```
header "pre_include_hpp" {
#include <antlr/TokenStreamSelector.hpp>
#include "utils.h"
}

header "pre_include_cpp" {
#pragma warning( disable : 4251 4267 4101 4267 )
}

options {
        language="Cpp";
        namespace="ff";
        namespaceStd="std";
        namespaceAntlr="antlr";
}

/** FjolnirForritLexer er lesgreinir sem vinnur á
 *  Fjölnis-forritum utan eininganna sjálfra. Hann
 *  lesgreinir einingaraðgerðir en um leið og hann rekst
 *  á einingu, sem afmarkast af opnum slaufusviga ('{')
 *  ýtir hann einingalesgreini (FjolnirEiningLexer) á stafla
 *  TokenStreamSelector eintaksins.
 */
class FjolnirForritLexer extends Lexer;
options {
        k = 2;
        charVocabulary = '\3'..'\377';
        exportVocab = FjolnirForrit;
}

{
protected:
        antlr::TokenStreamSelector* selector;

public:
        void initialize(antlr::TokenStreamSelector * sel) {
                selector = sel;
        }
}

EIN_SEMIKOMMA
        : ';'
        ;

EIN_JAFNTOG
        : '='
        ;

EIN_MINNA
        : '<'
        ;

EIN_SVIGIOPNA
        : '('
        ;

EIN_SVIGILOKA
        : ')'
        ;

/* Einingaaðgerðir */

EIN_ITRUN               :       '!' ;
EIN_INNFLUTT    :       '*' ;
EIN_SAMSETNING  :       ':' ;
EIN_HLIDSETNING :       '+' ;
EIN_ITRUDHLIDS  :       '&' ;

EIN_OPNASLAUFU
        : '{' { selector->push("eininglexer"); }
        ;

/* Nöfn og strengir */
EIN_STRENGUR
        : '"'! ( ~('"'|'\\') | StyriStafur[false] )* '"'!
        ;
```



```
protected
StyriStafur [bool expand=true]
        : ( '\\' ~'$' | "\\$" HexStafur HexStafur )
                {
                        std::string temp;
                        if (expand) {
                                temp = $getText;
                                $setText(styriStafur(temp));
                        }
                }
        ;

protected
HexStafur
        : '0'..'9' | 'A'..'F' | 'a'..'f'
        ;

protected
NafnStafur
        : ~( ' ' | '(' | ')' | '[' | ']' | '{' | '}' | ',' | ';' | '.'
           | '+' | '-' | '*' | '/' | '%' | '!' | '&' | '=' | '?' | '<' | '>' | '|' | ':'
| '@'
           | '\3'..'\31' | '0'..'9' | '"' )
        ;

EIN_NAFN
        : NafnStafur (NafnStafur /* skv. handbók eru tölust. nafnst. | '0'..'9' */ )*
        ;


/* Whitespace og comment */

WS
        : ('\t' | ' ' | EndOfLine {newline();} )+ {$setType(antlr::Token::SKIP);};

COMMENT
        : ";;" (~('\n' | '\r'))* EndOfLine {$setType(antlr::Token::SKIP); newline(); };

/* Hjálparreglur */

protected
EndOfLine
        : (options{generateAmbigWarnings = false;}:
                "\r\n"
        |       '\r'
        |       '\n'
        );
```

```
header "post_include_hpp" {
#include <antlr/TokenStreamSelector.hpp>
#include "utils.h"
}

header "pre_include_cpp" {
#pragma warning( disable : 4251 4267 4101 4267 )
}

options {
        language="Cpp";
        namespace="ff";
        namespaceStd="std";
        namespaceAntlr="antlr";
        genHashLines = true;
}

/** FjolnirEiningLexer er lesgreinir sem vinnur á einingaskilgreiningum
 *  í Fjölnis-forritum, þ.e. þess kóða sem er innan slaufusviga.
 *  Þegar hann sér lokun á óopnuðum slaufusviga (’}’) poppar hann af
 *  "input-selector" hlaðanum og með því færist stjórn lesgreiningarinnar
 *  aftur yfir til FjolnirForritLexer, sem lesgreinir tákn utan
 *  eininganna (s.s. einingaraðgerðir).
