Reliable Software

Term Analyser

Documentation de projet

Attiogbe

13 Juillet 2007

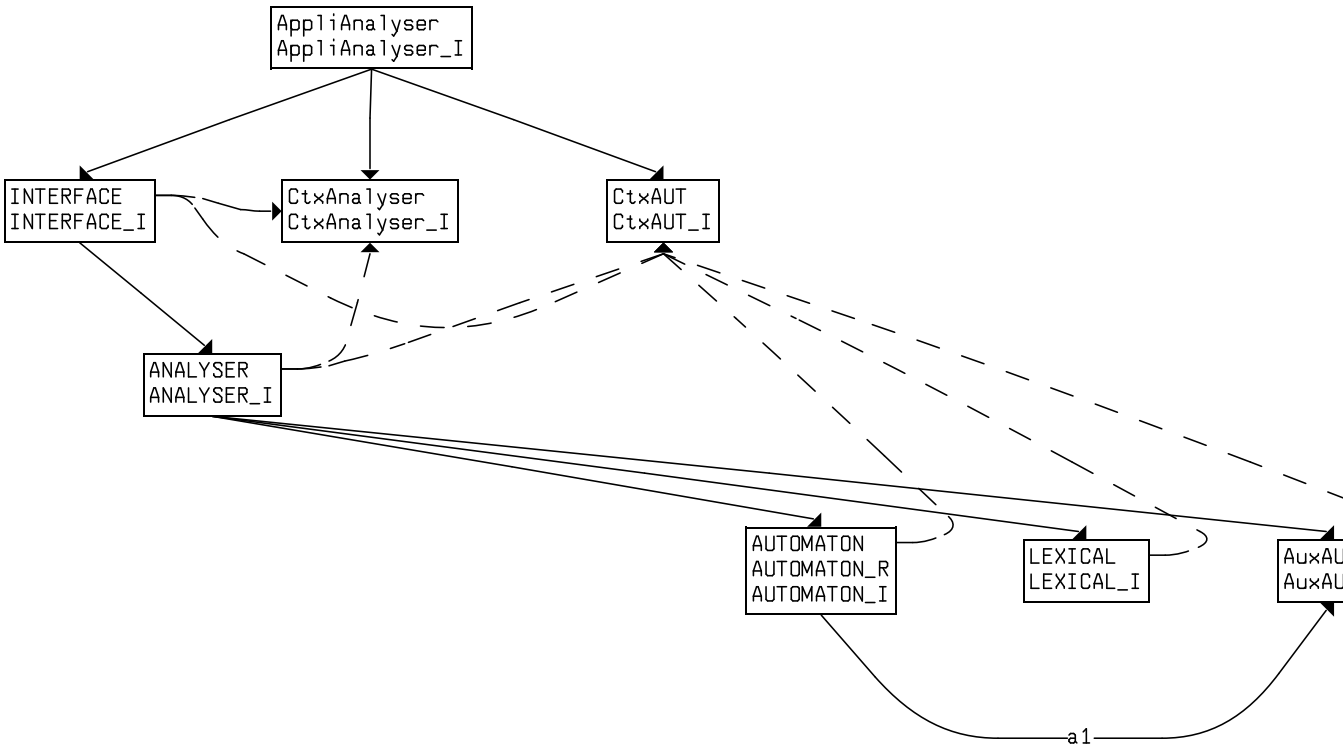
1 INTRODUCTION

La documentation est effectuée sur le projet : TermAnalyser.

2 STRUCTURE GENERALE

2.1 Introduction

2.2 Graphe de dépendance



2.3 Métrique

Project status

COMPONENT	TC	POG	Obv	nPO	nUn	%Pr	B0C	C	Ada	C++	HIA
ANALYSER	OK	OK	34	1	0	100	-				
ANALYSER_I	OK	OK	369	219	85	61	OK	OK	-	-	-
AUTOMATON	OK	OK	183	35	3	91	-				
AUTOMATON_I	OK	-					OK	OK	-	-	-
AUTOMATON_R	OK	OK	429	64	10	84	-				
AppliAnalyser	OK	OK	3	0	0	100	OK				
AppliAnalyser_I	OK	OK	49	72	0	100	OK	OK	-	-	-
AuxAUT	OK	OK	14	2	0	100	-				
AuxAUT_I	OK	OK	143	96	35	63	OK	OK	-	-	-
CtxAUT	OK	OK	1	0	0	100	-				
CtxAUT_I	OK	OK	16	2	0	100	OK	OK	-	-	-
CtxAnalyser	OK	OK	1	0	0	100	OK				
CtxAnalyser_I	OK	OK	6	0	0	100	OK	OK	-	-	-
INTERFACE	OK	OK	12	2	1	50	-				
INTERFACE_I	OK	OK	139	24	17	29	OK	OK	-	-	-
LEXICAL	OK	OK	11	4	0	100	-				
LEXICAL_I	OK	OK	35	32	12	62	OK	OK	-	-	-
TOTAL	OK	-	1445	553	163	70	-	OK	-	-	-

3 DESCRIPTION DES COMPOSANTS

3.1 Composant ANALYSER

3.1.1 Présentation

Le composant traité dans ce paragraphe est : ANALYSER.

3.1.2 Métrique

ANALYSER AutoProved /home/enseignants/attiogbe/SPEC/DESS/B/TermAnalyser/sources/ANALYSER

	NbObv	NbPO	NbPRi	NbPRa	NbUn	%Pr
InstanciatedConstraintsLemmas	0	0	0	0	0	100
Initialisation	2	0	0	0	0	100
add_state_automaton	3	0	0	0	0	100
add_transition_automaton	8	1	0	1	0	100
setInitialState	3	0	0	0	0	100
addFinalState	3	0	0	0	0	100
new_automaton	3	0	0	0	0	100
build_automaton	2	0	0	0	0	100
build_lexic	2	0	0	0	0	100
lexicalParseExpression	3	0	0	0	0	100
recogniseTerm	3	0	0	0	0	100
write_automaton	2	0	0	0	0	100
ANALYSER	34	1	0	1	0	100

3.1.3 Règles ajoutées à la preuve

Fichier /home/enseignants/attiogbe/SPEC/DESS/B/TermAnalyser/bdp/ANALYSER.pmm non existant.

3.1.4 Source B

```
/*_____________________________________*/
/*  Projet   : NaBLa (Nantes B Libraries)                  */
/*  Dossier  : Automata / Analyseur d'expressions          */
/*  C. Attiogbe, 2000-...                                  */
/*_____________________________________*/
MACHINE
      ANALYSER


SEES
   CtxAUT
,    CtxAnalyser /* nbWords */

INCLUDES
      AUTOMATON
,     LEXICAL

VARIABLES
```

$symbTerm$

INVARIANT
$symbTerm \in 1 \,..\, nbWords \rightarrow SYMBOL$

INITIALISATION
$symbTerm :\in 1 \,..\, nbWords \rightarrow SYMBOL$

OPERATIONS

add_state_automaton(st) =
PRE
$st \in STATE$
$\wedge\ st \notin Qqd$
THEN
add_state(st)
END
;
add_transition_automaton(ist, xx, fst) =
PRE
$ist \in STATE \wedge ist \in Qqd$
$\wedge\ fst \in STATE \wedge fst \in Qqd$
$\wedge\ xx \in SYMBOL \wedge xx \in Xad$
$\wedge\quad (ist \mapsto xx) \notin \mathbf{dom}(deltad)$
THEN
add_transition(ist, xx, fst)
END
;
setInitialState(ist) =
PRE
$ist \in STATE \wedge ist \in Qqd \wedge ist \notin Q0d$
THEN
add_initialState(ist)
END
;
addFinalState(st) =
PRE
$st \in STATE \wedge st \in Qqd \wedge st \notin Qfd$
THEN
add_finalState(st)
END
;

new_automaton =
clear_automaton
;
build_automaton = **skip**
/* to build the automaton for the current analyser */
;
build_lexic = **skip**
/* to build the lexic {(word, symbol)} of the current analyser */
;

/* lexical analysis of an expression */

```
    lexicalParseExpression(ch, ezis) =
        PRE
        ch ∈ 1 .. nbWords → WORD
      ∧ ezis ∈ NAT
      ∧ ezis > 0 ∧ ezis < nbWords
        THEN
           ANY
              sq
           WHERE
              sq ∈ 1 .. nbWords → SYMBOL
         ∧     ∀ (ii,wi).((ii ∈ NAT ∧ ii > 1 ∧ ii < nbWords ∧ wi = ch(ii))
                    ⇒ sq(ii) = lexCategory(wi))
           THEN
             symbTerm := sq
           END
        END
;
rr ←   recogniseTerm =
    /*  syntactic analysis of an expression */
    /* Check that a sequnce of symbols is accepted by the automaton */
    PRE
      card(symbTerm) > 0
    THEN
       rr :∈ BOOL
    END
;
    write_automaton = skip

END
```

3.2 Composant ANALYSER_I

3.2.1 Présentation

Le composant traité dans ce paragraphe est : ANALYSER_I.

3.2.2 Métrique

ANALYSER_I POGenerated /home/enseignants/attiogbe/SPEC/DESS/B/TermAnalyser/sources/ANALY

	NbObv	NbPO	NbPRi	NbPRa	NbUn	%Pr
ValuesLemmas	2	0	0	0	0	100
InstanciatedConstraintsLemmas	1	6	0	4	2	66
Initialisation	4	1	0	0	1	0
add_state_automaton	13	0	0	0	0	100
add_transition_automaton	18	1	0	1	0	100
setInitialState	13	0	0	0	0	100
addFinalState	13	0	0	0	0	100
new_automaton	9	0	0	0	0	100

	NbObv	NbPO	NbPRi	NbPRa	NbUn	%Pr
build_automaton	23	35	0	8	27	22
build_lexic	18	17	0	3	14	17
lexicalParseExpression	20	31	0	17	14	54
recogniseTerm	220	102	0	76	26	74
write_automaton	15	26	0	25	1	96
ANALYSER_I	369	219	0	134	85	61

3.2.3 Règles ajoutées à la preuve

Fichier /home/enseignants/attiogbe/SPEC/DESS/B/TermAnalyser/bdp/ANALYSER_I.pmm non existant.

