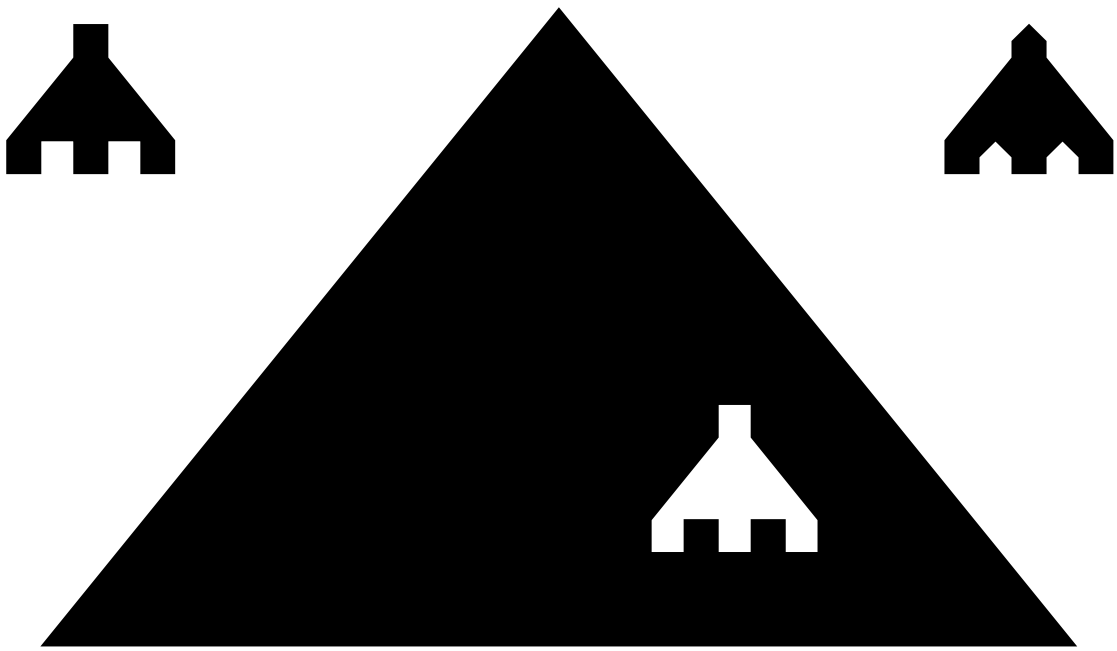
Computer Aided Language Engineering (CALE) Part II – Native Patterns

Alex Sellink
Chris Verhoef

overview

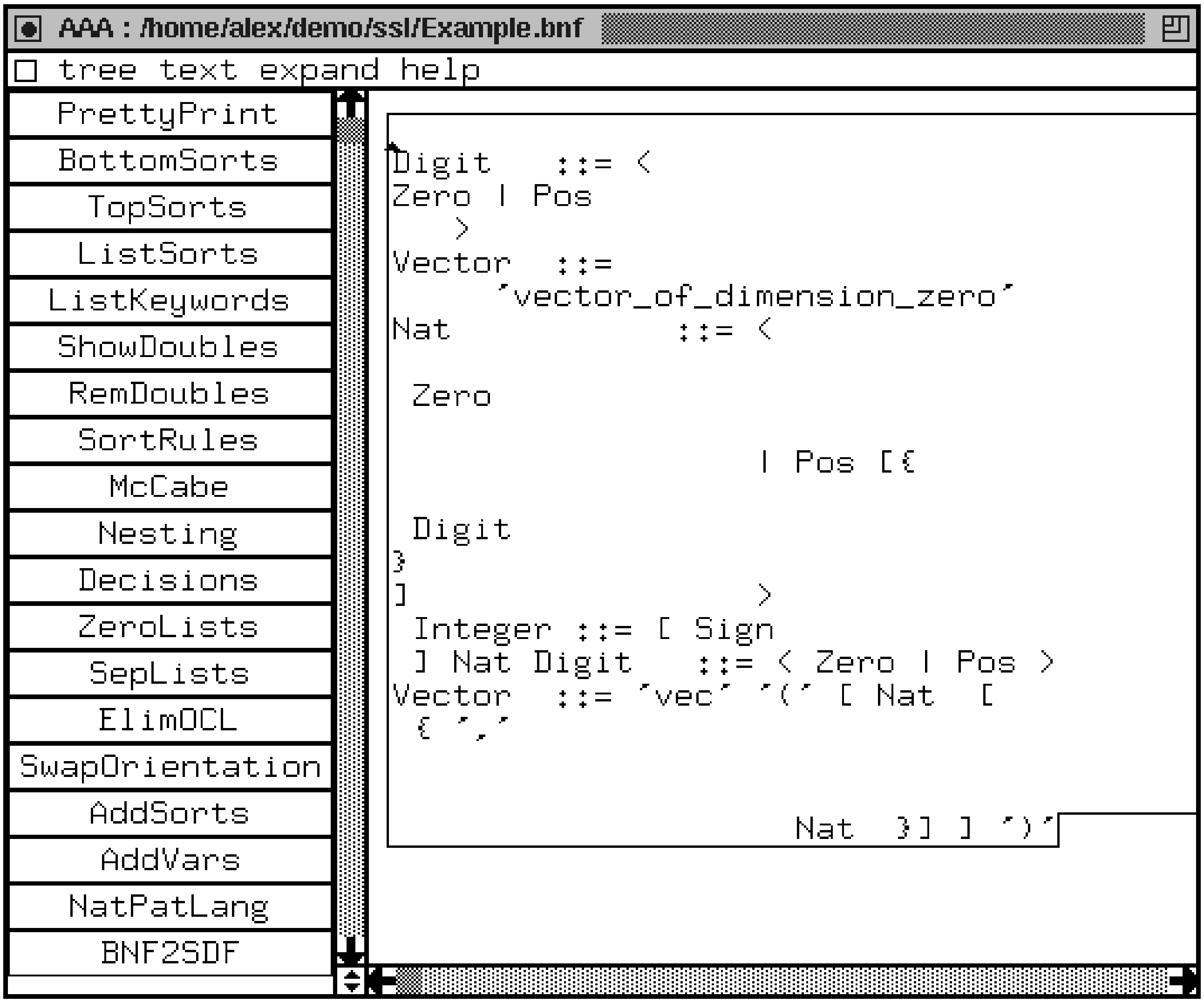
- motivation
- CALE tools: generation of a native pattern language



motivation

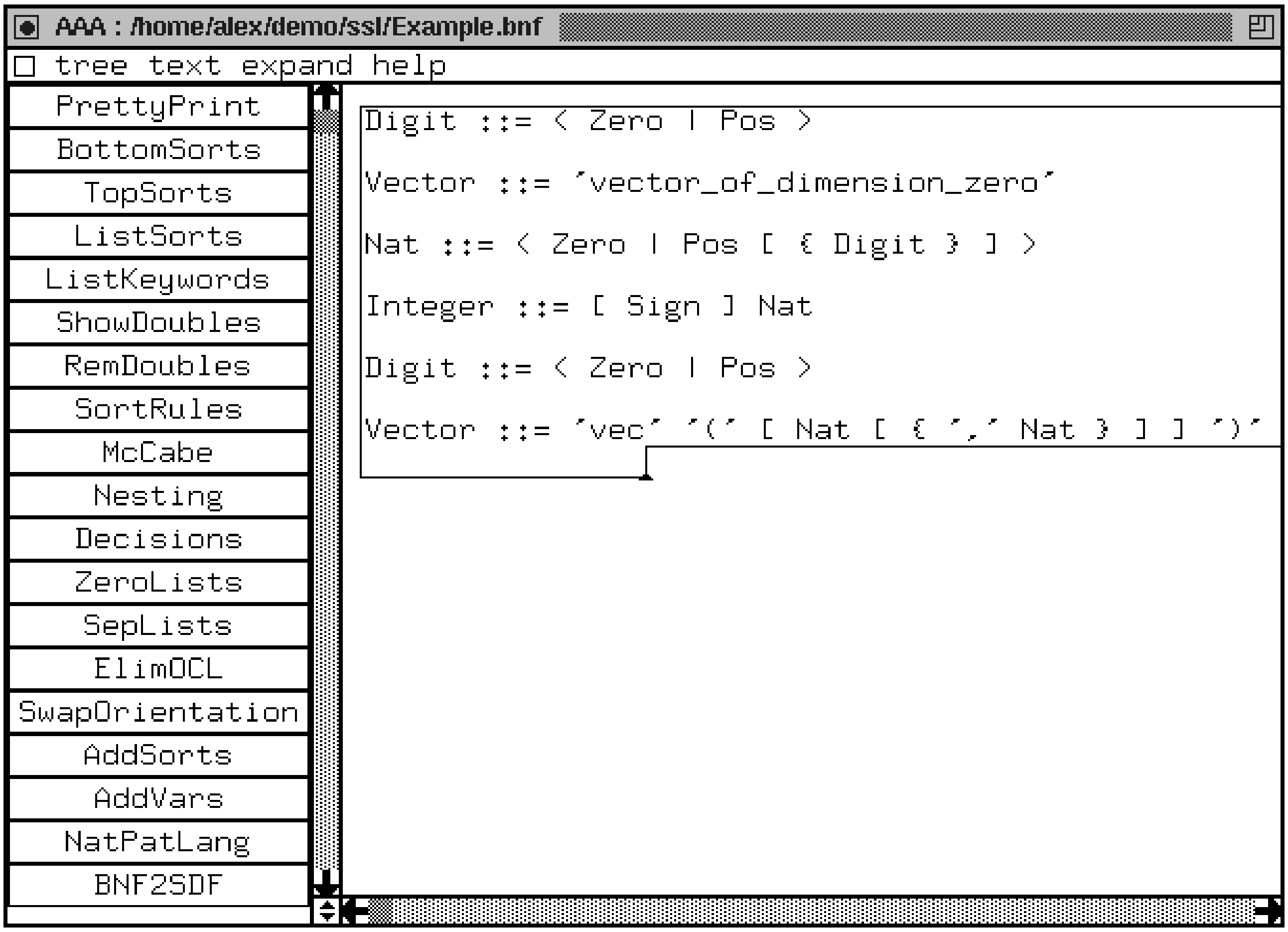
- careful and correct design
- no new language
- high productivity
- generic

