```
                                                        /* PROGRAM */

program:
        unit_sequence
;

unit:
        operator_definition
|       /* \Tfunction{F_n}{a, b}{...} */
        '\Tfunction' function_definition_formal_part
                '{' statement_sequence '}'
|       assignment
|       '\Toutput' output_limit_option '{' expression '}' optional_Tsc
;

                                                        /* DEFINITIONS */

/* Operator definitions */
operator_definition:
        /* \Tdefine{a \xi b, a \in N}{...} */
        '\Tdefine' '{' definee '}' '{' expression '}' optional_Tsc
;

definee:
        /* zeroadic operator */
        identifier
|       /* \xi_i \alpha */
        prinfix_operator_definee formal_operand
|       /* (a, b) \xi_i */
        formal_operand postfix_operator_token formal_subscript_option
|       /* (p, q) \xi_i (r, s) */
        formal_operand prinfix_operator_definee formal_operand
;

prinfix_operator_definee:
        prinfix_operator_token formal_subscript_option
;

function_definition_formal_part:
        '{' function_identifier formal_subscript_option '}'
                '{' formal_parameter_list '}'
;

                                                        /* FORMALS */

/* Functions have formal_parameters: */

formal_parameter:
        possibly_set_typed_identifier
```

```
|       constant                        /* atom for parameter matching */
|       formal_numeric_range
|       possibly_set_typed_formal_parameter_list_packed
|       formal_subsequence_proper
|       possibly_packed_formal_element_range
|       /* Y_{1...m,1..k} */
        identifier formal_index_range
;

/* Operators have formal_operands, which are more restricted: */

formal_operand:
        formal_operand_name_part type_definition_list_addition_option
;

formal_operand_name_part:
        possibly_set_typed_identifier
|       constant                        /* atom for parameter matching */
|       formal_numeric_range
|       possibly_set_typed_formal_parameter_list_packed
;

possibly_set_typed_identifier:
        identifier set_type_specification_option
;

possibly_set_typed_formal_parameter_list_packed:
        formal_parameter_list_pack set_type_specification_option
;

formal_numeric_range:
        /* 0 < i <= 10 */
        range_head range_tail
;

formal_subsequence_proper:
        /* a S a (at least two elements) */
        formal_subsequence_element formal_subsequence_head_option formal_subsequence_eleme
;

formal_subsequence_element:
        identifier
|       possibly_packed_formal_element_range
;

possibly_packed_formal_element_range:
        /* Y_{1,1} \ldots Y_{m,k} */
        /* Y_{1,1} Z_{1} \ldots Y_{m,k} Z{m} */
        formal_element_sequence from_to_token formal_element_sequence
|       formal_element_range_pack
```

```
;

formal_element:
        /* A[m,k] */
        identifier formal_subscript
;

formal_subscript:
        /* _{1, k} */
        '_' formal_index_form
|       '[' formal_index_list ']'
;

formal_index_form:
        formal_index
|       '{' formal_index_list '}'
;

formal_index:
        identifier
|       constant_identifier
|       number_constant
;

formal_index_range:
        /* _{ 1 .. m, 1 .. n } */
        '_' '{' formal_range_list '}'
|       /* [ 1 .. m, 1 .. n ] */
        '[' formal_range_list ']'
;

formal_range:
        /* 1 .. m */
        formal_index from_to_token formal_index
;

                                                        /* TYPE DEFINITIONS */

type_definition:
        /* a, b \in \mathbb{N} */
        identifier_list set_type_specification
|       /* 0 < i <= 10 */
        range_head range_tail
;

type_definition_list_addition:
        ',' type_definition_list
;

range_head:
```

```
        arithmetic_expression less_or_less_equal_comparison_operator identifier
;

range_tail:
        less_or_less_equal_comparison_operator arithmetic_expression
;

set_type_specification:
        /* \in \mathbb{N} */
        '\in' arithmetic_expression
;

                                                        /* ASSIGNMENTS */

assignment:
        '\Tassign' operator_extension_option
                '{' destination '}' '{' expression '}' optional_Tsc
;

operator_extension:
        /* for e.g. x +:= 1 */
        '[' homeotype_infix_operator ']'
;

homeotype_infix_operator:
        /* T op T -> T */
        or_operator_token
|       and_operator_token
|       term_operator
|       '\cap'
|       div_operator