3.2.4 Source B

```
/*_____________________________________*/
/*  Projet  : NaBLa (Nantes B Libraries)                  */
/*  Dossier : Automata / Analyseur d'expressions          */
/*  C. Attiogbe, 2000-...                                 */
/*_____________________________________*/
IMPLEMENTATION
      ANALYSER_I

REFINES
      ANALYSER

SEES
   CtxAUT /* SYMBOL, WORD, STATE */
,    CtxAnalyser /* nbWords */

IMPORTS
      AUTOMATON
,    AuxAUT
,      LEXICAL
,      r_seqSymb.L_SEQUENCE(maxSymbols,SYMBOL) /* result of a lexical parse */
,    r_seqState.L_SEQUENCE(maxStates,STATE) /* result of a syntactical parse */
, bio.BASIC_IO
INVARIANT
   r_seqSymb.seq_vrb ⊆ symbTerm


INITIALISATION
      r_seqSymb.CLR_SEQ
;      r_seqState.CLR_SEQ

OPERATIONS

   add_state_automaton(st) =
   BEGIN
   add_state(st)
   END
;
   add_transition_automaton(ist, xx, fst) =
   BEGIN
   add_transition(ist, xx, fst)
```

```
    END
;
    setInitialState(ist) =
    BEGIN
    add_initialState(ist)
    END
;
    addFinalState(st) =
    BEGIN
    add_finalState(st)
    END
;
    new_automaton =
    clear_automaton
;
        build_automaton =
        /* to build the automaton for the current analyser  */
        BEGIN
    VAR bb,s0, s1, s2, s3, s4,
    art, nc, vb, np
    IN
    r_seqSymb.CLR_SEQ;
        r_seqState.CLR_SEQ;
    bb, s0 ← newState;
    bb, s1 ← newState;
    bb, s2 ← newState;
    bb, s3 ← newState;
    bb, s4 ← newState;

        add_state(s0);
        add_state(s1);
        add_state(s2);
        add_state(s3);
        add_state(s4);

        add_initialState(s0);
        add_finalState(s4);

art := 20;
nc := 21;
vb := 22;
np := 23 ;

    add_symbol(art);
    add_symbol(nc);

        add_transition(s0, art, s1 );
    add_transition(s1, nc, s2);
        add_transition(s0, nc, s2 );
        add_transition(s2, vb, s3 );
    add_transition(s3, art, s4);
        add_transition(s3, np, s4)
        END
    END
```

```
;
      build_lexic =
      /* to build the lexic {(word, symbol)} of the current analyser  */
      BEGIN

add_lexicalMap(le, 200);
add_lexicalMap(la, 200);
add_lexicalMap(un, 200);
add_lexicalMap(chien, 201);
add_lexicalMap(point, 203)


      END
;
   lexicalParseExpression (ch, ezis) =
   /* syntactic analysis opf an expression  */
   VAR nbitem, cpt, word, symb
   IN
       r_seqSymb.CLR_SEQ;
       nbitem := ezis;
       cpt := 1;
         word := ch(cpt);
         symb ← lexicalParseWord(word);
        WHILE cpt < nbitem DO /* the stepwis lexical parsing */
         word := ch(cpt);
         symb ← lexicalParseWord(word);
         r_seqSymb.PUSH_SEQ(symb);
        cpt := cpt + 1
        INVARIANT
          cpt ∈ 1 .. nbitem
        ∧   nbitem ∈ NAT
        VARIANT
          nbitem + 1 - cpt
        END
   ; /* now the last value of ii */
    word := ch(cpt);
       symb ← lexicalParseWord(word);
       r_seqSymb.PUSH_SEQ(symb)
   END
;
rr ←   recogniseTerm =
   /*  syntactic analysis of an expression */
   /* Check that a sequence of symbols is accepted by the automaton */
   VAR
     nbSymb
   , cpt
   , cptS /* nb states of the recognition chain */
   , nbRecognisedState /* after the syntactic analysis */
   , bb

   IN
   rr := FALSE;
   nbSymb ← r_seqSymb.LEN_SEQ;
```

```
cpt := 1;
cptS := 0; bb := FALSE;
nbRecognisedState := 0;
IF nbSymb = 0
THEN
      rr := FALSE
ELSE
 VAR
   qi, sbli, bb1, sl1
 IN
   bb1 := FALSE; sl1 := 0;
   r_seqState.CLR_SEQ; /* sequence of reached states */
   qi ← getInitial; /* initial state */
   sbli ← r_seqSymb.VAL_SEQ(cpt); /* first symbol */
   WHILE cpt < nbSymb DO
     sbli ← r_seqSymb.VAL_SEQ(cpt);
     bb1 ← isTransition(qi, sbli);
   IF bb1 = FALSE
   THEN
        cpt := nbSymb;
  rr := FALSE
   ELSE
     sl1 ← r_seqState.LEN_SEQ;
     IF sl1>0
  THEN
  r_seqState.PUSH_SEQ(qi);
  qi ← direct_successor(qi, sbli);
  cpt := cpt+1
  END
   END
  INVARIANT
       cpt ∈ 1 .. nbSymb
  VARIANT
    nbSymb - cpt
  END
 ;/* now the last value of cpt =nbSymb */
 IF rr = TRUE
  THEN
    sbli ← r_seqSymb.VAL_SEQ(cpt);
    bb1 ← isTransition(qi, sbli);
    IF bb1 = FALSE
    THEN
     rr := FALSE
    ELSE
        sl1 ← r_seqState.LEN_SEQ;
         IF sl1>0
      THEN
       r_seqState.PUSH_SEQ(qi);
       qi ← direct_successor(qi, sbli)
      END
    END
  END
```

```
  END



;
 cptS ← r_seqState.LEN_SEQ;
      nbRecognisedState := nbSymb+1;
    IF cptS = nbRecognisedState /* we should have nbSymb+1 states */
    THEN
VAR is, fs, bb1, bb2 IN
      is ← r_seqState.VAL_SEQ(1); /* the initial state */
      fs ← r_seqState.VAL_SEQ(nbRecognisedState); /* final one */
      bb1  ← isInitialState(is);
      bb2  ← isFinalState(fs);

      IF (bb1 = TRUE)
   THEN
     IF (bb2 = TRUE)
         THEN
   rr := TRUE
     ELSE
   rr := FALSE
     END
   ELSE
     rr := FALSE
     END
END
    END
END
END
;
write_automaton =
VAR cpt IN
cpt := 0;
WHILE cpt < 6 DO
cpt := cpt+1;
VAR sti, sbl, stf IN
  sti,sbl, stf ← get_transition(cpt);
  bio.STRING_WRITE("\n transition : ");
   bio.INT_WRITE(sti);
   bio.STRING_WRITE(", ");
   bio.INT_WRITE(sbl) ;
   bio.STRING_WRITE(", ");
   bio.INT_WRITE(stf) ;
   bio.STRING_WRITE("\n")
END
INVARIANT
cpt ∈ 0 .. 6
VARIANT
6 - cpt
END
END
```

END

3.3 Composant AUTOMATON

3.3.1 Présentation

Le composant traité dans ce paragraphe est : AUTOMATON.

3.3.2 Métrique

AUTOMATON POGenerated /home/enseignants/attiogbe/SPEC/DESS/B/TermAnalyser/sources/AUTO

	NbObv	NbPO	NbPRi	NbPRa	NbUn	%Pr
Initialisation	2	6	0	6	0	100
clear_automaton	3	6	0	6	0	100
add_state	7	4	0	4	0	100
remove_state	5	5	0	5	0	100
add_symbol	8	2	0	2	0	100
remove_symbol	8	2	0	1	1	50
add_initialState	9	1	0	1	0	100
remove_initialState	8	2	0	2	0	100
add_finalState	11	0	0	0	0	100
remove_finalState	9	1	0	1	0	100
add_transition	8	2	0	2	0	100
remove_transition	8	2	0	1	1	50
modify_transition	8	2	0	1	1	50
get_transition	10	0	0	0	0	100
check_consistent	19	0	0	0	0	100
direct_successor	10	0	0	0	0	100
isInitialState	10	0	0	0	0	100
isFinalState	10	0	0	0	0	100
getInitial	10	0	0	0	0	100
setDummyState	10	0	0	0	0	100
isTransition	10	0	0	0	0	100
AUTOMATON	183	35	0	32	3	91

3.3.3 Règles ajoutées à la preuve

Fichier /home/enseignants/attiogbe/SPEC/DESS/B/TermAnalyser/bdp/AUTOMATON.pmm non existant.

