TruPL Grammar v3.0 - A Semantically-Attributed Version of TruPL 2.0

- Non-terminals (i.e. `PROGRAM`, `IF_STMT`) are shown in `ALL CAPS`.
- Keywords (i.e. *program*, *if*) are shown in *lowercase italics*.
- Non-keyword terminals (i.e. `identifier`, `mulop`) and special symbols (i.e. '`;`', '`+`') are shown in `typewriter font`.
- There are six variables that are global to the grammar (and hence to the parser):
  - "stab" is the symbol table.
  - "main_env" contains the name of the environment of the main program. It is set in parse_program() and never changes. `_EXTERNAL` is a special literal for the environment where the program name is defined.
  - "current_env" contains the name of the environment we are currently parsing. It will change as we enter and exit parse_procedure_decl.
  - "actual_parm_position" contains the position of an actual parameter as we parse it in a procedure call statement.
  - "formal_parm_position" contains the position of a formal parameter as we parse it in a procedure definition.
  - "procedure_name" contains the potential procedure name as we try to discover if we are looking at a procedure call.

PROGRAM	$\longrightarrow$	*program* `identifier`
		{stab.install(identifier.attr, "_EXTERNAL", program_t);
		current_env = identifier.attr;
		main_env = identifier.attr;}
		; DECL_LIST BLOCK ;
DECL_LIST	$\longrightarrow$	VARIABLE_DECL_LIST PROCEDURE_DECL_LIST
VARIABLE_DECL_LIST	$\longrightarrow$	VARIABLE_DECL ; VARIABLE_DECL_LIST
		| λ
VARIABLE_DECL	$\longrightarrow$	IDENTIFIER_LIST : STANDARD_TYPE
		{foreach (identifier i in stab such that i.type == unknown_t)
		stab.update_type (i, STANDARD_TYPE.type);}
PROCEDURE_DECL_LIST	$\longrightarrow$	PROCEDURE_DECL ; PROCEDURE_DECL_LIST

```
                           | λ
IDENTIFIER_LIST        →   identifier
                               {if (is_declared (identifier.attr, current_env))
                                   multiple_definition_error();
                                else
                                   stab.install (identifier.attr, current_env, unknown_t}
                           IDENTIFIER_LIST_PRM
IDENTIFIER_LIST_PRM    →   , identifier
                               {if (is_declared (identifier.attr, current_env))
                                   multiple_definition_error();
                                else
                                   stab.install (identifier.attr, current_env, unknown_t}
                           IDENTIFIER_LIST_PRM
                           | λ
STANDARD_TYPE          →   int
                               {STANDARD_TYPE.type = int_t;}
                           | bool
                               {STANDARD_TYPE.type = bool_t;}
BLOCK                  →   begin STMT_LIST end
PROCEDURE_DECL         →   procedure identifier
                               {if (is_declared(id, current_env))
                                   multiply_declared_id_error();
                                else
                                   stab.install (identifier.attr, current_env, procedure_t);
                                   current_env = identifier.attr;
                                   formal_parm_position = 0;}
                           ( PROCEDURE_ARGS ) VARIABLE_DECL_LIST BLOCK
                               {current_env = main_env;}
PROCEDURE_ARGS         →   FORMAL_PARM_LIST
                           | λ
FORMAL_PARM_LIST       →   identifier
                               {if (is_declared(identifier.attr, current_env))
                                   multiple_definition_error();
                                   else
                                       stab.install(identifier.attr, current_env, unknown_t, formal_parm_position);
                                       formal_parm_position++;}
                                 IDENTIFIER_LIST_PRM : STANDARD_TYPE
                                    {foreach (identifier i in stab such that i.type == unknown_t)
                                       stab.update_type (i, current_env, STANDARD_TYPE.type);}
                                 FORMAL_PARM_LIST_HAT
FORMAL_PARM_LIST_HAT    ⟶      ; FORMAL_PARM_LIST
                                 | λ
STMT_LIST                ⟶      STMT ; STMT_LIST_PRM
                                 | ; STMT_LIST_PRM
STMT_LIST_PRM            ⟶      STMT ; STMT_LIST_PRM
                                 | λ
```

```
STMT                    →   | IF_STMT
                            | WHILE_STMT
                            | PRINT_STMT
                            | identifier
                                {if (!is_declared(identifier.attr, current_env))
                                    undeclared_id_error();}
                                 else
                                    procedure_name = identifier.attr;}
                            AD_HOC_AS_PC_TAIL
                                {if ad_hoc_as_pc_tail.type != identifier.type
                                    type_error();}
AD_HOC_AS_PC_TAIL       →   := EXPR
                                {AD_HOC_AS_PC_TAIL.type = EXPR.type;}
                            | (
                                if (get_type (procedure_name, main_env) != procedure_t)
                                    type_error();
                                 actual_parm_position = 0;}
                            EXPR_LIST )