 */
class FjolnirEiningLexer extends Lexer;

options {
        k = 7; /* lengst er ANNARS,ANNARSEF */
        charVocabulary = ’\3’..’\377’;
        importVocab = FjolnirForrit;
        exportVocab = Fjolnir;
}

tokens {
        EIN_LOKASLAUFU;
        ADGERD_7; ADGERD_6; ADGERD_5; ADGERD_4; ADGERD_3; ADGERD_2; ADGERD_1;
}

{
protected:
        int nestedSlaufur;
        antlr::TokenStreamSelector* selector;

public:
        void initialize(antlr::TokenStreamSelector * sel) {
                selector = sel;
                nestedSlaufur = 0;
        }
}

/* Lykilorð. ANTLR sér sjálfkrafa um að strengfastar í lesgreins-
 * og mállýsingunni fari í sérstaka töflu og eru strengir úr henni
 * ávallt auðkenndir sem lykilorð frekar en nöfn.
 */
L_ANNARS        :       "annars"        ;
L_ANNARSEF      :       "annarsef"      ;
L_BREYTA        :       "breyta"        ;
L_EDA           :       "eða"           ;
L_EFLOK         :       "eflok"         ;
L_EF            :       "ef"            ;
L_EKKI          :       "ekki"          ;
L_FYRIR         :       "fyrir"         ;
L_INNFLUTT      :       "innflutt"      ;
L_KOSTUR        :       "kostur"        ;
L_LYKKJA        :       "lykkja"        ;
L_LYKKJULOK     :       "lykkjulok"     ;
L_MEDAN         :       "meðan"         ;
L_OG            :       "og"            ;
L_SKILA         :       "skila"         ;
L_STADVAER      :       "staðvær"       ;
L_STEF          :       "stef"          ;
L_STOFN         :       "stofn"         ;
L_STOFNLOK      :       "stofnlok"      ;
L_UR            :       "úr"            ;
L_UT            :       "út"            ;
L_VAL           :       "val"           ;
L_VALLOK        :       "vallok"        ;
L_THA           :       "þá"            ;
```



```
/* Sleppum Fjölnir 2 constructum í bili
L_HLUTUR        :       "hlutur"        ;
L_HLUTLOK       :       "hlutlok"       ;
L_ARFUR         :       "arfur"         ;
L_BOD           :       "boð"           ;
L_THESSI        :       "þessi"         ;
*/

/* Virkjar og frátekin tákn */

protected
INNSETNING              : "->"  ;

protected
GILDISVEITING   : ":="  ;

SVIGI_OPNA              : ’(’   ;
SVIGI_LOKA              : ’)’   ;
SEMIKOMMA               : ’;’   ;
KOMMA                   : ’,’   ;
HORNKLOFI_OPNA  : ’[’   ;
HORNKLOFI_LOKA  : ’]’   ;
PUNKTUR                 : ’.’   ;
PUNKTURPUNKTUR  : ".."  ;

/* Lesfastar */

STAFFASTI
        : ’\’’! ( ~’\\’ | StyriStafur ) ’\’’!
        ;

STRENGFASTI
        : ’"’! ( ~(’"’|’\\’) | StyriStafur[false] )* ’"’!
        ;

protected
StyriStafur [bool expand=true]
        : ( ’\\’ ~’$’ | "\\$" HexStafur HexStafur )
                {
                        std::string temp;
                        if (expand) {
                                temp = $getText;
                                $setText(styriStafur(temp));
                        }
                }
        ;

protected
HexStafur
        : ’0’..’9’ | ’A’..’F’ | ’a’..’f’
        ;

/* Þessi regla notar "Syntactic predicate" í ANTLR til að skoða
 * fram í strauminn og taka ákvörðun um hvort við lesgreinum heiltölu,
 * fleytitölu, "->", ":=", aðgerðarnafn eða fjöldatölu
 *
 * Þetta gerir okkur kleyft að lesgreina rétt fyrirbæri eins og "---",
 * "->>", ":=+-++" o.s.frv.