3.3.4 Source B

```
/*_____*/
/* Projet : NaBLa (Nantes B Libraries)                    */
/* Dossier : Automata / Analyseur d'expressions           */
/* C. Attiogbe, 2000-...                                  */
/*_____*/
```

MACHINE

AUTOMATON /* Determinitsic Finite State Automata */

SEES

CtxAUT /* Parameters: STATE, SYMBOL, etc */

SETS

$ConsistentSet = \{OK, NOK\}$

VARIABLES

Xad /* Alphabet of the DFA */
, *Qqd* /* set of states of the DFA */
, *Q0d* /* initial state (only one) of the DFA */
, *Qfd* /* final states aof the DFA */
, *deltad* /* transition function of the DFA */
, *consistentAutom* /* does the DFA consistent (connexity) */
, *dummyState*

INVARIANT

$Xad \in \mathcal{F}(SYMBOL)$
$\wedge$ $Qqd \subseteq STATE$
$\wedge$ $Q0d \subseteq Qqd$
$\wedge$ $\mathbf{card}(Q0d) \leq 1$
$\wedge$ $Qfd \subseteq Qqd$
$\wedge$ $deltad \in Qqd \times Xad \nrightarrow STATE$
$\wedge$ $\mathbf{ran}(deltad) \subseteq Qqd$
$\wedge$ $consistentAutom \in ConsistentSet$
$\wedge$ $dummyState \in STATE$

INITIALISATION

$Xad, Qqd, Q0d, Qfd, deltad := \emptyset , \emptyset , \emptyset , \emptyset , \emptyset$
|| $consistentAutom := NOK$
|| $dummyState :\in STATE$

OPERATIONS

clear_automaton =
BEGIN
$Xad, Qqd, Q0d, Qfd, deltad := \emptyset , \emptyset , \emptyset , \emptyset , \emptyset$
END
;

add_state(qq) =
/* add a state to the automaton */
PRE
$qq \in STATE$
$\wedge$ $qq \notin Qqd$
THEN
$Qqd := Qqd \cup \{qq\}$
|| $consistentAutom := NOK$
END
;

remove_state(qq) =
/* remove a state of the automaton */
PRE
$qq \in STATE$
$\wedge$ $qq \in Qqd$

$\wedge \quad qq \notin \mathbf{dom}(\mathbf{dom}(deltad))$
$\wedge \quad qq \notin \mathbf{ran}(deltad)$
$\wedge \quad qq \notin Qfd$
$\wedge \quad qq \notin Q0d$
$\wedge \quad deltad : (Qqd - \{qq\}) \times Xad \nrightarrow (Qqd - \{qq\})$
THEN
$Qqd := Qqd - \{qq\}$
END
;
add_symbol$(sx) =$
PRE
$sx \in SYMBOL$
$\wedge \quad sx \notin Xad$
THEN
$Xad := Xad \cup \{sx\}$
END
;
remove_symbol$(xx) =$

PRE
$xx \in SYMBOL$
$\wedge \quad xx \notin \mathbf{ran}(\mathbf{dom}(deltad))$
THEN
$Xad := Xad - \{xx\}$
END
;

add_initialState$(ii) =$
PRE
$ii \in STATE \wedge ii \in Qqd$
$\wedge \quad ii \notin Q0d$

THEN
$Q0d := \{ii\}$
END
;

remove_initialState$(ii) =$
PRE
$ii \in STATE \wedge ii \in Qqd$
$\wedge \quad ii \in Q0d$
$\wedge \quad \mathbf{card}(Q0d) = 1$

THEN
$Q0d := Q0d - \{ii\}$
END
;

add_finalState$(ff) =$
PRE
$ff \in STATE \wedge ff \in Qqd$
$\wedge \quad ff \notin Qfd$
THEN

$Qfd := Qfd \cup \{ff\}$

END

;

remove_finalState(ff) =

PRE

$ff \in STATE \wedge ff \in Qqd$

$\wedge \quad ff \in Qfd$

THEN

$Qfd := Qfd$ - $\{ff\}$

END

;

add_transition(qi, xx, qf) =

PRE

$qi \in STATE \wedge qf \in STATE$

$\wedge \quad qi \in Qqd$

$\wedge \quad xx \in SYMBOL$

$\wedge \quad xx \in Xad$

$\wedge \quad qf \in Qqd$

$\wedge \quad (qi \mapsto xx) \notin \mathbf{dom}(deltad \rhd \{qf\})$

THEN

$deltad := deltad \lhd\!\!\!- \{ (qi \mapsto xx) \mapsto qf\}$

END

;

remove_transition(qi , xx , qf) =

PRE

$qi \in STATE \wedge qi \in Qqd$

$\wedge \quad xx \in SYMBOL$

$\wedge \quad xx \in Xad$

$\wedge \quad qf \in STATE$

$\wedge \quad qf \in Qqd$

$\wedge \quad (qi \mapsto xx) \in \mathbf{dom}\ (deltad \rhd \{qf\})$

THEN

$deltad := deltad$ - $\{(qi \mapsto xx) \mapsto qf\}$

$||$ $consistentAutom := NOK$

END

;

modify_transition(qi, xx, qfa, qfn) =

PRE

$qi \in STATE$

$\wedge \quad qi \in Qqd$

$\wedge \quad xx \in SYMBOL$

$\wedge \quad xx \in Xad$

$\wedge \quad qfa \in STATE \wedge qfn \in STATE$

$\wedge \quad qfa \in Qqd$

$\wedge \quad qfn \in Qqd$

$\wedge \quad (\ qi \mapsto xx\ ) \mapsto qfa \in deltad$

$\wedge \quad (\ qi \mapsto xx\ ) \mapsto qfn \notin deltad$

THEN
$deltad := (deltad$ - $\{(qi \mapsto xx) \mapsto qfa\}) \lhd\!\!\!- \{\ (qi \mapsto xx) \mapsto qfn\}$
$||\quad consistentAutom := NOK$
END
;
$si,xi,sf \leftarrow \quad get\_transition(ii) =$
PRE
$ii \in \mathbf{NAT} \wedge ii > 0 \wedge ii < nbTransitions$
THEN
ANY ssi, xxi, ssf **WHERE**
$ssi \in STATE \wedge ssi \in Qqd$
$\wedge\ xxi \in SYMBOL \wedge xxi \in Xad$
$\wedge\ ssf \in STATE \wedge ssf \in Qqd$
THEN
$si := ssi\ ||\ xi := xxi\ ||\ sf := ssf$
END
END

;
check_consistent =

ANY
qq
WHERE
$qq \in STATE \wedge qq \in Qqd$
THEN
IF $qq \in \mathbf{dom}(\mathbf{dom}(deltad)) \cup \mathbf{ran}(deltad)$
THEN
$consistentAutom := OK$
ELSE
$consistentAutom := NOK$
END
END
;
$res \leftarrow$ **direct_successor**$(qq,\ aa) =$

PRE
$qq \in STATE$
$\wedge \quad aa \in SYMBOL$
$\wedge \quad qq \in Qqd$
$\wedge \quad aa \in Xad$
$\wedge \quad (qq \mapsto aa) \in \mathbf{dom}\ (deltad)$

THEN
$res := deltad(qq \mapsto aa)$
END
;
$res \leftarrow$ **isInitialState**$(st) =$

PRE
$st \in STATE \wedge st \in Qqd$
THEN
$res := \mathbf{bool}(st \in Q0d)$
END
;
$res \leftarrow \mathbf{isFinalState}(st) =$
PRE
$st \in STATE \wedge st \in Qqd$
THEN
$res := \mathbf{bool}(st \in Qfd)$
END
;
$st \leftarrow \mathbf{getInitial} =$
ANY qq
WHERE
$qq \in STATE \wedge qq \in Q0d$
THEN
$st := qq$
END
;
setDummyState(ds)=
PRE
$ds \in STATE$
THEN
$dummyState := ds$
END
;
$res \leftarrow \mathbf{isTransition}(qq, xx) =$
PRE
$qq \in STATE \quad \wedge \quad qq \in Qqd$
$\wedge \quad xx \quad \in SYMBOL \wedge \quad xx \in Xad$
THEN
$res := \mathbf{bool}((qq \mapsto xx) \in \mathbf{dom}\ (deltad))$
END

END

3.4 Composant AUTOMATON_I

3.4.1 Présentation

Le composant traité dans ce paragraphe est : AUTOMATON_I.

3.4.2 Métrique

AUTOMATON_I TypeChecked /home/enseignants/attiogbe/SPEC/DESS/B/TermAnalyser/sources/AUTO

3.4.3 Règles ajoutées à la preuve

Fichier /home/enseignants/attiogbe/SPEC/DESS/B/TermAnalyser/bdp/AUTOMATON_I.pmm non existant.

