CD23 Programming Language LL(1) Grammar

NPROG	<program> ::= CD23 <id> <global> <funcs> <mainbody>
NGLOB	<globals> ::= <consts> <types> <arrays>
Special	<consts> ::- constants <initlist> | ε
NILIST	<initlist> ::= <init> <opt_initlist>
Special	<opt_initlist> ::= , <initlist> | ε

We applied left factoring to this rule because it wasn't LL(1)

NINIT	<init> ::= <id> is <expr>
Special	<types> ::= types <typelist> | ε
Special	<arrays> ::= arrays <arrdecls> | ε
NFUNCS	<funcs> ::= <func> <funcs> | ε
NMAIN	<mainbody> ::= main <slist> begin <stats> end CD23 <id>
NSDLST	<slist> ::= <sdecl> <opt_slist>
Special	<opt_slist> ::= , <slist> | ε

We applied left factoring to this rule because it wasn't LL(1)

NTYPEL	<typelist> ::= <type> <opt_typelist>
Special	<opt_typelist> ::= <typelist> | ε
NRTYPE	<type> ::= <structed> is <fields> end
NATYPE	<type> ::= <typeid> is array [<expr>] of <structid> end
NFLIST	<fields> ::= <sdecl> <opt_fields>
Special	<opt_fields> ::= , <fields> | ε

We applied left factoring to this rule because it wasn't LL(1)

NSDECL	<sdecl> ::= <id> : <stype>
NALIST	<arrdecls> ::= <arrdecl> <opt_arrdecl>
Special	<opt_arrdecl> ::= , <arrdecls> | ε

We applied left factoring to this rule because it wasn't LL(1)

NARRD	<arrdecl> ::= <id> : <typeid>
NFUND	<func> ::= func <id> (<plist>):<rtype> <funcbody>
Special	<rtype> ::= <stype> | void
Special	<plist> ::= <params> | ε
NPLIST	<params> ::= <param> <opt_params>
Special	<opt_params> ::= , <params> | ε
NSIMP	<param> ::= <sdecl>
NARRP	<param> ::= <arrdecl>

NARRC	<param> ::= const <arrdecl>
Special	<funcbody> ::= <locals> begin <stats> end
Special	<locals> ::= <dlist> | ε
NDLIST	<dlist> ::= <decl> <opt_dlist>
Special	<opt_dlist> ::= , <dlist> | ε
Special	<decl> ::= <sdecl> | <arrdecl>
Special	<stype> ::= integer | real | Boolean
NSTATS	<stats> ::= <stat>; <opt_stats> | <strstat> <opt_stats>
Special	<opt_stats> ::= <stats> | ε
Special	<strstat> ::= <forstat> | <ifstat>
Special	<stat> ::= <reptstat> | <asngnstat> | <iostat>
Special	<stat> ::= <callstat> | <returnstat>
NFORL	<forstat> ::= for (<asgnlist> ; <bool>) <stats> end
NREPT	<repstat> ::= repeat (<asgnlist>) <stats> until <bool>
Special	<asgnlist> ::= <alist> | ε
NASGNS	<alist> ::= <asgnstat> <opt_alist>
Special	<opt_alist> ::= , <alist> | ε
NIFTH	<ifstat> ::= if (<bool>) <stats> <opt_else> end
NIFTE	<opt_else> ::= else <stats> | ε
Special	<asgnstat> ::= <var> <asgnop> <bool>
NASGN	<asgnop> ::= =
NPLEQ	<asgnop> ::= +=
NMNEQ	<asgnop> ::= -=
NSTEQ	<asgnop> ::= *=
NDVEQ	<asgnop> /=
NINPUT	<iostat> ::= In >> <vlist>
NOUTP	<iostat> ::= Out << <prlist> <opt_line>
Special	<opt_line> ::= ε
NOUTL	<opt_line> ::= <<Line
NOUTL	<iostat> ::= Out << Line
NCALL	<callstat> ::= <id> (<elist>) | <id> ()

We applied left factoring to this rule because it wasn't LL(1)

We applied left factoring to this rule because it wasn't LL(1)

We applied left factoring to these rules because they weren't LL(1)

We applied left factoring to these rules because they weren't LL(1).

Note that if <opt_line> is <<Line, the node is NOUTL, not NOUTP

NRETN	<returnstat> ::= return void | return <expr>
NVLIST	<vlist> ::= <var> <opt_vlist>
Special	<opt_vlist> ::= , <vlist> | ε

We applied left factoring to NVLIST because it wasn't LL(1)

NSIMV	<var> ::= <id> <opt_expr>
Special	<opt_expr> ::= ε
NAELT	<opt_expr> ::= [<expr>] <opt_id>
Special	<opt_id> ::= ε
NARRV	<opt_id> ::= . <id>

We applied left factoring to these rules because they weren't LL(1).

Note that if <opt_line> is <<Line, the node is NOUTL, not NOUTP

NEXPL	<elist> ::= <bool> <opt_elist>
Special	<opt_elist> ::= , <elist> | ε

We applied left factoring to NEXPL because it wasn't LL(1)

NBOOL	<bool> ::= <logop> <rel> <bool'> | ε
Special	<bool'> ::= <rel> <bool'>

We changed NBOOL because it was left recursive

NNOT	<rel> ::= ! <expr> <relop> <expr>
Special	<rel> ::= <expr> <opt_rel>
Special	<opt_rel> ::= <relop> <expr> | ε

We applied left factoring to this rule because it wasn't LL(1)

NAND	<logop> ::= &&
NOR	<logop> ::= ||
NXOR	<logop> ::= &|
NEQL	<relop> ::= ==
NNEQ	<relop> ::= !=
NGRT	<relop> ::= >
NLSS	<relop> ::= <
NLEQ	<relop> ::= <=
NGEQ	<relop> ::= >=
Special	<expr> ::= <term> <expr'>
NADD	<expr'> ::= +<term> <expr'>
NSUB	<expr'>::= - <term> <expr'>
Special	<expr'> ::= ε

We changed these rules because they were left recursive

Special	<term> ::= <fact> <term'>
NMUL	<term'> ::= *<fact> <term'>
NDIV	<term'> ::= /<fact> <term'>

We changed these rules because they were left recursive

NMOD	<term’> ::= % <fact> <term’>
Special	<term’> ::= ε
Special	<fact> ::= <exponent> <fact’>
NPOW	<fact’> ::= ^ <exponent> <fact’> | ε
Special	<exponent> ::= <var>
NILIT	<exponent> ::= <intlit>
NFLIT	<exponent> ::= <reallit>
Special	<exponent> ::= <fncall>
NTRUE	<exponent> ::= true
NFALS	<exponent> ::= false
Special	<exponent> ::= (<bool>)
NFCALL	<fncall> ::= <id> (<elist>) | <id> ()
NPRLST	<prlist> ::= <printitem> <opt_prlist>
Special	<opt_prlist> ::= , <prlist> | ε
Special	<printitem> ::= <expr>
NSTRG	<printitem> ::= <string>

We changed this rule because it was left recursive

We applied left factoring to this rule because it wasn’t LL(1)