Examples for Ebnf2ps – Automatic Railroad Diagram Drawing

Peter Thiemann, Michael Walter
Wilhelm-Schickard-Institut, Universität Tübingen
Sand 13, 72076 Tübingen
E-mail: {thiemann,walterm}@informatik.uni-tuebingen.de

May 18, 1996

Contents

1 gofer.BNF

This section gives a summary of the grammar for the language used by Gofer. Gofer's BNF is originally taken from Mark Jones's Gofer distribution. The diagrams have been produced with:

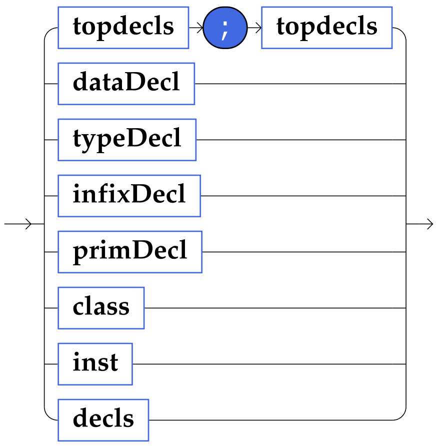
```
> Ebnf2ps -verbose
          -titleFont Palatino-BoldItalic
          -ntFont Palatino-Bold
          -ntBoxColor RoyalBlue
          -tFont Courier
          -tColor White
          -tBg RoyalBlue
          -fatLineWidth 50
          +simplify
          gofer.BNF '.*'

using colors:
        ntColor          (0,0,0)
        tColor           (255,255,255)
        lineColor        (0,0,0)
        ntBoxColor       (65,105,225)
        tBoxColor        (0,0,0)
        ntBg             (255,255,255)
        tBg              (65,105,225)
        titleColor       (0,0,0)
```

```
from rgbPathDefault: ["/usr/lib/X11", "/usr/local/X11R5/lib/X11"]
generating nonterminals: [".*"]
from gofer.BNF
using input path ["/home/walterm/tests/"]
using ebnfInput
produce layout and output
>
```