3.4.4 Source B

```
/*-------------------------------------------*/
/*  Projet   : NaBLa (Nantes B Libraries)                  */
/*  Dossier  : Automata / Analyseur d'expressions          */
/*  C. Attiogbe, 2000-...                                  */
/*-------------------------------------------*/
IMPLEMENTATION
      AUTOMATON_I  /* Determ. Finite State Automata */

REFINES
      AUTOMATON_R

SEES
      CtxAUT    /* sees STATE, SYMBOL, TRANSITION */
,   BASIC_IO

IMPORTS
   a1.AuxAUT
,      XadSet.L_SET(nbSymbols,SYMBOL)
,      QqdSet.L_SET(nbStates,STATE)
,      Q0dSet.L_SET(1,STATE)
,      QfdSet.L_SET(nbStates,STATE)
   /*     implementing transitions */
,   STi_transitions.L_SEQUENCE(nbTransitions,STATE)
,   SBL_transitions.L_SEQUENCE(nbTransitions,SYMBOL)
,   STf_transitions.L_SEQUENCE(nbTransitions,STATE)
,   rtransitions.L_SEQUENCE(nbTransitions,TRANSITION)
,   rdeltad.L_SEQUENCE(nbTransitions,STATE)

,      currentSymbols.L_SET(nbSymbols,SYMBOL)
,      connectedStates.L_SET(nbStates,STATE)

CONCRETE_VARIABLES
   lastTransition
,      consistentAutom
,   dummyState

INVARIANT
   lastTransition ∈ 0 .. 100
∧     dummyState ∈ STATE
∧       ran(XadSet.set_vrb)   = Xad
∧       ran(QqdSet.set_vrb)   = Qqd
∧       ran(Q0dSet.set_vrb)   = Q0d
∧       ran(QfdSet.set_vrb)   = Qfd
∧       consistentAutom ∈ ConsistentSet
∧     dom(transitions) ⊆ ran(STi_transitions.seq_vrb)
∧     ran(transitions) ⊆ ran(SBL_transitions.seq_vrb)
∧     ran(rdeltad) ⊆ ran(STf_transitions.seq_vrb)

∧     card(rdeltad) = size(STi_transitions.seq_vrb)
∧     card(rdeltad) = size(STf_transitions.seq_vrb)

INITIALISATION
```

```
    lastTransition := 0
;   dummyState := dums
;     consistentAutom := NOK
;   rtransitions.CLR_SEQ
;   rdeltad.CLR_SEQ
;   XadSet.CLR_SET
;   QqdSet.CLR_SET
;   Q0dSet.CLR_SET

OPERATIONS
    clear_automaton =
    BEGIN
    rtransitions.CLR_SEQ
;   rdeltad.CLR_SEQ
;   XadSet.CLR_SET
;   QqdSet.CLR_SET
;   Q0dSet.CLR_SET

    END
;

        add_state(qq) =
        VAR inqqd, ii IN
            inqqd, ii ← QqdSet.FIND_SET(qq);
                IF inqqd = FALSE
                THEN
                    VAR bb IN
                        bb ← QqdSet.IS_FULL_SET;
                        IF  bb = FALSE
                        THEN
                            QqdSet.INS_SET(qq);
                            consistentAutom := NOK
                        END
                    END
                ELSE
                    STRING_WRITE("\nERREUR: qq est déjà un état de l'automate \n")
                END
        END

;

        remove_state(qq) =
        VAR inqqd, ii IN
            inqqd, ii ← QqdSet.FIND_SET(qq);
            IF inqqd = TRUE
            THEN
                QqdSet.RMV_SET(qq)
            ELSE
                STRING_WRITE("\nERREUR: qq n'est pas un état de l'automate \n")
            END
        END
;
        add_symbol(sx) =
```

```
        VAR  inXad, ii IN
            inXad, ii ← XadSet.FIND_SET(sx);
              IF inXad = FALSE
              THEN
                 VAR bb IN
                    bb ← XadSet.IS_FULL_SET;
                    IF  bb = FALSE
                    THEN
                       XadSet.INS_SET(sx)
                    END
                 END

              END
        END
;

        remove_symbol(xx) =
        VAR inXad, ii IN
           inXad, ii ← XadSet.FIND_SET(xx);
           IF inXad = TRUE
           THEN
              XadSet.RMV_SET(xx)
           ELSE
              STRING_WRITE("\nERREUR: xx is not a known symbol \n")
           END
        END
;

        add_initialState(ii) =
        VAR inqqd, inq0d, jj, kk, tt IN
           inqqd, jj ← QqdSet.FIND_SET(ii);
           inq0d, kk ← Q0dSet.FIND_SET(ii);
           tt ← Q0dSet.CARD_SET;
           IF inqqd = TRUE
           THEN
              IF tt = 0
              THEN
                 Q0dSet.INS_SET(ii)
              ELSE
                 STRING_WRITE("\nERREUR: il existe déjà un état initial \n")
              END
           ELSE
              STRING_WRITE("\nERREUR: ii n'est pas un état de l'automate \n")
           END
        END
;

          remove_initialState(ii) =
        VAR inqqd, inq0d, jj, kk, tt IN
           inqqd, jj ← QqdSet.FIND_SET(ii);
           inq0d, kk ← Q0dSet.FIND_SET(ii);
           tt ← Q0dSet.CARD_SET;
           IF inqqd = TRUE
```

```
    THEN
      IF tt = 1
      THEN
        IF inq0d = TRUE
        THEN
          Q0dSet.RMV_SET(ii)
        ELSE
          STRING_WRITE("\nERREUR: ii n'est pas l'état initial \n")
        END
      ELSE
        STRING_WRITE("\nERREUR: il n'existe pas d'état initial \n")
      END
    ELSE
      STRING_WRITE("\nERREUR: ii n'est pas un état de l'automate \n")
    END
  END
;

  add_finalState(ff) =
  VAR inqqd, inqfd, jj, kk IN
    inqqd, jj ← QqdSet.FIND_SET(ff);
    inqfd, kk ← QfdSet.FIND_SET(ff);
    IF inqqd = TRUE
    THEN
      IF inqfd = FALSE
      THEN
        VAR bb IN
          bb ← QfdSet.IS_FULL_SET;
          IF  bb = FALSE
          THEN
            QfdSet.INS_SET(ff)
          END
        END

      END
    ELSE
      STRING_WRITE("\nERREUR: ff n'est pas un état de l'automate \n")
    END
  END
;

  remove_finalState(ff) =
  VAR inqqd, inqfd, jj, kk IN
    inqqd, jj ← QqdSet.FIND_SET(ff);
    inqfd, kk ← QfdSet.FIND_SET(ff);
    IF inqqd = TRUE
    THEN
      IF inqfd = TRUE
      THEN
        QfdSet.RMV_SET(ff)
      ELSE
        STRING_WRITE("\nERREUR: ff n'est pas un état final \n")
      END
```

```
        ELSE
           STRING_WRITE("\nERREUR: ff n'est pas un état de l'automate \n")
        END
      END
;
      add_transition(qi, xx, qf) =
      VAR inqqdi, inqqdf, inxad,
       indqi, indqf, indxx,
       inConnected, b0
    IN
       inqqdi, indqi ← QqdSet.FIND_SET(qi);
       inqqdf, indqf ← QqdSet.FIND_SET(qf);
       inxad, indxx  ← XadSet.FIND_SET(xx);
       IF inqqdi = FALSE
     THEN
              STRING_WRITE("\nERREUR: Unknown qi \n")
    ELSE
        IF inqqdf = FALSE
        THEN
              STRING_WRITE("\nERREUR: Unknown qf \n")
     ELSE
       IF inxad = FALSE
     THEN
              STRING_WRITE("\nERREUR: Unknown xx \n")
        ELSE
     VAR newt, it IN
      it := 1;
         b0, newt ← a1.newTransition
              ; b0 ← rtransitions.IS_FULL_SEQ;
            IF b0 = FALSE
          THEN
           rtransitions.PUSH_SEQ(newt) ;
           b0, it ← rtransitions.FIND_FIRST_SEQ(newt);
           STi_transitions.STR_SEQ(it,qi) ;
           SBL_transitions.STR_SEQ(it,xx) ;
           STf_transitions.STR_SEQ(it,qf) ;
           rdeltad.STR_SEQ(it, newt)
            ELSE
              STRING_WRITE("\nERREUR: Transitions Structrure FULL \n")
            END
     END
         END
       END
    END
    ;
    VAR inSbl IN
    inSbl , indxx ← currentSymbols.FIND_SET(xx);
    b0 ← currentSymbols.IS_FULL_SET;
    IF inSbl = FALSE
     THEN
      IF b0 = FALSE
       THEN
        currentSymbols.INS_SET(xx)
```

```
    ELSE
      STRING_WRITE("\nERREUR: Current symbol list is FULL \n")
    END
  END
  END
  ;
  inConnected , indxx ← connectedStates.FIND_SET(qi);
  b0 ← connectedStates.IS_FULL_SET;
  IF inConnected = FALSE
    THEN
     IF b0 = FALSE
       THEN
         connectedStates.INS_SET(qi)
       ELSE
         STRING_WRITE("\nERREUR: ConnectedStates is FULL \n")
     END
  END
  ;
     inConnected, indxx ← connectedStates.FIND_SET(qf);
  b0 ← connectedStates.IS_FULL_SET;
  IF inConnected = FALSE
    THEN
      IF b0 = FALSE
      THEN
        connectedStates.INS_SET(qf)
      ELSE
        STRING_WRITE("\nERREUR: ConnectedStates is FULL \n")
      END
       ELSE
         STRING_WRITE("\nERREUR: la transition existe déjà \n")
      END
  END
;
     remove_transition(qi, xx, qf) =


  VAR inqqdi, inqqdf, inxad,
      indqi, indqf, indxx
  IN
       inqqdi, indqi ← QqdSet.FIND_SET(qi);
       inqqdf, indqf ← QqdSet.FIND_SET(qf);
       inxad, indxx  ← XadSet.FIND_SET(xx);
       IF inqqdi = FALSE
    THEN
            STRING_WRITE("\nERREUR: Unknown qi \n")
    ELSE
        IF inqqdf = FALSE
        THEN
            STRING_WRITE("\nERREUR: Unknown qf \n")
     ELSE
       IF inxad = FALSE
     THEN
            STRING_WRITE("\nERREUR: Unknown xx \n")
```

```
        ELSE
      VAR seql, ii, is,sti, bb,sbl, rmv IN

       seql ← STi_transitions.LEN_SEQ;
       bb,is ← STi_transitions.FIND_FIRST_SEQ(qi);
       sti ← STi_transitions.VAL_SEQ(is);
       sbl ← SBL_transitions.VAL_SEQ(is);
       ii := is;
       rmv := FALSE;
      WHILE ii < seql DO
       sti ← STi_transitions.VAL_SEQ(ii);
       sbl ← SBL_transitions.VAL_SEQ(ii);
      IF (sti = qi) ∧ (sbl = xx)
      THEN
        rtransitions.RMV_SEQ(ii);
        STi_transitions.RMV_SEQ(ii);
        SBL_transitions.RMV_SEQ(ii);
          STf_transitions.RMV_SEQ(ii);
        rdeltad.RMV_SEQ(ii);
        rmv := TRUE
      ELSE
        ii := ii+1
      END
      INVARIANT
         ii ∈ 1 .. seql
      VARIANT
         seql - ii
      END
      ;
      IF rmv = FALSE
      THEN
       sti ← STi_transitions.VAL_SEQ(ii);
       sbl ← SBL_transitions.VAL_SEQ(ii);
        IF (sti = qi) ∧ (sbl = xx)
       THEN
        rtransitions.RMV_SEQ(ii);
        STi_transitions.RMV_SEQ(ii);
        SBL_transitions.RMV_SEQ(ii);
          STf_transitions.RMV_SEQ(ii)
       END
      END
        END
       END
      END
     END
      END
;


      modify_transition(qi, xx, qfa, qfn) =
      VAR inqqdi, inqqdfa, inqqdfn, inxad,
         inSbl, indqi, indqfa, indqfn, indxx, jj
    IN
```

```
        inqqdi, indqi  ← QqdSet.FIND_SET(qi);
        inqqdfa, indqfa ← QqdSet.FIND_SET(qfa);
        inqqdfn, indqfn ← QqdSet.FIND_SET(qfn);
        inxad, jj ← XadSet.FIND_SET(xx);
        inSbl , indxx ← currentSymbols.FIND_SET(xx);
        IF inqqdfa = TRUE
         THEN
          IF inxad = TRUE
           THEN
     skip
           ELSE
             STRING_WRITE("\nERREUR: qfa is not a known state \n")
       END
         END
       END
;
si,xi,sf ←     get_transition (ii)=
   BEGIN
      si ← STi_transitions.VAL_SEQ(ii);
      xi ← SBL_transitions.VAL_SEQ(ii);
      sf ← STi_transitions.VAL_SEQ(ii)
   END
;


      check_consistent =
      VAR nbState , cpt IN
         nbState ← QqdSet.CARD_SET;
         cpt := 0;
         consistentAutom := OK;
         WHILE cpt < nbState DO
            VAR bb ,ii , qq IN
               cpt := cpt + 1;
               qq ← QqdSet.VAL_SET(cpt);
               bb, ii ← connectedStates.FIND_SET(qq);
               IF bb = FALSE
               THEN
                  consistentAutom := NOK
               END
            END

         INVARIANT
            cpt ∈ 0 .. nbState

         VARIANT
      nbState - cpt
         END
      END
;

res ← direct_successor(qq, aa) =

   VAR inqqdi, inxad, indqi, indxx
```

```
    IN
        inqqdi, indqi ← QqdSet.FIND_SET(qq);
        inxad, indxx  ← XadSet.FIND_SET(aa);
      res := qq;
        IF inqqdi = FALSE
      THEN
              STRING_WRITE("\nERREUR: Unknown qi \n")
      ELSE
        IF inxad = FALSE
      THEN
              STRING_WRITE("\nERREUR: Unknown xx \n")
         ELSE
      VAR seql, ii, is,sti, bb,sbl IN

       seql ← STi_transitions.LEN_SEQ;
       bb,is ← STi_transitions.FIND_FIRST_SEQ(qq);
       sti ← STi_transitions.VAL_SEQ(is);
       sbl ← SBL_transitions.VAL_SEQ(is);
       ii := is;
      WHILE ii < seql DO
       sti ← STi_transitions.VAL_SEQ(ii);
       sbl ← SBL_transitions.VAL_SEQ(ii);
      IF (sti = qq) ∧ (sbl = aa)
      THEN
        res ← STf_transitions.VAL_SEQ(ii)
      ELSE
         ii := ii+1
      END
      INVARIANT
         ii ∈ 1 .. seql
      VARIANT
         seql - ii
      END
      END
       END
      END
     END

;
res ← isInitialState(st) =
    VAR ii IN
    res, ii ← Q0dSet.FIND_SET(st)
    END

;
res ← isFinalState(st) =
    VAR ii IN
    res, ii ← QfdSet.FIND_SET(st)
    END
;
st ← getInitial =
```

```
    st ← Q0dSet.VAL_SET(1)

;   setDummyState(ds)=
    BEGIN
    dummyState := ds
    END
;
res ← isTransition(qq, xx) =
    VAR inqqd, inxad, indqq , ll
    IN
          inqqd, indqq ← QqdSet.FIND_SET(qq);
          inxad, ll ← XadSet.FIND_SET(xx);
       res := FALSE;
          IF inqqd = TRUE
          THEN
             IF inxad = TRUE
             THEN
       VAR vq IN
                vq := TRUE
       END
       ELSE
          res := FALSE
       END
       ELSE
          res := FALSE
       END
        END

END
```

