Annexes Partie 3 Chapitre 9

A-3-9.1. Analyse avec OLMES

Première analyse avec traçage minimal prédéfini

```
OLMES>e

      OLMES 2.0 - Macintosh Common Lisp Version 2.0 -


      Chargement du savoir syntaxique et lexical

Texte analyse'  ? (SANS guillemets)
txt2.lisp
Dictionnaire compile'  ? (SANS guillemets)
dic2.lisp
Classes pour le dictionnaire compile'  ? (SANS guillemets)
dic2.lisp.ccmm.1
Grammaire compile'e  ? (SANS guillemets)
cfg3.lisp


(PROGN (FORMAT *STANDARD-OUTPUT* *SET-PROCESSING-ENVIRONMENT-FMT-STR*) (OPEN-TEXTUAL-INPUT A-PARSER)
(LOAD-LEXICAL-AND-SYNTACTIC-KNOWLEDGE A-PARSER) (FORMAT *STANDARD-OUTPUT* "~2%")) took 32893 milliseconds
(32.893 seconds) to run.
Of that, 15331 milliseconds (15.331 seconds) were spent in The Cooperative Multitasking Experience.
 181160 bytes of memory allocated.


      Voulez-vous les arbres correspondants aux groupes construits ?
      Re'ponses possibles : O oui   ou   N non
n
      Voulez-vous un suivi pas a pas SUR FICHIER  ?
      Re'ponses possibles : O oui   ou   N non
n
      Partie commune des noms des fichiers de sortie ?
      Une re'ponse de type SYMBOL est attendue
sortie-cfg3



           Phrase 1


      Noeud 0


0-1    1 moulin
0-1    2 N -> MOULIN : 1

      Noeud 1


1-2    3 a
1-2    4 PREP -> A : 3

      Noeud 2


2-3    5 paroles
2-3    6 N -> PAROLES : 5

      Noeud 3


      Arc(s) nouveau(x) (INITIALISATION)


2-2    8 SN ->    *N* <PREP> <N>
0-0    7 SN ->    *N* <PREP> <N>

      Arc extrait de l'agenda :
2-2    8 SN ->    *N* <PREP> <N>
       Arc(s) examine'(s) pour combinaison :
2-3    6 N -> PAROLES : 5
```

```
       Arc(s) nouveau(x) (COMBINAISON)


2-3    9 SN ->   N   *PREP* <N>
                 N=6

      Arc extrait de l'agenda :
2-3    9 SN ->   N   *PREP* <N>
                 N=6

      Arc extrait de l'agenda :
0-0    7 SN ->    *N* <PREP> <N>
       Arc(s) examine'(s) pour combinaison :
0-1    2 N -> MOULIN : 1

       Arc(s) nouveau(x) (COMBINAISON)


0-1    10 SN ->  N   *PREP* <N>
                 N=2

      Arc extrait de l'agenda :
0-1    10 SN ->  N   *PREP* <N>
                 N=2
       Arc(s) examine'(s) pour combinaison :
1-2    4 PREP -> A : 3

       Arc(s) nouveau(x) (COMBINAISON)


0-2    11 SN ->  N PREP   *N*
                 N=2  PREP=4

      Arc extrait de l'agenda :
0-2    11 SN ->  N PREP   *N*
                 N=2  PREP=4
       Arc(s) examine'(s) pour combinaison :
2-3    6 N -> PAROLES : 5

       Arc(s) nouveau(x) (COMBINAISON)


0-3    12 SN -> N : 2  PREP : 4  N : 6

      Arc extrait de l'agenda :
0-3    12 SN -> N : 2  PREP : 4  N : 6
       Arc(s) examine'(s) pour combinaison :
0-0    7 SN ->    *N* <PREP> <N>
(LET ((A-SCHEDULER (MAKE-SCHEDULER))) (ADD-ITEM (GET-ACTIVABLE-TASK-POOL A-SCHEDULER) (MAKE-TEXTUAL-DATA-
READER A-PARSER)) (SCHEDULE A-SCHEDULER A-PARSER) (VALUES)) took 6134 milliseconds (6.134 seconds) to run.
Of that, 71 milliseconds (0.071 seconds) were spent in The Cooperative Multitasking Experience.
 42648 bytes of memory allocated.


(PROGN (FORMAT *STANDARD-OUTPUT* "~2%") (PROCESSING-TEXTUAL-INPUT A-PARSER) (FORMAT *STANDARD-OUTPUT*
"~2%")) took 6684 milliseconds (6.684 seconds) to run.
Of that, 76 milliseconds (0.076 seconds) were spent in The Cooperative Multitasking Experience.
 43832 bytes of memory allocated.



           Fin du traitement




    Texte analyse'  : Macintosh HD:Jussieu:CLOS:Olmes:Code:data:txt:txt2.lisp
    Dictionnaire compile'  : Macintosh HD:Jussieu:CLOS:Olmes:Code:data:dic:dic2.lisp
   Classes pour le dictionnaire compile'  : Macintosh
HD:Jussieu:CLOS:Olmes:Code:data:dic:dic2∂.lisp∂.ccmm.1
    Grammaire compile'e  : Macintosh HD:Jussieu:CLOS:Olmes:Code:data:gram:cfg3.lisp




           Fichiers de sortie utilise's
             (Re'pertoire : Macintosh HD:Jussieu:CLOS:Olmes:Code:data:results:)


       phrases analyse'es  : 1

       mots traite's  : 3

       re`gles de'clenche'es  : 2

       re`gles ayant re'ussi  : 1

       arcs cre'e's  : 12


       Nombre moyen d'e'tapes pour traiter une phrase : 7
```

Accès à l'interface GASPAR (sous CLOS)

```
OLMES>d
;Compiler warnings :
;    Undeclared free variable MENU-TRACE, in an anonymous lambda form.
;Compiler warnings :
;    Undefined function OLMES::MAKE-NO-TRACE, in an anonymous lambda form.
;    Undefined function OLMES::LOAD-NO-TRACE, in an anonymous lambda form.
;    Undefined function OLMES::LOAD-TRACE, in an anonymous lambda form.

GASPAR>

 n'est pas un choix possible. Recommencez.

A  LOAD-TRACE
       GASPAR: Load Traceur-Gaspar de OLMES
```

```
B  LOAD-UNTRACE
       GASPAR: Detraceur-Gaspar de OLMES

C  ANALYSE-WITH-TRACEUR-GASPAR
       GASPAR: analyse with traceur-gaspar OLMES

D  HELP-GASPAR
       GASPAR: help on trace and untrace

Z  QUITTER
      Fin du travail avec GASPAR
```

Chargement des traceurs GASPAR

```
GASPAR>a
;Compiler warnings :
;    Undeclared free variable *HELP-TRACE*, in HELP-TRACE.
;Compiler warnings :
;    Undeclared free variable *RESULT-TREES*, in PRINT-TREES-FOR-CLIM.
;Compiler warnings :
;    Undeclared free variable *ABA*, in USE-AND-CREATE-HYPOTHESES.
;Compiler warnings :
;    Undeclared free variable *HELP-NETWORK-GRAMMAR* (2 references), in ADD-HELP.
;Compiler warnings :
;    Undeclared free variable *HELP-NETWORK-GRAMMAR*, in HELP-NETWORK-GRAMMAR.
;Compiler warnings :
;    Undeclared free variable *CURRENT-COMPILED-GRAMMAR*, in NETWORK-LIST.
;Compiler warnings :
;    Undeclared free variable *CURRENT-COMPILED-GRAMMAR*, in NETWORK-OBJECTS.
;Compiler warnings :
;    Unused lexical variable RULE, in an anonymous lambda form inside CALCULATE-STAT.
;    Undeclared free variable *SOMME* (3 references), in an anonymous lambda form inside CALCULATE-STAT.
;    Undeclared free variable *RULE-TABLE*, in an anonymous lambda form inside CALCULATE-STAT.
;    Undeclared free variable *SOMME*, in an anonymous lambda form inside an anonymous lambda form inside
CALCULATE-STAT.
;    Unused lexical variable A-RULE, in an anonymous lambda form inside an anonymous lambda form inside
CALCULATE-STAT.
;    Undeclared free variable *RULE-TABLE*, in CALCULATE-STAT.
;Compiler warnings :
;    Undeclared free variable *SOMME*, in ADD.
;Compiler warnings :
;    Undeclared free variable *RULE-TABLE*, in PRINT-STAT.
;Compiler warnings :
;    Undeclared free variable *CURRENT-COMPILED-GRAMMAR*, in DISPLAY-CURRENT-COMPILED-GRAMMAR.
;Compiler warnings :
;    Undefined function GET-RULE-WEIGHT, in an anonymous lambda form inside PRINT-STAT.
;    Undefined function GET-SUCCESS, in an anonymous lambda form inside an anonymous lambda form inside
CALCULATE-STAT.
;    Undefined function SET-RULE-WEIGHT, in an anonymous lambda form inside CALCULATE-STAT.
;    Undefined function GET-SUCCESS, in an anonymous lambda form inside CALCULATE-STAT.
;    Undefined function NEW-CONSTITUENT-RIGHT-HYPOTHESES, in CUSTOMIZE-NEW-EDGE.
;    Undefined function SET-RIGHT-HYPOTHESES, in CUSTOMIZE-NEW-EDGE.
;    Undefined function NEW-CONSTITUENT-LEFT-HYPOTHESES, in CUSTOMIZE-NEW-EDGE.
;    Undefined function SET-LEFT-HYPOTHESES, in CUSTOMIZE-NEW-EDGE.
;    Undefined function NEW-CONSTITUENT-RIGHT-HYPOTHESES, in CUSTOMIZE-NEW-EDGE.
;    Undefined function SET-RIGHT-HYPOTHESES, in CUSTOMIZE-NEW-EDGE.
;    Undefined function NEW-CONSTITUENT-LEFT-HYPOTHESES, in CUSTOMIZE-NEW-EDGE.
;    Undefined function SET-LEFT-HYPOTHESES, in CUSTOMIZE-NEW-EDGE.
;Compiler warnings :
;    Undeclared free variable *HELP-NO-TRACE*, in HELP-NO-TRACE.
```