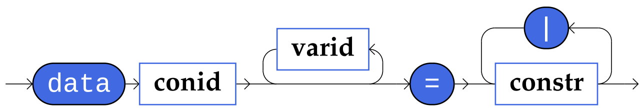
module

interp

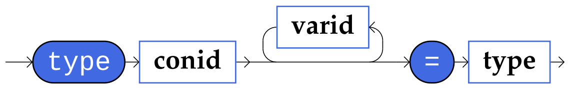
topdecls



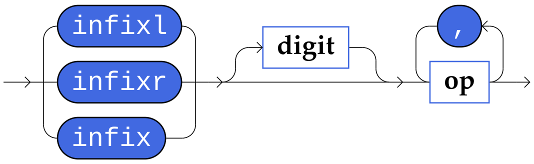
dataDecl



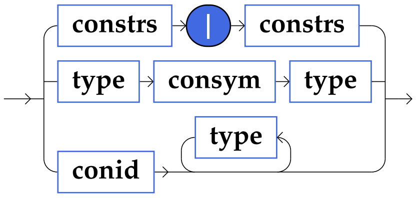
typeDecl



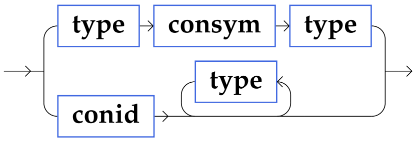
infixDecl



constrs

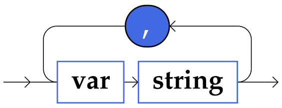


constr

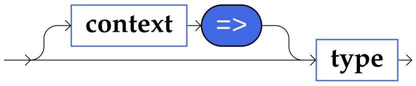


primDecl

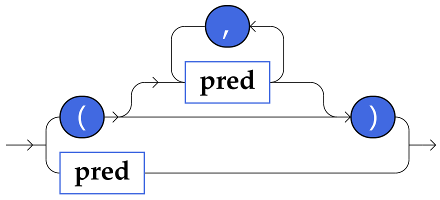
prims



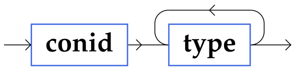
sigType



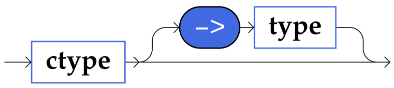
context



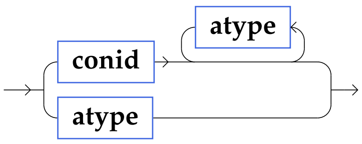
pred



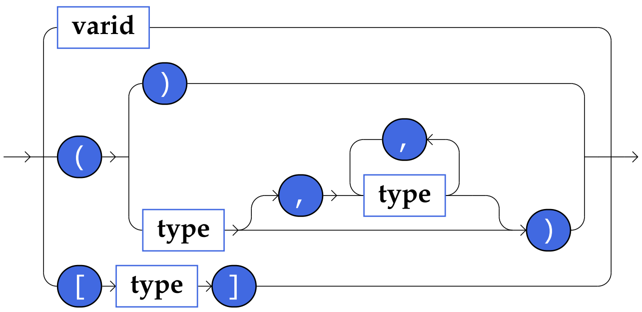
type



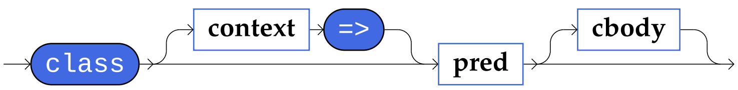
ctype



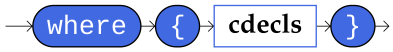
atype



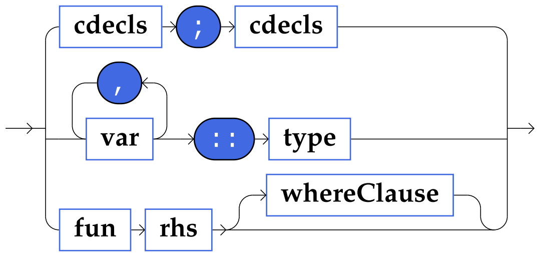
class



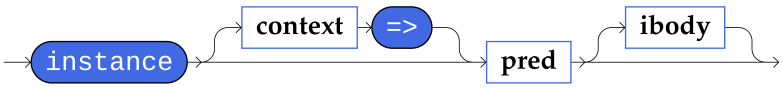
cbody



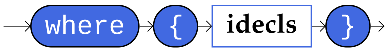
cdecls



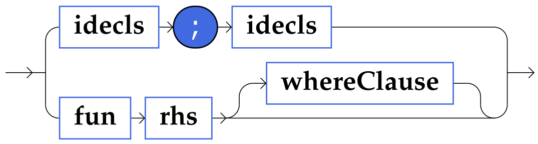
inst



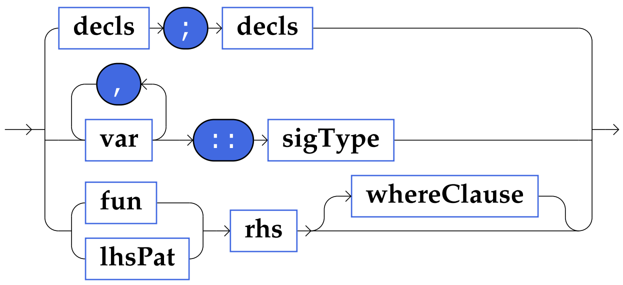
ibody



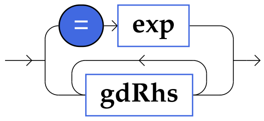
idecls



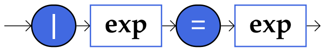
decls



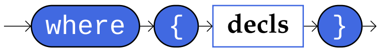
rhs



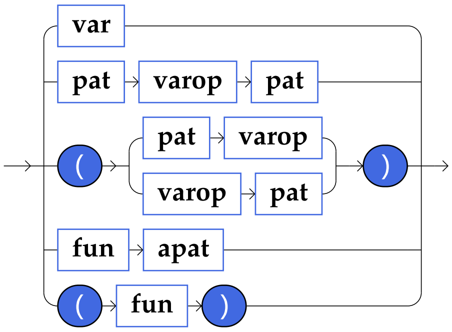
gdRhs



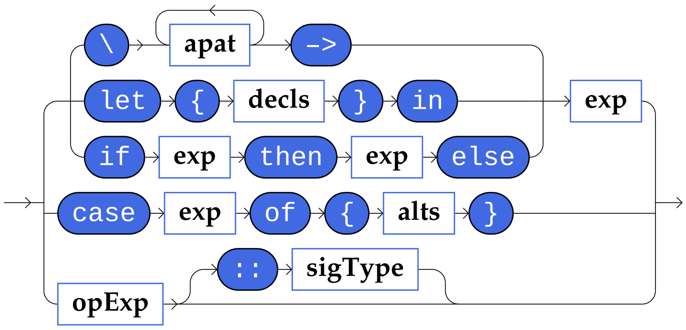
whereClause



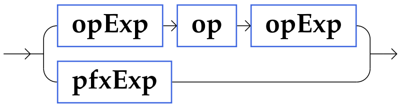
fun



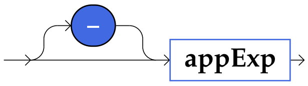
exp



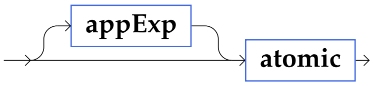
opExp



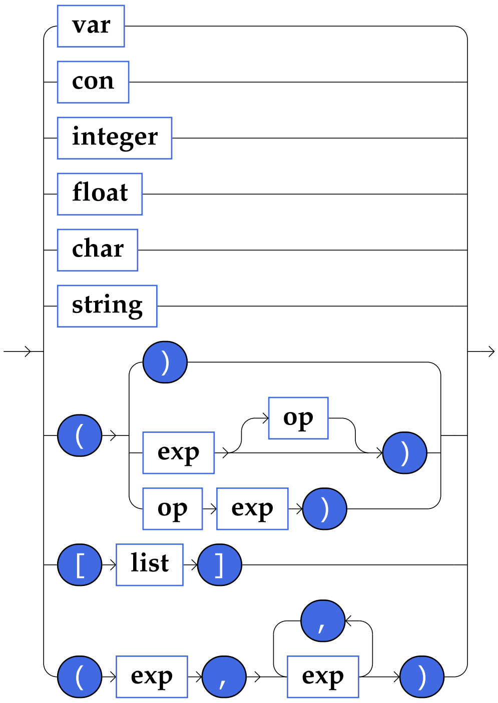
pfxExp



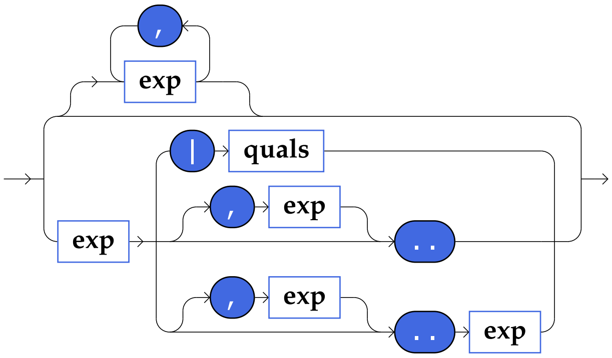
appExp



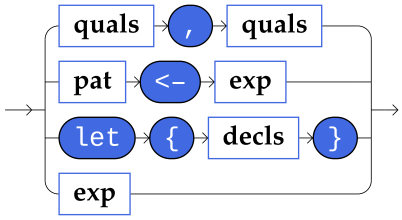
atomic



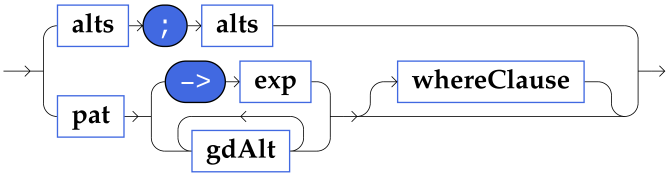
list



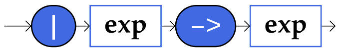
quals



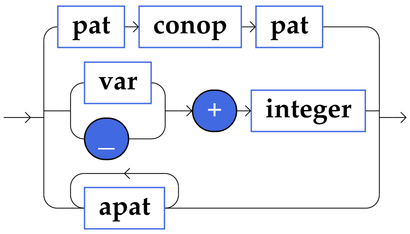
alts



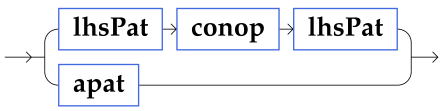
gdAlt



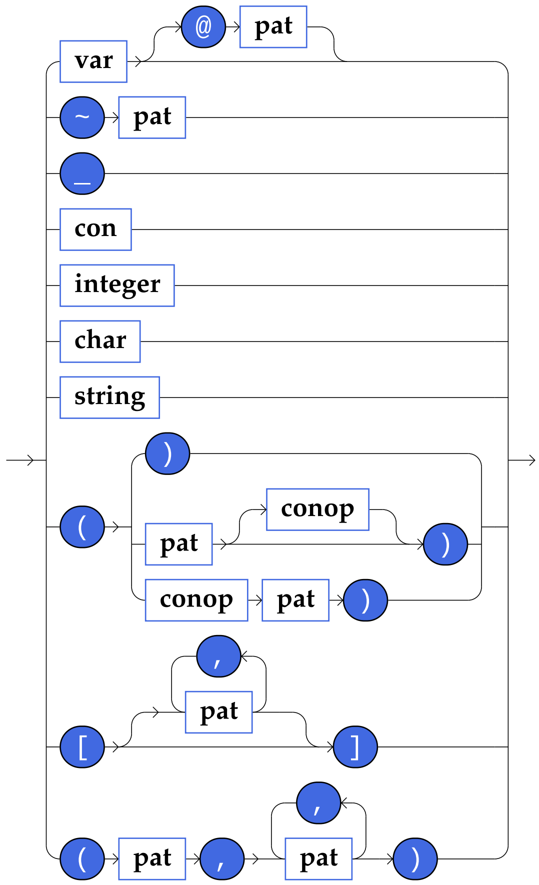
pat



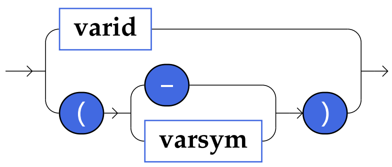
lhsPat



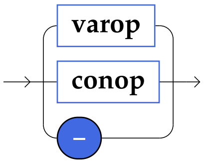
apat



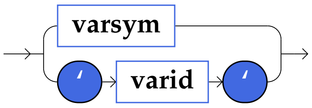
var



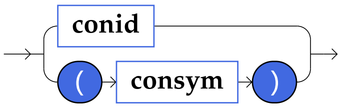
op



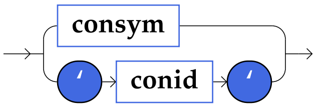
varop



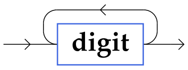
con



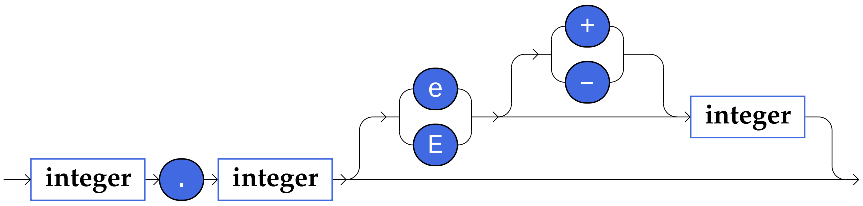
conop



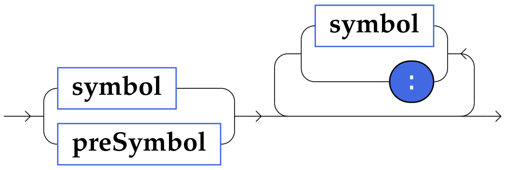
integer



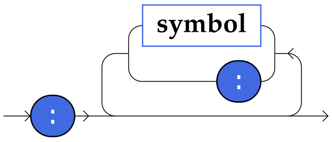
float



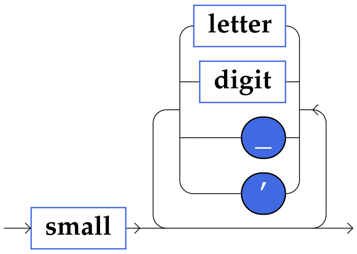
varsym



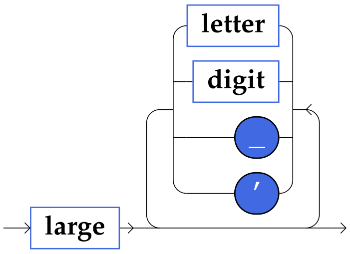
consym



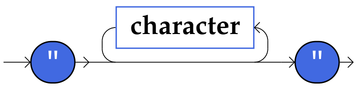
varid



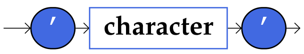
conid



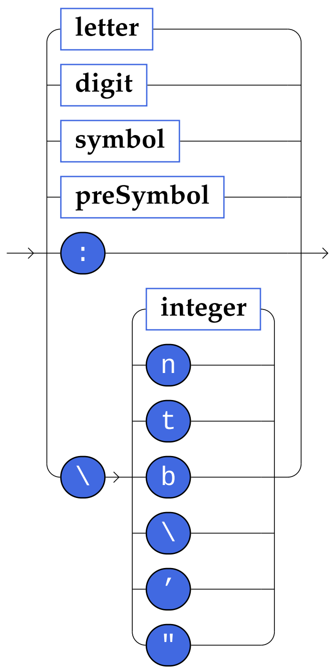
string



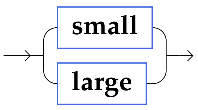
char



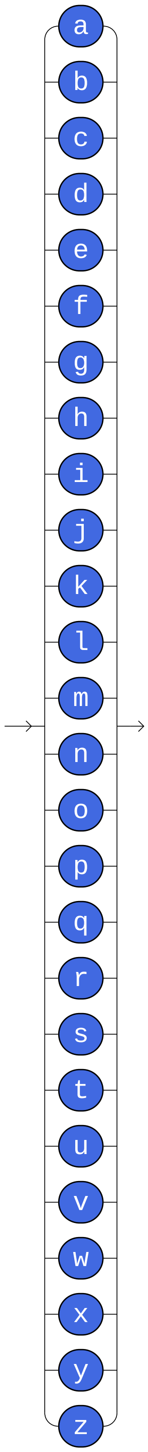
character



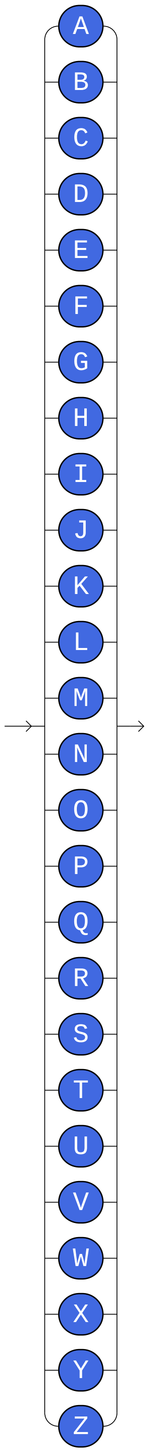
letter



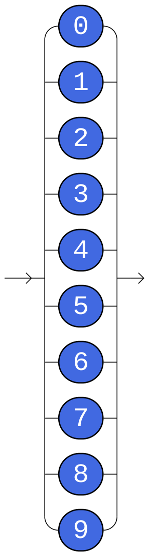
small



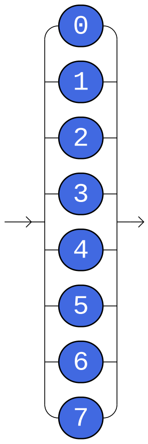
large



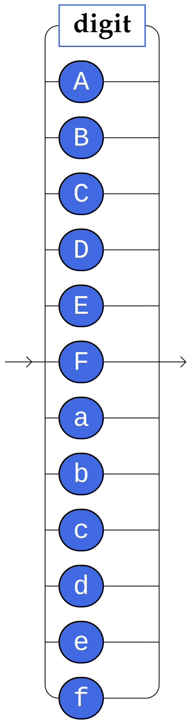
digit



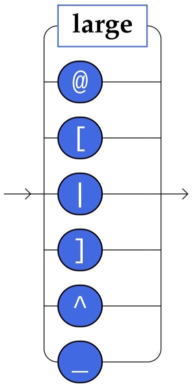
octit



