetch STACKPTR of S))))

(PUTPROPS ARITHPUSH MACRO (OPENLAMBDA (S NEW)
                            (add (fetch STACKPTR of S)
                                 1)
                            (SETA (fetch STACK of S)
                                  (fetch STACKPTR of S)
                                  NEW)))

(PUTPROPS ARITHPOP MACRO (OPENLAMBDA (S)
                           (PROG1 (ARITHTOP S)
                               (add (fetch STACKPTR of S)
```

```
                                    -1))))

(PUTPROPS ARITHSTACK MACRO (NIL (create ARITHSTACK
                                        STACKPTR _ 0
                                        STACK _ (ARRAY 7))))
)

(DECLARE: EVAL@COMPILE

(DATATYPE ARITHSTACK (STACKPTR STACK))
)

(/DECLAREDATATYPE (QUOTE ARITHSTACK)
       (QUOTE (POINTER POINTER))
       ;; ---field descriptor list elided by lister---
       (QUOTE 4))

(PUTPROPS PARSERG COPYRIGHT ("Xerox Corporation" 1986))
```

FUNCTION INDEX

MACRO INDEX

VARIABLE INDEX

RECORD INDEX