Annexes Partie 2 Chapitre 7

A-2-7.1. Illustration de l'héritage avec Self sur un exemple

On donne dans le code présenté ci-dessous une illustration de l'héritage multiple et dynamique sur un exemple simple.

```
Self 1> 'heritageMultiple.self' _RunScript
reading ./heritageMultiple.self...
<object 3>
```

Héritage Multiple

On dispose dans cet exemple deux types de comportement pour l'affichage des attributs des prototypes définis.

```
Self 2>  inspect: traits parent1
( | _ parent* = traits clonable.
    ^ copy = ( clone).
    ^ display = (
        _AddSlotsIfAbsent: (| displayParent1 |).
        displayParent1:( '      (Attribut 1: ', slot1  printString, ')').
        displayParent1 printLine.
        self.
    ). | )
<object 5>
Self 3>  inspect: traits parent2
( | ^ display = (
        _AddSlotsIfAbsent: (| displayParent1 |).
        displayParent1:( '      (Attribut 1: ', slot1  printString, ')').
        displayParent1 printLine.
        _AddSlotsIfAbsent: (| displayParent2 |).
        displayParent2:( '      (Attribut 2: ', slot2  printString, ')').
        displayParent2 printLine.
        self.
    ). | )
<object 7>
```

On construit des prototypes qui vont déléguer un comportement à l'un ou à l'autre de ces objets ou aux deux.

```
Self 4> inspect: filsParent1
( | _ parent* <- <object 5>.
    ^ slot1 <- 'valeur attribut slot1'.
    ^ slot2 <- 'valeur attribut slot2'. | )
<object 9>
Self 5> inspect: filsParent1EtParent2
( | _ root2* <- <object 7>.
    _ root1* <- <object 5>.
    ^ slot1 <- 'valeur attribut slot1'.
    ^ slot2 <- 'valeur attribut slot2'. | )
<object 11>
Self 6> inspect: filsParent1EtParent2Bis
( | _ root2* <- <object 7>.
    _ root1* <- <object 5>.
```



```
    ^ slot1 <- 'valeur attribut slot1'.
    ^ slot2 <- 'valeur attribut slot2'.
    ^ display = (
        _AddSlotsIfAbsent: (| displayParent2 |).
        displayParent2:( '      (Attribut 2: ', slot2  printString, ')').
        displayParent2 printLine.
        _AddSlotsIfAbsent: (| displayParent1 |).
        displayParent1:( '      (Attribut 1: ', slot1  printString, ')').
        displayParent1 printLine.
        self.
    ). | )
<object 13>
Self 7> inspect: filsParent1EtParent2Ter
( | _ root2* <- <object 7>.
    _ parent* <- <object 5>.
    ^ slot1 <- 'valeur attribut slot1'.
    ^ slot2 <- 'valeur attribut slot2'.
    ^ displayFils = ( parent.display). | )
<object 3>
```

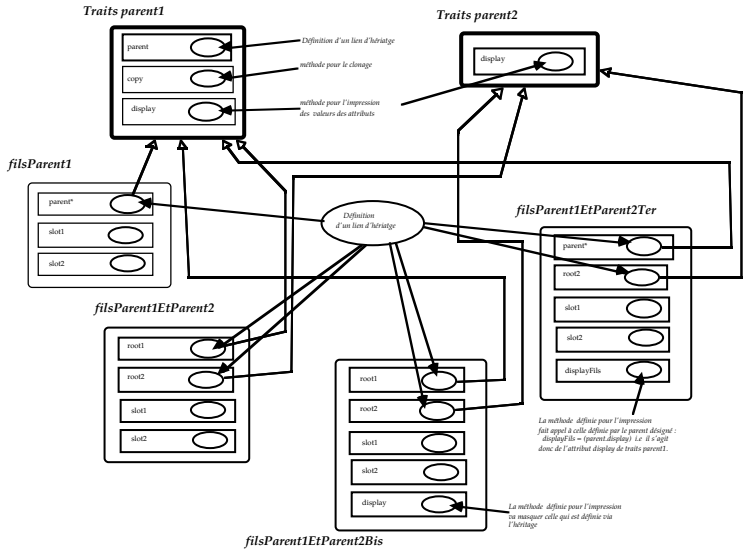

Figure a-2-7.1 : L'héritage et Self.