Example BNF

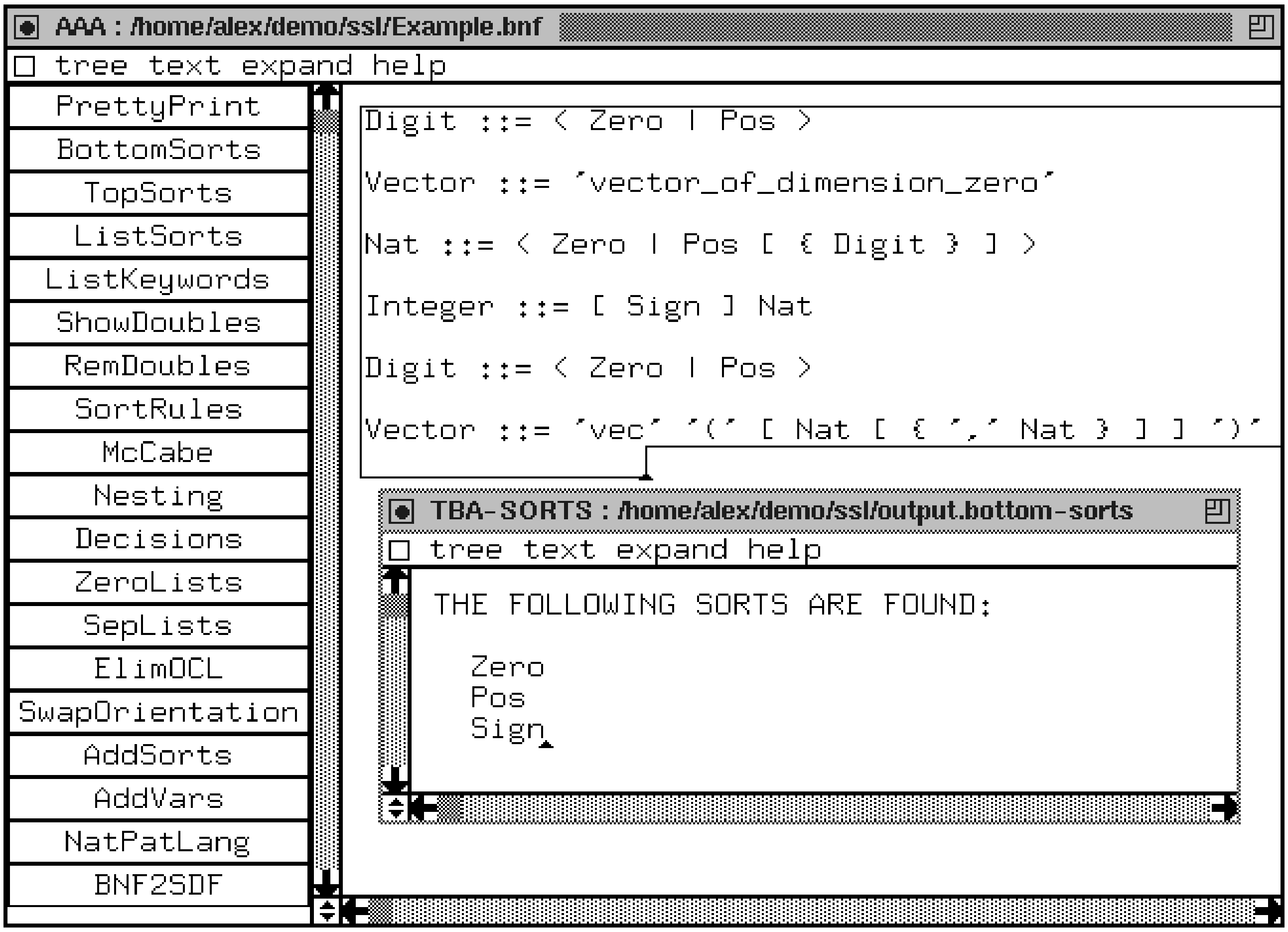


AAA : /home/alex/demo/ssl/Example.bnf
tree text expand help
PrettyPrint
BottomSorts
TopSorts
ListSorts
ListKeywords
ShowDoubles
RemDoubles
SortRules
McCabe
Nesting
Decisions
ZeroLists
SepLists
ElimOCL
SwapOrientation
AddSorts
AddVars
NatPatLang
BNF2SDF
Digit ::= <
Zero | Pos
>
Vector ::=
'vector_of_dimension_zero'
Nat ::= <
Zero
| Pos [{
Digit
}
] >
Integer ::= [Sign
] Nat Digit ::= < Zero | Pos >
Vector ::= 'vec' '(' [Nat [
{ ','
Nat }]] ')'