|       '\times'
|       applied_user_operator
;

destination:
        identifier
|       formal_subsequence_proper
|       /* the first n-1 indexes go to the primary_2,
           the last one goes to the lvalue
        */
        primary_2 actual_subscript
|       destination_list_pack
;

actual_subscript:
        /* _i, _{m+\omega,\Gamma k, 1..5} */
        '_' actual_index_form
|       '[' actual_index_list ']'
;
```

```
actual_index_form:
        primary_0
|       '{' actual_index_list '}'
;

output_limit:
        /* for sets only */
        '[' number_constant ']'
;

                                                        /* STATEMENTS */

statement:
        operator_definition
        /* Context check: constant definition only */
|       assignment
|       '\Tcall' '{' function_call '}' optional_Tsc
|       '\Tif' condition_action elseif_or_else_tail_option '\Tendif'
                /* is this what we want ???? */
|       '\Tfor'
        '{' set_generator '}' '{' statement_sequence '}'
        '\Tendfor'
|       '\Twhile' condition_action '\Tendwhile'
|       '\Treturn' '{' expression_option '}'
;

condition_action:
        '{' Boolean_expression type_definition_list_addition_option '}' '{' statement_sequ
;

elseif_or_else_tail:
        '\Telseif' condition_action elseif_or_else_tail_option
|       '\Telse' '{' statement_sequence '}'
;

                                                        /* EXPRESSIONS */

expression:
        /* top level of any expression */
        conditional_expression
|       Boolean_expression
;

conditional_expression:
        conditional_if_expression
|       conditional_if_expression ':' Boolean_expression
;

conditional_if_expression:
```

```
        Boolean_expression '?' Boolean_expression
;

Boolean_expression:
        Boolean_term
|       Boolean_expression or_operator_token Boolean_term
;

Boolean_term:
        Boolean_factor
|       Boolean_term and_operator_token Boolean_factor
;

Boolean_factor:
        arithmetic_expression
|       arithmetic_comparison
;

arithmetic_comparison:
        /* the operands cannot contain Boolean operators,
           hence arithmetic_comparison
        */
        /* i < j; i \not \in N */
        arithmetic_expression comparison_operator arithmetic_expression
|       /* X \prec Y \preceq Z */
        arithmetic_comparison comparison_operator arithmetic_expression
;

comparison_operator:
        /* T op T -> Bool */
        not_operator_token_option relational_operator
|       not_operator_token_option '\in'
;

relational_operator:
        '<'
|       '>'
|       '='
|       '\leq'
|       '\geq'
|       '\neq'
|       '\subset'
|       '\subseteq'
|       '\supset'
|       '\supseteq'
|       other_relational_operator
;

arithmetic_expression:
        term
```

```
|       arithmetic_expression term_operator term
;

term_operator:
        '+'
|       '-'
|       '\cup'
|       setminus_operator_token
;

                                                        /* TERMS */

term:
        term_0
|       term '\cap' term_0
;

term_0:
        /* a; a \times b \times c \times d / e */
        arithmetic_term
|       /* a \xi b c */
        user_term
;

arithmetic_term:
        /* a \times b \times c \times d */
        nominator
|       nominator div_operator operand
;

div_operator:
        '/'
|       '\mod'
;

nominator:
        factor
|       nominator '\times' factor
;

user_term:
        factor possibly_repeated_applied_user_operator factor
|       /* a \xi b c \phi_k d */
        user_term possibly_repeated_applied_user_operator factor
;

possibly_repeated_applied_user_operator:
        applied_user_operator exponent_option
;
```

```
applied_user_operator:
        prinfix_operator_token actual_subscript_option
;
                                                        /* FACTORS */

factor:
        operand
|       factor invisible_operator primary_4
        /* the second operand cannot be prefixed, or a -b would be ambiguous */
;

invisible_operator:
        /* 1. multiplication
           2. concatenation
           3. half a relation operation ????
        */
;
                                                        /* OPERANDS */

/* "Operands" are the basic operands of infix operators.  They come in six
   levels of precedence, as follows:

     -   \Sigma      x   _n   !   ^n
   6   5        4321   1    2   3    456

   For example, everything between the two 5s is produced by primary_5; etc.