On regarde maintenant comment se comporte chacun de ces prototypes par rapport aux parents qui leurs sont attachés.

```
Self 8>  filsParent1 display
      (Attribut 1: 'valeur attribut slot1')
<object 9>

"Le prototype précédent délègue le comportement visé à un seul parent"

Self 9>  filsParent1EtParent2 display
More than one 'display' slot was found in filsParent1EtParent2 <11>.
Sending method holder is shell <17>.
The matching slots are: traits parent1 <5> and traits parent2 <7>.
## Stack trace for process <18>
##____Rcvr___Selector/Block_____File:line_____Method-holder/location_____
#5    <11>   display                <error>:1      filsParent1EtParent2
#6    <17>   <top level expr>       <the prompt>:1 shell
```

Le prototype précédent délègue le même comportement à deux parents, d'où le message d'erreur produit. Pour éviter de tels conflits, il faut soit les résoudre au momemt de la définition des objets en prenant soin de ne pas générer de telles ambiguïtés. On dispose aussi de primitives qui permettent de selectionner le parent désiré pour la délégation.

```
Self 10>  filsParent1EtParent2Bis display
```

```
      (Attribut 2: 'valeur attribut slot2')
      (Attribut 1: 'valeur attribut slot1')
<object 8>
```

Le prototype précédent possède le même nom d'attribut que celui qu'il pourrait déléguer à ces deux parents, c'est donc cet attribut qui est activé (activation du slot de plus bas niveau).

```
Self 11>  filsParent1EtParent2Ter display
More than one 'display' slot was found in filsParent1EtParent2Ter <3>.
Sending method holder is shell <17>.
The matching slots are: traits parent1 <5> and traits parent2 <7>.
## Stack trace for process <21>
##____Rcvr___Selector/Block_________File:line_____Method-holder/location_____
#5    <3>    display               <error>:1      filsParent1EtParent2Ter
#6    <17>   <top level expr>       <the prompt>:1 shell
```

Le prototype précédent délègue le même comportement à deux parents, d'où le message d'erreur produit. Ce même prototype permet aussi de passer outre cette ambiguïté: l'attribut `displayfils` permet de réaliser un renvoi dirigé vers un de ses parents.

```
Self 12>  filsParent1EtParent2Ter displayFils
      (Attribut 1: 'valeur attribut slot1')
filsParent1EtParent2Ter
```

De même on peut réaliser un renvoi de message pour les deux prototypes précédents auxquels les messages d'affichage qui leur étaient adressés ne pouvaient être exécutés. On utilise pour cela des primitives du langage de la manière suivante:

```
Self 13>  'display' sendTo:  filsParent1EtParent2 DelegatingTo: filsParent1EtParent2 root1
      (Attribut 1: 'valeur attribut slot1')
filsParent1EtParent2
Self 14>  'display' sendTo:  filsParent1EtParent2 DelegatingTo: filsParent1EtParent2 root2
      (Attribut 1: 'valeur attribut slot1')
      (Attribut 2: 'valeur attribut slot2')
filsParent1EtParent2
Self 15>  'display' sendTo:  filsParent1EtParent2Ter DelegatingTo: filsParent1EtParent2Ter root2
      (Attribut 1: 'valeur attribut slot1')
      (Attribut 2: 'valeur attribut slot2')
filsParent1EtParent2Ter
Self 16>  'display' sendTo:  filsParent1EtParent2Ter DelegatingTo: filsParent1EtParent2Ter parent
      (Attribut 1: 'valeur attribut slot1')
filsParent1EtParent2Ter
```