Nouvelle analyse avec traceurs GASPAR

```
GASPAR>c

       OLMES 2.0 - Macintosh Common Lisp Version 2.0 -


       Chargement du savoir syntaxique et lexical

Texte analyse'  ? (SANS guillemets)
txt2.lisp
Dictionnaire compile'  ? (SANS guillemets)
dic2.lisp
Classes pour le dictionnaire compile'  ? (SANS guillemets)
dic2.lisp.ccmm.1
Grammaire compile'e  ? (SANS guillemets)
cfg3.lisp


(PROGN (FORMAT *STANDARD-OUTPUT* *SET-PROCESSING-ENVIRONMENT-FMT-STR*) (OPEN-TEXTUAL-INPUT A-PARSER)
(LOAD-LEXICAL-AND-SYNTACTIC-KNOWLEDGE A-PARSER) (FORMAT *STANDARD-OUTPUT* "~2%")) took 26393 milliseconds
(26.393 seconds) to run.
Of that, 17862 milliseconds (17.862 seconds) were spent in The Cooperative Multitasking Experience.
 73984 bytes of memory allocated.


        Voulez-vous les arbres correspondants aux groupes construits ?
       Re'ponses possibles : O oui    ou    N non
n
        Voulez-vous un suivi pas a pas SUR FICHIER  ?
       Re'ponses possibles : O oui    ou    N non
n
        Partie commune des noms des fichiers de sortie ?
        Une re'ponse de type SYMBOL est attendue
sortie-cfg3-gaspar



             Phrase 1



      Noeud 0
```

```
0-1    1 moulin
0-1    2 N -> MOULIN : 1

      Noeud 1


1-2    3 a
1-2    4 PREP -> A : 3

      Noeud 2


2-3    5 paroles
2-3    6 N -> PAROLES : 5

      Noeud 3



======================================================================


======================================================================



;;;;Compiled-network-grammar

 Here's the complete compiled network :


 Regle n.1:
 SN0 --> (N0 PREP0 N1)



;;;;Phrase en cours d'analyse
 0 MOULIN  1 A  2 PAROLES



======================================================================



       Arc(s) nouveau(x) (INITIALISATION)



2-2    8 SN ->    *N* <PREP> <N>
0-0    7 SN ->    *N* <PREP> <N>

      Arc extrait de l'agenda :
2-2    8 SN ->    *N* <PREP> <N>

======================================================================

;;;;Try-to-continue-edges  :before


Nouvel arc a construire

Noeud de depart: 2
Noeud de fin: 2
Label de l'arc: N
Numero de l'arc: 8

---Arcs disponibles


 --Internals Arc disponible:


Noeud de depart: 2
Noeud de fin: 3
Label de l'arc: N
Numero de l'arc: 6
======================================================================

======================================================================

;;;;make-combined-edges  :before

internals active-edge

Noeud de depart: 2
Noeud de fin: 2
Label de l'arc: N
Numero de l'arc: 8

internals inactive-edge

Noeud de depart: 2
Noeud de fin: 3
Label de l'arc: N
Numero de l'arc: 6

;;;;Regle Construite


Documentation attachee a la partie gauche de la regle qui suit:
NIL

Nom de la classe associee au dernier element de la regle qui vient d'etre construite:N0


Label de la classe qui vient d'etre construite:N

======================================================================



       Arc(s) nouveau(x) (COMBINAISON)



2-3    9 SN ->   N   *PREP* <N>
                 N=6

======================================================================
```

```
;;;;Internals-edges

Noeud de depart: 2
Noeud de fin: 2
Label de l'arc: N
Numero de l'arc: 8

;;;;Construction d'un Rhs element


Documentation attachee au dernier elememt pointe de la regle qui suit:
NIL

Label de cet element:N


Nom de la classe associee a cet element:N0


Prochaine classe necessaire:PREP0

Arbres construits --in progress--

0 (N MOULIN)

1 (PREP A)

2 (N PAROLES)

======================================================================


      Arc extrait de l'agenda :
2-3    9 SN ->   N  *PREP* <N>
               N=6

======================================================================

;;;;Internals-edges

Noeud de depart: 2
Noeud de fin: 3
Label de l'arc: PREP
Numero de l'arc: 9

;;;;Construction d'un Rhs element


Documentation attachee au dernier elememt pointe de la regle qui suit:
NIL

Label de cet element:PREP


Nom de la classe associee a cet element:PREP0


Prochaine classe necessaire:N1

Arbres construits --in progress--

0 (N MOULIN)

1 (PREP A)

2 (N PAROLES)

======================================================================


      Arc extrait de l'agenda :
0-0    7 SN ->    *N* <PREP> <N>

======================================================================

;;;;Try-to-continue-edges :before


Nouvel arc a construire

Noeud de depart: 0
Noeud de fin: 0
Label de l'arc: N
Numero de l'arc: 7

---Arcs disponibles


  --Internals Arc disponible:


Noeud de depart: 0
Noeud de fin: 1
Label de l'arc: N
Numero de l'arc: 2
======================================================================

======================================================================

;;;;make-combined-edges :before

internals active-edge

Noeud de depart: 0
Noeud de fin: 0
Label de l'arc: N
Numero de l'arc: 7

internals inactive-edge

Noeud de depart: 0
```

```
Noeud de fin: 1
Label de l'arc: N
Numero de l'arc: 2

;;;;Regle Construite


Documentation attachee a la partie gauche de la regle qui suit:
NIL

Nom de la classe associee au dernier element de la regle qui vient d'etre construite:N0


Label de la classe qui vient d'etre construite:N

======================================================================


      Arc(s) nouveau(x) (COMBINAISON)


0-1    10 SN ->  N  *PREP* <N>
               N=2

======================================================================


;;;;Internals-edges

Noeud de depart: 0
Noeud de fin: 0
Label de l'arc: N
Numero de l'arc: 7

;;;;Construction d'un Rhs element


Documentation attachee au dernier elememt pointe de la regle qui suit:
NIL

Label de cet element:N


Nom de la classe associee a cet element:N0


Prochaine classe necessaire:PREP0

Arbres construits --in progress--

0 (N MOULIN)

1 (PREP A)

2 (N PAROLES)

======================================================================


      Arc extrait de l'agenda :
0-1    10 SN ->  N  *PREP* <N>
               N=2

======================================================================

;;;;Try-to-continue-edges :before


Nouvel arc a construire

Noeud de depart: 0
Noeud de fin: 1
Label de l'arc: PREP
Numero de l'arc: 10

---Arcs disponibles


  --Internals Arc disponible:


Noeud de depart: 1
Noeud de fin: 2
Label de l'arc: PREP
Numero de l'arc: 4
======================================================================


      Arc(s) nouveau(x) (COMBINAISON)


0-2    11 SN ->  N PREP  *N*
               N=2  PREP=4

======================================================================


;;;;Internals-edges

Noeud de depart: 0
Noeud de fin: 1
Label de l'arc: PREP
Numero de l'arc: 10

;;;;Construction d'un Rhs element


Documentation attachee au dernier elememt pointe de la regle qui suit:
NIL

Label de cet element:PREP


Nom de la classe associee a cet element:PREP0
```

```
Prochaine  classe  necessaire:N1

Arbres  construits  --in  progress--

0  (N  MOULIN)

1  (PREP  A)

2  (N  PAROLES)

=====================================================================


      Arc  extrait  de  l'agenda  :
0-2    11  SN  ->   N  PREP   *N*
                N=2   PREP=4

=====================================================================

;;;;Try-to-continue-edges  :before


Nouvel  arc  a  construire

Noeud  de  depart:  0
Noeud  de  fin:  2
Label  de  l'arc:  N
Numero  de  l'arc:  11

---Arcs  disponibles


  --Internals  Arc  disponible:


Noeud  de  depart:  2
Noeud  de  fin:  3
Label  de  l'arc:  N
Numero  de  l'arc:  6
=====================================================================

=====================================================================

;;;;make-combined-edges  :before

internals  active-edge

Noeud  de  depart:  0
Noeud  de  fin:  2
Label  de  l'arc:  N
Numero  de  l'arc:  11

internals  inactive-edge

Noeud  de  depart:  2
Noeud  de  fin:  3
Label  de  l'arc:  N
Numero  de  l'arc:  6

;;;;Regle  Construite


Documentation  attachee  a  la  partie  gauche  de  la  regle  qui  suit:
NIL

Nom  de  la  classe  associee  au  dernier  element  de  la  regle  qui  vient  d'etre  construite:N1


Label  de  la  classe  qui  vient  d'etre  construite:N

=====================================================================


      Arc(s)  nouveau(x)  (COMBINAISON)


0-3    12  SN  ->  N  :  2   PREP  :  4   N  :  6

      Arc  extrait  de  l'agenda  :
0-3    12  SN  ->  N  :  2   PREP  :  4   N  :  6

=====================================================================

;;;;Try-to-continue-edges  :before


Nouvel  arc  a  construire

Noeud  de  depart:  0
Noeud  de  fin:  3
Label  de  l'arc:  SN
Numero  de  l'arc:  12

---Arcs  disponibles


  --Internals  Arc  disponible:


Noeud  de  depart:  0
Noeud  de  fin:  0
Label  de  l'arc:  N
Numero  de  l'arc:  7
=====================================================================

=====================================================================

Arbres  construits  terminaux
#<ACTIVE-CHART  #x438B29>
Class:  #<STANDARD-CLASS  ACTIVE-CHART>
Wrapper:  #<CCL::CLASS-WRAPPER  ACTIVE-CHART  #x3F5C21>
```

```
Instance  slots
NUMBER-OF-EDGES:  12
NUMBER-OF-WORDS:  3
SUCCESSES:  NIL
FIRST-VERTEX:  #<VERTEX-FOR-BIDIRECTIONAL-ACTIVE-CHART  #x438B51>
AGENDA:  #<DEPTH-FIRST-AGENDA  #x438B89>
SCHEDULER::WAITING:  NIL
NUMBER:  1

0  (SN  (N  MOULIN)  (PREP  A)  (N  PAROLES))

0  (N  MOULIN)

1  (PREP  A)

2  (N  PAROLES)

Arbres  construits  terminaux  for  CLIM
#<ACTIVE-CHART  #x438B29>
Class:  #<STANDARD-CLASS  ACTIVE-CHART>
Wrapper:  #<CCL::CLASS-WRAPPER  ACTIVE-CHART  #x3F5C21>
Instance  slots
NUMBER-OF-EDGES:  12
NUMBER-OF-WORDS:  3
SUCCESSES:  NIL
FIRST-VERTEX:  #<VERTEX-FOR-BIDIRECTIONAL-ACTIVE-CHART  #x438B51>
AGENDA:  #<DEPTH-FIRST-AGENDA  #x438B89>
SCHEDULER::WAITING:  NIL
NUMBER:  1

=====================================================================
  Taper:  (help)  a  la  fin  d'une  analyse  pour  disposer  des  outils  de
            visualisation  des  resultats  de  la  compilation  ou  pour  le
           Tracage/No-tracage

=====================================================================
(LET  ((A-SCHEDULER  (MAKE-SCHEDULER)))  (ADD-ITEM  (GET-ACTIVABLE-TASK-POOL  A-SCHEDULER)  (MAKE-TEXTUAL-DATA-
READER  A-PARSER))  (SCHEDULE  A-SCHEDULER  A-PARSER)  (VALUES))  took  30668  milliseconds  (30.668  seconds)  to
run.
Of  that,  364  milliseconds  (0.364  seconds)  were  spent  in  The  Cooperative  Multitasking  Experience.
  53592  bytes  of  memory  allocated.


(PROGN  (FORMAT  *STANDARD-OUTPUT*  "~2%")  (PROCESSING-TEXTUAL-INPUT  A-PARSER)  (FORMAT  *STANDARD-OUTPUT*
"~2%"))  took  31166  milliseconds  (31.166  seconds)  to  run.
Of  that,  368  milliseconds  (0.368  seconds)  were  spent  in  The  Cooperative  Multitasking  Experience.
  54976  bytes  of  memory  allocated.



           Fin  du  traitement




     Texte  analyse'   :  Macintosh  HD:Jussieu:CLOS:Olmes:Code:data:txt:txt2.lisp
     Dictionnaire  compile'   :  Macintosh  HD:Jussieu:CLOS:Olmes:Code:data:dic:dic2.lisp
    Classes  pour  le  dictionnaire  compile'   :  Macintosh
HD:Jussieu:CLOS:Olmes:Code:data:dic:dic2∂.lisp∂.ccmm.1
     Grammaire  compile'e   :  Macintosh  HD:Jussieu:CLOS:Olmes:Code:data:gram:cfg3.lisp




            Fichiers  de  sortie  utilise's
              (Re'pertoire  :  Macintosh  HD:Jussieu:CLOS:Olmes:Code:data:results:)


       phrases  analyse'es   :  1

       mots  traite's   :  3

       re`gles  de'clenche'es   :  2

       re`gles  ayant  re'ussi   :  1

       arcs  cre'e's   :  12


      Nombre  moyen  d'e'tapes  pour  traiter  une  phrase  :  7
```

Outils de l'interface GASPAR

```
GASPAR>d

GASPAR-HELP>a

.....................................................................

Trois  HELP  sont  disponibles:


.....................................................................

  Le  premier  concerne  la  grammaire  compilee:  (help-network-grammar)


.....................................................................

  Le  second  concerne  le  tracage  des  methodes  au  cours  de  l'analyse:
(help-trace)


.....................................................................

  Le  troisieme  concerne  le  non-tracage  des  methodes  au  cours  de  l'analyse:
(help-no-trace)
```

```
...................................................................

GASPAR-HELP>b

.....................................................................
TRACAGE:  the  interface  functions  and  their  documentation.
  *  SCHEDULER:RUN
 NIL

  *  OLMES::COMBINE
 NIL

  *  OLMES::TRY-TO-CONTINUE-EDGES
 NIL

  *  OLMES::MAKE-COMBINED-EDGE
 NIL

  *  OLMES::CUSTOMIZE-NEW-EDGE
 NIL

.....................................................................

GASPAR-HELP>c

.....................................................................
NO-TRACAGE:  the  interface  functions  and  their  documentation.
  *  OLMES::MAKE-NO-TRACE-ON-RUN
 "no-trace  sur  la  methode  run"

  *  OLMES::MAKE-NO-TRACE-ON-COMBINE
 "no-trace  sur  la  methode  combine"

  *  OLMES::MAKE-NO-TRACE-ON-TRY-TO-CONTINUE-EDGES
 "no-trace  sur  la  methode  try-to-continue-edges,  types  d'argument:
new-constituent-maker,  building-edge  ou  edge-entity"

  *  OLMES::MAKE-NO-TRACE-ON-MAKE-COMBINED-EDGE
 "no-trace  sur  la  methode  make-combined-edge  l'argument  peut  etre
   new-constituent-maker"

  *  OLMES::MAKE-NO-TRACE-ON-USE-AND-CREATE-HYPOTHESES
 "no-trace  sur  la  methode  use-and-create-hypotheses"

  *  OLMES::MAKE-NO-TRACE
 "No-tracage  pour  les  methodes  contenues  dans  *help-no-tracage*"

.....................................................................
GASPAR-HELP>z

"The  end"
GASPAR>z

"The  end"

Retour  à  l'interface   principale  pour  OLMES:

OLMES>

 n'est  pas  un  choix  possible.  Recommencez.

A  COMPILE
      Compilation  de  OLMES

B  LOAD-PARSER
      load  du  parser  de   OLMES

C  LOAD-AND-ANALYSE
      ANALYSE  with  OLMES

D  TRACE
      GASPAR  :  outil  de  trace  fleurien

E  ANALYSE
      ANALYSE  with  OLMES

F  LOAD-CLIM-INTERFACE
      Load  Clim-Interface  for  OLMES

G  CLIM-INTERFACE
      Clim-Interface  for  OLMES

Q  QUITTER
      Fin  du  travail  avec  OLMES
```