3.5 Composant AUTOMATON_R

3.5.1 Présentation

Le composant traité dans ce paragraphe est : AUTOMATON_R.

3.5.2 Métrique

AUTOMATON_R POGenerated /home/enseignants/attiogbe/SPEC/DESS/B/TermAnalyser/sources/AUT

	NbObv	NbPO	NbPRi	NbPRa	NbUn	%Pr
Initialisation	6	7	0	7	0	100
clear_automaton	8	7	0	7	0	100
add_state	17	5	0	5	0	100
remove_state	17	8	0	8	0	100
add_symbol	18	3	0	3	0	100
remove_symbol	17	4	0	4	0	100
add_initialState	21	1	0	1	0	100
remove_initialState	20	3	0	3	0	100
add_finalState	22	1	0	1	0	100

	NbObv	NbPO	NbPRi	NbPRa	NbUn	%Pr
remove_finalState	20	2	0	2	0	100
add_transition	24	5	0	4	1	80
remove_transition	21	5	0	2	3	40
modify_transition	24	6	0	3	3	50
get_transition	30	0	0	0	0	100
check_consistent	36	2	0	0	2	0
direct_successor	23	2	0	2	0	100
isInitialState	20	1	0	1	0	100
isFinalState	20	1	0	1	0	100
setDummyState	20	0	0	0	0	100
isTransition	23	1	0	0	1	0
getInitial	22	0	0	0	0	100
AUTOMATON_R	429	64	0	54	10	84

3.5.3 Règles ajoutées à la preuve

Fichier /home/enseignants/attiogbe/SPEC/DESS/B/TermAnalyser/bdp/AUTOMATON_R.pmm non existant.

3.5.4 Source B

/*——————————————————————————*/
/* Projet : NaBLa (Nantes B Libraries) */
/* Dossier : Automata / Analyseur d'expressions */
/* C. Attiogbe, 2000-... */
/*——————————————————————————*/

REFINEMENT
AUTOMATON_R

SEES
CtxAUT /* sees STATE, SYMBOL, etc */

REFINES
AUTOMATON

VARIABLES
Xad
, *Qqd*
, *Q0d*
, *Qfd*
, *rdeltad*
, *consistentAutom*
, *transitions*
, *dummyState*

INVARIANT
$Xad \in \mathcal{F}(SYMBOL)$
$\wedge$ $Qqd \subseteq STATE$
$\wedge$ $Q0d \subseteq Qqd$
$\wedge$ $\mathbf{card}(Q0d) \leq 1$
$\wedge$ $Qfd \subseteq Qqd$
$\wedge$ $transitions \subseteq Qqd \times Xad$
$\wedge$ $rdeltad \in transitions \rightarrow Qqd$
$\wedge$ $transitions = \mathbf{dom}(deltad)$

$\wedge$ $\mathbf{ran}(rdeltad) = \mathbf{ran}(deltad)$
$\wedge$ $deltad = rdeltad$ /* refinement if transition */
$\wedge$ $consistentAutom \in ConsistentSet$
$\wedge$ $dummyState \in STATE$

INITIALISATION
$Xad, Qqd, Q0d, Qfd, rdeltad, consistentAutom, transitions := \emptyset , \emptyset , \emptyset , \emptyset , \emptyset , NOK, \emptyset$
|| $dummyState :\in STATE$

OPERATIONS
clear_automaton =
BEGIN
$Xad := \emptyset$
|| $Qqd := \emptyset$
|| $Q0d := \emptyset$
|| $Qfd := \emptyset$
|| $rdeltad := \emptyset$
|| $transitions := \emptyset$
END
;

add_state(qq) =
PRE
$qq \in STATE$
$\wedge$ $qq \notin Qqd$
THEN
$Qqd := Qqd \cup \{qq\}$
|| $consistentAutom := NOK$
END
;

remove_state(qq) =
PRE
$qq \in STATE$
$\wedge$ $qq \in Qqd$
$\wedge$ $qq \notin \mathbf{dom}(transitions)$
$\wedge$ $qq \notin \mathbf{ran}(rdeltad)$
$\wedge$ $qq \notin Qfd$
$\wedge$ $qq \notin Q0d$
THEN
$Qqd := Qqd$ - $\{qq\}$
END
;
add_symbol(sx) =
PRE
$sx \in SYMBOL$
$\wedge$ $sx \notin Xad$
THEN
$Xad := Xad \cup \{sx\}$
END
;

remove_symbol(xx) =

PRE
$xx \in SYMBOL$
$\wedge \quad xx \notin \mathbf{ran}(transitions)$
THEN
$Xad := Xad$ - $\{xx\}$
END
;

add_initialState$(ii) =$
PRE
$ii \in STATE \wedge ii \in Qqd$
$\wedge \quad ii \notin Q0d$

THEN
$Q0d := \{ii\}$
END
;

remove_initialState$(ii) =$
PRE
$ii \in STATE \wedge ii \in Qqd$
$\wedge \quad ii \in Q0d$
$\wedge \quad \mathbf{card}(Q0d) = 1$

THEN
$Q0d := Q0d$ - $\{ii\}$
END
;

add_finalState$(ff) =$
PRE
$ff \quad \in STATE \wedge ff \in Qqd$
$\wedge \quad ff \notin Qfd$
THEN
$Qfd := Qfd \cup \{ff\}$
END
;

remove_finalState$(ff) =$
PRE
$ff \in STATE \wedge ff \in Qqd$
$\wedge \quad ff \in Qfd$
THEN
$Qfd := Qfd$ - $\{ff\}$
END
;

add_transition$(qi, xx, qf) =$
PRE
$qi \in STATE \wedge qf \in STATE$
$\wedge \quad qi \in Qqd$
$\wedge \quad xx \in SYMBOL$

$\wedge \quad xx \in Xad$
$\wedge \quad qf \in Qqd$
$\wedge \quad (qi \mapsto xx) \notin transitions$
$\wedge \quad (qi \mapsto xx) \notin \mathbf{dom}(rdeltad \rhd \{qf\})$