Héritage Dynamique

Il est possible de modifier les comportements attachés à un prototype donné. Il convient pour cela de définir le lien qui existe entre le prototype et son père avec un slot d'assignation: c'est le cas pour tous les prototypes définis ici. On peut donc modifier les liens qui existent entre les objets et leurs ancêtres.

```
Self 17>  filsParent1 parent
traits parent1
Self 18> filsParent1 parent: traits parent2
filsParent1
Self 19>  filsParent1 parent
traits parent2
Self 20>  filsParent1 display
      (Attribut 1: 'valeur attribut slot1')
      (Attribut 2: 'valeur attribut slot2')
filsParent1
Self 21>
```

Le comportement attaché au prototype précédent a été modifié.

A-2-7.2. Génération dynamique de prototypes lexicaux (1)

On donne ci-dessous une première illustration de la génération de contraintes associées à des représentations prototypiques d'éléments lexicaux. Les représentations pré-définies ici sont du même type que celles que nous avons déjà présentées. On présente tout d'abord le fichier qui contient les contraintes recueillies:

```
#Dc
pontage
Nom
Sn
(('Det' & 'Nom') asList)
N1prepN2
(('Nom' & 'Prep' & 'Nom') asList)
N1prepV
(('Nom' & 'Prep' & 'Verbe') asList)
AdjN
(('Adj' & 'Nom') asList)
NAdj
(('Nom' & 'Adj') asList)
PrepN
(('Prep' & 'Nom') asList)
Fc#
#Dc
lesion
Nom
Sn
(('Det' & 'Nom') asList)
N1prepN2
(('Nom' & 'Prep' & 'Nom') asList)
N1prepV
(('Nom' & 'Prep' & 'Verbe') asList)
AdjN
(('Adj' & 'Nom') asList)
NAdj
(('Nom' & 'Adj') asList)
PrepN
(('Prep' & 'Nom') asList)
Fc#
#EndOfContraintes#
```

La première contrainte sur `pontage` peut se lire de la manière suivante:

```
<pontage,Nom> -> {Sn (Det Nom), NPrepN (N1PrepN2), AdjN (Adj Nom), NAdj (Nom Adj),
PrepN (Prep Nom)}
```