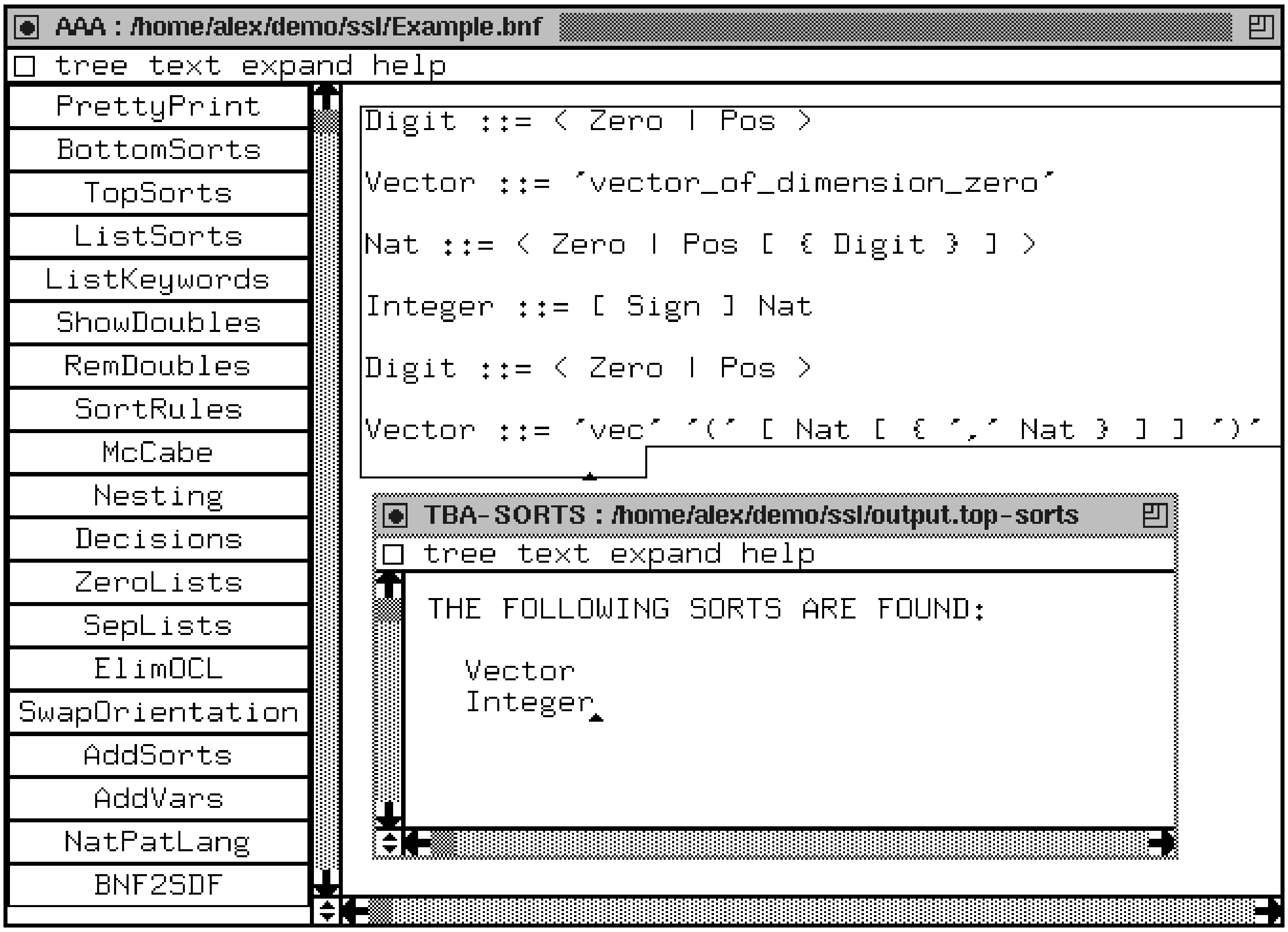
pretty printed BNF



bottom sorts

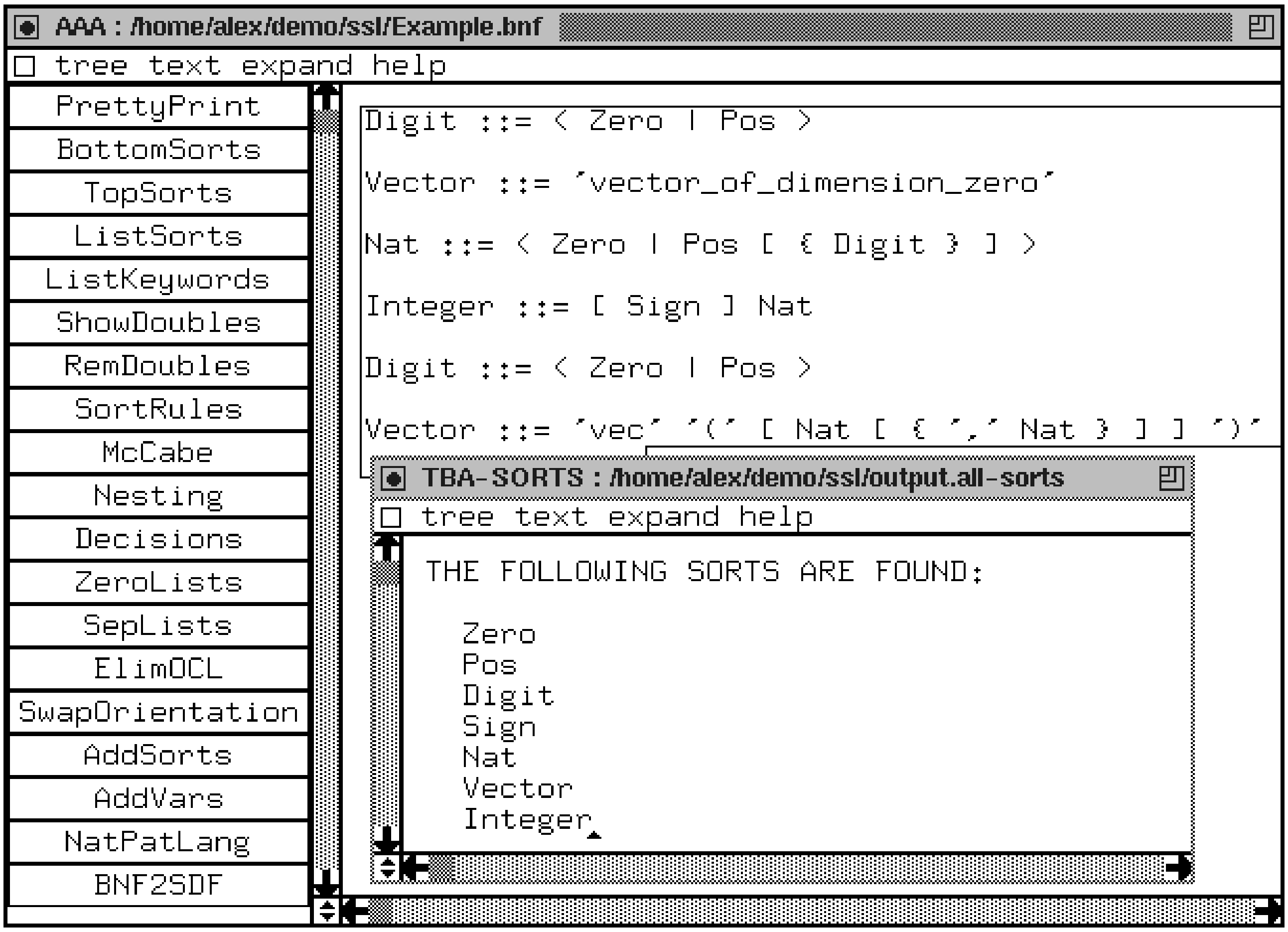


top sorts

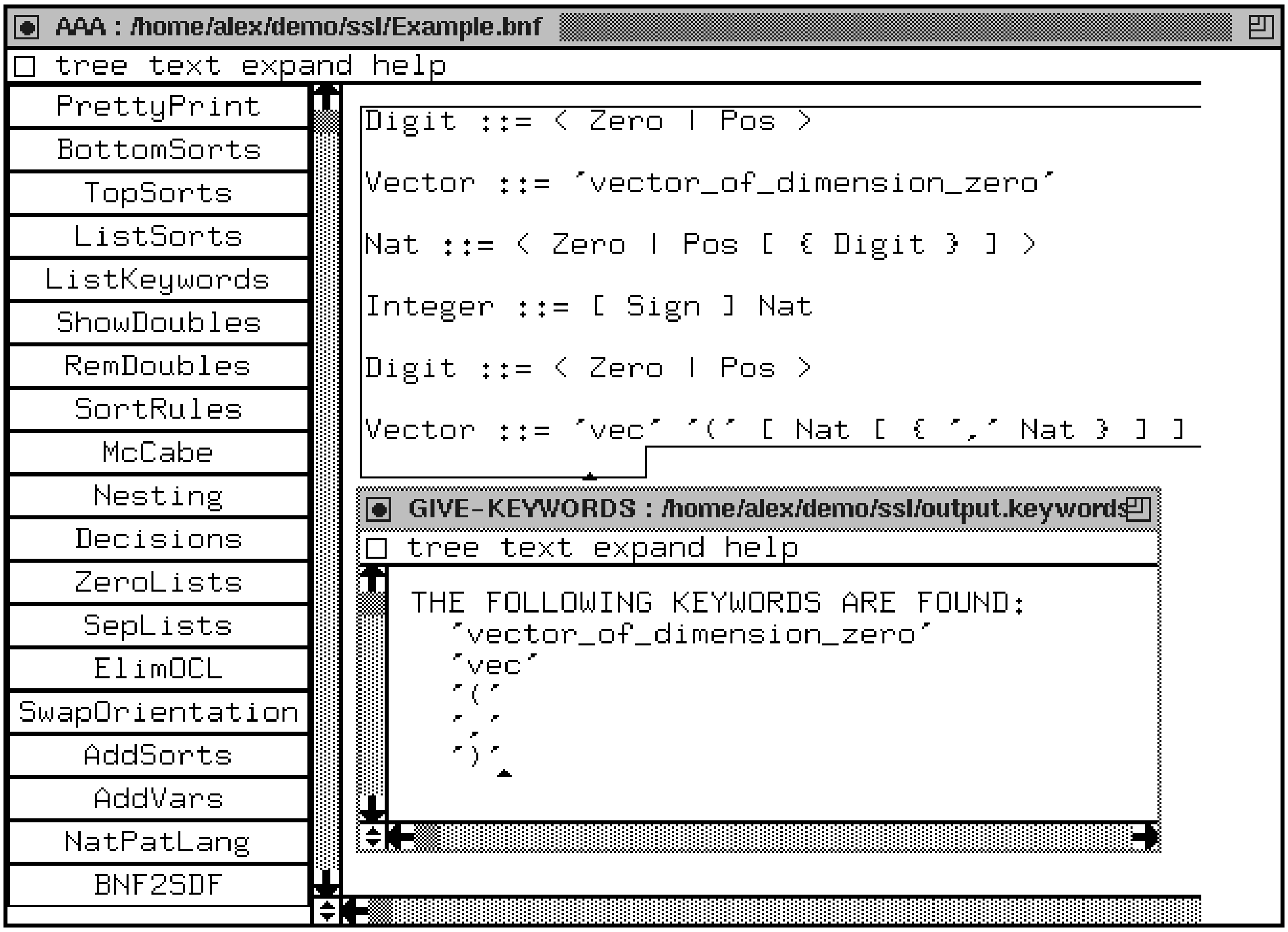


AAA : /home/alex/demo/ssl/Example.bnf
tree text expand help
PrettyPrint
BottomSorts
TopSorts
ListSorts
ListKeywords
ShowDoubles
RemDoubles
SortRules
McCabe
Nesting
Decisions
ZeroLists
SepLists
ElimOCL
SwapOrientation
AddSorts
AddVars
NatPatLang
BNF2SDF
Digit ::= < Zero | Pos >
Vector ::= 'vector_of_dimension_zero'
Nat ::= < Zero | Pos [{ Digit }] >
Integer ::= [Sign] Nat
Digit ::= < Zero | Pos >
Vector ::= 'vec' '(' [Nat [{ ',' Nat }]] ')'
TBA-SORTS : /home/alex/demo/ssl/output.top-sorts
tree text expand help
THE FOLLOWING SORTS ARE FOUND:
Vector
Integer