THEN
$transitions := transitions \cup \{qi \mapsto xx\}$
$||$ $rdeltad := rdeltad \lhd\!\!\!- \{ (qi \mapsto xx) \mapsto qf\}$
END
;
remove_transition$(qi\ ,\ xx\ ,\ qf) =$
PRE
$qi \in STATE \wedge qi \in Qqd$
$\wedge \quad xx \in SYMBOL$
$\wedge \quad xx \in Xad$
$\wedge \quad qf \in STATE$
$\wedge \quad qf \in Qqd$
$\wedge \quad (qi\ ,\ xx) \in \mathbf{dom}\ (rdeltad \rhd \{qf\})$
THEN
$transitions := transitions$ - $\{qi \mapsto xx\}$
$||$ $rdeltad := rdeltad$ - $\{(qi \mapsto xx) \mapsto qf\}$
$||$ $consistentAutom := NOK$
END
;

modify_transition$(qi,\ xx,\ qfa,\ qfn\ ) =$
PRE
$qi \in STATE$
$\wedge \quad qi \in Qqd$
$\wedge \quad xx \in SYMBOL$
$\wedge \quad xx \in Xad$
$\wedge \quad qfa \in STATE \wedge qfn \in STATE$
$\wedge \quad qfa \in Qqd$
$\wedge \quad qfn \in Qqd$
$\wedge \quad (qi,\ xx) \in transitions$
$\wedge \quad (\ qi \mapsto xx\ ) \mapsto qfa \in rdeltad$

$\wedge \quad (\ qi \mapsto xx\ ) \mapsto qfn \notin rdeltad$

THEN
$rdeltad := (rdeltad$ - $\{(qi\ ,\ xx) \mapsto qfa\}) \lhd\!\!\!- \{\ (qi,\ xx) \mapsto qfn\}$
$||$ $consistentAutom := NOK$
END
;
$si,xi,sf \leftarrow get\_transition\ (ii) =$
PRE $ii \in \mathbf{NAT} \wedge ii > 0 \wedge ii < nbTransitions$
THEN
ANY $ssi,\ xxi,\ ssf$
WHERE
$ssi \in STATE \wedge ssi \in Qqd$

```
    ∧ xxi ∈ SYMBOL ∧ xxi ∈ Xad
    ∧ ssf ∈ STATE ∧ ssf ∈ Qqd
    ∧ (ssi ↦ xxi) ∈ transitions
    ∧ ssf = rdeltad(ssi ↦ xxi)
  THEN
  si := ssi || xi := xxi || sf := ssf
  END
  END

;


    check_consistent =
    ANY
      qq
    WHERE
      qq ∈ STATE ∧ qq ∈ Qqd
    THEN
      IF qq ∈ dom(transitions) ∪ ran(rdeltad)
      THEN
        consistentAutom := OK
      ELSE
        consistentAutom := NOK
      END
    END
;

res ← direct_successor(qq, aa) =
    PRE
      qq ∈ STATE
  ∧   aa  ∈ SYMBOL
    ∧   qq  ∈ Qqd
    ∧   aa  ∈ Xad
    ∧   (qq ↦ aa) ∈ transitions

    THEN
      res := rdeltad(qq ↦ aa)
    END
;
res ← isInitialState(st) =
  PRE
  st ∈ STATE
  THEN
    res := bool(st ∈ Q0d)
  END
;
res ← isFinalState(st) =
  PRE
  st ∈ STATE
  THEN
    res := bool(st ∈ Qfd)
  END
;
```

setDummyState(*ds*)=
PRE
ds ∈ *STATE*
THEN
dummyState := *ds*
END
;
res ← **isTransition**(*qq*, *xx*) =
PRE
qq ∈ *STATE* ∧ *qq* ∈ *Qqd*
∧ *xx* ∈ *SYMBOL* ∧ *xx* ∈ *Xad*
THEN
res := **bool**((*qq*, *xx*) ∈ *transitions*)
END

END

3.6 Composant AppliAnalyser

3.6.1 Présentation

Le composant traité dans ce paragraphe est : AppliAnalyser.

3.6.2 Métrique

AppliAnalyser AutoProved /home/enseignants/attiogbe/SPEC/DESS/B/TermAnalyser/sources/AppliAnaly

	NbObv	NbPO	NbPRi	NbPRa	NbUn	%Pr
Initialisation	1	0	0	0	0	100
main	2	0	0	0	0	100
AppliAnalyser	3	0	0	0	0	100

3.6.3 Règles ajoutées à la preuve

Fichier /home/enseignants/attiogbe/SPEC/DESS/B/TermAnalyser/bdp/AppliAnalyser.pmm non existant.

3.6.4 Source B

MACHINE
AppliAnalyser
OPERATIONS
main = **skip**
END

3.7 Composant AppliAnalyser_I

3.7.1 Présentation

Le composant traité dans ce paragraphe est : AppliAnalyser_I.

3.7.2 Métrique

AppliAnalyser_I AutoProved /home/enseignants/attiogbe/SPEC/DESS/B/TermAnalyser/sources/AppliAn

	NbObv	NbPO	NbPRi	NbPRa	NbUn	%Pr
ValuesLemmas	1	0	0	0	0	100
InstanciatedConstraintsLemmas	0	0	0	0	0	100
Initialisation	2	0	0	0	0	100
main	46	72	0	72	0	100
AppliAnalyser_I	49	72	0	72	0	100

3.7.3 Règles ajoutées à la preuve

Fichier /home/enseignants/attiogbe/SPEC/DESS/B/TermAnalyser/bdp/AppliAnalyser_I.pmm non existant.

3.7.4 Source B

IMPLEMENTATION
AppliAnalyser_I

REFINES
AppliAnalyser
/* alraedy impotrted in INTERFACE */
IMPORTS
INTERFACE
, *CtxAUT*
, *CtxAnalyser*
, *BASIC_IO*

OPERATIONS
main =
BEGIN
STRING_WRITE("\t.....Nabla Project .... Analyser ....\n");
VAR *rr*, *xx* **IN**
xx := 20
; *rr* := 1
; **WHILE** (*xx* > 0)
DO
STRING_WRITE("\t+————————————————+\n")
; **STRING_WRITE**("\t+ Analyser: Application Menu +\n")
; **STRING_WRITE**("\t+————————————————+\n")
; **STRING_WRITE**("\t New Automaton : 1\n")
; **STRING_WRITE**("\t New Lexic : 2\n")
; **STRING_WRITE**("\t New Expression (seq of words) : 3\n")
; **STRING_WRITE**("\t Analyse Current Expression : 4\n")
; **STRING_WRITE**("\t Display Current Automaton : 5\n")
; **STRING_WRITE**("\t Display Current Expression) : 6\n")
; **STRING_WRITE**("\t+————————————————+ \n")
; **STRING_WRITE**("\t Quit : 0 + \n")
; **STRING_WRITE**("\t+————————————————+ \n\n")

```
    ;  STRING_WRITE("\t choix ? ")

        ;  rr ← INTERVAL_READ(0,6)
        ;  CASE rr OF
           EITHER 0 THEN

                  xx := 1
           OR  1 THEN

                  buildAnAutomaton
           OR  2 THEN

                  buildLexic
           OR  3 THEN

                  inputAnExpression
           OR  4 THEN

                  analyseTerm
           OR  5 THEN

                  displayAutomaton
           OR  6 THEN

                  displayExpression
              END
           END


        ;  STRING_WRITE(" \n")
        ;  STRING_WRITE(" \n")

    ;  xx := xx - 1
        INVARIANT
              xx ∈ NAT
 ∧      xx ∈ 0 .. 20
        VARIANT
              xx
        END
    END

  END
END
```

3.8 Composant AuxAUT

3.8.1 Présentation

Le composant traité dans ce paragraphe est : AuxAUT.

3.8.2 Métrique

AuxAUT AutoProved /home/enseignants/attiogbe/SPEC/DESS/B/TermAnalyser/sources/AuxAUT.mch

	NbObv	NbPO	NbPRi	NbPRa	NbUn	%Pr
Initialisation	0	2	0	2	0	100
newState	4	0	0	0	0	100
newTransition	4	0	0	0	0	100
Int2Sstate	3	0	0	0	0	100
nat2WORD	3	0	0	0	0	100
AuxAUT	14	2	0	2	0	100

3.8.3 Règles ajoutées à la preuve

Fichier /home/enseignants/attiogbe/SPEC/DESS/B/TermAnalyser/bdp/AuxAUT.pmm non existant.

3.8.4 Source B

```
MACHINE
    AuxAUT        /* Auxilliary operations on AUTOMATON */
SEES
    CtxAUT        /*    STATES,SYMBOL,TRANSITION */

VARIABLES
    States
,     Transitions

INVARIANT
    States <: STATE
&     Transitions <: TRANSITION

INITIALISATION

    States := {}
||    Transitions := {}

OPERATIONS
bb, st <– newState =
    ANY ss WHERE
    ss : STATE - States
    THEN
    st := ss
    || States := States \/ {ss}
    || bb :: BOOL
    END
;
bb, tr <– newTransition =
    ANY trt WHERE
    trt : TRANSITION
    THEN
    tr := trt
    || Transitions := Transitions \/ {trt}
    || bb :: BOOL
    END
;
res <– Int2Sstate (ii) =
    PRE
```

ii : INT
THEN
res :: STATE
END
;
res <– nat2WORD(vv) = /∗? necessary to perform typeconversion from input ∗/
PRE $vv \in$ **NAT**
$\wedge$ $vv \leq 9$
THEN
ANY *rr* **WHERE**
$rr \in WORD$
THEN $res := rr$
END
END
END

3.9 Composant AuxAUT_I

3.9.1 Présentation

Le composant traité dans ce paragraphe est : AuxAUT_I.

3.9.2 Métrique

AuxAUT_I POGenerated /home/enseignants/attiogbe/SPEC/DESS/B/TermAnalyser/sources/AuxAUT_I.

	NbObv	NbPO	NbPRi	NbPRa	NbUn	%Pr
ValuesLemmas	1	0	0	0	0	100
InstanciatedConstraintsLemmas	1	6	0	4	2	66
Initialisation	7	45	0	32	13	71
newState	21	24	0	16	8	66
newTransition	21	19	0	9	10	47
Int2Sstate	6	1	0	0	1	0
nat2WORD	86	1	0	0	1	0
AuxAUT_I	143	96	0	61	35	63

3.9.3 Règles ajoutées à la preuve

Fichier /home/enseignants/attiogbe/SPEC/DESS/B/TermAnalyser/bdp/AuxAUT_I.pmm non existant.