 */
INNSETN_GILDISV_ADGERD_EDA_TALA
        { int ft; }
        : ( (’-’)* (’0’..’9’)+ ’.’ )=> FLEYTITALA {$setType(FLEYTITALA); }
        | ( (’-’)* (’0’..’9’ | ’$’))=> HEILTALA {$setType(HEILTALA);}
        | ( "->" EkkiAdgerdaStafur)=> INNSETNING {$setType(INNSETNING);}
        | ( ":=" EkkiAdgerdaStafur)=> GILDISVEITING {$setType(GILDISVEITING);}
        | ft=ADGERD {$setType(ft);}
        | FJOLDATALA {$setType(FJOLDATALA);}
        ;

protected
FJOLDATALA
        : ’$’ (HexStafur)+
        | (’0’..’9’)+
        ;

protected
HEILTALA
```

```
        : ('-')* FJOLDATALA
        ;

protected
FLEYTITALA
        : ('-')* ('0'..'9')+ '.' ('0'..'9')* (('e'|'E') ('+'|'-')? ('0'..'9')+)?
        ;

TOMAGILDI
        : "[]"
        ;


/* Nöfn og aðgerðanöfn */

protected
AdgerdaStafur
        : ( '+' | '-' | '*' | '/' | '%' | '!' | '&' | '='
          | '?' | '<' | '>' | '|' | ':' | '^' | '@' )
        ;

/* Hjálparregla f. predicates */
protected
EkkiAdgerdaStafur
        : ~( '+' | '-' | '*' | '/' | '%' | '!' | '&' | '='
          | '?' | '<' | '>' | '|' | ':' | '^' | '@' )
        ;

protected
ADGERD returns [int forgangsToken]
        : (AdgerdaStafur)+
                {
                        switch (text[_begin]) {
                        case '^': forgangsToken = ADGERD_7; break;
                        case '!':
                        case '?':
                        case '@': forgangsToken = ADGERD_6; break;
                        case '*':
                        case '/':
                        case '%':
                        case '&': forgangsToken = ADGERD_5; break;
                        case '+':
                        case '-':
                        case '|': forgangsToken = ADGERD_4; break;
                        case '<':
                        case '>':
                        case '=': forgangsToken = ADGERD_3; break;
                        case ':': forgangsToken = ADGERD_2; break; /* vinstri tengin aðgerð */
                        default:  forgangsToken = ADGERD_1; break;
                        }
                }
        | '\\'! ( NafnStafur | AdgerdaStafur )+ { forgangsToken = ADGERD_1; }
        ;

protected
NafnStafur
        : ~( ' ' | '(' | ')' | '[' | ']' | '{' | '}' | ',' | ';' | '.'
           | '+' | '-' | '*' | '/' | '%' | '!' | '&' | '=' | '?' | '<' | '>' | '|' | ':' | '^'
| '@'
           | '\'' /* viðbót arnar */
           | '\3'..'\31' | '0'..'9' | '"' )
        ;

protected
EkkiNafnStafur
        : ( ' ' | '(' | ')' | '[' | ']' | '{' | '}' | ',' | ';' | '.'