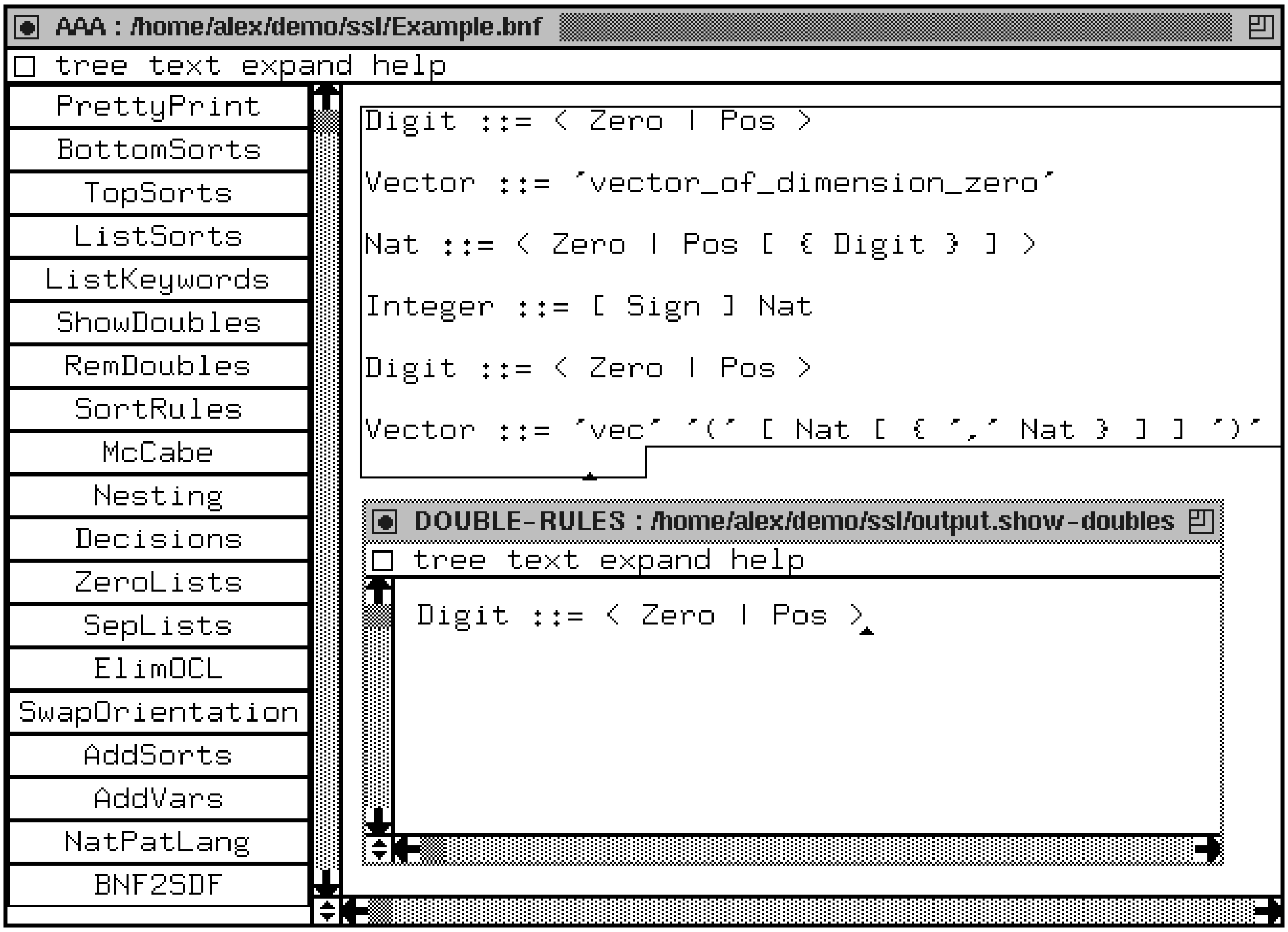
list all sorts



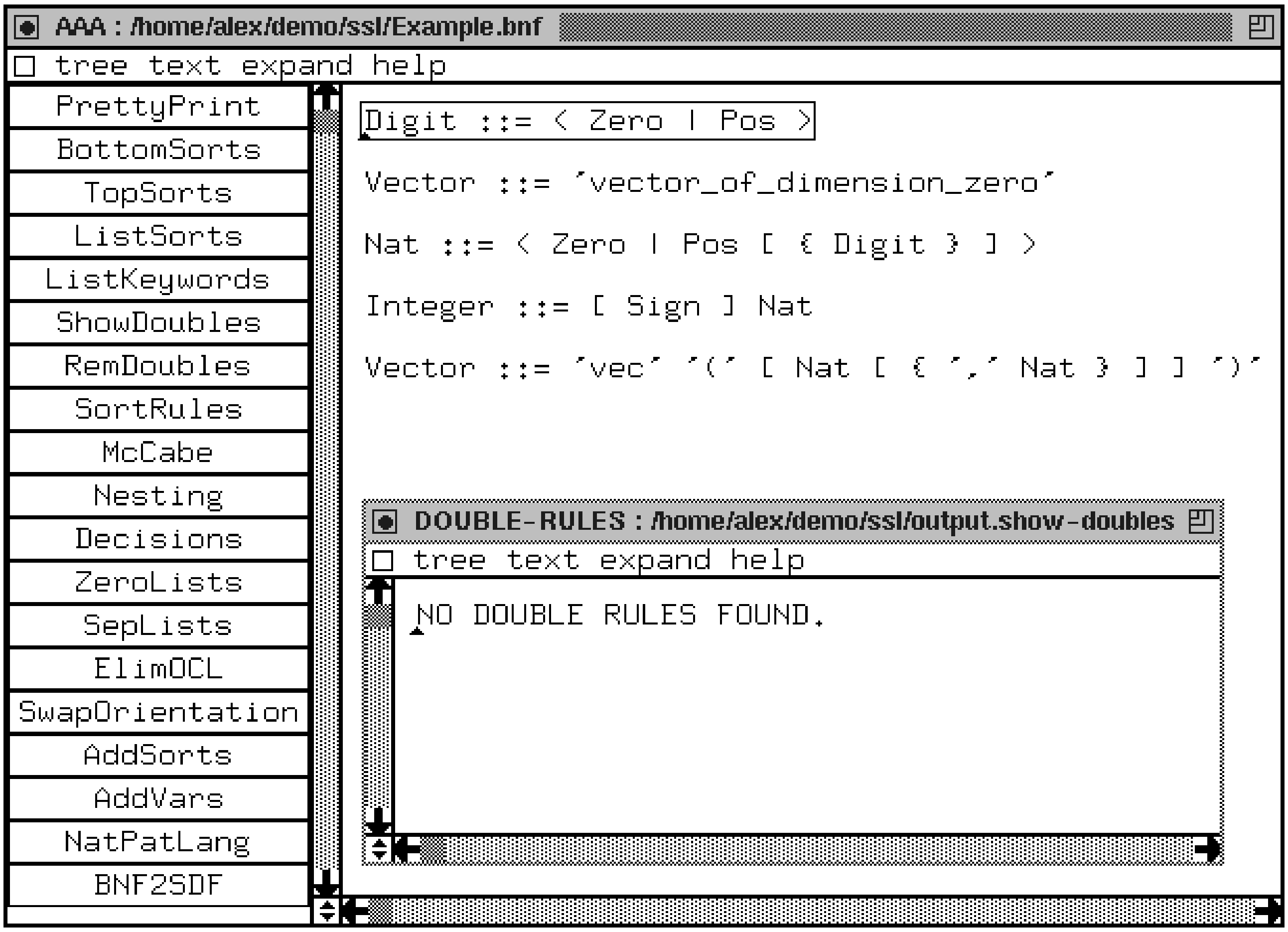
list all keywords



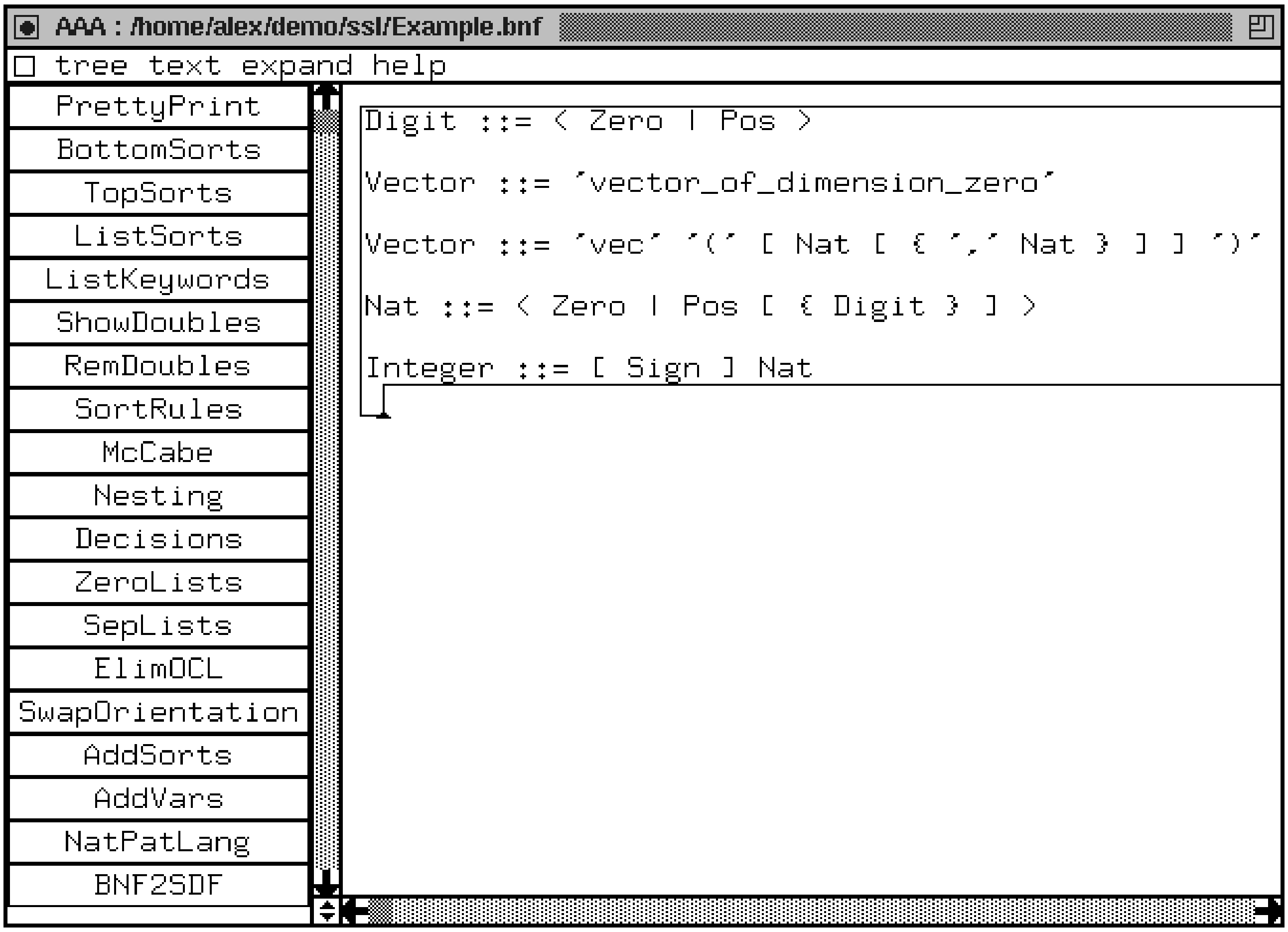
show double BNF rules