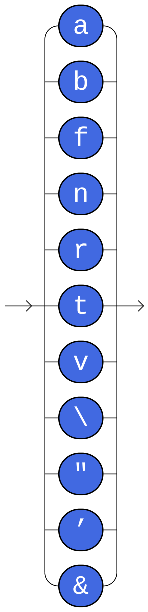
hexit



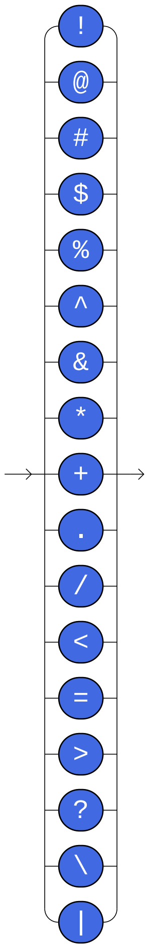
cntrl



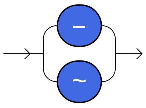
charesc



symbol



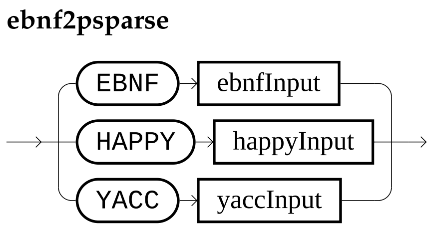
preSymbol

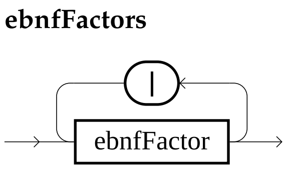


2 Ebnf2psParser.ly

`Ebnf2psParser.ly` is the Happy parser for Ebnf2ps. The diagrams have been produced with

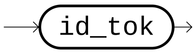
```
> Ebnf2ps -verbose
          -titleFont AGaramondExp-Bold
          -tFont Courier
          -borderDistX 650
          +simplify
          +ebnf
          -happy
          Ebnf2psParser.ly '.*'

using colors:
        ntColor          (0,0,0)
        tColor           (0,0,0)
        lineColor        (0,0,0)
        ntBoxColor       (0,0,0)
        tBoxColor        (0,0,0)
        ntBg             (255,255,255)
        tBg              (255,255,255)
        titleColor       (0,0,0)
from rgbPathDefault: ["/usr/lib/X11", "/usr/local/X11R5/lib/X11"]
generating nonterminals: [".*"]
from Ebnf2psParser.ly
using input path ["/home/walterm/tests/"]
using happyInput
empty productions
produce EBNF file for happyInput
        Ebnf2psParser.ly.BNF
produce layout and output
>
```



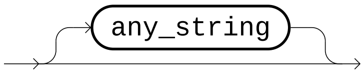


ebnfInput

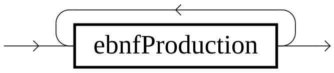
ebnfNonterminal



ebnfOptString

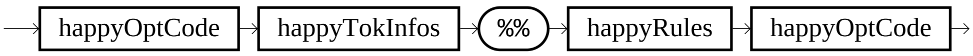


ebnfProductions



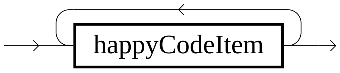
ebnfProduction

happyInput



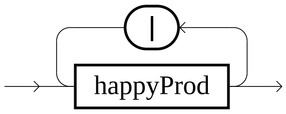
happyCode

happyCodeBody

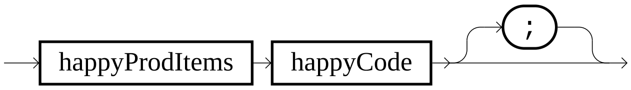


happyOptCode

happyProds

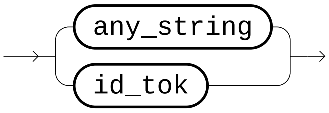


happyProd

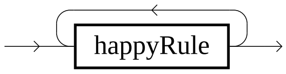


happyProdItems

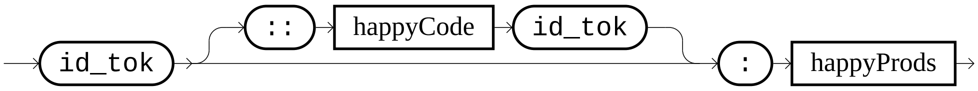
happyProdItem



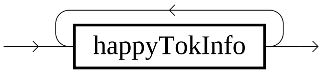
happyRules



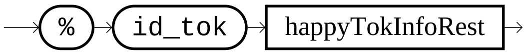
happyRule



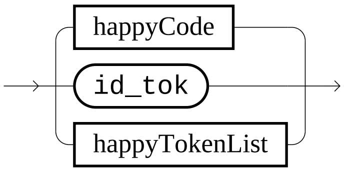
happyTokInfos



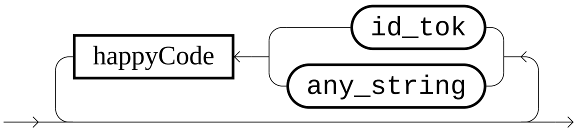
happyTokInfo



happyTokInfoRest

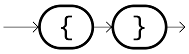


happyTokenList

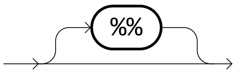


yaccInput

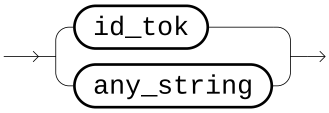
yaccAction



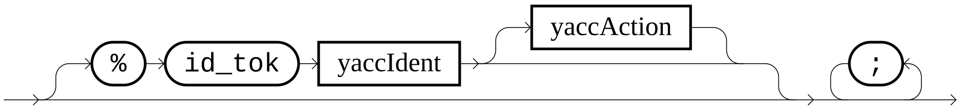
yaccEnd



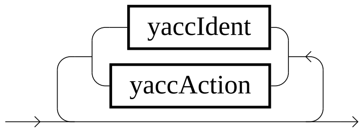
yaccIdent



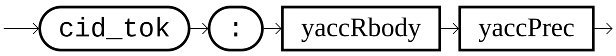
yaccPrec



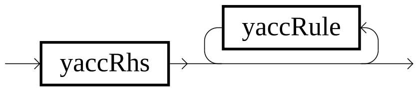
yaccRbody



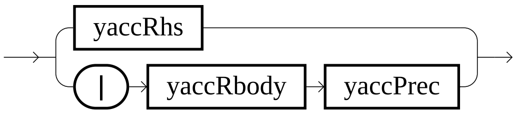
yaccRhs



yaccRules



yaccRule



3 hsparser.y

This section gives a summary of the LALR(1) syntax for Haskell 1.2. The yacc file hsparser.y by Maria M. Gutierrez is originally taken from a Glasgow Haskell Compiler distribution (The AQUA Team, Departement of Computer Science, University of Glasgow, Scotland: Glasgow FP group page URL `http://www.dcs.gla.ac.uk/fp/`). The diagrams have been produced with:

