The Concrete Syntax of VDM

Roger Bishop Jones

ICL Defence Systems

1. INTRODUCTION

This is a tranbscription of the concrete syntax of VDM into sml. The material is transcribed from STC document no.725 07863 Ed 3. The metalanguage is described in DBC/RBJ/060.

2. MODULES

```
module_spec     =   interface [specification|] | specification;
```

3. INTERFACES

```
interface     =   NAME itype "-INTERFACE" "IS" {"IMPORT" NAME import_list}
                  ["EXPORT" "(" {qual_and_type?";"} ")"|];
itype         =   "DECLNS" | "ADT";
import_list   =   "(" { "("{import_name ? ","}")" qualification ? ","}      |
                  "ALL";
import_name       =   NAME | LCNAME | UCNAME;
qualification     =   "FN" | "OP" | "CN" | "TY";
qual_and_type     =   "FN" LCNAME [":" fntype|]    |
                      "OP" UCNAME [":" [xlist|] "->>" [xlist|] |] |
                      "CN" {LCNAME?","} [":" type|]                          |
                      "TY" {NAME?","} [rest_type|];
```

4. SPECIFICATIONS

```
specification     =   NAME "ADT_SPEC" "IS" {var_decl} [init|] spec_body;
init              =   "INIT-" NAME "(" pattern ")" "==" expr;
spec_body         =   decl1 [decl2|] "END";
decl1             =   decl_item {decl_item};
decl_item         =   function_decl | const_decl | type_decl | invar_decl;
decl2             =   operation {operation | function_decl};
type_decl         =   {NAME?","} rest_type;
rest_type         =   "::" record | "=" [type|expr] | "IS" "NOT_YET_DEFINED";
var_decl          =   {UCNAME?","} ":" type;
function_decl     =   LCNAME ":" fn_type opt_fpre post_or_explicit;
fn_type           =   [xlist|] "->>" target_type;
xlist             =   { [type | "("fn_type")"] ? "X"};
target_type       =   type | fn_type | "(" fn_type ")";
opt_fpre          =   "PRE" LCNAME {"("[pattern_list|]")"};
pattern_list  =   {pattern ? ","};
pattern           =   LCNAME | "_" | simple_literal |
                      list_pattern | record_pattern;
invar_decl        =   "INV-" NAME "(" pattern ")" "==" expr;
operation         =   UCNAME "(" par_list ")" par_list opn_spec;
par_list          =   {{UCNAME?","} ":" type ? ","}      |;
opn_spec          =   opt_ext {let_clause} {"ERR" UCNAME expr}
                      ["PRE" expr|] "POST" expr {"ERRPOST" UCNAME expr};
opt_ext           =   "EXT" {{UCNAME?","} ":" ext_qual}      |;
ext_qual          =   ["RD"|"WR"] type;
type              =   explicit_type | union_type | map_type;
```

```
explicit_type   =   basic_type {"-set"|"-list"|"-list1"};
basic_type      =   "@" NAME | dname | "Bool" | "Nat" | "Nat0" |
                    "Int" | "Real" | "Char" | "Text" | "(" type ")";
union_type      =   {[type|UCNAME] ? "|"} | "[" type "]";
map_type        =   explicit_type ["->"|"<->"] explicit_type;
record          =   [ {type} | {UCNAME ":" type} ] "END";
expr            =   {{prefix_op} element ? infix_op}{"WHERE" whereclse_block}      |
                    expr ["IN"|"NOTIN"] explicit_type;

infix_op        =   "+" | "-" | "*" | "/" | "MOD" | "//" |
                    "**" | "=>" | "<=>" | "OR" | "AND" | "=" |
                    "~=" | "<=" | "<" | ">" | ">=" | "SSUB" |
                    "SUB" | "IN" | "NOTIN" | "UNION" | "DIFF" |
                    "INTER" | "CONC" | "OVERWR" | "MUNION" |
                    "RESTR" | "REM" | "o" | "^" | "WHERE";
prefix_op       =   "+" | "LEN" | "CARD" | "~" | "DUNION" |
                    "DOM" | "RNG" | "INDS" | "ELEMS" | "DINTER" |
                    "TL" | "DCONC" | "HD";
element         =   "(" expr ")"                                |
                    {"("["ALL"|"EXISTS"|"EXISTS1"]{quant_var?","}")"}
                          "(" expr ")"                          |
                    "PRE-"      ducname     "(" exprlist ")"    |
                    "POST-"     ducname     "(" exprlist ")"    |
                    "PRE-"      dlcname     "(" optexprl ")"    |
                    "POST-"     dlcname     "(" exprlist ")"    |
                    "INV-"      dname               "(" expr ")"|
                    "INIT-"     dname               "(" expr ")"|
                    set_constr | list_constr | map_constr |
                    element "MODIFY" "(" mod_list ")"           |
                    rec_constr  |       "NIL"|       LCNAME "'"      |
                    constant    |       cond_expr   |
                    UCNAME "AS" expr                                |
                    element "(" optexprl ")"                |
                    let_expr;
modlist         =   {UCNAME "AS" expr ? ","};
constant        =   INTVAL | REALVAL | STRINGVAL | CHARVAL |
                    boolean | dlcname | UCNAME;
type_or_expr    =   explicit_type | simple_expr;
let_expr        =   let_clause expr;
let_clause      =   "LET" let_clause_block "WITHIN";
let_clause_block =  {clause_item ? "ANDLET"};
where_clse_block =  {clause_item ? "ANDWHERE"};
clause_item     =   function_decl                               |
                    "LCNAME" [":" type|] ["=="|"IN"] expr       |
                    pattern_alt [":" type|] ["=="|"IN"] expr;
```