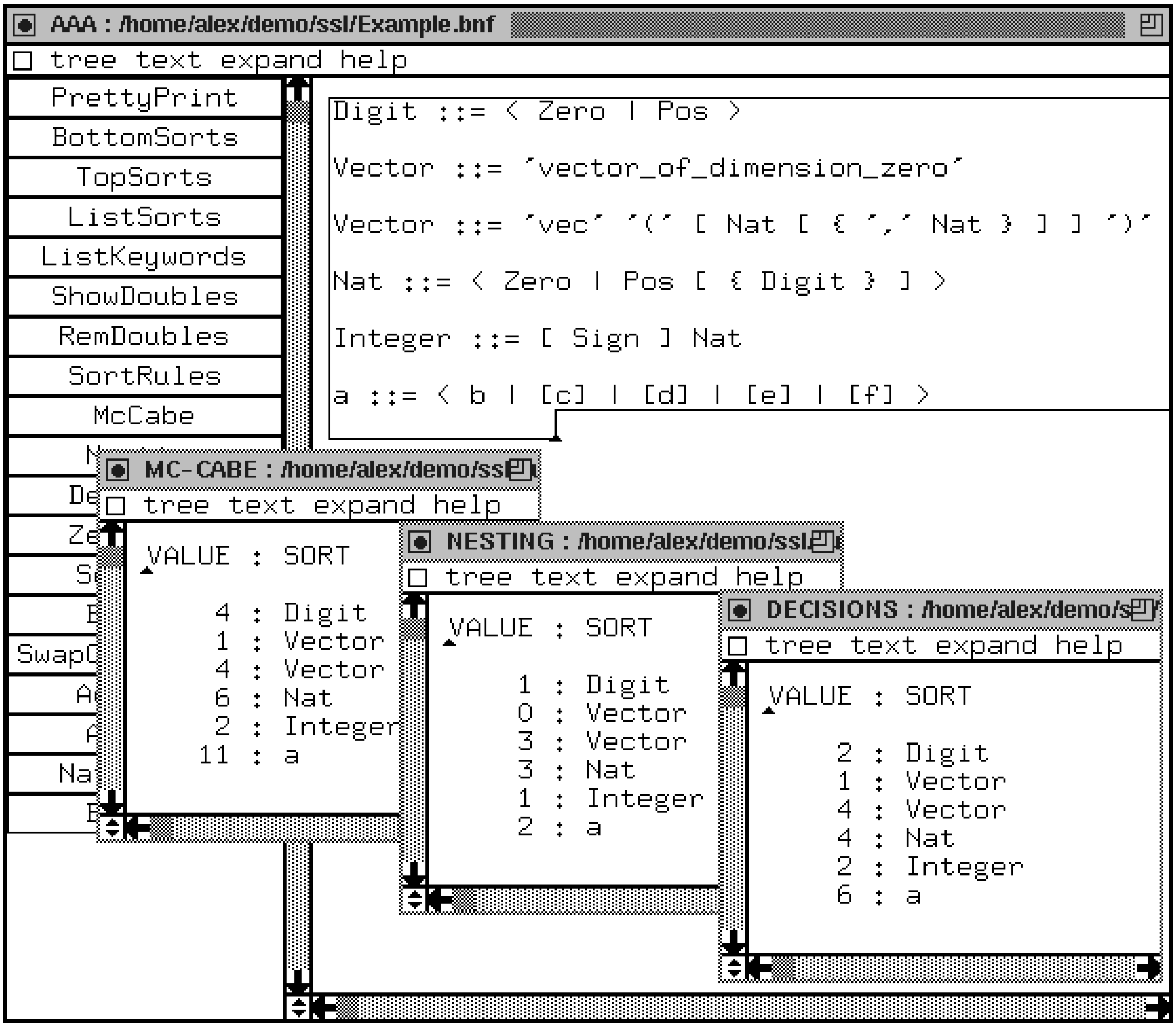
remove double BNF rules



sort BNF rules



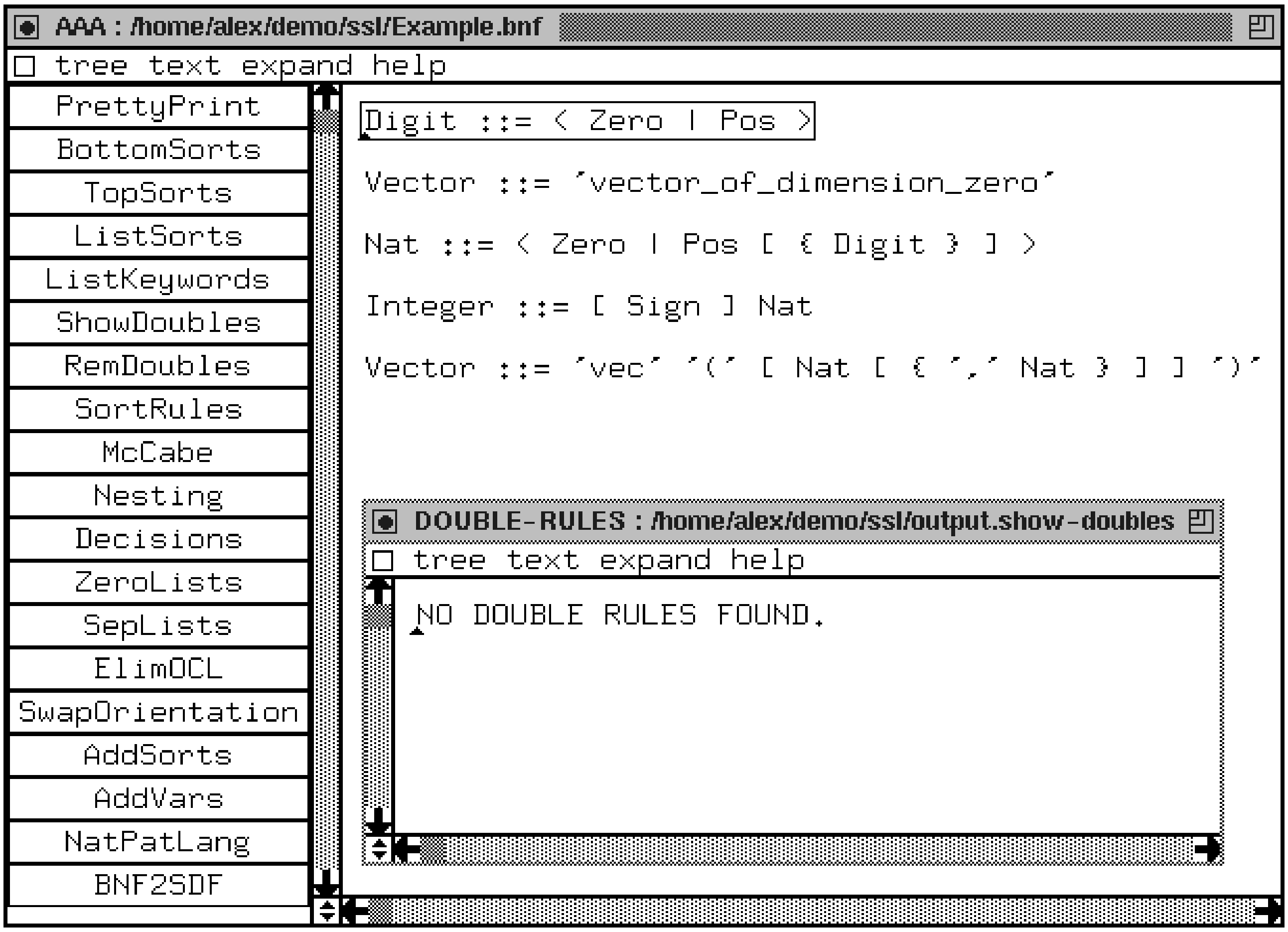
metrics on BNF



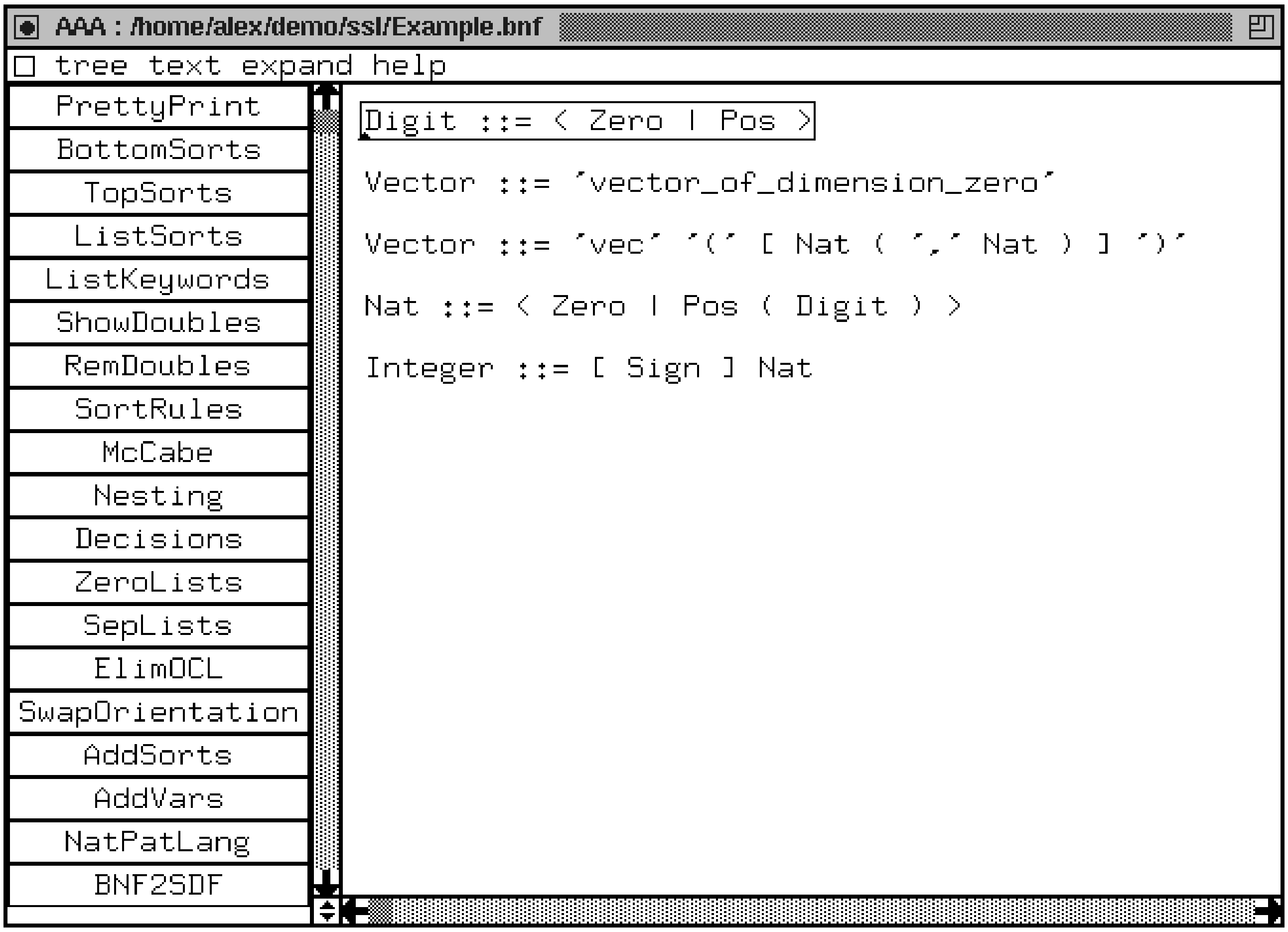
assembly line for generating native pattern languages

- introduce zero lists
- introduce separated lists
- eliminate optionals, choices and complex list items
- turn BNF into context-free functions
- declare sorts
- declare pattern variables