           | '+' | '-' | '*' | '/' | '%' | '!' | '&' | '=' | '?' | '<' | '>' | '|' | ':' | '^'
| '@'
           | '\'' /* viðbót arnar */
           | '\3'..'\31' | '0'..'9' | '"' )
        ;

NAFN
        : NafnStafur (NafnStafur | '0'..'9' )*
        ;

/* Lok einingar og innri einingar */
```



```
SLAUFA_OPNA
        : '{' { nestedSlaufur++; }
        ;

SLAUFA_LOKA
        : '}' {
                if (0 == nestedSlaufur) {
                        selector->pop();
                        $setType(EIN_LOKASLAUFU);
                } else {
                        nestedSlaufur--;
                }
        };

/*EIN_LOKASLAUFU
        : '}' { selector->pop(); }
        ;*/


/* Whitespace og comment */

WS
        : ('\t' | ' ' | EndOfLine {newline();} )+ {$setType(antlr::Token::SKIP);};

COMMENT
        : ";;" (~('\n' | '\r'))* EndOfLine {$setType(antlr::Token::SKIP); newline(); };

/* Hjálparreglur */

protected
EndOfLine
        : (options{generateAmbigWarnings = false;}:
                "\r\n"
        |       '\r'
        |       '\n'
        );
```

```
header "pre_include_cpp" {
#pragma warning( disable : 4251 4267 4101 4267 )
}

options {
        language="Cpp";
        namespace="ff";
        namespaceStd="std";
        namespaceAntlr="antlr";
        genHashLines = true;
}

class FjolnirParser extends Parser;

options {
        k = 3;
        importVocab = Fjolnir;
        buildAST = true;
}
tokens {
        NAFNARUNA; SKILGREININGAR; SEGDARUNA; LYKKJUSKILYRDI;
        EINUNDARADGERD; LISTI; FYLKISGILDISVEITING;
}

forrit
        : ( (forritsVeiting | einingarVeiting) EIN_SEMIKOMMA! )* EOF
        ;

forritsVeiting
        : EIN_STRENGUR EIN_MINNA^ EIN_NAFN eining
        ;

einingarVeiting
        : (EIN_STRENGUR | EIN_NAFN) EIN_JAFNTOG^ eining
        ;

eining
        : einHlidsett
        ;

einItrunarhlidsett
        : einHlidsett (EIN_ITRUDHLIDS^ einHlidsett)*
        ;

einHlidsett
        : einSamsett (EIN_HLIDSETNING^ einSamsett)*
        ;

einSamsett
        : einInnflutt (EIN_SAMSETNING^ einInnflutt)*
        ;

einInnflutt
        : einItrud (EIN_INNFLUTT^ einItrud)*
        ;

einItrud
        : EIN_ITRUN^ einItrud | grunnEining
        ;

grunnEining
        : EIN_STRENGUR
        | EIN_NAFN
        | EIN_SVIGIOPNA! eining EIN_SVIGILOKA!
        | EIN_OPNASLAUFU^ (vorpun)* EIN_LOKASLAUFU!
        ;

protected
adgerd
        : (ADGERD_1 | ADGERD_2 | ADGERD_3 | ADGERD_4 | ADGERD_5 | ADGERD_6 | ADGERD_7)
              { #adgerd->setType(ADGERD); }
        ;

vorpun
        : (NAFN | adgerd) INNSETNING^
              ( NAFN | adgerd | L_BREYTA /*| L_BOD*/ | stefskilgreining /*| hlutskilgreining*
/ )
        ;
```



```
stefskilgreining
        : L_STEF^ SVIGI_OPNA! nafnaruna SEMIKOMMA! nafnaruna SVIGI_LOKA!
              skilgreiningar
              L_STOFN! segdaruna L_STOFNLOK!
              (innriTextaeining)?
        ;

nafnaruna
        : (NAFN (KOMMA! NAFN)*)?
              {#nafnaruna = #([NAFNARUNA,"nafnaruna"], #nafnaruna); }
        ;

skilgreiningar
        : (skilgr_innflutt | skilgr_stadvaer)*
              { #skilgreiningar = #([SKILGREININGAR,"skilgreiningar"], #skilgreiningar)
        ;

skilgr_innflutt
        : L_INNFLUTT^ nafnaruna_ekkitom
        ;

skilgr_stadvaer
        : L_STADVAER^ frumstillingaruna
        ;

nafnaruna_ekkitom
        : NAFN (KOMMA! NAFN)*
        ;

frumstillingaruna
        : (NAFN | skilgrOgGildisv) (KOMMA! (NAFN | skilgrOgGildisv))*
        ;

skilgrOgGildisv
        : NAFN GILDISVEITING^ segd
        ;

innriTextaeining
        : SLAUFA_OPNA^ (innraStef)* SLAUFA_LOKA!