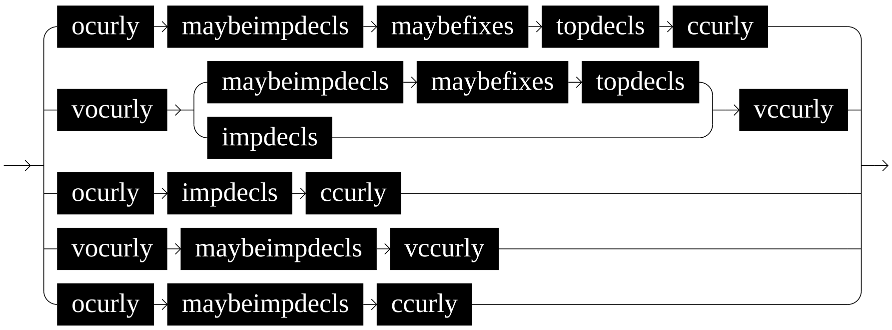
```
> Ebnf2ps -verbose
          +unfold
          +simplify
          -yacc
          -titleScale 12
          -ntBg black
          -ntColor white
          -tFont Courier-Bold
          -tScale 8
          -fatLineWidth 40
          hsparser.y '.*' -- '.*key'

using colors:
        ntColor          (255,255,255)
        tColor           (0,0,0)
        lineColor        (0,0,0)
        ntBoxColor       (0,0,0)
        tBoxColor        (0,0,0)
        ntBg             (0,0,0)
        tBg              (255,255,255)
        titleColor       (0,0,0)
from rgbPathDefault: ["/usr/lib/X11", "/usr/local/X11R5/lib/X11"]
generating nonterminals: [".*"]
from hsparser.y
using input path ["/home/walterm/tests/"]
using yaccInput
empty productions
        layout
unfold nonterminals
        classkey
        datakey
        defaultkey
        importkey
        instkey
        lazykey
        minuskey
        modulekey
        obrackkey
```

```
        oparenkey
        typekey
in the following productions
        conk
        vark
        aapatk
        dpatk
        iinstd
        iclassd
        idatad
        ityped
        defaultd
        instd
        classd
        datad
        typed
        iimport
        module
produce layout and output
>
```