A-3-9.2. Analyse avec le prototype `Gaspar`

A-3-9.2.1. Réflexion de structure opérée avec Self

On donne ci-dessous une illustration interactive de l'utilisation de la réflexion de structure en utilisant la notion de miroir mise en place par Self. Après création de l'attribut `displayGaspar`, on associe à cet attribut l'objet `miroir` obtenu sur `Gaspar` via la primitive `reflect:`

```
"Self  31"  _AddSlots:  (|  displayGaspar  |)
shell

"Self  32"  displayGaspar:  (reflect:  gaspar)
shell

"Self  33"  inspect:  displayGaspar
(  |  prototype  =  (  mirrors  slots).
     parent*  =  traits  mirrors  slots.  |  )
Module:  mirror
a  mirrors  slots(on  an  object)

"Self  34"  inspect:  displayGaspar  reflectee

(  |   parent*  =  traits  clonable.
     comment  =  (  'Gaspar  est  le  prototype  qui  centralise  toutes  les  activites  de  gestion  des  elements  mis
en  oeuvre  pour  une  analyse  automatique  prototypique:  chargement  des  donnees,  affichage  du  lexique,  ajout
d\'elements  lexicaux,  modification  d\'elements  lexicaux,...,  et  enfin  analyse.\n  Pour  le  moment,
l\'analyse  travaille  sur  quatre  fichiers  qui  contiennent  chacun  leur  phrase  a  analyser  et  l\'appel  au
manager  pour  l\'analyse.  Pour  analyser  une  phrase  suivre  le  modele  que  constitue  chacun  des  fichiers
donnes  dans  PhaseX/txt.  L\'acces  au  menu  de  gaspar  se  fait  via  la  syntaxe  suivante:  gaspar  menu.  Suivant
la  tache  a  effectuer,  on  fait  ensuite  appel  aux  menus  associes  aux  taches  visees.\n').
      gasparDirectory  =  '/home/fleury/Self/source/Self-4.0/Phase2/'.
     lexiqueGaspar  =  an  object.
     metaGaspar  =  an  object.
     toolsGaspar  =  an  object.
     parserGaspar  =  an  object.
     dictGaspar  =  an  object.
     categorieGaspar  =  an  object.
      categorieGasparDynamic  =  an  object.
     menusGaspar  =  an  object.  |  )
Module:
an  object
```

On peut ensuite, en utilisant des primitives prédéfinies, manipuler les `miroirs` obtenus et ainsi interroger la structure de l'objet `Gaspar`.

```
"Self  38"   ((reflect:  gaspar)  at:  (displayGaspar  at:  3)  name)
a  slots  plain({  'Category:  lexiqueGaspar'
   lexiqueGaspar  =  an  object
})

"Self  39"   ((reflect:  gaspar)  at:  (displayGaspar  at:  1)  name)
a  slots  method(comment  =  (  'Gaspar  est  le  prototype  qui  centralise  toutes  les  activites  de  gestion  des
elements  mis  en  oeuvre  pour  une  analyse  automatique  prototypique:  chargement  des  donnees,  affichage  du
lexique,  ajout  d\'elements  lexicaux,  modification  d\'elements  lexicaux,...,  et  enfin  analyse.\n  Pour  le
moment,  l\'analyse  travaille  sur  quatre  fichiers  qui  contiennent  chacun  leur  phrase  a  analyser  et  l\'appel
au  manager  pour  l\'analyse.  Pour  analyser  une  phrase  suivre  le  modele  que  constitue  chacun  des  fichiers
donnes  dans  PhaseX/txt.  L\'acces  au  menu  de  gaspar  se  fait  via  la  syntaxe  suivante:  gaspar  menu.  Suivant
la  tache  a  effectuer,  on  fait  ensuite  appel  aux  menus  associes  aux  taches  visees.\n'))

"Self  40"  (displayGaspar  at:  8)  name
'categorieGaspar'

"Self  41"  ((reflect:  gaspar)  at:  (displayGaspar  at:  8)  name)  isAssignable
false

"Self  42"   ((reflect:  gaspar)  at:  (displayGaspar  at:  8)  name)  isParent
false

})

"Self  43"  ((reflect:  gaspar)  at:  (displayGaspar  at:  0)  name)
a  slots  parent(parent*  =  traits  clonable)

"Self  44"  ((reflect:  gaspar)  at:  (displayGaspar  at:  0)  name)  isParent
true
```

A moins de connaître tous les protocoles définis via Self pour manipuler ce type de processus, il reste difficile à un non spécialiste du langage de manipuler ce type de processus. On a donc mis en place une interface méta qui active des outils manipulant ces processus de manière transparente pour l'utilisateur.

A-3-9.2.2. Un méta-niveau pour inspecter les objets

Dans l'exemple qui suit on utilise l'interface méta construite pour interroger la structure des objets construits. Dans cet exemple, on effectue une requête d'affichage des attributs d'un prototype `<Nom: lesion>`. S'il existe une représentation prototypique adéquate, on obtient une présentation détaillée de la structure de ce prototype.

```
-------------------------------------------------------
<1: Display Meta Rules>
<2: Display Bugs Report>
<3: Build Delegation>
<4: Remove Delegation>
<5: Build Prototypes>
<6: Look For Delegation>
<7: Meta Trace>
<8: Add Meta Rule>
<9: Remove Meta Rule>
<10: Add Bugs Report>
<11: Remove Bugs Report>
<12: Display Slot Of Selected Prototypes>
<13: Menu AddSlots>
<14: Compteur de Formes lexicales>
<15: Empty Waste Gaspar>

Select a number or q to quit MenuMeta:12
-------------------------------------------------------
-----------------------------------------------------------------------------------------------
Quel type de prototype pour la  selection
1. Prototypes/Arbres Construits (recherche sur forme lexicale)
2. Prototypes/Arbres Non Construits (recherche sur Categorie)
3. Tout type de Prototypes/Arbres
Votre Choix: 1


Recherche d'un Arbre Elementaire Construit
-------------------------------------------
Forme de l'Arbre Elementaire Cherche: lesion
Categorie de tete de l'Arbre Elementaire Cherche: Nom
-----------------------------------------------------------------------------------------------
 Prototype : ['lesion'|'Nom']
Selection de ce prototype
(repondre par y ou Y): y
----------------------------------------------------------------------------------------------
-----------------------------------------------------------------------------------------------
 Prototype : ['lesion'|'Nom']
Selection de ce prototype
(repondre par y ou Y): n
----------------------------------------------------------------------------------------------
------------------------------------------------------------------
Description des Slots du Prototype : <lesion|Nom>
------------------------------------------------------------------
           Nom du Slot: parent
            Contenu du Slot parent : parent* = an object
                 Ce slot est un ParentSlot
           Nom du Slot: protoBuilt
            Contenu du Slot protoBuilt : protoBuilt = true
                 Ce slot est un AssignableSlot
           Nom du Slot: protoBuilt:
            Contenu du Slot protoBuilt: : protoBuilt: = <-
           Nom du Slot: protoToBuild
            Contenu du Slot protoToBuild : protoToBuild = an object
                 Ce slot est un AssignableSlot
           Nom du Slot: protoToBuild:
            Contenu du Slot protoToBuild: : protoToBuild: = <-
           Nom du Slot: elemTree
            Contenu du Slot elemTree : elemTree = a list(an object, an object)
                 Ce slot est un AssignableSlot
           Nom du Slot: elemTree:
            Contenu du Slot elemTree: : elemTree: = <-
            Nom du Slot: listCat_ElemTree
             Contenu du Slot listCat_ElemTree : listCat_ElemTree = a list('N1PrepNPivot', 'NAdj')
                 Ce slot est un AssignableSlot
            Nom du Slot: listCat_ElemTree:
             Contenu du Slot listCat_ElemTree: : listCat_ElemTree: = <-
           Nom du Slot: compTree
            Contenu du Slot compTree : compTree = an object
                 Ce slot est un AssignableSlot
           Nom du Slot: compTree:
            Contenu du Slot compTree: : compTree: = <-
           Nom du Slot: treeUp
            Contenu du Slot treeUp : treeUp = an object
                 Ce slot est un AssignableSlot
           Nom du Slot: treeUp:
            Contenu du Slot treeUp: : treeUp: = <-
           Nom du Slot: treeDown
            Contenu du Slot treeDown : treeDown = an object
                 Ce slot est un AssignableSlot
           Nom du Slot: treeDown:
            Contenu du Slot treeDown: : treeDown: = <-
           Nom du Slot: roleTree
            Contenu du Slot roleTree : roleTree = 'TermTree'