c'est-à-dire que le prototype lexical `pontage` de catégorie `Nom` peut initialiser la construction des prototypes `Sn`, `NPrepN`... avec les composants associés. Idem pour le prototype `lésion`. On présente ci-dessous la trace de la phase de lecture de ces contraintes puis de leurs affectations aux prototypes concernés.

```
------------------------------------------------------------
Gaspar-Load  Shut  Down


------------------------------------------------------------
<Gaspar  Location:/home/info/fleury/Self/source/Self-3.0/Phase13/...>

------------------------------------------------------------
<1: Menu Load >
<2:  Menu  Parsing>
<3:  Menu  Lexical  (Display-Create)>
<4:  Menu  Define  Prototype  From  Corpus>
<5:  Menu  Help>
<6:  PontageN2-Semantic-Rules>
<7:  ui  start>
<8:  ui  quit>
<9:  Tools>

Select  a  number  or  q  to  quit  Gaspar:2
------------------------------------------------------------
------------------------------------------------------------
<1:  GasparParser  Menu>
<2:  GasparContraints  Menu>

Select  a  number  or  q  to  quit  MenuParser:2
------------------------------------------------------------
------------------------------------------------------------
<1:  Define  Contraints  From  Corpus>
<2:  Display  All  Defined  Word-Contraints  From  Corpus>
```

```
<3: Display One Defined Word-Contraints From Corpus>
<4: Display Pre-Defined Cat-Contraints>

Select a number or q to quit MenuContraints:1
-----------------------------------------------------------


      Vidage du dictionnaire des contraintes


      Recreation du dictionnaire des contraintes


      Dictionnaire Vide des contraintes disponible


 Lecture du Fichier Disponible de Contraintes sur les Mots.

/home/info/fleury/Self/source/Self-3.0/Phase13/work/inputContraintes.self


Contrainte[1] dans le fichier des contraintes :

-------------------------------------------------------------------------------
Contrainte sur : <pontage>
Valeur categorielle du mot contraint : <Nom>
--> Verification de l'existence d'un prototype portant la valeur categorielle <Nom>
Nombre de Prototype ayant un attribut forme :158
--> Verification terminee
--> Il existe un Prototype qui porte cette valeur categorielle, on continue!

Constituants de la Contrainte : <(('Det' & 'Nom') asList)>
Categorie de la contrainte lue : <Sn>
--> Verification de l'existence d'un prototype portant la valeur categorielle <Sn>
Nombre de Prototype ayant un attribut forme :158
--> Verification terminee
--> Il existe un Prototype qui porte cette valeur categorielle, on continue!

Constituants de la Contrainte : <(('Nom' & 'Prep' & 'Nom') asList)>
Categorie de la contrainte lue : <N1prepN2>
--> Verification de l'existence d'un prototype portant la valeur categorielle <N1prepN2>
Nombre de Prototype ayant un attribut forme :158
--> Verification terminee
--> Il existe un Prototype qui porte cette valeur categorielle, on continue!

Constituants de la Contrainte : <(('Nom' & 'Prep' & 'Verbe') asList)>
Categorie de la contrainte lue : <N1prepV>
--> Verification de l'existence d'un prototype portant la valeur categorielle <N1prepV>
Nombre de Prototype ayant un attribut forme :158
--> Verification terminee
--> Il existe un Prototype qui porte cette valeur categorielle, on continue!

Constituants de la Contrainte : <(('Adj' & 'Nom') asList)>
Categorie de la contrainte lue : <AdjN>
--> Verification de l'existence d'un prototype portant la valeur categorielle <AdjN>
Nombre de Prototype ayant un attribut forme :158
--> Verification terminee
--> Il existe un Prototype qui porte cette valeur categorielle, on continue!

Constituants de la Contrainte : <(('Nom' & 'Adj') asList)>
Categorie de la contrainte lue : <NAdj>
--> Verification de l'existence d'un prototype portant la valeur categorielle <NAdj>
Nombre de Prototype ayant un attribut forme :158
--> Verification terminee
--> Il existe un Prototype qui porte cette valeur categorielle, on continue!

Constituants de la Contrainte : <(('Prep' & 'Nom') asList)>
Categorie de la contrainte lue : <PrepN>
--> Verification de l'existence d'un prototype portant la valeur categorielle <PrepN>
Nombre de Prototype ayant un attribut forme :158
--> Verification terminee
--> Il existe un Prototype qui porte cette valeur categorielle, on continue!


Creation des arbres elementaires associes a la contrainte.
-----------------------------------------------------------


/home/info/fleury/Self/source/Self-3.0/Phase13/work/contraintsOn_Corpus.self
/home/info/fleury/Self/source/Self-3.0/Phase13/work/contraintsOn_Corpus.self