        ;

innraStef
        : NAFN INNSETNING^ stefskilgreining
        ;

/*
hlutskilgreining
        : L_HLUTUR^ SVIGI_OPNA! SEMIKOMMA! nafnaruna SVIGI_LOKA!
              arfskilgreining
              (bodskilgreining)*
              L_HLUTLOK!
        ;

arfskilgreining
        : (L_ARFUR! NAFN)? {#arfskilgreining = #([L_ARFUR,"arfur"], #arfskilgreining); }
        ;

bodskilgreining
        : L_BOD^ NAFN SVIGI_OPNA! nafnaruna SEMIKOMMA! nafnaruna SVIGI_LOKA!
              skilgreiningar
              L_STOFN! segdaruna L_STOFNLOK!
              (innriTextaeining)?
        ;
*/

segdaruna
        : (segd (KOMMA! segd)*)?
              {#segdaruna = #([SEGDARUNA,"segðaruna"], #segdaruna); }
        ;

segd
        : ogSegd ( options {greedy=true;}: L_EDA^ ogSegd)*
        ;

ogSegd
        : ekkiSegd ( options {greedy=true;}: L_OG^ ekkiSegd)*
        ;
```

```
    ekkiSegd
            : L_EKKI^ ekkiSegd | adgerd_1
            ;

    /* virkjar og forgangur þeirra */

    adgerd_1
            : adgerd_2 ( options {greedy=true;}: ADGERD_1^ adgerd_2 { ##->setType(ADGERD); } )*
            ;

    adgerd_2
            /* vinstri tengið */
            : adgerd_3 ( options {greedy=true;}: ADGERD_2^ adgerd_2 { ##->setType(ADGERD); } )?
            ;

    adgerd_3
            : adgerd_4 ( options {greedy=true;}: ADGERD_3^ adgerd_4 { ##->setType(ADGERD); } )*
            ;

    adgerd_4
            : adgerd_5 ( options {greedy=true;}: ADGERD_4^ adgerd_5 { ##->setType(ADGERD); } )*
            ;

    adgerd_5
            : adgerd_6 ( options {greedy=true;}: ADGERD_5^ adgerd_6 { ##->setType(ADGERD); } )*
            ;

    adgerd_6
            : adgerd_7 ( options {greedy=true;}: ADGERD_6 adgerd_7 { ##->setType(ADGERD); } )*
            ;

    adgerd_7
            : hlutfylkissegd ( options {greedy=true;}: ADGERD_7^ hlutfylkissegd { ##->setType(ADGER
    D); } )*
            ;

    hlutfylkissegd
            : (smasegd HORNKLOFI_OPNA)=> smasegd
                    ( options {greedy=true;}: HORNKLOFI_OPNA^ segdaruna HORNKLOFI_LOKA! )+
                    ( GILDISVEITING^ segd {#GILDISVEITING->setType(FYLKISGILDISVEITING); } )?
            | smasegd
            ;

    /* Fyrir Fjölnir 2 samhæfingu:
    hlutfylkissegd
            : smasegd ( (HORNKLOFI_OPNA | PUNKTUR)=>
                    ( options {greedy=true;}:
                      HORNKLOFI_OPNA^ segdaruna HORNKLOFI_LOKA!
                    | PUNKTUR^ NAFN
                            (SVIGI_OPNA^ segdaruna SEMIKOKMMA! segdaruna SVIGI_LOKA! )?
                    )+
                    ( GILDISVEITING^ segd )?
            )?
            ;
    */

    smasegd
            : nafnsegd
            | (adgerd SVIGI_OPNA)=> adgerd SVIGI_OPNA^ SEMIKOMMA! segdaruna SVIGI_LOKA!
            | einundaradgerd
            | efsegd
            | HORNKLOFI_OPNA^ segdaruna HORNKLOFI_LOKA! { ##->setType(LISTI); }
            | lykkjusegd
            | L_STOFN^ segdaruna L_STOFNLOK!
            | valsegd
            | L_UT
            | L_SKILA^ segd
            | SVIGI_OPNA! segd SVIGI_LOKA!
            | STRENGFASTI
            | STAFFASTI | FJOLDATALA | HEILTALA | FLEYTITALA | TOMAGILDI
            | L_STEF^ NAFN SVIGI_OPNA! FJOLDATALA SEMIKOMMA! FJOLDATALA SVIGI_LOKA!