pmodule

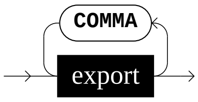
module

body

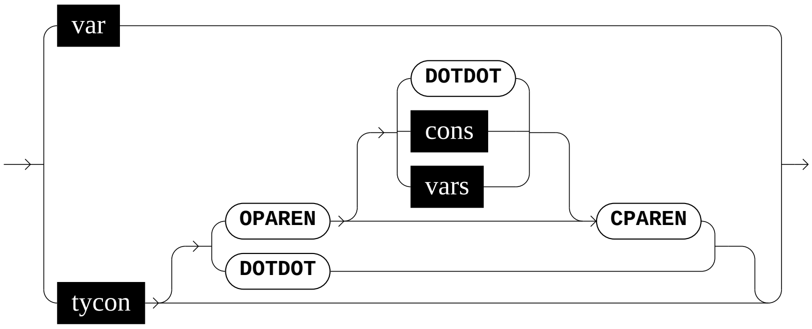


maybeexports

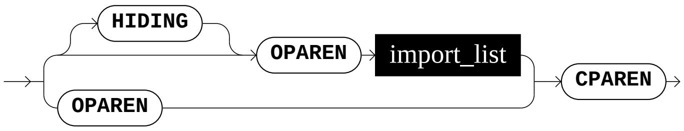
export_list



export

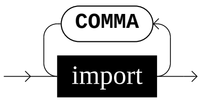


impspec

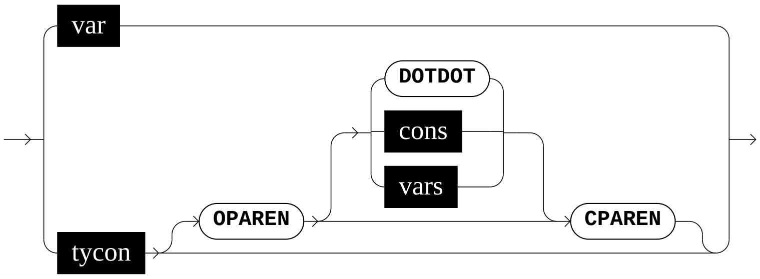


maybeimpspec

import_list



import

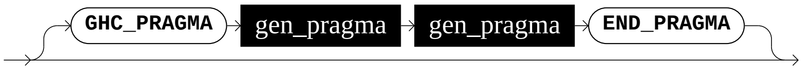


idata_pragma

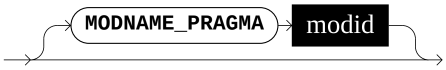
itype_pragma

iclas_pragma

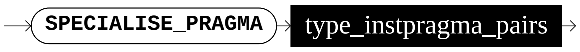
iclasop_pragma



modname_pragma

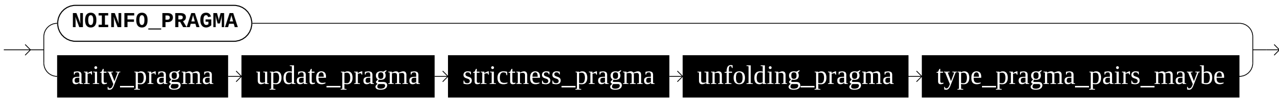


restof_iinst_spec

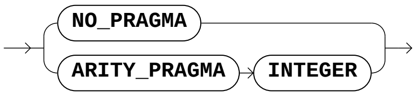


ival_pragma

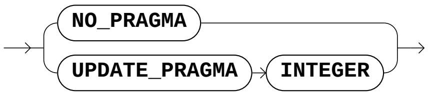
gen_pragma



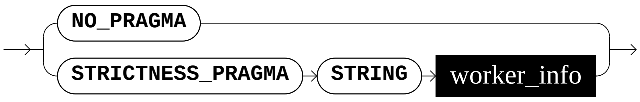
arity_pragma



update_pragma

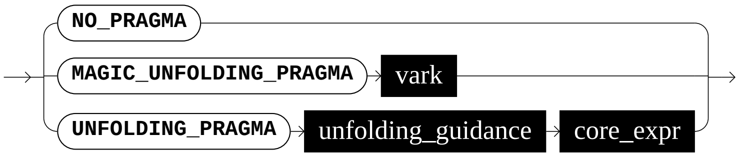


strictness_pragma

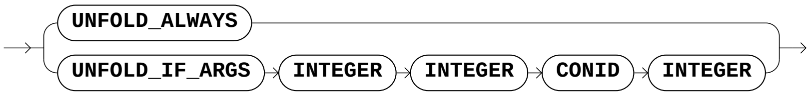


worker_info

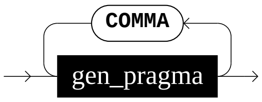
unfolding_pragma



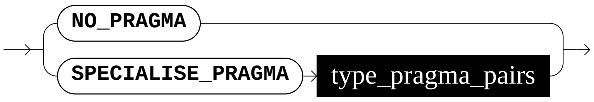
unfolding_guidance



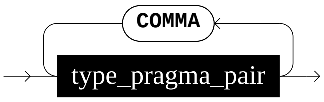
gen_pragma_list



type_pragma_pairs_maybe

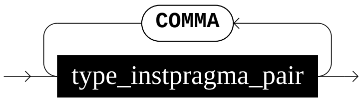


type_pragma_pairs



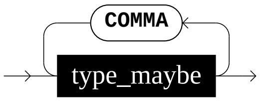
type_pragma_pair

type_instpragma_pairs

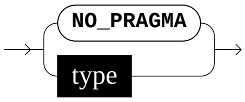


type_instpragma_pair

type_maybes

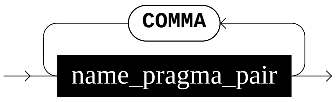


type_maybe

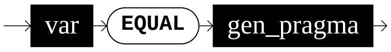


maybe_name_pragma_pairs

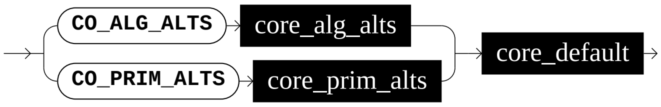
name_pragma_pairs



name_pragma_pair

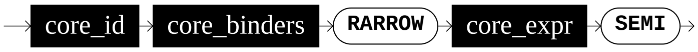


core_case_alts



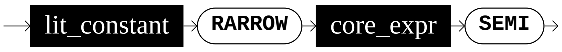
core_alg_alts

core_alg_alt

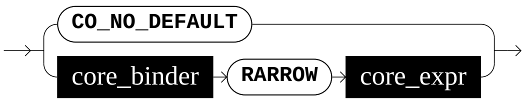


core_prim_alts

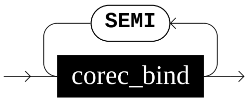
core_prim_alt



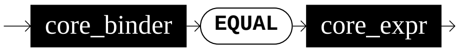
core_default



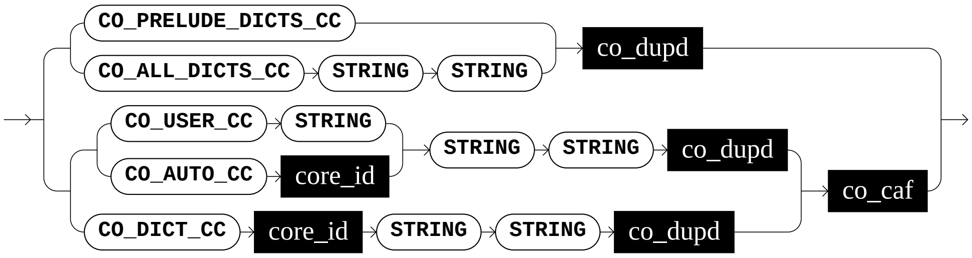
corec_binds



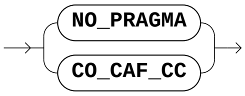
corec_bind



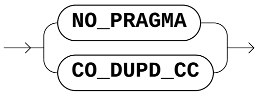
co_scc



co_caf



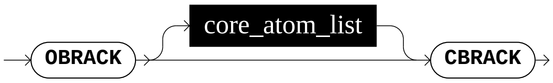
co_dupd



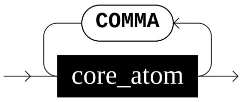