3.9.4 Source B

IMPLEMENTATION
AuxAUT_I

REFINES
AuxAUT

SEES
CtxAUT

IMPORTS

$States.L\_SET(nbStates, STATE)$
, $freeStates.L\_SET(nbStates, STATE)$
, $Transitions.L\_SET(nbTransitions, TRANSITION)$
, $freeTransitions.L\_SET(nbTransitions, TRANSITION)$

INVARIANT
$States = \mathbf{ran}(States.set\_vrb)$
$\wedge$ $\mathbf{ran}(freeStates.set\_vrb) = STATE$ - $\mathbf{ran}(States.set\_vrb)$

$\wedge$ $Transitions = \mathbf{ran}(Transitions.set\_vrb)$
$\wedge$ $\mathbf{ran}(freeTransitions.set\_vrb) = TRANSITION$ - $\mathbf{ran}(Transitions.set\_vrb)$

INITIALISATION
$States.CLR\_SET$
; $Transitions.CLR\_SET$
; **VAR** $ss1$ **IN**
$ss1 := minSTATES$
; **WHILE** $(ss1 < maxSTATES)$ **DO**
freeStates.INS_SET$(ss1)$
; $ss1 := ss1 + 1$
INVARIANT
$ss1 \in STATE$
$\wedge$ $\mathbf{card}(\mathbf{ran}(freeStates.set\_vrb)) \leq maxSTATES$
VARIANT
$maxSTATES$-$ss1$
END
END
;
VAR tt **IN**
$tt := minTransitions$
; **WHILE** $(tt < maxTransitions)$ **DO**
freeTransitions.INS_SET(tt)
; $tt := tt + 1$
INVARIANT
$tt \in TRANSITION$
VARIANT
$maxTransitions$-tt
END
END

OPERATIONS
$bb, st \leftarrow newState =$
VAR bl **IN**
$bl \leftarrow$ **States.IS_FULL_SET**
; **IF** $bl =$ **TRUE**
THEN
$bb :=$ **FALSE**
; $st := 0$
ELSE
$st \leftarrow$ **freeStates.VAL_SET**(1)
; $bb :=$ **TRUE**
; **freeStates.RMV_SET**(st)
; **States.INS_SET**(st)

```
    END
    END
;

bb, tr ← newTransition =
    VAR br IN
    br ← Transitions.IS_FULL_SET
    ; IF br = TRUE
    THEN
        bb := FALSE
    ;    tr := 0
    ELSE
        tr ← freeStates.VAL_SET(1)
    ;    bb := TRUE
    ;    freeTransitions.RMV_SET(tr)
    ;     Transitions.INS_SET(tr)
    END
    END
;
res ← Int2Sstate (ii) =
    BEGIN
    res := ii
    END
;
res ← nat2WORD(vv) =
    BEGIN
     CASE vv OF
                EITHER 1 THEN
           res :=  le
                OR  2 THEN

           res := la
                OR  3 THEN

                      res := un
                OR  10 THEN

           res := ciel
                OR  11 THEN

           res := terre
                OR  12 THEN

           res := chien
                OR  13 THEN

           res :=  chat
                OR  14 THEN

           res := oiseau
                OR  40 THEN

           res := est
```

OR 41 **THEN**

res := *possede*
OR 42 **THEN**

res := *vole*
OR 43 **THEN**

res := *court*
OR 70 **THEN**

res := *gris*
OR 80 **THEN**

res := *medor*
OR 81 **THEN**

res := *nenette*
OR 100 **THEN**

res := *point*
END
END
END
END

3.10 Composant CtxAUT

3.10.1 Présentation

Le composant traité dans ce paragraphe est : CtxAUT.

3.10.2 Métrique

CtxAUT AutoProved /home/enseignants/attiogbe/SPEC/DESS/B/TermAnalyser/sources/CtxAUT.mch

	NbObv	NbPO	NbPRi	NbPRa	NbUn	%Pr
Initialisation	1	0	0	0	0	100
CtxAUT	1	0	0	0	0	100

3.10.3 Règles ajoutées à la preuve

Fichier /home/enseignants/attiogbe/SPEC/DESS/B/TermAnalyser/bdp/CtxAUT.pmm non existant.

3.10.4 Source B

MACHINE
CtxAUT /* seen by all spec of AUTOMATON */

SETS

$WORD = \{le, la, un, ciel, terre, chien, chat, oiseau, est, possede, vole, court, gris, medor, nenette, point\}$

CONSTANTS
$STATE$
, $TRANSITION$
, $SYMBOL$
, $dums$
, $nbSymbols$
, $nbStates$
, $nbTransitions$
, $minSTATES$ /* bound of state interval */
, $maxSTATES$
, $minSYMBOLS$
, $maxSYMBOLS$
, $minTransitions$ /* bound of transitions */
, $maxTransitions$
PROPERTIES
$STATE = 100 \,..\, 199$
$\wedge$ $TRANSITION = 300 \,..\, 399$
$\wedge$ $SYMBOL = 200 \,..\, 299$
$\wedge$ $dums \in STATE$

$\wedge$ $nbSymbols \in \mathbf{NAT}$
$\wedge$ $nbStates \in \mathbf{NAT}$
$\wedge$ $nbTransitions \in \mathbf{NAT}$
$\wedge$ $minSTATES \in \mathbf{NAT}$
$\wedge$ $maxSTATES \in \mathbf{NAT}$
$\wedge$ $minTransitions \in \mathbf{NAT}$
$\wedge$ $maxTransitions \in \mathbf{NAT}$
$\wedge$ $minSYMBOLS \in \mathbf{NAT}$
$\wedge$ $maxSYMBOLS \in \mathbf{NAT}$
END

3.11 Composant CtxAUT_I

3.11.1 Présentation

Le composant traité dans ce paragraphe est : CtxAUT_I.

3.11.2 Métrique

CtxAUT_I AutoProved /home/enseignants/attiogbe/SPEC/DESS/B/TermAnalyser/sources/CtxAUT_I.im

	NbObv	NbPO	NbPRi	NbPRa	NbUn	%Pr
ValuesLemmas	14	1	0	1	0	100
Initialisation	2	1	0	1	0	100
CtxAUT_I	16	2	0	2	0	100

	NbObv	NbPO	NbPRi	NbPRa	NbUn	%Pr

3.11.3 Règles ajoutées à la preuve

Fichier /home/enseignants/attiogbe/SPEC/DESS/B/TermAnalyser/bdp/CtxAUT_I.pmm non existant.

3.11.4 Source B

IMPLEMENTATION
CtxAUT_I /* seen by all spec of AUTOMATON */
REFINES
CtxAUT

VALUES

$STATE = 100 .. 199$
; $SYMBOL = 200 .. 299$
; $TRANSITION = 300 .. 399$

; $dums = 100$

; $nbSymbols = 10$
; $nbStates = 20$
; $nbTransitions = 20$

; $minSTATES = 100$
; $maxSTATES = 199$
; $minTransitions = 300$
; $maxTransitions = 399$
; $minSYMBOLS = 200$
; $maxSYMBOLS = 299$

CONCRETE_VARIABLES
cst

INVARIANT
$cst \in STATE$

INITIALISATION
$cst := 199$

END

3.12 Composant CtxAnalyser

3.12.1 Présentation

Le composant traité dans ce paragraphe est : CtxAnalyser.

3.12.2 Métrique

CtxAnalyser AutoProved /home/enseignants/attiogbe/SPEC/DESS/B/TermAnalyser/sources/CtxAnalyser

	NbObv	NbPO	NbPRi	NbPRa	NbUn	%Pr
Initialisation	1	0	0	0	0	100
CtxAnalyser	1	0	0	0	0	100

3.12.3 Règles ajoutées à la preuve

Fichier /home/enseignants/attiogbe/SPEC/DESS/B/TermAnalyser/bdp/CtxAnalyser.pmm non existant.

3.12.4 Source B

MACHINE
CtxAnalyser /* seen by all spec of ANALYSER... */

CONSTANTS
nbWords
, *maxSymbols*
, *maxStates*
PROPERTIES
nbWords $\in$ **NAT** $\wedge$ *nbWords* <1000
$\wedge$ *maxStates* $\in$ **NAT** $\wedge$ *maxStates* <1000
$\wedge$ *maxSymbols* $\in$ **NAT** $\wedge$ *maxSymbols* <1000

END

3.13 Composant CtxAnalyser_I

3.13.1 Présentation

Le composant traité dans ce paragraphe est : CtxAnalyser_I.

3.13.2 Métrique

CtxAnalyser_I AutoProved /home/enseignants/attiogbe/SPEC/DESS/B/TermAnalyser/sources/CtxAnalys

	NbObv	NbPO	NbPRi	NbPRa	NbUn	%Pr
ValuesLemmas	5	0	0	0	0	100
Initialisation	1	0	0	0	0	100
CtxAnalyser_I	6	0	0	0	0	100

3.13.3 Règles ajoutées à la preuve

Fichier /home/enseignants/attiogbe/SPEC/DESS/B/TermAnalyser/bdp/CtxAnalyser_I.pmm non existant.

3.13.4 Source B

IMPLEMENTATION
CtxAnalyser_I /* seen by all spec of ANALYSER... */

REFINES

CtxAnalyser

VALUES
nbWords = 10
; *maxStates* = 100
; *maxSymbols* = 100
END

3.14 Composant INTERFACE

3.14.1 Présentation

Le composant traité dans ce paragraphe est : INTERFACE.