    /*      | L_THESSI */
    /*      | L_ARFUR */
            ;

    einundaradgerd
            :
```



```
            ( ADGERD_1^ | ADGERD_2^ | ADGERD_3^ | ADGERD_4^ | ADGERD_5^ | ADGERD_6^ | ADGERD_
                    smasegd { #einundaradgerd->setType(EINUNDARADGERD); }
            ;

    nafnsegd
            : NAFN
            | NAFN GILDISVEITING^ segd
            | NAFN SVIGI_OPNA^ nafnaruna SEMIKOMMA! segdaruna SVIGI_LOKA!
            ;

    lykkjusegd
            : L_LYKKJA^ segdaruna L_LYKKJULOK!
            | L_MEDAN^ segd L_LYKKJA! segdaruna L_LYKKJULOK!
            | L_FYRIR^ SVIGI_OPNA! segdaruna SEMIKOMMA! segd SEMIKOMMA! segdaruna SVIGI_LOKA!
                    L_LYKKJA! segdaruna L_LYKKJULOK!
            ;

    efsegd
            : L_EF^ segd L_THA! segdaruna (annarsef)* (L_ANNARS! segdaruna)? L_EFLOK!
            ;

    annarsef
            : L_ANNARSEF^ segd L_THA! segdaruna
            ;

    valsegd
            : L_VAL^ segd L_UR! (valkostur)* (L_ANNARS! segdaruna)? L_VALLOK!
            ;

    valkostur
            : L_KOSTUR^ valfasti_range L_THA! segdaruna
            ;

    valfasti_range
            : valfasti (PUNKTURPUNKTUR^ valfasti)?
            ;

    /* má ekki vera fjöldatala hér líka? */
    valfasti
            : STAFFASTI | HEILTALA
            ;
```

```
header "pre_include_cpp" {
#pragma warning( disable : 4251 4267 4101 4267 )
}

options {
        language="Cpp";
        namespace="ff";
        namespaceStd="std";
        namespaceAntlr="antlr";
        genHashLines = true;
}

class FjolnirTransformer extends TreeParser;

options {
        k = 3;
        importVocab = FjolnirParser;
        buildAST = true;
}

forrit
        : ( veiting )*
        ;

veiting
        : #(EIN_MINNA EIN_STRENGUR EIN_NAFN eining)
        | #(EIN_JAFNTOG (EIN_STRENGUR | EIN_NAFN) eining)
        ;

eining
        : #(EIN_ITRUD eining)
        | #(EIN_ITRUDHLIDS eining eining)
        | #(EIN_HLIDSETNING eining eining)
        | #(EIN_SAMSETNING eining eining)
        | #(EIN_INNFLUTT eining eining)
        | EIN_STRENGUR
        | EIN_NAFN
        | #(EIN_OPNASLAUFU (vorpun)*)
        ;

vorpun
        : #(INNSETNING (NAFN | ADGERD) minnissvaedi)
        ;

minnissvaedi
        : NAFN | ADGERD | L_BREYTA | L_BOD | stefskilgreining | hlutskilgreining
        ;

stefskilgreining
        : #(L_STEF nafnaruna nafnaruna skilgreiningar segdaruna (innriTextaeining)?)
        ;

nafnaruna
        : #(NAFNARUNA (NAFN)*)
        ;

nafnaruna_ekkitom
        : (NAFN)+
        ;

/* Eftirfarandi regla sameinar mörg tilvik af "staðvær" og "innflutt"
   breytuyfirlýsingarrunum í eitt tilvik af hvorri */
skilgreiningar!