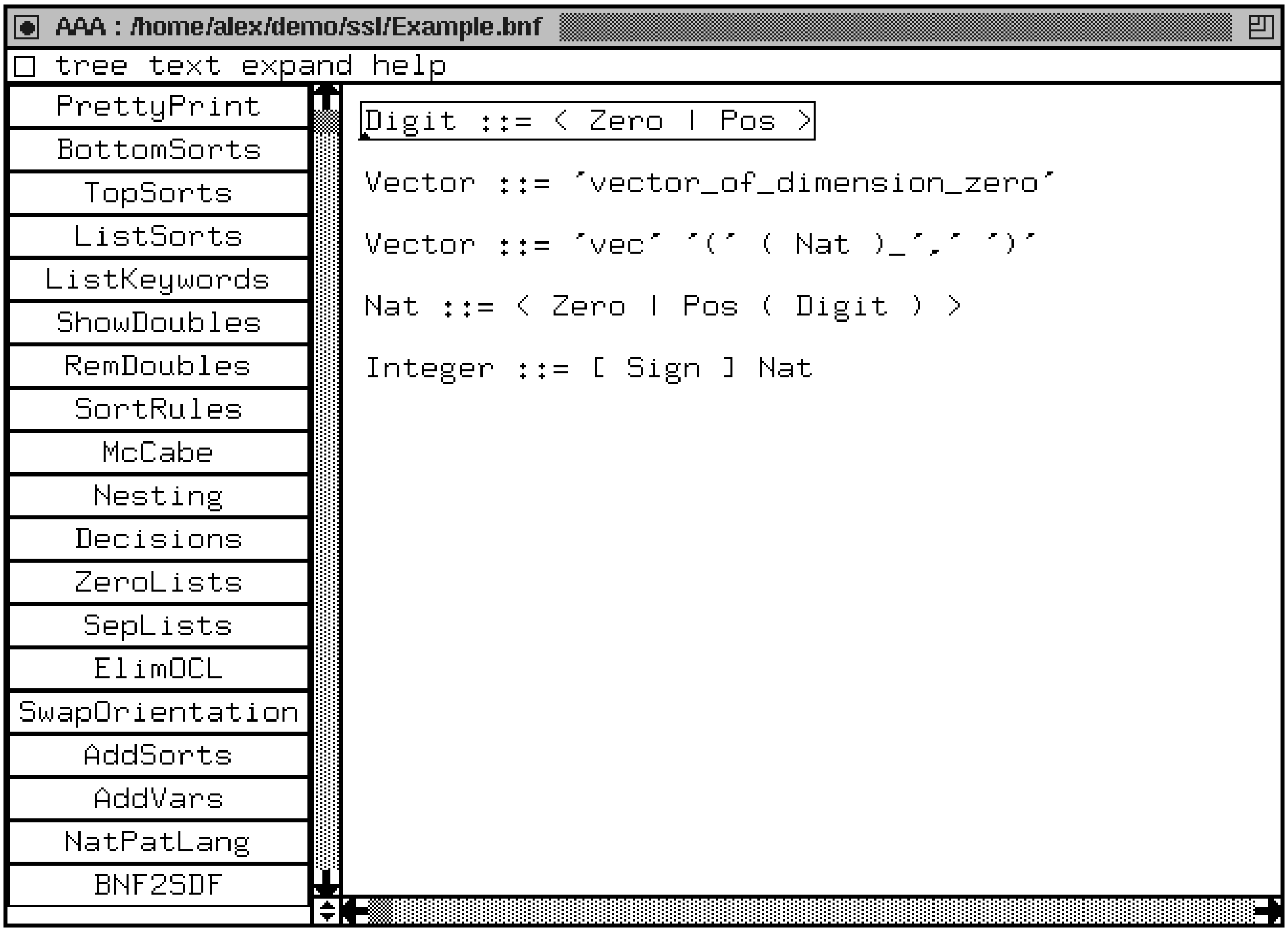
example without doubles



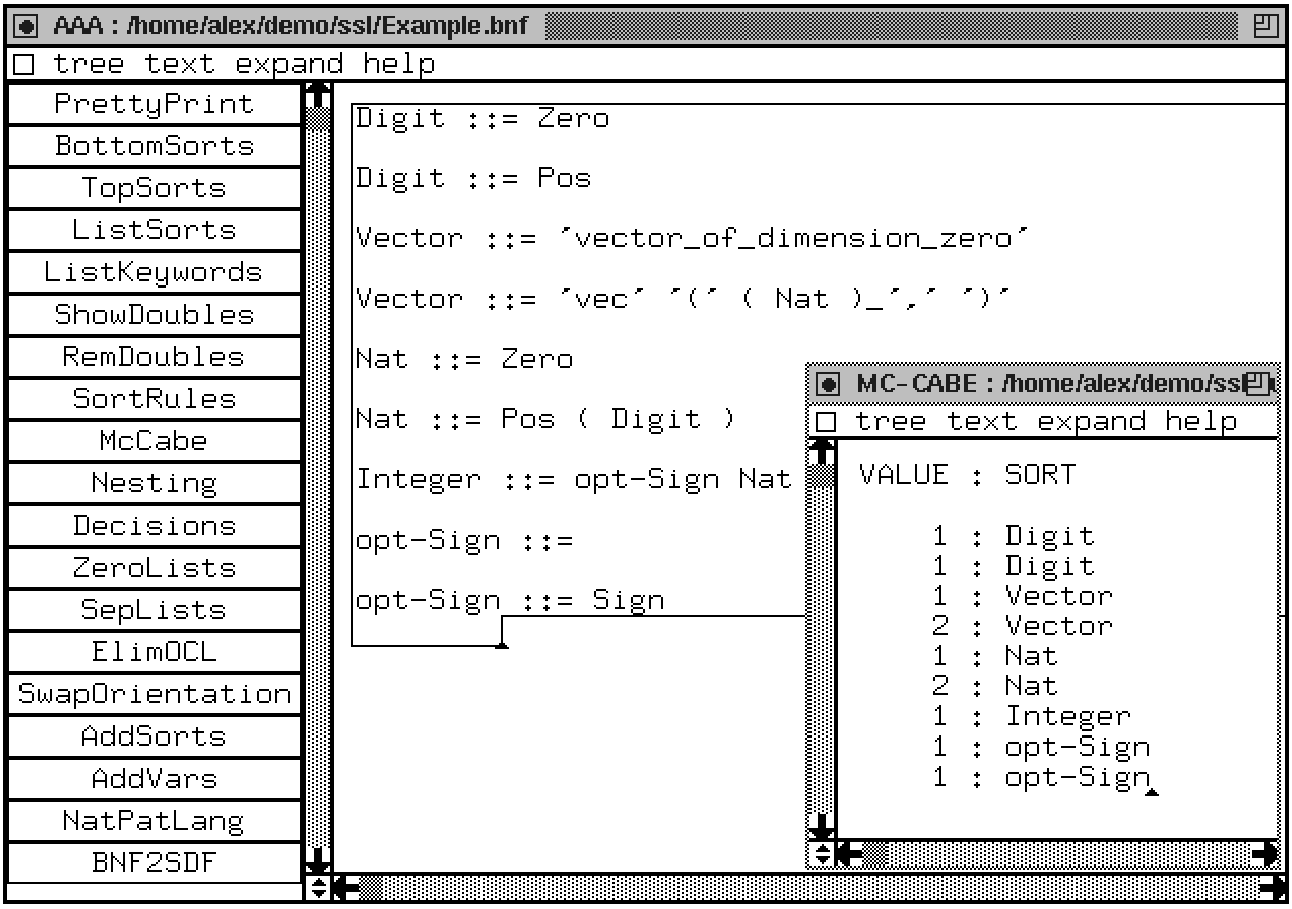
zero lists introduced



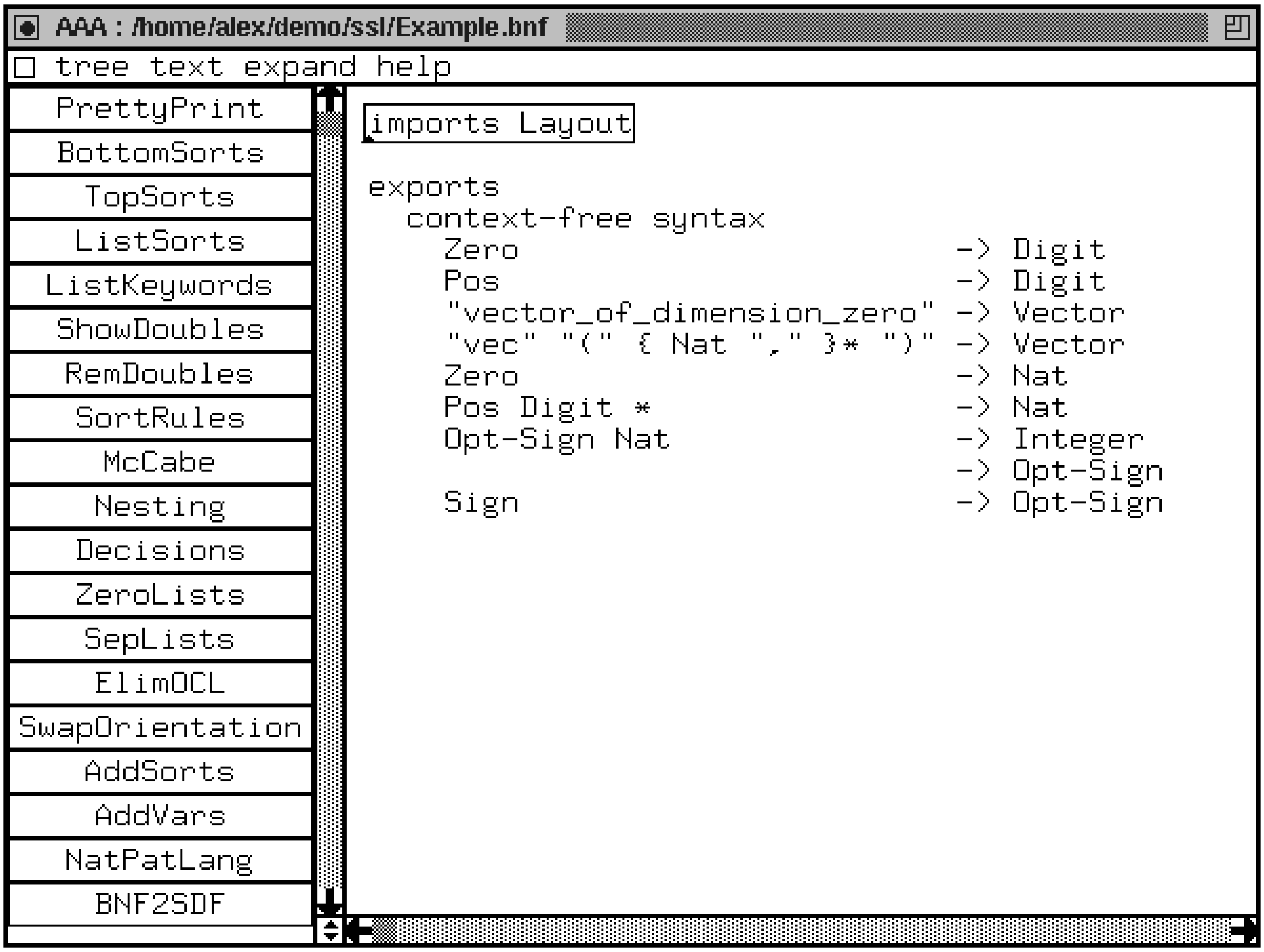
separated lists introduced



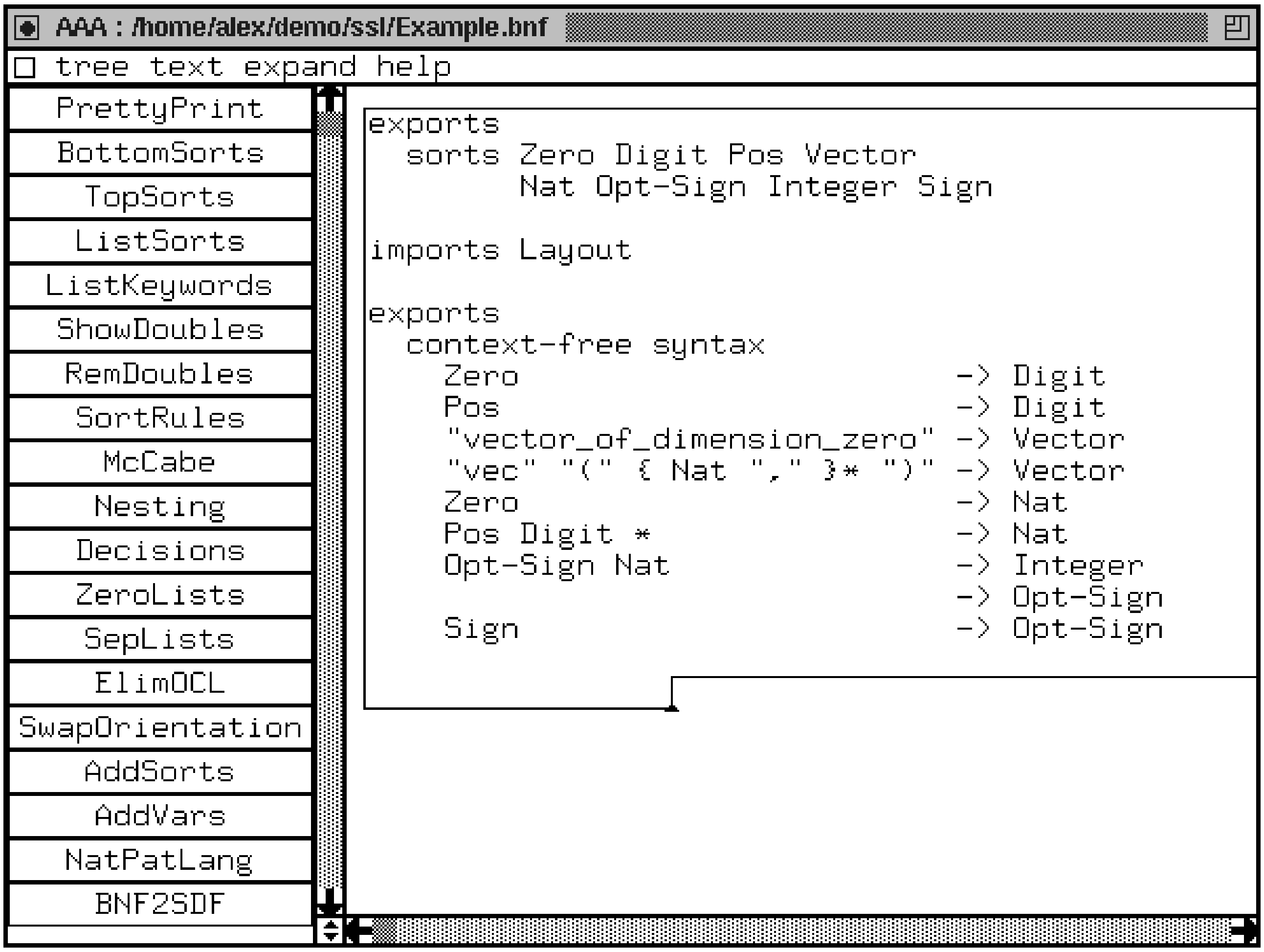
simplification of rules



BNF to context-free functions



make SDF


AAA : /home/alex/demo/ssl/Example.bnf
tree text expand help
PrettyPrint
BottomSorts
TopSorts
ListSorts
ListKeywords
ShowDoubles
RemDoubles
SortRules
McCabe
Nesting
Decisions
ZeroLists
SepLists
ElimOCL
SwapOrientation
AddSorts
AddVars
NatPatLang
BNF2SDF
exports
sorts Zero Digit Pos Vector
Nat Opt-Sign Integer Sign
imports Layout
exports
context-free syntax
Zero -> Digit
Pos -> Digit
"vector_of_dimension_zero" -> Vector
"vec" "(" { Nat "," }* ")" -> Vector
Zero -> Nat
Pos Digit * -> Nat
Opt-Sign Nat -> Integer
-> Opt-Sign
Sign -> Opt-Sign

native pattern language

ADD-VARS : /home/alex/demo/ssl/MuSSL.syn

tree text expand help

```
exports
  sorts Zero Digit Pos Vector
        Nat Opt-Sign Integer Sign

imports Layout

exports
  context-free syntax
    Zero                               -> Digit