3.14.2 Métrique

INTERFACE POGenerated /home/enseignants/attiogbe/SPEC/DESS/B/TermAnalyser/sources/INTERF

	NbObv	NbPO	NbPRi	NbPRa	NbUn	%Pr
InstanciatedConstraintsLemmas	0	0	0	0	0	100
Initialisation	0	2	0	1	1	50
buildAnAutomaton	2	0	0	0	0	100
buildLexic	2	0	0	0	0	100
inputAnExpression	2	0	0	0	0	100
analyseTerm	2	0	0	0	0	100
displayAutomaton	2	0	0	0	0	100
displayExpression	2	0	0	0	0	100
INTERFACE	12	2	0	1	1	50

3.14.3 Règles ajoutées à la preuve

Fichier /home/enseignants/attiogbe/SPEC/DESS/B/TermAnalyser/bdp/INTERFACE.pmm non existant.

3.14.4 Source B

MACHINE
INTERFACE
SEES
CtxAUT
, *CtxAnalyser*

INCLUDES
ANALYSER

CONCRETE_VARIABLES
anExpression

INVARIANT

$anExpression \in 1 \,..\, nbWords \rightarrow WORD$

INITIALISATION

ANY *ww* **WHERE**
$ww \in WORD$
THEN
$anExpression := (1 \,..\, nbWords) \times \{ww\}$
END

OPERATIONS

buildAnAutomaton =
BEGIN
skip
END
;
buildLexic =
skip
;
inputAnExpression =
BEGIN
skip
END
;
analyseTerm =
BEGIN
skip
END
;
displayAutomaton = **skip**
;
displayExpression = **skip**
END

3.15 Composant INTERFACE_I

3.15.1 Présentation

Le composant traité dans ce paragraphe est : INTERFACE_I.

3.15.2 Métrique

INTERFACE_I POGenerated /home/enseignants/attiogbe/SPEC/DESS/B/TermAnalyser/sources/INTER

	NbObv	NbPO	NbPRi	NbPRa	NbUn	%Pr
ValuesLemmas	2	0	0	0	0	100
InstanciatedConstraintsLemmas	0	0	0	0	0	100
Initialisation	9	1	0	0	1	0
buildAnAutomaton	18	1	0	1	0	100

	NbObv	NbPO	NbPRi	NbPRa	NbUn	%Pr
buildLexic	18	0	0	0	0	100
inputAnExpression	19	12	0	1	11	8
analyseTerm	37	7	0	2	5	28
displayAutomaton	18	1	0	1	0	100
displayExpression	18	2	0	2	0	100
INTERFACE_I	139	24	0	7	17	29

3.15.3 Règles ajoutées à la preuve

Fichier /home/enseignants/attiogbe/SPEC/DESS/B/TermAnalyser/bdp/INTERFACE_I.pmm non existant.

3.15.4 Source B

IMPLEMENTATION
INTERFACE_I

REFINES
INTERFACE

SEES
CtxAUT
, *CtxAnalyser*
, *BASIC_IO*

IMPORTS
ANALYSER

CONCRETE_VARIABLES
exprSize

INVARIANT
$exprSize \in \mathbf{NAT}$
$\wedge$ $exprSize \geq 0$

INITIALISATION

$anExpression := (1 \,..\, nbWords) \times \{point\}$
; $exprSize := 0$
OPERATIONS
buildAnAutomaton =
BEGIN
STRING_WRITE("\n Building automaton....\n");

build_automaton

END
;
buildLexic =
BEGIN
build_lexic
END
;
inputAnExpression =

```
   BEGIN

anExpression(1) := le;
anExpression(2) := ciel;
anExpression(3) := est;
anExpression(4) := gris;
anExpression(5) := point;
anExpression(6) := point;
anExpression(7) := point;
anExpression(8) := point;
anExpression(9) := point;
anExpression(10) := point

   ; exprSize := nbWords
   END
;
   analyseTerm = /* To analyse a given term */
   VAR rr IN
         lexicalParseExpression(anExpression,exprSize);
      rr ← recogniseTerm;
         IF rr = TRUE
         THEN
            STRING_WRITE("\n The expression is well-formed \n")
         ELSE
            STRING_WRITE("\n The expression is not well-formed \n")
         END
      END
;
   displayAutomaton =
   BEGIN
        STRING_WRITE("\n The current Automaton is: \n");
   write_automaton
   END
;
   displayExpression =
   BEGIN
        STRING_WRITE("\n The current Expression is: \n");
        STRING_WRITE("\n ...not yet implemented ... \n")
   END
END
```

3.16 Composant LEXICAL

3.16.1 Présentation

Le composant traité dans ce paragraphe est : LEXICAL.

3.16.2 Métrique

LEXICAL AutoProved /home/enseignants/attiogbe/SPEC/DESS/B/TermAnalyser/sources/LEXICAL.mc

	NbObv	NbPO	NbPRi	NbPRa	NbUn	%Pr
Initialisation	0	3	0	3	0	100
add_lexicalMap	3	1	0	1	0	100
lexicalParse	4	0	0	0	0	100
lexicalParseWord	4	0	0	0	0	100
LEXICAL	11	4	0	4	0	100

3.16.3 Règles ajoutées à la preuve

Fichier /home/enseignants/attiogbe/SPEC/DESS/B/TermAnalyser/bdp/LEXICAL.pmm non existant.

3.16.4 Source B

MACHINE
$LEXICAL$

SEES
$CtxAUT$
DEFINITIONS

$Litem(mm) == lexCategory(mm)$

VARIABLES
$Symbs$,
$Words$,
$lexCategory$

INVARIANT
$Symbs \in \mathcal{F}\ (SYMBOL)$
$\wedge \quad Words \in \mathcal{F}\ (WORD)$
$\wedge \quad lexCategory \in Words \nrightarrow Symbs$

INITIALISATION
$Symbs,\ Words,\ lexCategory := \emptyset\ ,\ \emptyset\ ,\ \emptyset$

OPERATIONS

add_lexicalMap$(mm\ ,\ sb) =$
PRE
$mm \in WORD \wedge mm \in Words$
$\wedge \quad sb \in SYMBOL \wedge sb \in Symbs$
$\wedge \quad (mm \mapsto sb) \notin lexCategory$
THEN
$lexCategory := lexCategory \lhd\!\!\!\!- \{mm \mapsto sb\}$
END

;

$ss \leftarrow$ **lexicalParse**$(ch,\ nbitem) =$
PRE
$ch \in 1\ ..\ 100 \rightarrow WORD \wedge \mathbf{ran}(ch) \subseteq Words$

$\wedge \quad ss \in 1 \,..\, 100 \rightarrow SYMBOL \wedge \mathbf{ran}(ss) \subseteq Symbs$
$\wedge \quad nbitem \in 1 \,..\, 100$
THEN
ANY *ssy* **WHERE**
$ssy \in \mathbf{seq}\ (SYMBOL) \wedge \mathbf{ran}(ssy) \subseteq Symbs$
$\wedge \quad \forall\ (ii).(\ ii \in \mathbf{dom}(ch) \Rightarrow ssy(ii) = Litem(ch(ii)))$
THEN
$ss := ssy$
END
END

;

$sb \leftarrow$ **lexicalParseWord**$(mot) =$
PRE
$mot \in WORD \wedge mot \in Words$
$\wedge \quad sb \in SYMBOL \wedge sb \in Symbs$
THEN
$sb := Litem(mot)$
END
END

3.17 Composant LEXICAL_I

3.17.1 Présentation

Le composant traité dans ce paragraphe est : LEXICAL_I.

3.17.2 Métrique

LEXICAL_I POGenerated /home/enseignants/attiogbe/SPEC/DESS/B/TermAnalyser/sources/LEXICAL

	NbObv	NbPO	NbPRi	NbPRa	NbUn	%Pr
ValuesLemmas	1	0	0	0	0	100
InstanciatedConstraintsLemmas	3	4	0	4	0	100
Initialisation	1	4	0	4	0	100
add_lexicalMap	8	2	0	1	1	50
lexicalParse	15	19	0	11	8	57
lexicalParseWord	7	3	0	0	3	0
LEXICAL_I	35	32	0	20	12	62

3.17.3 Règles ajoutées à la preuve

Fichier /home/enseignants/attiogbe/SPEC/DESS/B/TermAnalyser/bdp/LEXICAL_I.pmm non existant.

3.17.4 Source B

IMPLEMENTATION
LEXICAL_I

SEES
CtxAUT

REFINES
LEXICAL

IMPORTS
ensSymbs.L_SET(100,*SYMBOL*)
, *ensWords.L_SET*(100,*WORD*)
, *lexCategory.L_ARRAY_COLLECTION*(1,*WORD*,*SYMBOL*)
, *ss.L_SEQUENCE*(100,*SYMBOL*)

CONCRETE_VARIABLES
indtab

INVARIANT
indtab ∈ **NAT**
∧ **ran**(*ensSymbs.set_vrb*) = *Symbs*
∧ **ran**(*ensWords.set_vrb*) = *Words*

INITIALISATION
BEGIN
ss.CLR_SEQ;
VAR *bb* **IN**
bb := **TRUE**;
indtab,*bb* ← *lexCategory.CRE_ARR_COL*
END
END

OPERATIONS

add_lexicalMap(*mm* , *sb*) =
lexCategory.STR_ARR_COL(1,*mm*,*sb*)

;

ss ← **lexicalParse**(*ch*,*nbitem*) =
VAR *ii* **IN**
ii := 1;
ss.CLR_SEQ;
WHILE *ii* ≤ *nbitem* **DO**
VAR *mm*, *vv* **IN**
mm := *ch*(*ii*);
vv ← **lexCategory.VAL_ARR_COL**(*indtab*,*mm*);
ss.PUSH_SEQ(*vv*);
ii := *ii* + 1
END
INVARIANT
ii ∈ **NAT**

```
        ∧    nbitem ∈ NAT
      VARIANT
        nbitem + 1 - ii
      END

    END

;


sb ←  lexicalParseWord(mot) =
      sb ← lexCategory.VAL_ARR_COL(indtab,mot)

END
```

4 DICTIONNAIRE

Contents