core_binders

core_binder

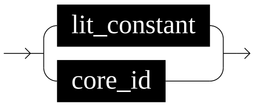
core_atoms



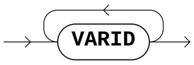
core_atom_list



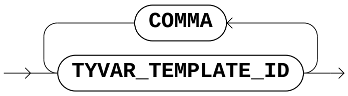
core_atom



core_tyvars

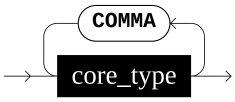


core_tv_templates



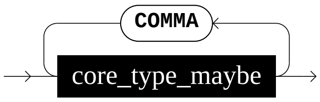
core_types

core_type_list

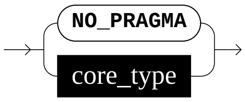


core_type

core_type_maybes



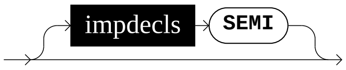
core_type_maybe



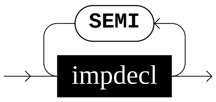
readpreludecore

readprelude

maybeimpdecls

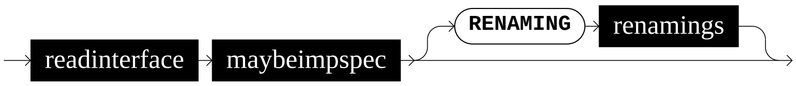


impdecls



impdecl

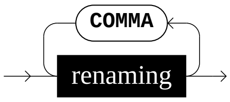
impdecl_rest



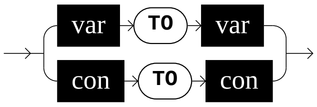
readinterface

renamings

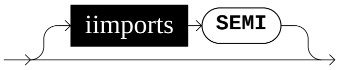
renaming_list



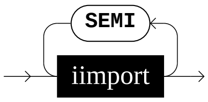
renaming



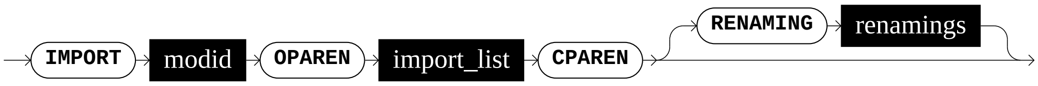
maybeiimports



iimports

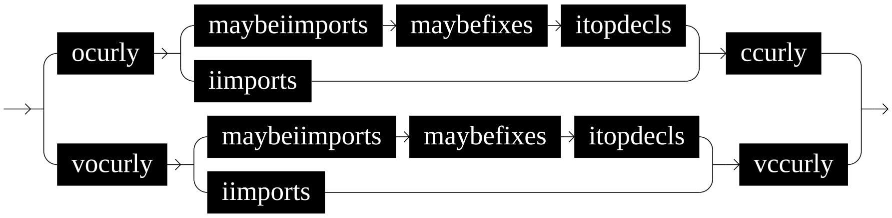


iimport

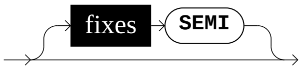


interface

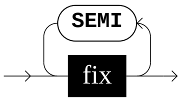
ibody



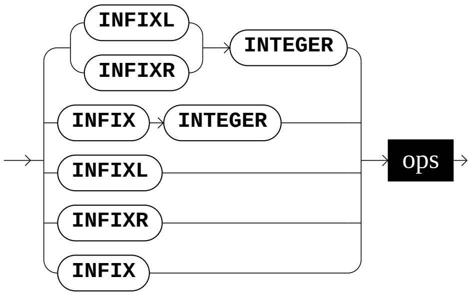
maybefixes



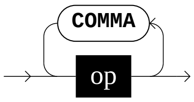
fixes



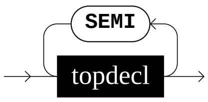
fix



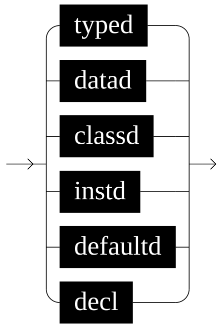
ops



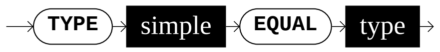
topdecls



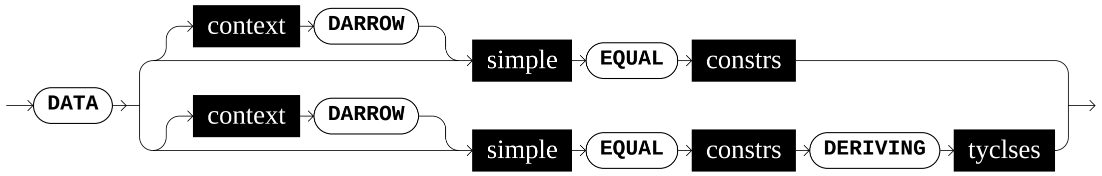
topdecl



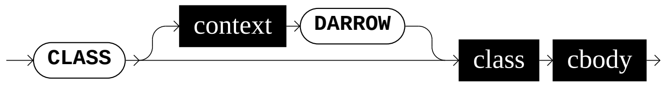
typed



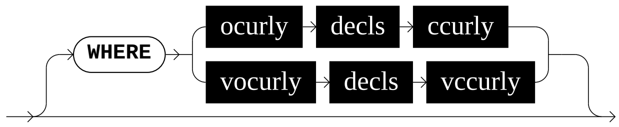
datad



classd

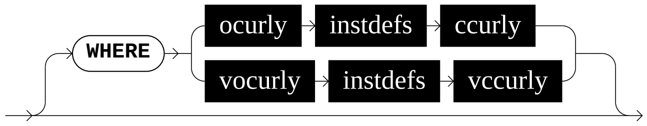


cbody

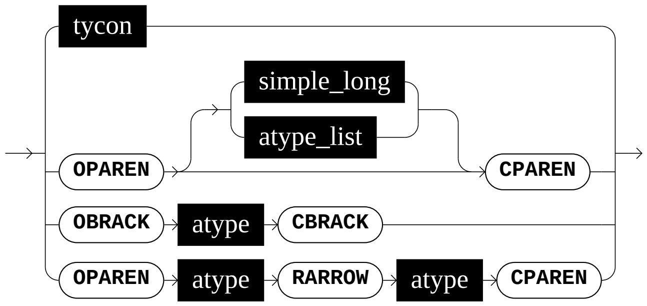


instd

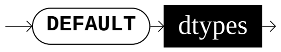
rinst



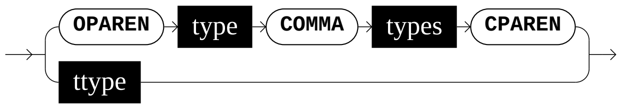
inst



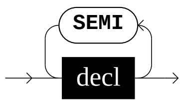
defaultd



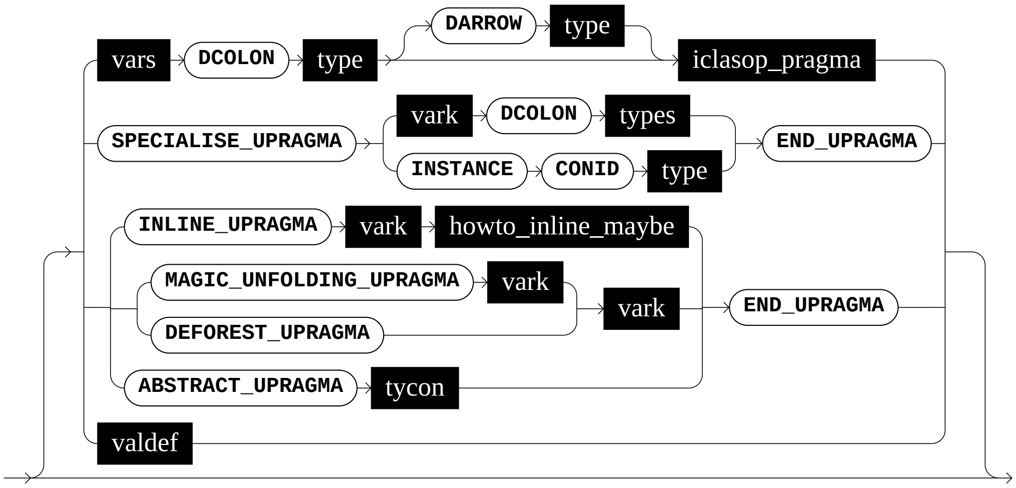
dtypes



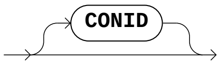
decls



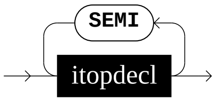
decl



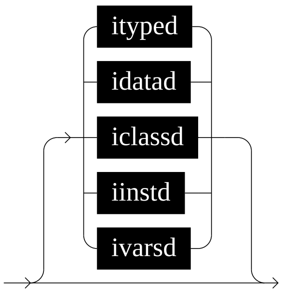
howto_inline_maybe



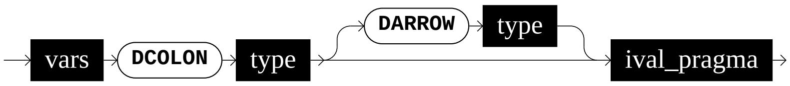
itopdecls



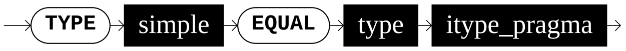
itopdecl