        {
                antlr::RefAST innflutt = #([L_INNFLUTT,"innflutt"]);
                antlr::RefAST stadvaerar = #([L_STADVAER,"staðvær"]);
        }
        : #(dummy:SKILGREININGAR
                (       #(L_INNFLUTT nr:nafnaruna_ekkitom)
                        { innflutt->addChild(#nr); }
                |       #(L_STADVAER fr:frumstillingaruna)
                        { stadvaerar->addChild(#fr); }
                )*
                {
                        ## = #([SKILGREININGAR,"skilgreiningar"]);
                        ##->setFirstChild(innflutt);
                        innflutt->setNextSibling(stadvaerar);
                        stadvaerar->setNextSibling(antlr::nullAST);
                }
        )
        ;

frumstillingaruna
        : ( NAFN | #(GILDISVEITING NAFN segd) )+
        ;

innriTextaeining
        : #(SLAUFA_OPNA (innraStef)* )
        ;

innraStef
        : #(INNSETNING NAFN stefskilgreining)
        ;

hlutskilgreining
        : #(L_HLUTUR nafnaruna #(L_ARFUR (NAFN)?) (bodskilgreining)* )
        ;

bodskilgreining
        : #(L_BOD NAFN nafnaruna nafnaruna skilgreiningar segdaruna (innriTextaeining)?)
        ;

segdaruna
        : #(SEGDARUNA (segd)*)
        ;

segd
        : #(L_OG segd segd)
        | #(L_EDA segd segd)
        | #(L_EKKI segd)
        |! #(ADGERD s1:segd s2:segd)
                {
                        #ADGERD->setType(NAFN);
                        ## = #([SVIGI_OPNA,"("], ADGERD, #([NAFNARUNA,"inn-út"]), #([SEGD
"aðgerðarvið"], s1, s2));
                }
        | hlutfylkissegd
        | #(GILDISVEITING NAFN segd)
        |! #(EINUNDARADGERD s:segd)
                {       #EINUNDARADGERD->setType(NAFN);
                        ## = #([SVIGI_OPNA,"("], EINUNDARADGERD, #([NAFNARUNA,"inn-út"]),
GDARUNA,"einundaraðgerðarvið"],s));
                }
        |! (#(SVIGI_OPNA ADGERD))=> #(foo:SVIGI_OPNA ADGERD sr:segdaruna)
                {       #ADGERD->setType(NAFN);
                        ## = #([SVIGI_OPNA,"("], ADGERD, #([NAFNARUNA,"inn-út"]), sr);
                }
        | #(SVIGI_OPNA NAFN nafnaruna segdaruna)
        |! #(LISTI slist:segdaruna)
        {
            /* breytir [a,b,c] í :(;a,:(;b,c)) */
            antlr::RefAST current = #slist->getFirstChild();
                        if (current == antlr::nullAST) {
                                ## = #([TOMAGILDI,"[]"]);
                        } else {
                                antlr::RefAST subtree = #([SEGDARUNA,"gildisvið"]);
                                antlr::RefAST root = #([SVIGI_OPNA,"("], [NAFN,":"], #([N
NA,"inn-út"]), subtree);
                                while (current != antlr::nullAST) {
                                        subtree->addChild(current);
                                        antlr::RefAST temp1 = current->getNextSibling();
                                        current->setNextSibling(antlr::nullAST);
                                        current = temp1;
                                        if (current != antlr::nullAST) {
                                                antlr::RefAST newSubtree = #([SEGDARUNA,"
viðf"]);
                                                antlr::RefAST temp2 = #([SVIGI_OPNA,"("],
,":"], #([NAFNARUNA,"inn-út"]), newSubtree);
                                                subtree->addChild(temp2);
                                                subtree = newSubtree;
                                        }
                                }
                                subtree->addChild( #([TOMAGILDI,"[]"]) );
                                ## = root;
                        }
        }
```

```
        | #(L_STOFN segdaruna)
        | #(L_EF segd segdaruna (#(L_ANNARSEF segd segdaruna))* (efannars:segdaruna)?)
                { if (antlr::nullAST == efannars) ##->addChild(#([SEGDARUNA,"segðaruna"], [TOMA
GILDI,"[]"])); }
        | lykkjusegd
        | #(L_VAL segd (#(L_KOSTUR valfasti_range segdaruna))* (valannars:segdaruna)?)