                 Ce slot est un AssignableSlot
           Nom du Slot: roleTree:
            Contenu du Slot roleTree: : roleTree: = <-
           Nom du Slot: elts
            Contenu du Slot elts : elts = a list('Nom')
                 Ce slot est un AssignableSlot
           Nom du Slot: elts:
           Contenu du Slot elts: : elts: = <-
           Nom du Slot: nbElts
           Contenu du Slot nbElts : nbElts = 1
                 Ce slot est un AssignableSlot
           Nom du Slot: nbElts:
            Contenu du Slot nbElts: : nbElts: = <-
           Nom du Slot: morphologie
            Contenu du Slot morphologie : morphologie = an object
                 Ce slot est un AssignableSlot
           Nom du Slot: morphologie:
            Contenu du Slot morphologie: : morphologie: = <-
           Nom du Slot: catName
            Contenu du Slot catName : catName = 'Nom'
                 Ce slot est un AssignableSlot
           Nom du Slot: catName:
            Contenu du Slot catName: : catName: = <-
           Nom du Slot: accord
```

```
            Contenu du Slot accord : accord = an object
                 Ce slot est un AssignableSlot
           Nom du Slot: accord:
            Contenu du Slot accord: : accord: = <-
            Nom du Slot: checkStructure
             Contenu du Slot checkStructure : checkStructure = true
                 Ce slot est un AssignableSlot
            Nom du Slot: checkStructure:
             Contenu du Slot checkStructure: : checkStructure: = <-
           Nom du Slot: semantique
            Contenu du Slot semantique : semantique = a list(an object)
                 Ce slot est un AssignableSlot
           Nom du Slot: semantique:
            Contenu du Slot semantique: : semantique: = <-
            Nom du Slot: addSemantique:
             Contenu du Slot addSemantique: : addSemantique: t = ( semantique addLast: t)
           Nom du Slot: resultCopy
            Contenu du Slot resultCopy : resultCopy = an object
                 Ce slot est un AssignableSlot
           Nom du Slot: resultCopy:
            Contenu du Slot resultCopy: : resultCopy: = <-
           Nom du Slot: copy
           Contenu du Slot copy : copy = (
     resultCopy: clone.
     resultCopy protoToBuild: nil.
      resultCopy morphologie: globals gaspar categorieGaspar morphologie copy.
     resultCopy semantique: list copy.
      (semantique = nil) || (semantique size = 0) || (semantique = '')
         ifFalse:[(0 to: ((semantique size) - 1)
                    Do:[ | :i |
                          resultCopy semantique addLast:
                                  (semantique at:i) copy.
                        ]).
                ].
     (accord = nil) || [accord = '']
             ifTrue:[
                       resultCopy accord: gaspar categorieGaspar accord copy.
                       resultCopy accord genre: gaspar categorieGaspar genre copy.
                       resultCopy accord nombre: gaspar categorieGaspar nombre copy.
                   ]
              False:[resultCopy accord: accord copy.
                     resultCopy accord genre:  accord genre copy.
                     resultCopy accord nombre: accord nombre copy.
                   ].

      resultCopy.
)
            Nom du Slot: inputSemantique
             Contenu du Slot inputSemantique : inputSemantique = a list(a list)
                 Ce slot est un AssignableSlot
            Nom du Slot: inputSemantique:
             Contenu du Slot inputSemantique: : inputSemantique: = <-
           Nom du Slot: inputMorph
            Contenu du Slot inputMorph : inputMorph = a list('lesion', a list, 'Nom')
                 Ce slot est un AssignableSlot
           Nom du Slot: inputMorph:
            Contenu du Slot inputMorph: : inputMorph: = <-
           Nom du Slot: inputForme
            Contenu du Slot inputForme : inputForme = 'lesion'
                 Ce slot est un AssignableSlot
           Nom du Slot: inputForme:
            Contenu du Slot inputForme: : inputForme: = <-
           Nom du Slot: inputLemme
            Contenu du Slot inputLemme : inputLemme = 'lesion'
                 Ce slot est un AssignableSlot
           Nom du Slot: inputLemme:
            Contenu du Slot inputLemme: : inputLemme: = <-
           Nom du Slot: inputFormeNue
            Contenu du Slot inputFormeNue : inputFormeNue = 'lesion'
                 Ce slot est un AssignableSlot
           Nom du Slot: inputFormeNue:
            Contenu du Slot inputFormeNue: : inputFormeNue: = <-
           Nom du Slot: inputGenre
            Contenu du Slot inputGenre : inputGenre = 'Fem'
                 Ce slot est un AssignableSlot
           Nom du Slot: inputGenre:
            Contenu du Slot inputGenre: : inputGenre: = <-
           Nom du Slot: inputNombre
            Contenu du Slot inputNombre : inputNombre = 'Sing'
                 Ce slot est un AssignableSlot
           Nom du Slot: inputNombre:
            Contenu du Slot inputNombre: : inputNombre: = <-
           Nom du Slot: inputClasse
            Contenu du Slot inputClasse : inputClasse = 'Nom A Pluriel en S'
                 Ce slot est un AssignableSlot
           Nom du Slot: inputClasse:
            Contenu du Slot inputClasse: : inputClasse: = <-
            Nom du Slot: checkSemDescription:
             Contenu du Slot checkSemDescription: : checkSemDescription: t = (
     (t = nil) ifTrue:[ inputSemantique: list copy.]
                False:[ inputSemantique: t.].
)
            Nom du Slot: checkMorphDescription:
             Contenu du Slot checkMorphDescription: : checkMorphDescription: t = (
     (t = nil) ifTrue:[ inputMorph: list copy.]
                False:[ inputMorph: t.].
     (inputMorph size = 0)
                ifTrue:[
                        inputForme: '-'.
                         inputFormeNue: '-'.
                        inputLemme: '-'.
                        inputGenre: '-'.
                        inputNombre: '-'.
                        inputClasse: '-'.
                       ].
     (inputMorph size = 1)
           ifTrue:[
                   inputForme: inputMorph at: 0.
                  inputFormeNue: '-'.
                  inputLemme: '-'.
                  inputGenre: '-'.
                  inputNombre: '-'.
                  inputClasse: '-'.
                 ].
```

```
     (inputMorph size != 0) && (inputMorph size != 1)
           ifTrue:[
                    inputForme: inputMorph at: 0.
                     inputFormeNue: (inputMorph at: 1) at: 0.
                     inputLemme: (inputMorph at: 1) at: 1.
                     inputGenre: (inputMorph at: 1) at: 2.
                     inputNombre: (inputMorph at: 1) at: 3.
                     inputClasse: (inputMorph at: 1) at: 4.
                  ].
     morphologie forme: inputForme.
     morphologie formeNue: inputFormeNue.
     morphologie lemme: inputLemme.
     (accord = '')  || ( accord = nil)
                      ifTrue:[accord: globals gaspar categorieGaspar accord copy.
                             accord genre:  globals gaspar categorieGaspar genre copy.
                             accord nombre:  globals gaspar categorieGaspar nombre copy.
                          ].
     accord genre type_genre: inputGenre.
     accord nombre type_nombre: inputNombre.
     morphologie classe: inputClasse.
)

Le Prototype : <lesion|Nom> a :

24 slot(s) de type AssignableSlot
0 slot(s) de type ArgumentSlot
1 slot(s) de type parentSlot

-------------------------------------------------------
<1: Display Meta Rules>
<2: Display Bugs Report>
<3: Build Delegation>
<4: Remove Delegation>
<5: Build Prototypes>
<6: Look For Delegation>
<7: Meta Trace>
<8: Add Meta Rule>
<9: Remove Meta Rule>
<10: Add Bugs Report>
<11: Remove Bugs Report>
<12: Display Slot Of Selected Prototypes>
<13: Menu AddSlots>
<14: Compteur de Formes lexicales>
<15: Empty Waste Gaspar>

Select a number or q to quit MenuMeta:
```