    Pos                                -> Digit
    "vector_of_dimension_zero"         -> Vector
    "vec" "(" { Nat "," }* ")"         -> Vector
    Zero                               -> Nat
    Pos Digit *                        -> Nat
    Opt-Sign Nat                       -> Integer
                                       -> Opt-Sign
    Sign                               -> Opt-Sign

exports
  variables
    Zero      [0-9] +     -> Zero
    Zero      [0-9] + "+" -> Zero +
    Zero      [0-9] + "*" -> Zero *
    Digit     [0-9] +     -> Digit
    Digit     [0-9] + "+" -> Digit +
    Digit     [0-9] + "*" -> Digit *
    Pos       [0-9] +     -> Pos
    Pos       [0-9] + "+" -> Pos +
    Pos       [0-9] + "*" -> Pos *
    Vector    [0-9] +     -> Vector
    Vector    [0-9] + "+" -> Vector +
    Vector    [0-9] + "*" -> Vector *
    Nat       [0-9] +     -> Nat
    Nat       [0-9] + "+" -> Nat +
    Nat       [0-9] + "*" -> Nat *
    Opt-Sign [0-9] +     -> Opt-Sign
    Opt-Sign [0-9] + "+" -> Opt-Sign
                              +
    Opt-Sign [0-9] + "*" -> Opt-Sign
                              *
    Integer   [0-9] +     -> Integer
```

the end