                { if (antlr::nullAST == valannars) ##->addChild(#([SEGDARUNA,"segðaruna"], [TOM
AGILDI,"[]"])); }
        | #(L_SKILA segd)
        | L_UT
        | NAFN
        | STRENGFASTI | STAFFASTI | FJOLDATALA | HEILTALA | FLEYTITALA | TOMAGILDI
        | #(L_STEF NAFN FJOLDATALA FJOLDATALA)
/*      | L_THESSI */
/*      | L_ARFUR  */
        ;

hlutfylkissegd!
        : #(HORNKLOFI_OPNA s:segd r:segdaruna)
                {
                        antlr::RefAST sr = #([SEGDARUNA], #s);
                        int i = 0;
                        antlr::RefAST hali = sr->getFirstChild();
                        antlr::RefAST c = #r->getFirstChild();
                        while (antlr::nullAST != c) {
                                hali->setNextSibling(c);
                                i++;
                                hali = c;
                                c = c->getNextSibling();
                        }
                        char fallnafn[64];
                        ::_snprintf(fallnafn, 64, "fylkisækja%d", i);
                        ## = #([SVIGI_OPNA], [NAFN,fallnafn], #([NAFNARUNA]), sr);
                }
        | #(FYLKISGILDISVEITING #(HORNKLOFI_OPNA ss:segd rr:segdaruna) gildi:segd)
                {
                        antlr::RefAST sr = #([SEGDARUNA], #ss);
                        int i = 0;
                        antlr::RefAST hali = sr->getFirstChild();
                        antlr::RefAST c = #rr->getFirstChild();
                        while (antlr::nullAST != c) {
                                hali->setNextSibling(c);
                                i++;
                                hali = c;
                                c = c->getNextSibling();
                        }
                        hali->setNextSibling(gildi);
                        char fallnafn[64];
                        ::_snprintf(fallnafn, 64, "fylkisetja%d", i);
                        ## = #([SVIGI_OPNA], [NAFN,fallnafn], #([NAFNARUNA]), sr);
                }
/*      | #(PUNKTUR segd NAFN (#(SVIGI_OPNA segdaruna segdaruna))? ) */
        ;

lykkjusegd
        : #(L_LYKKJA segdaruna)
        | #(L_MEDAN segd segdaruna)
        | #(L_FYRIR segdaruna segd segdaruna segdaruna)
                {
                        /* breytum "fyrir" lykkju í "meðan" lykkju */
                        ##->setType(L_MEDAN);
                        antlr::RefAST ini = ##->getFirstChild();
                        antlr::RefAST tst = ini->getNextSibling();
                        antlr::RefAST incr = tst->getNextSibling();
                        antlr::RefAST body = incr->getNextSibling();
                        ##->setFirstChild(tst);
                        antlr::RefAST lastBody = body->getFirstChild();
                        if (antlr::nullAST != lastBody) {
                                antlr::RefAST incr1 = incr->getFirstChild();
                                incr->setFirstChild(lastBody);
                                while (antlr::nullAST != lastBody->getNextSibling())
                                        lastBody = lastBody->getNextSibling();
                                lastBody->setNextSibling(incr1);
                        }
                        incr->setNextSibling(antlr::nullAST);

                        antlr::RefAST lastInit = ini->getFirstChild();
                        if (antlr::nullAST != lastInit) {
```



```
                                while (antlr::nullAST != lastInit->getNextSibling())
                                        lastInit = lastInit->getNextSibling();
                                lastInit->setNextSibling(##);
                                ## = lastInit;
                        }
                }
        ;

lykkjuskilyrdi
        : (#(LYKKJUSKILYRDI L_MEDAN))=> #(LYKKJUSKILYRDI L_MEDAN segd)
        | (#(LYKKJUSKILYRDI L_FYRIR))=> #(LYKKJUSKILYRDI L_FYRIR segdaruna segd segdaruna
        | LYKKJUSKILYRDI
        ;

valfasti_range
        : STAFFASTI
        | HEILTALA
        | #(PUNKTURPUNKTUR valfasti_range valfasti_range) /* þáttari skilar aldrei (a..b)
        ;
```