A-3-9.2.3. Rechercher des liens de délégation

Dans l'exemple qui suit on utilise l'interface méta construite pour rechercher des liens de délégation définis sur le prototype `<Nom: lesion>`. S'il en existe, on obtient une présentation détaillée des attributs délégués dans chacun des parents trouvés.

```
-------------------------------------------------------
<1: Display Meta Rules>
<2: Display Bugs Report>
<3: Build Delegation>
<4: Remove Delegation>
<5: Build Prototypes>
<6: Look For Delegation>
<7: Meta Trace>
<8: Add Meta Rule>
<9: Remove Meta Rule>
<10: Add Bugs Report>
<11: Remove Bugs Report>
<12: Display Slot Of Selected Prototypes>
<13: Menu AddSlots>
<14: Compteur de Formes lexicales>
<15: Empty Waste Gaspar>

Select a number or q to quit MenuMeta:6
-------------------------------------------------------
---------------------------------------------------------------------------------------------
Quel type de prototype pour la  selection
1. Prototypes/Arbres Construits (recherche sur forme lexicale)
2. Prototypes/Arbres Non Construits (recherche sur Categorie)
3. Tout type de Prototypes/Arbres
Votre Choix: 1


Recherche d'un Arbre Elementaire Construit
------------------------------------------
Forme de l'Arbre Elementaire Cherche: lesion
Categorie de tete de l'Arbre Elementaire Cherche: Nom
---------------------------------------------------------------------------------------------
 Prototype : ['lesion'|'Nom']
Selection de ce prototype
(repondre par y ou Y): y
--------------------------------------------------------------------------------------------
---------------------------------------------------------------------------------------------
 Prototype : ['lesion'|'Nom']
Selection de ce prototype
(repondre par y ou Y): n
--------------------------------------------------------------------------------------------
---------------------------------------------------------------------------------------------
 Prototype : ['lesion'|'Nom']
Selection de ce prototype
(repondre par y ou Y): n
--------------------------------------------------------------------------------------------
---------------------------------------------------------------------------------------------
 Prototype : ['lesion'|'Nom']
Selection de ce prototype
(repondre par y ou Y): n
--------------------------------------------------------------------------------------------
-------------------------------------------------------------------
```

```
Description des Slots Parents et Herites du Prototype : <lesion|Nom>
------------------------------------------------------------------
           Nom du Slot: parent
            Contenu du Slot parent : parent* = an object
                Ce slot est un ParentSlot
 ----------> Recherche des Slots Herites via ce Parent
-->             Nom du Slot: parent
-->              Contenu du Slot parent : parent* = traits clonable
-->             Nom du Slot: categorie
-->              Contenu du Slot categorie : categorie = an object
-->                   Ce slot est un AssignableSlot
-->             Nom du Slot: categorie:
-->              Contenu du Slot categorie: : categorie: = <-
-->             Nom du Slot: elementaryTree
-->              Contenu du Slot elementaryTree : elementaryTree = an object
-->                   Ce slot est un AssignableSlot
-->             Nom du Slot: elementaryTree:
-->              Contenu du Slot elementaryTree: : elementaryTree: = <-
-->             Nom du Slot: complexTree
-->              Contenu du Slot complexTree : complexTree = an object
-->                   Ce slot est un AssignableSlot
-->             Nom du Slot: complexTree:
-->              Contenu du Slot complexTree: : complexTree: = <-
-->             Nom du Slot: inside
-->              Contenu du Slot inside : inside = 'Traits catNom: comportements partages par tous les Noms'
-->             Nom du Slot: display
-->             Contenu du Slot display : display = (
      ('---------------CatNom display-----------\n') printLine.
      displayCategorie printLine.
     categorie display.
     displayMorph printLine.
     morphologie display.
     displayAccord printLine.
     accord display.
     displaySemantique printLine.
     (semantique _Eq: nil)
      ifFalse:[
                (0 to: ((semantique size) - 1)
                 Do: [ | :i |
                        ('     Semantique [', (i + 1) printString,'] attachee au Prototype N',morphologie
forme,': \n',
                        '     -------------------------------------- \n') print.
                       (semantique at: i) display.
                     ]).
             ].
      ('--------------- End  display-----------\n') printLine.

)
-->             Nom du Slot: file
-->             Contenu du Slot file : file = an object
-->                  Ce slot est un AssignableSlot
-->             Nom du Slot: file:
-->             Contenu du Slot file: : file: = <-
-->             Nom du Slot: uiDisplay:
-->             Contenu du Slot uiDisplay: : uiDisplay: t = (
    ui add: self At: t.
     (semantique _Eq: nil)
                   ifFalse:[ ui show: 'semantique' On:reflect:self.
                            (0 to: ((semantique size) - 1)
                              Do: [ | :i |

                                    (semantique at: i) uiDisplay: t.
                                  ]).
                        ].
     (morphologie _Eq: '')
                   ifFalse:[ ui show: 'morphologie' On:reflect:self.
                            "morphologie uiDisplay: t."
                        ].
     (catName _Eq: '')
                   ifFalse:[ui show: 'catName' On:reflect:self.
                            ].
     (accord _Eq: '')
                   ifFalse:[ui show: 'accord' On:reflect:self.
                           "accord uiDisplay: t."]
)
-->             Nom du Slot: displayOut:
-->             Contenu du Slot displayOut: : displayOut: t = (
    file: t.
     file write: ('---------------CatNom display-----------\n').
     file write: displayCategorie.
     categorie displayOut: file.
     file write: displayMorph.
     morphologie displayOut: file.
     file write: displayAccord.
     accord displayOut: file.
     file write: displaySemantique.
     (semantique _Eq: nil)
      ifFalse:[
                (0 to: ((semantique size) - 1)
                 Do: [ | :i |
                        file write: ('     Semantique [', (i + 1) printString,'] attachee au Prototype
N',morphologie forme,': \n',
                        '     -------------------------------------- \n').
                       (semantique at: i) displayOut: file.
                     ]).
             ].
     file write: ('--------------- End  display-----------\n').

)
           Nom du Slot: protoBuilt
            Contenu du Slot protoBuilt : protoBuilt = true
           Nom du Slot: protoBuilt:
            Contenu du Slot protoBuilt: : protoBuilt: = <-
           Nom du Slot: protoToBuild
            Contenu du Slot protoToBuild : protoToBuild = an object
           Nom du Slot: protoToBuild:
            Contenu du Slot protoToBuild: : protoToBuild: = <-
           Nom du Slot: elemTree
            Contenu du Slot elemTree : elemTree = a list(an object, an object)
           Nom du Slot: elemTree:
            Contenu du Slot elemTree: : elemTree: = <-
           Nom du Slot: listCat_ElemTree
             Contenu du Slot listCat_ElemTree : listCat_ElemTree = a list('N1PrepNPivot', 'NAdj')
           Nom du Slot: listCat_ElemTree:
```

```
             Contenu du Slot listCat_ElemTree: : listCat_ElemTree: = <-
            Nom du Slot: compTree
             Contenu du Slot compTree : compTree = an object
            Nom du Slot: compTree:
             Contenu du Slot compTree: : compTree: = <-
            Nom du Slot: treeUp
             Contenu du Slot treeUp : treeUp = an object
            Nom du Slot: treeUp:
             Contenu du Slot treeUp: : treeUp: = <-
            Nom du Slot: treeDown
             Contenu du Slot treeDown : treeDown = an object
            Nom du Slot: treeDown:
             Contenu du Slot treeDown: : treeDown: = <-
            Nom du Slot: roleTree
             Contenu du Slot roleTree : roleTree = 'TermTree'
            Nom du Slot: roleTree:
             Contenu du Slot roleTree: : roleTree: = <-
            Nom du Slot: elts
             Contenu du Slot elts : elts = a list('Nom')
            Nom du Slot: elts:
             Contenu du Slot elts: : elts: = <-
            Nom du Slot: nbElts
             Contenu du Slot nbElts : nbElts = 1
            Nom du Slot: nbElts:
             Contenu du Slot nbElts: : nbElts: = <-
            Nom du Slot: morphologie
             Contenu du Slot morphologie : morphologie = an object
            Nom du Slot: morphologie:
             Contenu du Slot morphologie: : morphologie: = <-
            Nom du Slot: catName
             Contenu du Slot catName : catName = 'Nom'
            Nom du Slot: catName:
             Contenu du Slot catName: : catName: = <-
            Nom du Slot: accord
             Contenu du Slot accord : accord = an object
            Nom du Slot: accord:
             Contenu du Slot accord: : accord: = <-
            Nom du Slot: checkStructure
             Contenu du Slot checkStructure : checkStructure = true
            Nom du Slot: checkStructure:
             Contenu du Slot checkStructure: : checkStructure: = <-
            Nom du Slot: semantique
             Contenu du Slot semantique : semantique = a list(an object)
            Nom du Slot: semantique:
             Contenu du Slot semantique: : semantique: = <-
            Nom du Slot: addSemantique:
             Contenu du Slot addSemantique: : addSemantique: t = ( semantique addLast: t)
            Nom du Slot: resultCopy
             Contenu du Slot resultCopy : resultCopy = an object
            Nom du Slot: resultCopy:
             Contenu du Slot resultCopy: : resultCopy: = <-
            Nom du Slot: copy
            Contenu du Slot copy : copy = (
     resultCopy: clone.
     resultCopy protoToBuild: nil.
     resultCopy morphologie: globals gaspar categorieGaspar morphologie copy.
     resultCopy semantique: list copy.
      (semantique = nil) || (semantique size = 0) || (semantique = '')
          ifFalse:[(0 to: ((semantique size) - 1)
                     Do:[ | :i |
                           resultCopy semantique addLast:
                                   (semantique at:i) copy.
                         ]).
                 ].
     (accord = nil) || [accord = '']
             ifTrue:[
                        resultCopy accord: gaspar categorieGaspar accord copy.
                        resultCopy accord genre: gaspar categorieGaspar genre copy.
                        resultCopy accord nombre: gaspar categorieGaspar nombre copy.
                    ]
              False:[resultCopy accord: accord copy.
                     resultCopy accord genre:  accord genre copy.
                     resultCopy accord nombre: accord nombre copy.
                    ].

      resultCopy.
)
             Nom du Slot: inputSemantique
              Contenu du Slot inputSemantique : inputSemantique = a list(a list)
             Nom du Slot: inputSemantique:
              Contenu du Slot inputSemantique: : inputSemantique: = <-
            Nom du Slot: inputMorph
             Contenu du Slot inputMorph : inputMorph = a list('lesion', a list, 'Nom')
            Nom du Slot: inputMorph:
             Contenu du Slot inputMorph: : inputMorph: = <-
            Nom du Slot: inputForme
             Contenu du Slot inputForme : inputForme = 'lesion'
            Nom du Slot: inputForme:
             Contenu du Slot inputForme: : inputForme: = <-
            Nom du Slot: inputLemme
             Contenu du Slot inputLemme : inputLemme = 'lesion'
            Nom du Slot: inputLemme:
             Contenu du Slot inputLemme: : inputLemme: = <-
            Nom du Slot: inputFormeNue
             Contenu du Slot inputFormeNue : inputFormeNue = 'lesion'
            Nom du Slot: inputFormeNue:
             Contenu du Slot inputFormeNue: : inputFormeNue: = <-
            Nom du Slot: inputGenre
             Contenu du Slot inputGenre : inputGenre = 'Fem'
            Nom du Slot: inputGenre:
             Contenu du Slot inputGenre: : inputGenre: = <-
            Nom du Slot: inputNombre
             Contenu du Slot inputNombre : inputNombre = 'Sing'
            Nom du Slot: inputNombre:
             Contenu du Slot inputNombre: : inputNombre: = <-
            Nom du Slot: inputClasse
             Contenu du Slot inputClasse : inputClasse = 'Nom A Pluriel en S'
            Nom du Slot: inputClasse:
             Contenu du Slot inputClasse: : inputClasse: = <-
             Nom du Slot: checkSemDescription:
              Contenu du Slot checkSemDescription: : checkSemDescription: t = (
     (t = nil) ifTrue:[ inputSemantique: list copy.]
                False:[ inputSemantique: t.].
)
```

```
            Nom du Slot: checkMorphDescription:
             Contenu du Slot checkMorphDescription: : checkMorphDescription: t = (
     (t = nil) ifTrue:[ inputMorph: list copy.]
                False:[ inputMorph: t.].
     (inputMorph size = 0)
                ifTrue:[
                         inputForme: '-'.
                          inputFormeNue: '-'.
                         inputLemme: '-'.
                         inputGenre: '-'.
                         inputNombre: '-'.
                         inputClasse: '-'.
                        ].
     (inputMorph size = 1)
           ifTrue:[
                    inputForme: inputMorph at: 0.
                   inputFormeNue: '-'.
                   inputLemme: '-'.
                   inputGenre: '-'.
                   inputNombre: '-'.
                   inputClasse: '-'.
                  ].
     (inputMorph size != 0) && (inputMorph size != 1)
           ifTrue:[
                     inputForme: inputMorph at: 0.
                      inputFormeNue: (inputMorph at: 1) at: 0.
                      inputLemme: (inputMorph at: 1) at: 1.
                      inputGenre: (inputMorph at: 1) at: 2.
                      inputNombre: (inputMorph at: 1) at: 3.
                      inputClasse: (inputMorph at: 1) at: 4.
                   ].
     morphologie forme: inputForme.
     morphologie formeNue: inputFormeNue.
     morphologie lemme: inputLemme.
     (accord = '')  || ( accord = nil)
                       ifTrue:[accord: globals gaspar categorieGaspar accord copy.
                               accord genre:  globals gaspar categorieGaspar genre copy.
                               accord nombre:  globals gaspar categorieGaspar nombre copy.
                           ].
     accord genre type_genre: inputGenre.
     accord nombre type_nombre: inputNombre.
     morphologie classe: inputClasse.
)

Le Prototype : <lesion|Nom> a :

1 slot(s) de type parentSlot

----------------------------------------------------------
<1: Display Meta Rules>
<2: Display Bugs Report>
<3: Build Delegation>
<4: Remove Delegation>
<5: Build Prototypes>
<6: Look For Delegation>
<7: Meta Trace>
<8: Add Meta Rule>
<9: Remove Meta Rule>
<10: Add Bugs Report>
<11: Remove Bugs Report>
<12: Display Slot Of Selected Prototypes>
<13: Menu AddSlots>
<14: Compteur de Formes lexicales>
<15: Empty Waste Gaspar>

Select a number or q to quit MenuMeta:
```