*/

operand:
        /* basic operand of infix operators */
        primary_6
;

primary_6:
        /* includes all prefixed things */
        primary_5
|       prefix_operator primary_6
        /* postfixing comes before prefixing:
           -a! == -(a!); -a^2 == -(a^2)
        */
;

prefix_operator:
        possibly_repeated_simple_prefix_operator
|       prinfix_operator_token actual_subscript_option exponent_option
;

possibly_repeated_simple_prefix_operator:
        simple_prefix_operator
```

```
|       simple_prefix_operator exponent
;

simple_prefix_operator:
        '+'
|       '-'
|       '\neg'
;

primary_5:
        primary_4
|       big_operator '_' '{' set_generator '}' big_to_specification_option primary_5
;

big_operator:
        '\bigcup'
|       '\bigwedge'
|       '\bigvee'
|       '\sum'
|       '\prod'
;

big_to_specification:
        '^' '{' expression '}'
;

primary_4:
        /* includes all powers */
        primary_3
|       primary_4 exponent
;

exponent:
        finite_exponent
|       Kleene_exponent
;

finite_exponent:
        '^' primary_0
|       '^' '{' expression '}'
;

Kleene_exponent:
        '^' '*'
|       '^' '+'
;


primary_3:
        /* includes all postfixed things */
```

```
        primary_2
|       primary_3 postfix_operator
;

postfix_operator:
        '!'
|       postfix_operator_token actual_subscript_option
;

primary_2:
        /* includes all subscripted things */
        primary_1
|       primary_2 actual_subscript
;

                                                        /* PRIMARY_1 */

primary_1:
        /* includes all self-contained things */
        primary_0
|       expression_pack
|       tuple_pack
|       set
|       /* |V| */
        '|' expression '|'
|       function_call
|       /* \forall \{a , b \in Q\} ((a, b) \in \omega) */
        sourced_quantifier primary_1
|       /* \frac{355}{113} */
        '\frac' '{' expression '}' '{' expression '}'
|       /* \lfloor n / 2 + 1 \rfloor */
        '\lfloor' expression '\rfloor'
|       /* \lceil n / 2 \rceil */
        '\lceil' expression '\rceil'
|       from_to_token
;

expression_list_proper:
        expression ',' expression_list
;

set:
        /* \{ \}, \{ 42 \}, \{ 1, 4, 9 \} */
        '\{' set_element_list_option '\}'
|       expression_set
|       /* \{ (a, 2) \in \Theta | a \% 2 = 0 \} */
        '\{' pattern_generator '\}'
;

expression_set:
```

```
        /* \{ a^2, a \in \mathbb{N} | a \mod 2 = 0 \} */
        '\{' good_or_bad_expression_generator filter '\}'
|       /* \{ a^2, a \in \mathbb{N} \} */
        '\{' expression_generator '\}'
;

good_or_bad_expression_generator:
        expression_generator
|       /* erroneous: source_definition_list missing */
        arithmetic_expression
;

expression_generator:
        /* (a^2, b), a \in \mathbb{N}, b \in Q */
        arithmetic_expression ',' source_definition_list
;

filter:
        /* | a \mod 2 = 0 */
        '|' Boolean_expression_list
;

pattern_generator:
        /* (a, 2) \in \Theta */
        formal_operand source_specification
|       range_source_definition
;

function_call:
        function_identifier actual_subscript_option expression_list_pack
;

sourced_quantifier:
        /* \forall \{ a , b \in Q \} */
        quantifier '\{' source_definition '\}'
|       /* \forall _ { a , b \in Q } */
        quantifier '_' '{' source_definition '}'
;

quantifier:
        '\forall'
|       '\exists'
;

set_generator:
        /* a b c, a \in V */
        expression_generator
|       pattern_generator
;
```

```
                                                        /* SOURCE DEFINITIONS */

source_definition:
        /* a, b \in \mathbb{N} */
        identifier_list source_specification
|       range_source_definition
;

range_source_definition:
        /* 0 < i <= 10 */
        arithmetic_expression less_or_less_equal_comparison_operator identifier less_or_le
;

less_or_less_equal_comparison_operator:
        '<'
|       '\leq'
;

source_specification:
        /* \in \mathbb{N} */
        '\in' arithmetic_expression
;

primary_0:
        /* so small it can be the right operand of ^ and _ without {}*/
        constant
|       identifier
;

constant:
        number_constant
|       '\emptyset'
|       '\varepsilon'
|       '\mathbb{N}'
|       '\mathbb{Z}'
|       '\mathbb{Q}'
|       '\infty'
|       '\Ttrue'
|       '\Tfalse'
|       delayed_numeric_token          /* for debugging */
|       bad_numeric_token              /* for debugging */
|       loop_token                     /* for debugging */
;

                                                        /* IDENTIFIERS */

identifier:
        small_identifier
|       large_identifier
;
```

```
optional_Tsc:
        '\Tsc'
|
;
```