                                {AD_HOC_AS_PC_TAIL.type = procedure_t;}
IF_STMT                 →   if EXPR
                                {if (EXPR.type != bool_t)
                                    type_error();}
                            then BLOCK IF_STMT_HAT
IF_STMT_HAT             →   else BLOCK
                            | λ
WHILE_STMT              →   while EXPR
                                {if (EXPR.type != bool_t)
                                    type_error();}
                            loop BLOCK
PRINT_STMT              →   print EXPR
                                {if (EXPR.type != int_t | | EXPR.type != bool_t)
                                    type_error();}
EXPR_LIST               →   ACTUAL_PARM_LIST
```

```
                                | λ
ACTUAL_PARM_LIST         ⟶    EXPR
                                    if (stab.get_type (procedure_name, main_env, actual_parm_position) != EXPR.type)
                                        type_error();
                                    actual_parm_position++;}
                                ACTUAL_PARM_LIST_HAT
ACTUAL_PARM_LIST_HAT     ⟶    , ACTUAL_PARM_LIST
                                | λ
EXPR                     ⟶    SIMPLE_EXPR EXPR_HAT
                                    {if (EXPR_HAT.type == no_t)
                                        EXPR.type = SIMPLE_EXPR.type;
                                     else if (SIMPLE_EXPR.type == int_t && EXPR_HAT.type == int_t)
                                        EXPR.type = bool_t;
                                     else
                                        type_error();}
EXPR_HAT                 ⟶    relop SIMPLE_EXPR
                                    {if (SIMPLE_EXPR.type == int_t)
                                        EXPR_HAT.type = int_t;
                                     else
                                        type_error();}
                                | λ
                                    {EXPR_HAT.type == no_t;}
SIMPLE_EXPR              ⟶    TERM SIMPLE_EXPR_PRM
                                    {if (SIMPLE_EXPR_PRM.type == no_t)
                                        SIMPLE_EXPR.type = TERM.type;
                                     else if (TERM.type == SIMPLE_EXPR_PRM.type)
                                        SIMPLE_EXPR.type = TERM.type;
                                     else
                                        type_error();}
SIMPLE_EXPR_PRM0         ⟶    addop TERM SIMPLE_EXPR_PRM1
                                    {if (SIMPLE_EXPR_PRM1.type == no_t)
                                        if (addop.type == TERM.type)
                                            SIMPLE_EXPR_PRM0.type = addop.type;
```

```
                                          else
                                              type_error();
                                      else if (addop.type == TERM.type && TERM.type == SIMPLE_EXPR_PRM$_1$.type)
                                          SIMPLE_EXPR_PRM$_0$.type = addop.type;
                                      else
                                          type_error();}
                                | $\lambda$
                                     {SIMPLE_EXPR_PRM$_0$.type = no_t;}
TERM                    $\longrightarrow$  FACTOR TERM_PRM
                                     {if (TERM_PRM.type == no_t)
                                          TERM.type = FACTOR.type;
                                      else if (FACTOR.type == TERM_PRM.type)
                                          TERM.type = FACTOR.type;
                                      else
                                          type_error();}
TERM_PRM$_0$              $\longrightarrow$  mulop FACTOR TERM_PRM$_1$
                                     {if (TERM_PRM$_1$.type == no_t && mulop.type == FACTOR.type)
                                          TERM_PRM$_0$.type = mulop.type;
                                     else if (mulop.type == FACTOR.type && FACTOR.type == TERM_PRM$_1$.type)
                                          TERM_PRM$_0$.type = mulop.type;
                                     else
                                          type_error();}
                                | $\lambda$
                                     {TERM_PRM$_0$.type = no_t}
FACTOR$_0$                $\longrightarrow$  identifier
                                     {if (!is_declared(identifier.attr, current_env))
                                          undeclared_id_error();
                                      else
                                          FACTOR$_0$.type = stab.get_type(identifier.attr, current_env);}
                                | num
                                     {FACTOR$_0$.type = int_t;}
                                | ( EXPR )
                                     {FACTOR$_0$.type = EXPR.type}
```

```
                    | SIGN FACTOR$_1$
                        {if (SIGN.type != FACTOR$_1$.type)
                            type_error();
                         else
                            FACTOR$_0$.type = FACTOR$_1$.type}
SIGN          ⟶    +
                        {SIGN.type = int_t}
                    | -
                        {SIGN.type = int_t}
                    | not
                        {SIGN.type = bool_t}
```