A-3-9.2.4. Ajouter des liens de délégation

Dans l'exemple qui suit on utilise l'interface méta pour construire un lien de délégation entre le prototype `<Adj: petite>` et le prototype `traits catAdj`. Cet objet est porteur des comportements communs à la catégorie des adjectifs. On associe donc à ce prototype d'adjectif l'objet porteur des comportements partagés par toute la famille des adjectifs.

```
----------------------------------------------------------
<1: Display Meta Rules>
<2: Display Bugs Report>
<3: Build Delegation>
<4: Remove Delegation>
<5: Build Prototypes>
<6: Look For Delegation>
<7: Meta Trace>
<8: Add Meta Rule>
<9: Remove Meta Rule>
<10: Add Bugs Report>
<11: Remove Bugs Report>
<12: Display Slot Of Selected Prototypes>
<13: Menu AddSlots>
<14: Compteur de Formes lexicales>
<15: Empty Waste Gaspar>

Select a number or q to quit MenuMeta:3
----------------------------------------------------------
----------------------------------------------------------------------------------------------------
Quel type de prototype pour la  selection
1. Prototypes/Arbres Construits (recherche sur forme lexicale)
2. Prototypes/Arbres Non Construits (recherche sur Categorie)
3. Tout type de Prototypes/Arbres
Votre Choix: 1


Recherche d'un Arbre Elementaire Construit
-------------------------------------------
Forme de l'Arbre Elementaire Cherche: petite
Categorie de tete de l'Arbre Elementaire Cherche: Adj
----------------------------------------------------------------------------------------------------
```

```
 Prototype  :  ['petite'|'Adj']
Selection  de  ce  prototype
(repondre  par  y  ou  Y):  y
-----------------------------------------------------------------------------------------------
------------------------------------------------------------------------------------------------
 Prototype  :  ['petite'|'Adj']
Selection  de  ce  prototype
(repondre  par  y  ou  Y):  n
-----------------------------------------------------------------------------------------------
------------------------------------------------------------------------------------------------
 Prototype  :  ['petite'|'Adj']
Selection  de  ce  prototype
(repondre  par  y  ou  Y):  n
-----------------------------------------------------------------------------------------------
------------------------------------------------------------------------------------------------
 Prototype  :  ['petite'|'Adj']
Selection  de  ce  prototype
(repondre  par  y  ou  Y):  n
-----------------------------------------------------------------------------------------------


Recherche  des  Objets  Traits
---------------------------

/..../

Objet  Trait  Disponible  :<Traits  catAdj:  comportements  partages  par  tous  les  Adj>
Selection  de  ce  Trait
(repondre  par  y  ou  Y):  y

/..../

------------------------------------------------------------------
Description  des  Slots  Parents  et  Herites  du  Prototype  :  <petite|Adj>
------------------------------------------------------------------
            Nom  du  Slot:  protoBuilt
             Contenu  du  Slot  protoBuilt  :  protoBuilt  =  true
            Nom  du  Slot:  protoBuilt:
             Contenu  du  Slot  protoBuilt:  :  protoBuilt:  =  <-
            Nom  du  Slot:  protoToBuild
             Contenu  du  Slot  protoToBuild  :  protoToBuild  =  an  object
            Nom  du  Slot:  protoToBuild:
             Contenu  du  Slot  protoToBuild:  :  protoToBuild:  =  <-
            Nom  du  Slot:  elemTree
             Contenu  du  Slot  elemTree  :  elemTree  =  an  object
            Nom  du  Slot:  elemTree:
             Contenu  du  Slot  elemTree:  :  elemTree:  =  <-
            Nom  du  Slot:  listCat_ElemTree
             Contenu  du  Slot  listCat_ElemTree  :  listCat_ElemTree  =  an  object
            Nom  du  Slot:  listCat_ElemTree:
             Contenu  du  Slot  listCat_ElemTree:  :  listCat_ElemTree:  =  <-
            Nom  du  Slot:  compTree
             Contenu  du  Slot  compTree  :  compTree  =  an  object
            Nom  du  Slot:  compTree:
             Contenu  du  Slot  compTree:  :  compTree:  =  <-
            Nom  du  Slot:  treeUp
             Contenu  du  Slot  treeUp  :  treeUp  =  an  object
            Nom  du  Slot:  treeUp:
             Contenu  du  Slot  treeUp:  :  treeUp:  =  <-
            Nom  du  Slot:  treeDown
             Contenu  du  Slot  treeDown  :  treeDown  =  an  object
            Nom  du  Slot:  treeDown:
             Contenu  du  Slot  treeDown:  :  treeDown:  =  <-
            Nom  du  Slot:  roleTree
             Contenu  du  Slot  roleTree  :  roleTree  =  'TermTree'
            Nom  du  Slot:  roleTree:
             Contenu  du  Slot  roleTree:  :  roleTree:  =  <-
            Nom  du  Slot:  elts
             Contenu  du  Slot  elts  :  elts  =  a  list('Adj')
            Nom  du  Slot:  elts:
             Contenu  du  Slot  elts:  :  elts:  =  <-
            Nom  du  Slot:  nbElts
             Contenu  du  Slot  nbElts  :  nbElts  =  1
            Nom  du  Slot:  nbElts:
             Contenu  du  Slot  nbElts:  :  nbElts:  =  <-
            Nom  du  Slot:  morphologie
             Contenu  du  Slot  morphologie  :  morphologie  =  an  object
            Nom  du  Slot:  morphologie:
             Contenu  du  Slot  morphologie:  :  morphologie:  =  <-
            Nom  du  Slot:  catName
             Contenu  du  Slot  catName  :  catName  =  'Adj'
            Nom  du  Slot:  catName:
             Contenu  du  Slot  catName:  :  catName:  =  <-
            Nom  du  Slot:  semantique
             Contenu  du  Slot  semantique  :  semantique  =  a  list
            Nom  du  Slot:  semantique:
             Contenu  du  Slot  semantique:  :  semantique:  =  <-
            Nom  du  Slot:  addSemantique:
             Contenu  du  Slot  addSemantique:  :  addSemantique:  t  =  (  semantique  addLast:  t)
            Nom  du  Slot:  accord
             Contenu  du  Slot  accord  :  accord  =  an  object
            Nom  du  Slot:  accord:
             Contenu  du  Slot  accord:  :  accord:  =  <-
            Nom  du  Slot:  checkStructure
             Contenu  du  Slot  checkStructure  :  checkStructure  =  true
            Nom  du  Slot:  checkStructure:
             Contenu  du  Slot  checkStructure:  :  checkStructure:  =  <-
            Nom  du  Slot:  resultCopy
             Contenu  du  Slot  resultCopy  :  resultCopy  =  an  object
            Nom  du  Slot:  resultCopy:
             Contenu  du  Slot  resultCopy:  :  resultCopy:  =  <-
            Nom  du  Slot:  copy
            Contenu  du  Slot  copy  :  copy  =  (
     resultCopy:  clone.
     resultCopy  protoToBuild:  nil.
      resultCopy  morphologie:  globals  gaspar  categorieGaspar  morphologie  copy.
     resultCopy  semantique:  list  copy.
      (semantique  =  nil)  ||  (semantique  size  =  0)  ||  (semantique  =  '')
         ifFalse:[(0  to:  ((semantique  size)  -  1)
                     Do:[  |  :i  |
                           resultCopy  semantique  addLast:
                                   (semantique  at:i)  copy.
                        ]).
               ].
```

```
      (accord  =  nil)  ||  [accord  =  '']
              ifTrue:[
                         resultCopy  accord:  gaspar  categorieGaspar  accord  copy.
                         resultCopy  accord  genre:  gaspar  categorieGaspar  genre  copy.
                         resultCopy  accord  nombre:  gaspar  categorieGaspar  nombre  copy.
                     ]
               False:[resultCopy  accord:  accord  copy.
                       resultCopy  accord  genre:   accord  genre  copy.
                       resultCopy  accord  nombre:  accord  nombre  copy.
                     ].

       resultCopy.
)
           Nom  du  Slot:  inputSemantique
            Contenu  du  Slot  inputSemantique  :  inputSemantique  =  an  object
           Nom  du  Slot:  inputSemantique:
            Contenu  du  Slot  inputSemantique:  :  inputSemantique:  =  <-
           Nom  du  Slot:  inputMorph
            Contenu  du  Slot  inputMorph  :  inputMorph  =  an  object
           Nom  du  Slot:  inputMorph:
            Contenu  du  Slot  inputMorph:  :  inputMorph:  =  <-
           Nom  du  Slot:  inputForme
            Contenu  du  Slot  inputForme  :  inputForme  =  an  object
           Nom  du  Slot:  inputForme:
            Contenu  du  Slot  inputForme:  :  inputForme:  =  <-
           Nom  du  Slot:  inputLemme
            Contenu  du  Slot  inputLemme  :  inputLemme  =  an  object
           Nom  du  Slot:  inputLemme:
            Contenu  du  Slot  inputLemme:  :  inputLemme:  =  <-
           Nom  du  Slot:  inputFormeNue
            Contenu  du  Slot  inputFormeNue  :  inputFormeNue  =  an  object
           Nom  du  Slot:  inputFormeNue:
            Contenu  du  Slot  inputFormeNue:  :  inputFormeNue:  =  <-
           Nom  du  Slot:  inputGenre
            Contenu  du  Slot  inputGenre  :  inputGenre  =  an  object
           Nom  du  Slot:  inputGenre:
            Contenu  du  Slot  inputGenre:  :  inputGenre:  =  <-
           Nom  du  Slot:  inputNombre
            Contenu  du  Slot  inputNombre  :  inputNombre  =  an  object
           Nom  du  Slot:  inputNombre:
            Contenu  du  Slot  inputNombre:  :  inputNombre:  =  <-
           Nom  du  Slot:  inputClasse
            Contenu  du  Slot  inputClasse  :  inputClasse  =  an  object
           Nom  du  Slot:  inputClasse:
            Contenu  du  Slot  inputClasse:  :  inputClasse:  =  <-
           Nom  du  Slot:  checkSemDescription:
            Contenu  du  Slot  checkSemDescription:  :  checkSemDescription:  t  =  (
      (t  =  nil)  ifTrue:[  inputSemantique:  list  copy.]
                    False:[  inputSemantique:  t.].
)
           Nom  du  Slot:  checkMorphDescription:
            Contenu  du  Slot  checkMorphDescription:  :  checkMorphDescription:  t  =  (
      (t  =  nil)  ifTrue:[  inputMorph:  list  copy.]
                    False:[  inputMorph:  t.].
      (inputMorph  size  =  0)
                    ifTrue:[
                             inputForme:  '-'.
                              inputFormeNue:  '-'.
                             inputLemme:  '-'.
                             inputGenre:  '-'.
                              inputNombre:  '-'.
                        inputClasse:  '-'.
                             ].
      (inputMorph  size  =  1)
              ifTrue:[
                         inputForme:  inputMorph  at:  0.
                        inputFormeNue:  '-'.
                        inputLemme:  '-'.
                        inputGenre:  '-'.
                        inputNombre:  '-'.
                        inputClasse:  '-'.
                       ].
      (inputMorph  size  !=  0)  &&  (inputMorph  size  !=  1)
              ifTrue:[
                          inputForme:  inputMorph  at:  0.
                           inputFormeNue:  (inputMorph  at:  1)  at:  0.
                           inputLemme:  (inputMorph  at:  1)  at:  1.
                           inputGenre:  (inputMorph  at:  1)  at:  2.
                           inputNombre:  (inputMorph  at:  1)  at:  3.
                           inputClasse:  (inputMorph  at:  1)  at:  4.
                        ].
      morphologie  forme:  inputForme.
      morphologie  formeNue:  inputFormeNue.
      morphologie  lemme:  inputLemme.
      (accord  =  '')  ||  (  accord  =  nil)
                         ifTrue:[accord:  globals  gaspar  categorieGaspar  accord  copy.
                                 accord  genre:   globals  gaspar  categorieGaspar  genre  copy.
                                 accord  nombre:   globals  gaspar  categorieGaspar  nombre  copy.
                             ].
      accord  genre  type_genre:  inputGenre.
      accord  nombre  type_nombre:  inputNombre.
      morphologie  classe:  inputClasse.
)
```

Le Prototype : <petite|Adj> a :