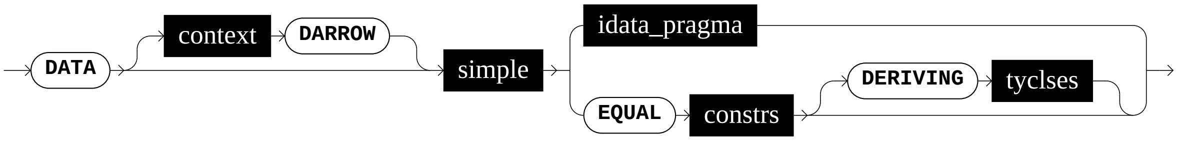
ivarsd



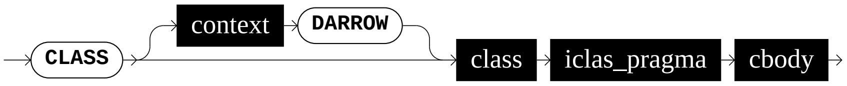
ityped



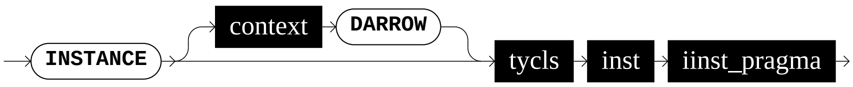
idatad



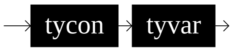
iclassd



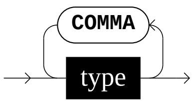
iinstd



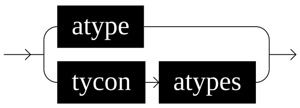
class



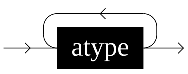
types



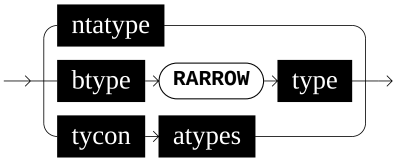
btype



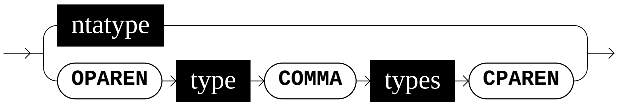
atypes



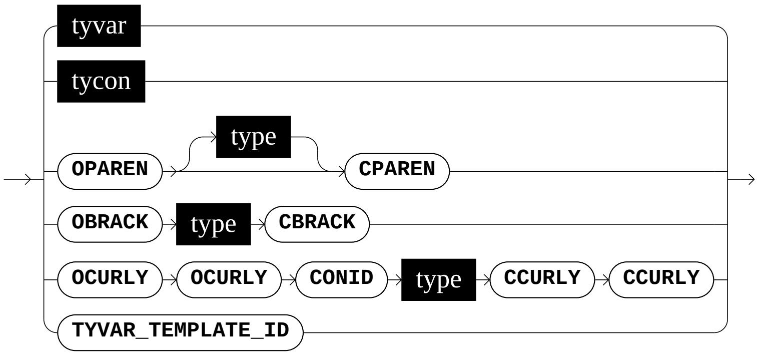
ttype



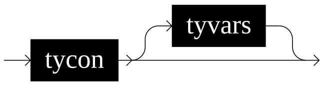
atype



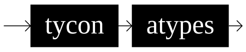
ntatype



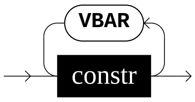
simple



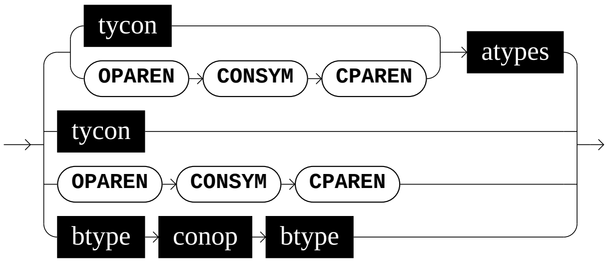
simple_long



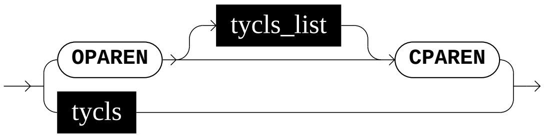
constrs



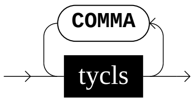
constr



tyclses

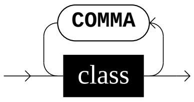


tycls_list

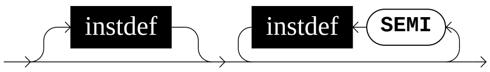


context

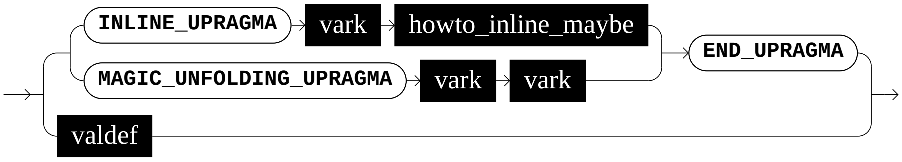
context_list



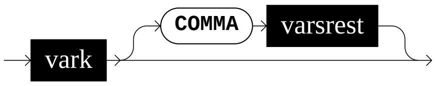
instdefs



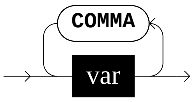
instdef



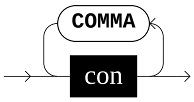
vars



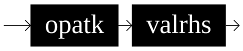
varsrest



cons

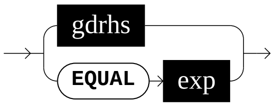


valdef

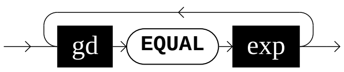


valrhs

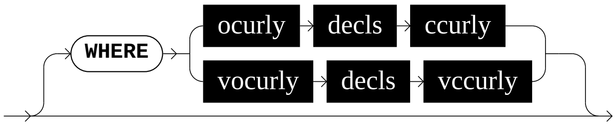
valrhs1



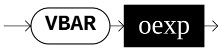
gdrhs



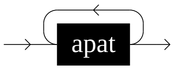
maybe_where



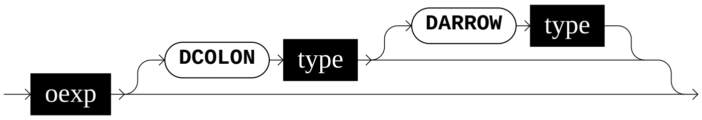
gd



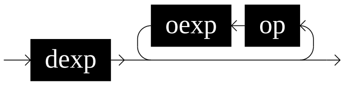
lampats



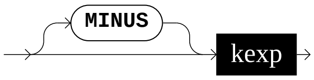
exp



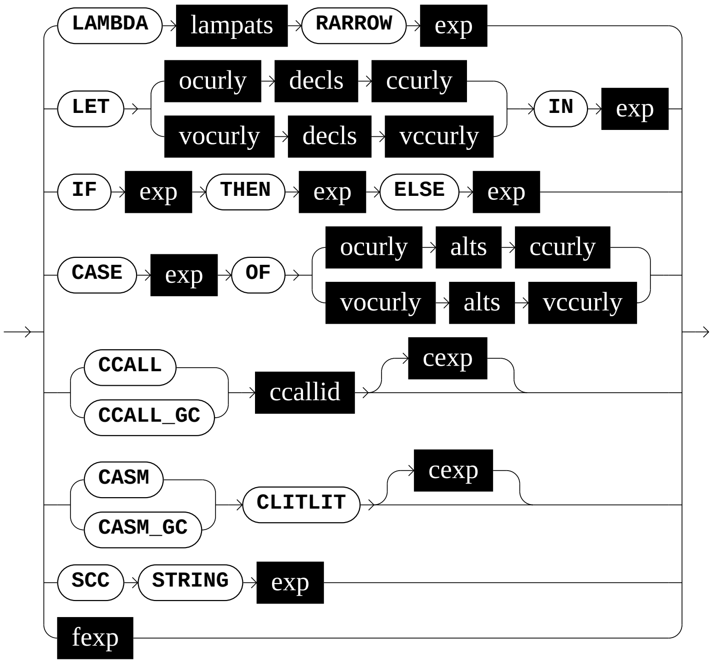
oexp



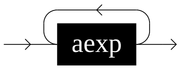
dexp



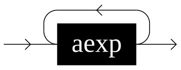
kexp



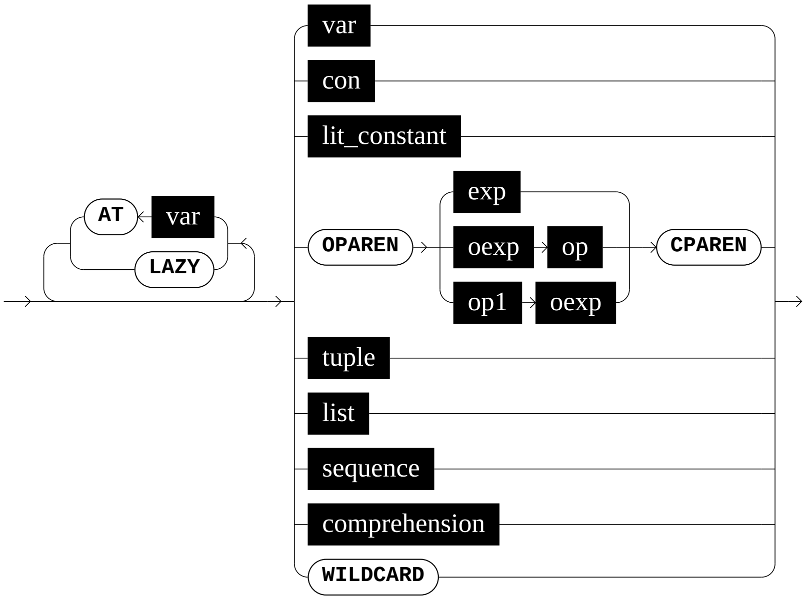
fexp



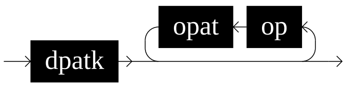
cexp



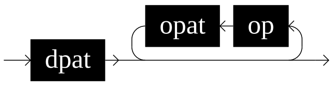
aexp



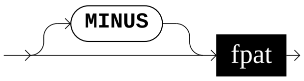
opatk



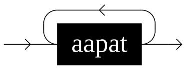
opat



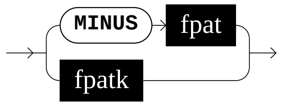
dpat



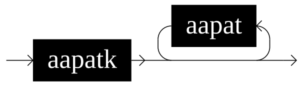