0 slot(s) de type parentSlot

```
-->0
/home/fleury/Self/source/Self-4.0/Phase2/work/buildDelegation.self
/home/fleury/Self/source/Self-4.0/Phase2/work/buildDelegation.self
reading  /home/fleury/Self/source/Self-4.0/Phase2/work/buildDelegation.self...

------------------------------------------------------------------
Description  des  Slots  Parents  et  Herites  du  Prototype  :  <petite|Adj>
------------------------------------------------------------------
            Nom  du  Slot:  protoBuilt
             Contenu  du  Slot  protoBuilt  :  protoBuilt  =  true
            Nom  du  Slot:  protoBuilt:
             Contenu  du  Slot  protoBuilt:  :  protoBuilt:  =  <-
            Nom  du  Slot:  protoToBuild
             Contenu  du  Slot  protoToBuild  :  protoToBuild  =  an  object
            Nom  du  Slot:  protoToBuild:
             Contenu  du  Slot  protoToBuild:  :  protoToBuild:  =  <-
```

```
            Nom du Slot: elemTree
             Contenu du Slot elemTree : elemTree = an object
            Nom du Slot: elemTree:
             Contenu du Slot elemTree: : elemTree: = <-
             Nom du Slot: listCat_ElemTree
              Contenu du Slot listCat_ElemTree : listCat_ElemTree = an object
             Nom du Slot: listCat_ElemTree:
              Contenu du Slot listCat_ElemTree: : listCat_ElemTree: = <-
            Nom du Slot: compTree
             Contenu du Slot compTree : compTree = an object
            Nom du Slot: compTree:
             Contenu du Slot compTree: : compTree: = <-
            Nom du Slot: treeUp
             Contenu du Slot treeUp : treeUp = an object
            Nom du Slot: treeUp:
             Contenu du Slot treeUp: : treeUp: = <-
            Nom du Slot: treeDown
             Contenu du Slot treeDown : treeDown = an object
            Nom du Slot: treeDown:
             Contenu du Slot treeDown: : treeDown: = <-
            Nom du Slot: roleTree
             Contenu du Slot roleTree : roleTree = 'TermTree'
            Nom du Slot: roleTree:
             Contenu du Slot roleTree: : roleTree: = <-
            Nom du Slot: elts
             Contenu du Slot elts : elts = a list('Adj')
            Nom du Slot: elts:
             Contenu du Slot elts: : elts: = <-
            Nom du Slot: nbElts
             Contenu du Slot nbElts : nbElts = 1
            Nom du Slot: nbElts:
             Contenu du Slot nbElts: : nbElts: = <-
            Nom du Slot: morphologie
             Contenu du Slot morphologie : morphologie = an object
            Nom du Slot: morphologie:
             Contenu du Slot morphologie: : morphologie: = <-
            Nom du Slot: catName
             Contenu du Slot catName : catName = 'Adj'
            Nom du Slot: catName:
             Contenu du Slot catName: : catName: = <-
            Nom du Slot: semantique
             Contenu du Slot semantique : semantique = a list
            Nom du Slot: semantique:
             Contenu du Slot semantique: : semantique: = <-
            Nom du Slot: addSemantique:
             Contenu du Slot addSemantique: : addSemantique: t = ( semantique addLast: t)
            Nom du Slot: accord
             Contenu du Slot accord : accord = an object
            Nom du Slot: accord:
             Contenu du Slot accord: : accord: = <-
            Nom du Slot: checkStructure
             Contenu du Slot checkStructure : checkStructure = true
            Nom du Slot: checkStructure:
             Contenu du Slot checkStructure: : checkStructure: = <-
            Nom du Slot: resultCopy
             Contenu du Slot resultCopy : resultCopy = an object
            Nom du Slot: resultCopy:
             Contenu du Slot resultCopy: : resultCopy: = <-
            Nom du Slot: copy
            Contenu du Slot copy : copy = (
     resultCopy: clone.
     resultCopy protoToBuild: nil.
     resultCopy morphologie: globals gaspar categorieGaspar morphologie copy.
     resultCopy semantique: list copy.
      (semantique = nil) || (semantique size = 0) || (semantique = '')
         ifFalse:[(0 to: ((semantique size) - 1)
                    Do:[ | :i |
                          resultCopy semantique addLast:
                                  (semantique at:i) copy.
                        ]).
               ].
     (accord = nil) || [accord = '']
            ifTrue:[
                      resultCopy accord: gaspar categorieGaspar accord copy.
                      resultCopy accord genre: gaspar categorieGaspar genre copy.
                      resultCopy accord nombre: gaspar categorieGaspar nombre copy.
                  ]
             False:[resultCopy accord: accord copy.
                    resultCopy accord genre:  accord genre copy.
                    resultCopy accord nombre: accord nombre copy.
                  ].

      resultCopy.
)
             Nom du Slot: inputSemantique
              Contenu du Slot inputSemantique : inputSemantique = an object
             Nom du Slot: inputSemantique:
              Contenu du Slot inputSemantique: : inputSemantique: = <-
            Nom du Slot: inputMorph
             Contenu du Slot inputMorph : inputMorph = an object
            Nom du Slot: inputMorph:
             Contenu du Slot inputMorph: : inputMorph: = <-
            Nom du Slot: inputForme
             Contenu du Slot inputForme : inputForme = an object
            Nom du Slot: inputForme:
             Contenu du Slot inputForme: : inputForme: = <-
            Nom du Slot: inputLemme
             Contenu du Slot inputLemme : inputLemme = an object
            Nom du Slot: inputLemme:
             Contenu du Slot inputLemme: : inputLemme: = <-
            Nom du Slot: inputFormeNue
             Contenu du Slot inputFormeNue : inputFormeNue = an object
            Nom du Slot: inputFormeNue:
             Contenu du Slot inputFormeNue: : inputFormeNue: = <-
            Nom du Slot: inputGenre
             Contenu du Slot inputGenre : inputGenre = an object
            Nom du Slot: inputGenre:
             Contenu du Slot inputGenre: : inputGenre: = <-
            Nom du Slot: inputNombre
             Contenu du Slot inputNombre : inputNombre = an object
            Nom du Slot: inputNombre:
             Contenu du Slot inputNombre: : inputNombre: = <-
            Nom du Slot: inputClasse
             Contenu du Slot inputClasse : inputClasse = an object
```

```
            Nom du Slot: inputClasse:
             Contenu du Slot inputClasse: : inputClasse: = <-
             Nom du Slot: checkSemDescription:
              Contenu du Slot checkSemDescription: : checkSemDescription: t = (
     (t = nil) ifTrue:[ inputSemantique: list copy.]
                 False:[ inputSemantique: t.].
)
             Nom du Slot: checkMorphDescription:
              Contenu du Slot checkMorphDescription: : checkMorphDescription: t = (
     (t = nil) ifTrue:[ inputMorph: list copy.]
                 False:[ inputMorph: t.].
     (inputMorph size = 0)
                 ifTrue:[
                          inputForme: '-'.
                           inputFormeNue: '-'.
                          inputLemme: '-'.
                          inputGenre: '-'.
                          inputNombre: '-'.
                    inputClasse: '-'.
                          ].
     (inputMorph size = 1)
           ifTrue:[
                    inputForme: inputMorph at: 0.
                   inputFormeNue: '-'.
                   inputLemme: '-'.
                   inputGenre: '-'.
                   inputNombre: '-'.
                   inputClasse: '-'.
                  ].
     (inputMorph size != 0) && (inputMorph size != 1)
           ifTrue:[
                     inputForme: inputMorph at: 0.
                      inputFormeNue: (inputMorph at: 1) at: 0.
                      inputLemme: (inputMorph at: 1) at: 1.
                      inputGenre: (inputMorph at: 1) at: 2.
                      inputNombre: (inputMorph at: 1) at: 3.
                      inputClasse: (inputMorph at: 1) at: 4.
                   ].
     morphologie forme: inputForme.
     morphologie formeNue: inputFormeNue.
     morphologie lemme: inputLemme.
     (accord = '') || ( accord = nil)
                       ifTrue:[accord: globals gaspar categorieGaspar accord copy.
                              accord genre:  globals gaspar categorieGaspar genre copy.
                              accord nombre:  globals gaspar categorieGaspar nombre copy.
                           ].
     accord genre type_genre: inputGenre.
     accord nombre type_nombre: inputNombre.
     morphologie classe: inputClasse.
)
            Nom du Slot: parentNumber
             Contenu du Slot parentNumber : parentNumber = 0
            Nom du Slot: parentNumber:
             Contenu du Slot parentNumber: : parentNumber: = <-
            Nom du Slot: parent0
             Contenu du Slot parent0 : parent0* = an object
                 Ce slot est un ParentSlot
 ----------> Recherche des Slots Herites via ce Parent
-->            Nom du Slot: parent
-->             Contenu du Slot parent : parent* = traits clonable
-->            Nom du Slot: categorie
-->             Contenu du Slot categorie : categorie = an object
-->                  Ce slot est un AssignableSlot
-->            Nom du Slot: categorie:
-->             Contenu du Slot categorie: : categorie: = <-
-->            Nom du Slot: elementaryTree
-->             Contenu du Slot elementaryTree : elementaryTree = an object
-->                  Ce slot est un AssignableSlot
-->            Nom du Slot: elementaryTree:
-->             Contenu du Slot elementaryTree: : elementaryTree: = <-
-->            Nom du Slot: complexTree
-->             Contenu du Slot complexTree : complexTree = an object
-->                  Ce slot est un AssignableSlot
-->            Nom du Slot: complexTree:
-->             Contenu du Slot complexTree: : complexTree: = <-
-->            Nom du Slot: displayLigne
-->             Contenu du Slot displayLigne : displayLigne =
'..............................................
'
-->            Nom du Slot: displayDoubleLigne
-->             Contenu du Slot displayDoubleLigne : displayDoubleLigne =
'::::::::::::::::::::::::::::::::::::::::::::::
'
-->            Nom du Slot: displayNewLine
-->             Contenu du Slot displayNewLine : displayNewLine = '
'
-->             Nom du Slot: displayPrototypeLexical
-->              Contenu du Slot displayPrototypeLexical : displayPrototypeLexical = 'Prototype Lexical:
'
-->            Nom du Slot: displayConstituants
-->             Contenu du Slot displayConstituants : displayConstituants = 'Constituant:
'
-->            Nom du Slot: displayCategorie
-->             Contenu du Slot displayCategorie : displayCategorie = '    Categorie:
'
-->            Nom du Slot: displayMorph
-->             Contenu du Slot displayMorph : displayMorph = '    Morph:
 '
-->            Nom du Slot: displaySemantique
-->             Contenu du Slot displaySemantique : displaySemantique = '    Semantique:
'
-->            Nom du Slot: displayAccord
-->             Contenu du Slot displayAccord : displayAccord = '    Accord: '
-->            Nom du Slot: inside
-->             Contenu du Slot inside : inside = 'Traits catAdj: comportements partages par tous les Adj'
-->            Nom du Slot: display
-->            Contenu du Slot display : display = (
      ('---------------CatAdj display-----------\n') printLine.
       displayCategorie printLine.
      categorie display.
      displayMorph printLine.
      morphologie display.
      displayAccord printLine.
     accord display.
```

```
        ('--------------- End   display-----------\n') printLine.
)
-->           Nom du Slot: file
-->            Contenu du Slot file : file = an object
-->                 Ce slot est un AssignableSlot
-->           Nom du Slot: file:
-->            Contenu du Slot file: : file: = <-
-->            Nom du Slot: uiDisplay:
-->             Contenu du Slot uiDisplay: : uiDisplay: t = (
     ui add: self At: t.
      (semantique _Eq: nil)
                     ifFalse:[ ui show: 'semantique' On:reflect:self.
                               (0 to: ((semantique size) - 1)
                                 Do: [ | :i |
                                       (semantique at: i) uiDisplay: t.
                                    ]).
                            ].
      (morphologie _Eq: '')
                     ifFalse:[ ui show: 'morphologie' On:reflect:self.
                               "morphologie uiDisplay: t."
                            ].
      (catName _Eq: '')
                     ifFalse:[ui show: 'catName' On:reflect:self.
                               ].
      (accord _Eq: '')
                     ifFalse:[ui show: 'accord' On:reflect:self.
                                "accord uiDisplay:t."
                               ].
)
-->            Nom du Slot: displayOut:
-->             Contenu du Slot displayOut: : displayOut: t = (
     file: t.
      file write: ('---------------CatAdj display-----------\n').
      file write: displayCategorie.
      categorie displayOut: file.
      file write: displayMorph.
      morphologie displayOut: file.
      file write: displayAccord.
      accord displayOut: file.
      file write: ('--------------- End   display-----------\n').
)
           Nom du Slot: parent0:
            Contenu du Slot parent0: : parent0: = <-

Le Prototype : <petite|Adj> a :

1 slot(s) de type parentSlot

-------------------------------------------------------
<1: Display Meta Rules>
<2: Display Bugs Report>
<3: Build Delegation>
<4: Remove Delegation>
<5: Build Prototypes>
<6: Look For Delegation>
<7: Meta Trace>
<8: Add Meta Rule>
<9: Remove Meta Rule>
<10: Add Bugs Report>
<11: Remove Bugs Report>
<12: Display Slot Of Selected Prototypes>
<13: Menu AddSlots>
<14: Compteur de Formes lexicales>
<15: Empty Waste Gaspar>

Select a number or q to quit MenuMeta:
```