fpat



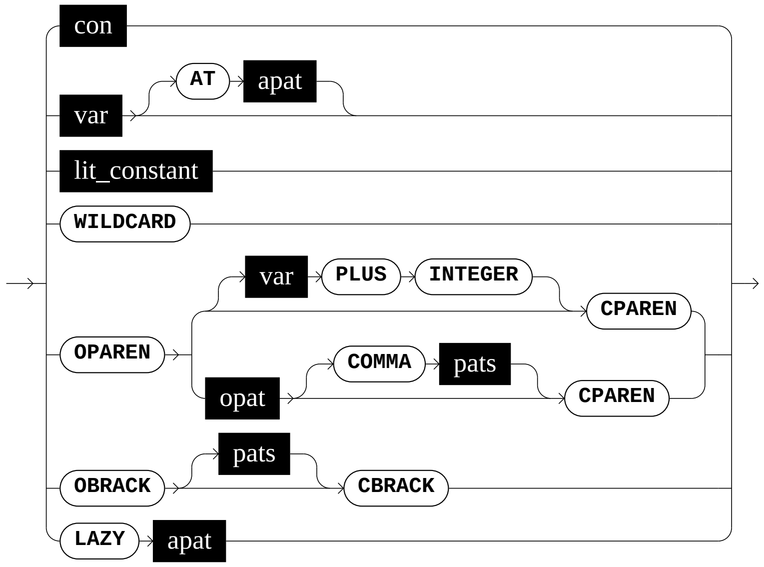
dpatk



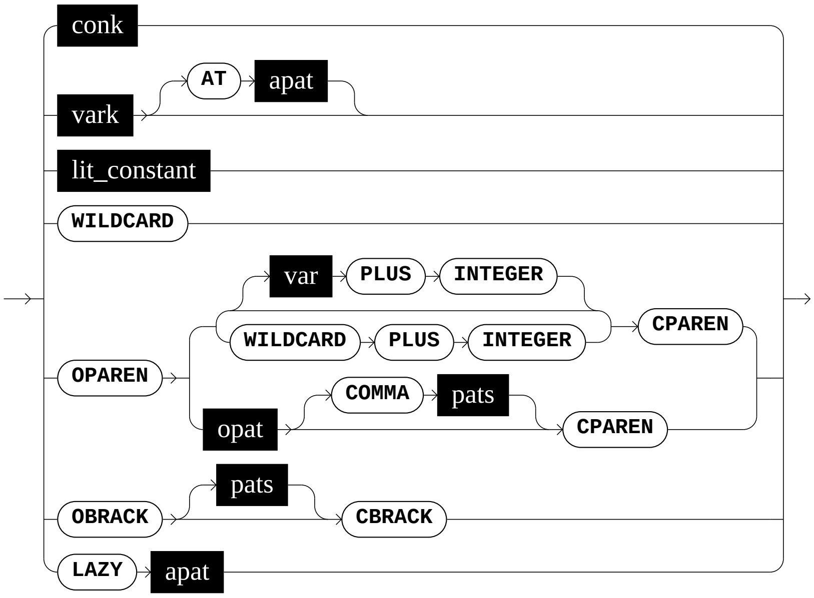
fpatk



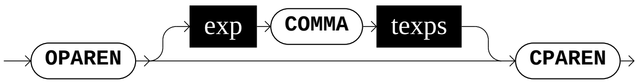
aapat



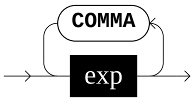
aapatk



tuple

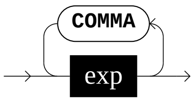


texps

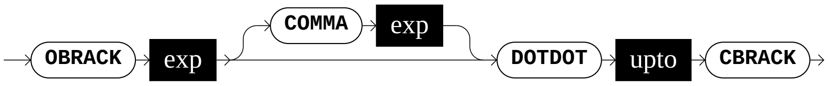


list

list_exps

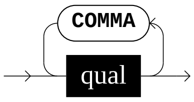


sequence

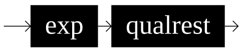


comprehension

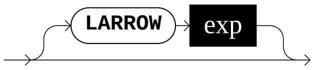
quals



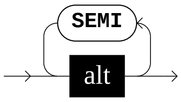
qual



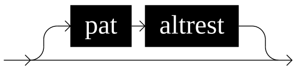
qualrest



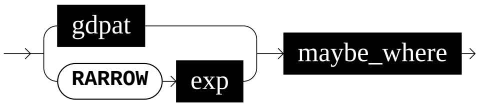
alts



alt

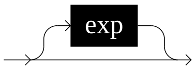


altrest

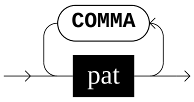


gdpat

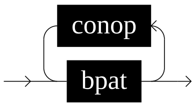
upto



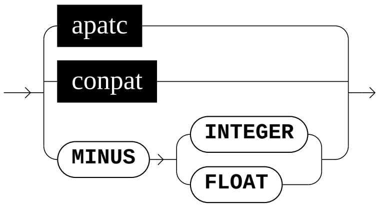
pats



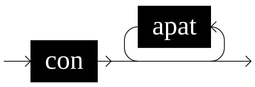
pat



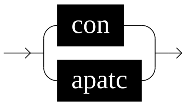
bpat



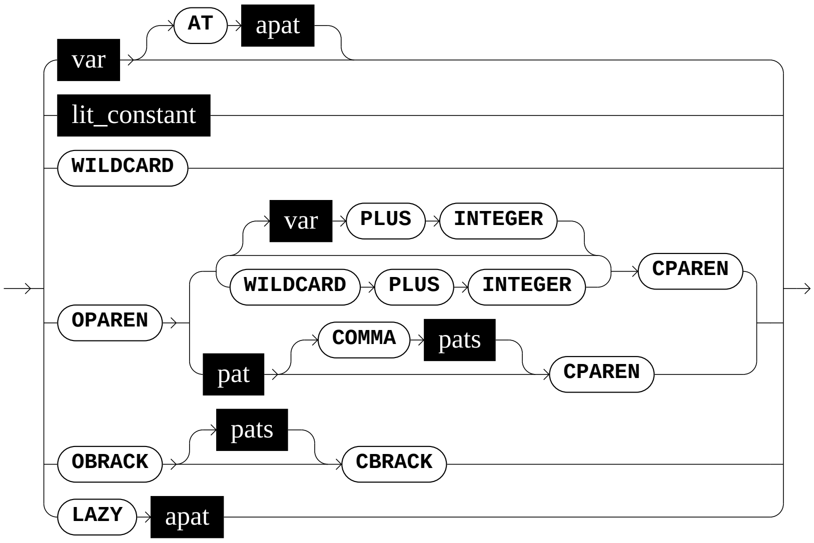
conpat



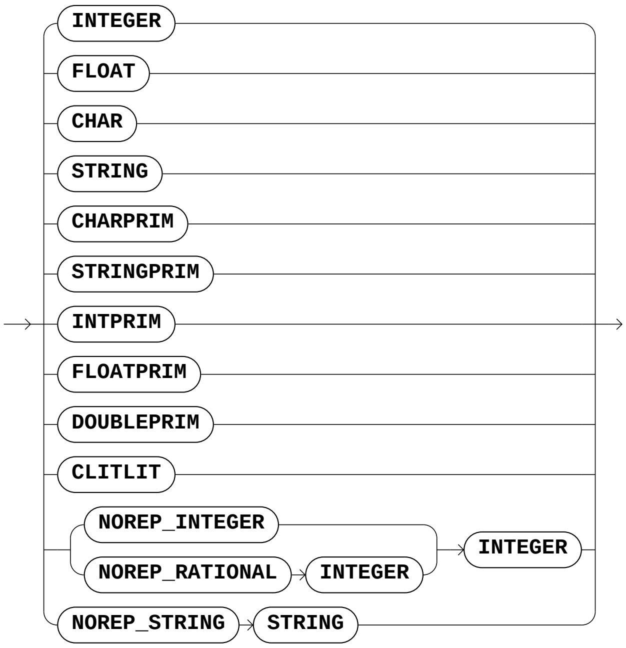
apat



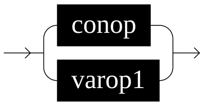
apatc



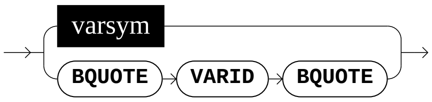
lit_constant



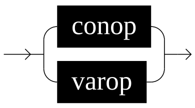
op1



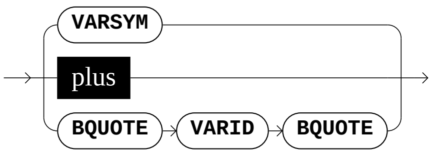
varop



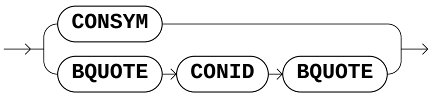
op



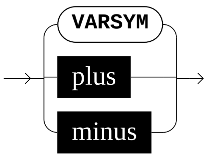
varop1



conop



varsym



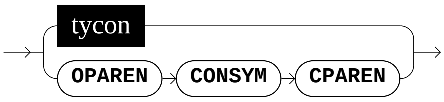
minus

plus

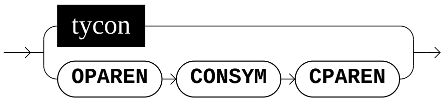
var

vark

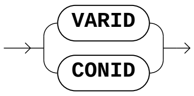
con



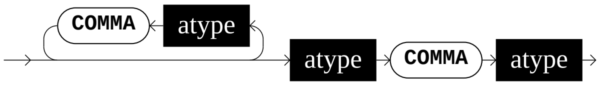
conk



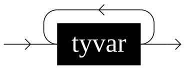
ccallid



atype_list



tyvars



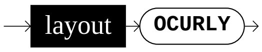
tyvar

tycls

tycon

modid

ocurly



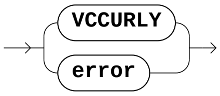
vocurly

layout <empty production>

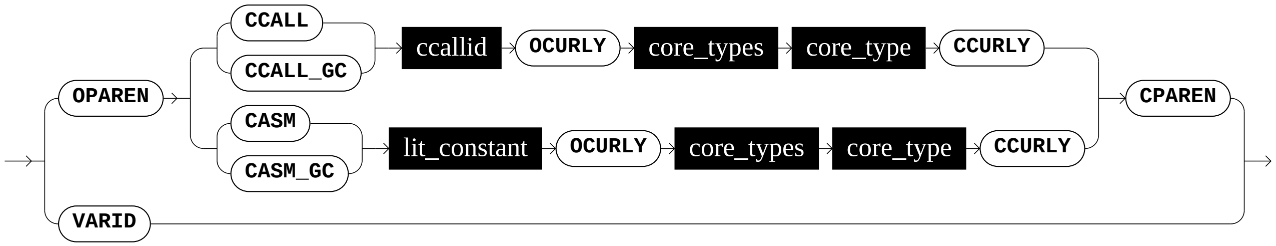
ccurly

vccurly

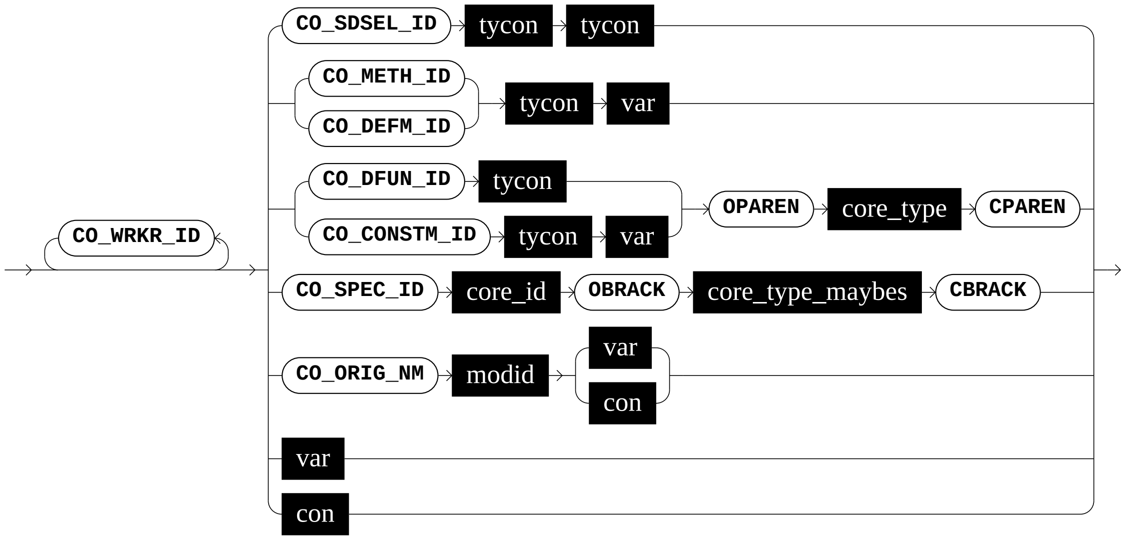
vccurly1



co_primop


OPAREN
CCALL
CCALL_GC
ccallid
OCURLY
core_types
core_type
CCURLY
CASM
CASM_GC
lit_constant
OCURLY
core_types
core_type
CCURLY
CPAREN
VARID

core_id



iinst_pragma