A-3-9.2.5. Supprimer des liens de délégation

Dans l'exemple qui suit on utilise l'interface méta pour supprimer le lien de délégation créé supra entre le prototype <`Adj: petite`> et le prototype `traits catAdj`.

```
-------------------------------------------------------
<1: Display Meta Rules>
<2: Display Bugs Report>
<3: Build Delegation>
<4: Remove Delegation>
<5: Build Prototypes>
<6: Look For Delegation>
<7: Meta Trace>
<8: Add Meta Rule>
<9: Remove Meta Rule>
<10: Add Bugs Report>
<11: Remove Bugs Report>
<12: Display Slot Of Selected Prototypes>
<13: Menu AddSlots>
<14: Compteur de Formes lexicales>
<15: Empty Waste Gaspar>

Select a number or q to quit MenuMeta:4
-------------------------------------------------------
-----------------------------------------------------------------------------------------------
Quel type de prototype pour la  selection
1. Prototypes/Arbres Construits (recherche sur forme lexicale)
2. Prototypes/Arbres Non Construits (recherche sur Categorie)
3. Tout type de Prototypes/Arbres
Votre Choix: 1


Recherche d'un Arbre Elementaire Construit
------------------------------------------
Forme de l'Arbre Elementaire Cherche: petite
Categorie de tete de l'Arbre Elementaire Cherche: Adj
-----------------------------------------------------------------------------------------------
 Prototype : ['petite'|'Adj']
Selection de ce prototype
(repondre par y ou Y): y
----------------------------------------------------------------------------------------------
```

```
------------------------------------------------------------------
Description des Slots Parents et Herites du Prototype : <petite|Adj>
------------------------------------------------------------------
           Nom du Slot: parent
            Contenu du Slot parent : parent* = an object
                 Ce slot est un ParentSlot
 ----------> Recherche des Slots Herites via ce Parent
-->            Nom du Slot: parent
-->             Contenu du Slot parent : parent* = traits clonable
-->            Nom du Slot: categorie
-->             Contenu du Slot categorie : categorie = an object
-->                  Ce slot est un AssignableSlot
-->            Nom du Slot: categorie:
-->             Contenu du Slot categorie: : categorie: = <-
-->            Nom du Slot: elementaryTree
-->             Contenu du Slot elementaryTree : elementaryTree = an object
-->                  Ce slot est un AssignableSlot
-->            Nom du Slot: elementaryTree:
-->             Contenu du Slot elementaryTree: : elementaryTree: = <-
-->            Nom du Slot: complexTree
-->             Contenu du Slot complexTree : complexTree = an object
-->                  Ce slot est un AssignableSlot
-->            Nom du Slot: complexTree:
-->             Contenu du Slot complexTree: : complexTree: = <-
-->            Nom du Slot: displayLigne
-->             Contenu du Slot displayLigne : displayLigne =
'..................................................
'
-->            Nom du Slot: displayDoubleLigne
-->             Contenu du Slot displayDoubleLigne : displayDoubleLigne =
'::::::::::::::::::::::::::::::::::::::::::::::::::
'
-->            Nom du Slot: displayNewLine
-->             Contenu du Slot displayNewLine : displayNewLine = '
'
-->             Nom du Slot: displayPrototypeLexical
-->              Contenu du Slot displayPrototypeLexical : displayPrototypeLexical = 'Prototype Lexical:
'
-->            Nom du Slot: displayConstituants
-->             Contenu du Slot displayConstituants : displayConstituants = 'Constituant:
'
-->            Nom du Slot: displayCategorie
-->             Contenu du Slot displayCategorie : displayCategorie = '    Categorie:
'
-->            Nom du Slot: displayMorph
-->             Contenu du Slot displayMorph : displayMorph = '    Morph:
 '
-->            Nom du Slot: displaySemantique
-->             Contenu du Slot displaySemantique : displaySemantique = '    Semantique:
'
-->            Nom du Slot: displayAccord
-->             Contenu du Slot displayAccord : displayAccord = '    Accord: '

-->             Contenu du Slot inside : inside = 'Traits catAdj: comportements partages par tous les Adj'
-->            Nom du Slot: display
-->             Contenu du Slot display : display = (
      ('---------------CatAdj display-----------\n') printLine.
        displayCategorie printLine.
      categorie display.
      displayMorph printLine.
      morphologie display.
      displayAccord printLine.
      accord display.
        ('--------------- End   display-----------\n') printLine.
)
-->           Nom du Slot: file
-->            Contenu du Slot file : file = an object
-->                 Ce slot est un AssignableSlot
-->           Nom du Slot: file:
-->            Contenu du Slot file: : file: = <-
-->            Nom du Slot: uiDisplay:
-->             Contenu du Slot uiDisplay: : uiDisplay: t = (
     ui add: self At: t.
      (semantique _Eq: nil)
                     ifFalse:[ ui show: 'semantique' On:reflect:self.
                               (0 to: ((semantique size) - 1)
                                 Do: [ | :i |
                                       (semantique at: i) uiDisplay: t.
                                    ]).
                            ].
      (morphologie _Eq: '')
                     ifFalse:[ ui show: 'morphologie' On:reflect:self.
                               "morphologie uiDisplay: t."
                            ].
      (catName _Eq: '')
                     ifFalse:[ui show: 'catName' On:reflect:self.
                               ].
      (accord _Eq: '')
                     ifFalse:[ui show: 'accord' On:reflect:self.
                                "accord uiDisplay:t."
                               ].
)
-->            Nom du Slot: displayOut:
-->             Contenu du Slot displayOut: : displayOut: t = (
     file: t.
      file write: ('---------------CatAdj display-----------\n').
      file write: displayCategorie.
      categorie displayOut: file.
      file write: displayMorph.
      morphologie displayOut: file.
      file write: displayAccord.
      accord displayOut: file.
      file write: ('--------------- End   display-----------\n').
)

-------------------------
Suppression de cet attribut parent
(repondre par y ou Y): y
-------------------------

           Nom du Slot: protoBuilt
            Contenu du Slot protoBuilt : protoBuilt = true
           Nom du Slot: protoBuilt:
            Contenu du Slot protoBuilt: : protoBuilt: = <-
```

```
           Nom du Slot: protoToBuild
            Contenu du Slot protoToBuild : protoToBuild = an object
           Nom du Slot: protoToBuild:
            Contenu du Slot protoToBuild: : protoToBuild: = <-
           Nom du Slot: elemTree
            Contenu du Slot elemTree : elemTree = an object
           Nom du Slot: elemTree:
            Contenu du Slot elemTree: : elemTree: = <-
           Nom du Slot: listCat_ElemTree
            Contenu du Slot listCat_ElemTree : listCat_ElemTree = an object
           Nom du Slot: listCat_ElemTree:
            Contenu du Slot listCat_ElemTree: : listCat_ElemTree: = <-
           Nom du Slot: compTree
            Contenu du Slot compTree : compTree = an object
           Nom du Slot: compTree:
            Contenu du Slot compTree: : compTree: = <-
           Nom du Slot: treeUp
            Contenu du Slot treeUp : treeUp = an object
           Nom du Slot: treeUp:
            Contenu du Slot treeUp: : treeUp: = <-
           Nom du Slot: treeDown
            Contenu du Slot treeDown : treeDown = an object
           Nom du Slot: treeDown:
            Contenu du Slot treeDown: : treeDown: = <-
           Nom du Slot: roleTree
            Contenu du Slot roleTree : roleTree = 'TermTree'
           Nom du Slot: roleTree:
            Contenu du Slot roleTree: : roleTree: = <-
           Nom du Slot: elts
            Contenu du Slot elts : elts = a list('Adj')
           Nom du Slot: elts:
            Contenu du Slot elts: : elts: = <-
           Nom du Slot: nbElts
            Contenu du Slot nbElts : nbElts = 1
           Nom du Slot: nbElts:
            Contenu du Slot nbElts: : nbElts: = <-
           Nom du Slot: morphologie
            Contenu du Slot morphologie : morphologie = an object
           Nom du Slot: morphologie:
            Contenu du Slot morphologie: : morphologie: = <-
           Nom du Slot: catName
            Contenu du Slot catName : catName = 'Adj'
           Nom du Slot: catName:
            Contenu du Slot catName: : catName: = <-
           Nom du Slot: semantique
            Contenu du Slot semantique : semantique = a list
           Nom du Slot: semantique:
            Contenu du Slot semantique: : semantique: = <-
           Nom du Slot: addSemantique:
            Contenu du Slot addSemantique: : addSemantique: t = ( semantique addLast: t)
           Nom du Slot: accord
            Contenu du Slot accord : accord = an object
           Nom du Slot: accord:
            Contenu du Slot accord: : accord: = <-
           Nom du Slot: checkStructure
            Contenu du Slot checkStructure : checkStructure = true
           Nom du Slot: checkStructure:
            Contenu du Slot checkStructure: : checkStructure: = <-
           Nom du Slot: resultCopy
            Contenu du Slot resultCopy : resultCopy = an object
           Nom du Slot: resultCopy:
            Contenu du Slot resultCopy: : resultCopy: = <-
           Nom du Slot: copy
           Contenu du Slot copy : copy = (
     resultCopy: clone.
     resultCopy protoToBuild: nil.
      resultCopy morphologie: globals gaspar categorieGaspar morphologie copy.
     resultCopy semantique: list copy.
       (semantique = nil) || (semantique size = 0) || (semantique = '')
          ifFalse:[(0 to: ((semantique size) - 1)
                     Do:[ | :i |
                          resultCopy semantique addLast:
                                  (semantique at:i) copy.
                        ]).
                ].
     (accord = nil) || [accord = '']
             ifTrue:[
                       resultCopy accord: gaspar categorieGaspar accord copy.
                       resultCopy accord genre: gaspar categorieGaspar genre copy.
                       resultCopy accord nombre: gaspar categorieGaspar nombre copy.
                   ]
              False:[resultCopy accord: accord copy.
                     resultCopy accord genre:  accord genre copy.
                     resultCopy accord nombre: accord nombre copy.
                   ].

      resultCopy.
)
           Nom du Slot: inputSemantique
            Contenu du Slot inputSemantique : inputSemantique = an object
           Nom du Slot: inputSemantique:
            Contenu du Slot inputSemantique: : inputSemantique: = <-
           Nom du Slot: inputMorph
            Contenu du Slot inputMorph : inputMorph = an object
           Nom du Slot: inputMorph:
            Contenu du Slot inputMorph: : inputMorph: = <-
           Nom du Slot: inputForme
            Contenu du Slot inputForme : inputForme = an object
           Nom du Slot: inputForme:
            Contenu du Slot inputForme: : inputForme: = <-
           Nom du Slot: inputLemme
            Contenu du Slot inputLemme : inputLemme = an object
           Nom du Slot: inputLemme:
            Contenu du Slot inputLemme: : inputLemme: = <-
           Nom du Slot: inputFormeNue
            Contenu du Slot inputFormeNue : inputFormeNue = an object
           Nom du Slot: inputFormeNue:
            Contenu du Slot inputFormeNue: : inputFormeNue: = <-
           Nom du Slot: inputGenre
            Contenu du Slot inputGenre : inputGenre = an object
           Nom du Slot: inputGenre:
            Contenu du Slot inputGenre: : inputGenre: = <-
           Nom du Slot: inputNombre
            Contenu du Slot inputNombre : inputNombre = an object
```

```
           Nom du Slot: inputNombre:
            Contenu du Slot inputNombre: : inputNombre: = <-
           Nom du Slot: inputClasse
            Contenu du Slot inputClasse : inputClasse = an object
           Nom du Slot: inputClasse:
            Contenu du Slot inputClasse: : inputClasse: = <-
            Nom du Slot: checkSemDescription:
             Contenu du Slot checkSemDescription: : checkSemDescription: t = (
     (t = nil) ifTrue:[ inputSemantique: list copy.]
                False:[ inputSemantique: t.].
)
            Nom du Slot: checkMorphDescription:
             Contenu du Slot checkMorphDescription: : checkMorphDescription: t = (
     (t = nil) ifTrue:[ inputMorph: list copy.]
                False:[ inputMorph: t.].
     (inputMorph size = 0)
                ifTrue:[
                        inputForme: '-'.
                         inputFormeNue: '-'.
                        inputLemme: '-'.
                        inputGenre: '-'.
                        inputNombre: '-'.
                  inputClasse: '-'.
                       ].
     (inputMorph size = 1)
           ifTrue:[
                   inputForme: inputMorph at: 0.
                  inputFormeNue: '-'.
                  inputLemme: '-'.
                  inputGenre: '-'.
                  inputNombre: '-'.
                  inputClasse: '-'.
                 ].
     (inputMorph size != 0) && (inputMorph size != 1)
           ifTrue:[
                     inputForme: inputMorph at: 0.
                    inputFormeNue: (inputMorph at: 1) at: 0.
                    inputLemme: (inputMorph at: 1) at: 1.
                    inputGenre: (inputMorph at: 1) at: 2.
                    inputNombre: (inputMorph at: 1) at: 3.
                    inputClasse: (inputMorph at: 1) at: 4.
                  ].
     morphologie forme: inputForme.
     morphologie formeNue: inputFormeNue.
     morphologie lemme: inputLemme.
     (accord = '') || ( accord = nil)
                      ifTrue:[accord: globals gaspar categorieGaspar accord copy.
                             accord genre:  globals gaspar categorieGaspar genre copy.
                             accord nombre:  globals gaspar categorieGaspar nombre copy.
                          ].
     accord genre type_genre: inputGenre.
     accord nombre type_nombre: inputNombre.
     morphologie classe: inputClasse.
)

Le Prototype : <petite|Adj> a :

0 slot(s) de type parentSlot

-------------------------------------------------------
<1: Display Meta Rules>
<2: Display Bugs Report>
<3: Build Delegation>
<4: Remove Delegation>
<5: Build Prototypes>
<6: Look For Delegation>
<7: Meta Trace>
<8: Add Meta Rule>
<9: Remove Meta Rule>
<10: Add Bugs Report>
<11: Remove Bugs Report>
<12: Display Slot Of Selected Prototypes>
<13: Menu AddSlots>
<14: Compteur de Formes lexicales>
<15: Empty Waste Gaspar>

Select a number or q to quit MenuMeta:
```

A-3-9.2.6. Maintenir une cohérence dans les savoirs représentés

Dans l'exemple qui suit on utilise l'interface méta pour rechercher tous les objets définis via `Gaspar`. On examine ensuite leur structure interne et soit on garde ces objets (leur structure est cohérente) soit on les détruit (leur structure n'est pas complète). En fait chaque objet défini via Gaspar est porteur dès sa contruction d'un attribut qui indiquera la cohérence de l'objet visé. C'est cet attribut qui est scruté ici.

```
-------------------------------------------------------
<1: Display Meta Rules>
<2: Display Bugs Report>
<3: Build Delegation>
<4: Remove Delegation>
<5: Build Prototypes>
<6: Look For Delegation>
<7: Meta Trace>
<8: Add Meta Rule>
<9: Remove Meta Rule>
<10: Add Bugs Report>
<11: Remove Bugs Report>
<12: Display Slot Of Selected Prototypes>
<13: Menu AddSlots>
<14: Compteur de Formes lexicales>
<15: Empty Waste Gaspar>

Select a number or q to quit MenuMeta:15
```

```
-------------------------------------------------------


Recherche des Objets Gaspar
---------------------------

Nombre d'objets Gaspar definis correctement : 180
Nombre d'objets Gaspar Poubellises : 60
Gaspar is going to kill !!! Keep Cool...
Kill(0) in Progress....
Kill(59) is done!


Nombre d'objets Gaspar definis correctement : 180
Nombre d'objets Gaspar Poubellises : 60
-------------------------------------------------------
<1: Display Meta Rules>
<2: Display Bugs Report>
<3: Build Delegation>
<4: Remove Delegation>
<5: Build Prototypes>
<6: Look For Delegation>
<7: Meta Trace>
<8: Add Meta Rule>
<9: Remove Meta Rule>
<10: Add Bugs Report>
<11: Remove Bugs Report>
<12: Display Slot Of Selected Prototypes>
<13: Menu AddSlots>
<14: Compteur de Formes lexicales>
<15: Empty Waste Gaspar>

Select a number or q to quit MenuMeta:
```