reading  /home/info/fleury/Self/source/Self-3.0/Phase13/work/contraintsOn_Corpus.self...
/home/info/fleury/Self/source/Self-3.0/Phase13/work/constituantOnContraints.self
/home/info/fleury/Self/source/Self-3.0/Phase13/work/constituantOnContraints.self
reading  /home/info/fleury/Self/source/Self-3.0/Phase13/work/constituantOnContraints.self...
Arbre elementaire en Cours de Construction :
<list{'pontage', 'Nom', list{catSn, catN1prepN2, catN1prepV, catAdjN, catNAdj, catPrepN, list{list, list,
list, list, list, list}}>


--> Verification de l'existence d'un prototype portant la valeur categorielle <'Sn'> et des composants
associes <list{'Det', 'Nom'}>
Nombre de Prototype ayant un attribut forme :159
--> Verification terminee
--> Il existe un Prototype qui porte cette valeur categorielle avec ces composants, on continue!
--> Verification de l'existence d'un prototype portant la valeur categorielle <'N1prepN2'> et des
composants associes <list{'Nom', 'Prep', 'Nom'}>
Nombre de Prototype ayant un attribut forme :159
--> Verification terminee
--> Il existe un Prototype qui porte cette valeur categorielle avec ces composants, on continue!
--> Verification de l'existence d'un prototype portant la valeur categorielle <'N1prepV'> et des
composants associes <list{'Nom', 'Prep', 'Verbe'}>
Nombre de Prototype ayant un attribut forme :159
--> Verification terminee
--> Il existe un Prototype qui porte cette valeur categorielle avec ces composants, on continue!
--> Verification de l'existence d'un prototype portant la valeur categorielle <'SAdjN'> et des composants
associes <list{'Adj', 'Nom'}>
Nombre de Prototype ayant un attribut forme :159
--> Verification terminee
--> Il existe un Prototype qui porte cette valeur categorielle avec ces composants, on continue!
--> Verification de l'existence d'un prototype portant la valeur categorielle <'NSAdj'> et des composants
associes <list{'Nom', 'Adj'}>
Nombre de Prototype ayant un attribut forme :159
--> Verification terminee
--> Il existe un Prototype qui porte cette valeur categorielle avec ces composants, on continue!
```



```
--> Verification de l'existence d'un prototype portant la valeur categorielle <'PrepN'> et des composants
associes <list{'Prep', 'Nom'}>
Nombre de Prototype ayant un attribut forme :159
--> Verification terminee
--> Il existe un Prototype qui porte cette valeur categorielle avec ces composants, on continue!

Contrainte[2] dans le fichier des contraintes :

-------------------------------------------------------------------------------
Contrainte sur : <lesion>
Valeur categorielle du mot contraint : <Nom>
--> Verification de l'existence d'un prototype portant la valeur categorielle <Nom>
Nombre de Prototype ayant un attribut forme :161
--> Verification terminee
--> Il existe un Prototype qui porte cette valeur categorielle, on continue!

Constituants de la Contrainte : <(('Det' & 'Nom') asList)>
Categorie de la contrainte lue : <Sn>
--> Verification de l'existence d'un prototype portant la valeur categorielle <Sn>
Nombre de Prototype ayant un attribut forme :161
--> Verification terminee
--> Il existe un Prototype qui porte cette valeur categorielle, on continue!

Constituants de la Contrainte : <(('Nom' & 'Prep' & 'Nom') asList)>
Categorie de la contrainte lue : <N1prepN2>
--> Verification de l'existence d'un prototype portant la valeur categorielle <N1prepN2>
Nombre de Prototype ayant un attribut forme :161
--> Verification terminee
--> Il existe un Prototype qui porte cette valeur categorielle, on continue!

Constituants de la Contrainte : <(('Nom' & 'Prep' & 'Verbe') asList)>
Categorie de la contrainte lue : <N1prepV>
--> Verification de l'existence d'un prototype portant la valeur categorielle <N1prepV>
Nombre de Prototype ayant un attribut forme :161
--> Verification terminee
--> Il existe un Prototype qui porte cette valeur categorielle, on continue!

Constituants de la Contrainte : <(('Adj' & 'Nom') asList)>
Categorie de la contrainte lue : <AdjN>
--> Verification de l'existence d'un prototype portant la valeur categorielle <AdjN>
Nombre de Prototype ayant un attribut forme :161
--> Verification terminee
--> Il existe un Prototype qui porte cette valeur categorielle, on continue!

Constituants de la Contrainte : <(('Nom' & 'Adj') asList)>
Categorie de la contrainte lue : <NAdj>
--> Verification de l'existence d'un prototype portant la valeur categorielle <NAdj>
Nombre de Prototype ayant un attribut forme :161
--> Verification terminee
--> Il existe un Prototype qui porte cette valeur categorielle, on continue!

Constituants de la Contrainte : <(('Prep' & 'Nom') asList)>
Categorie de la contrainte lue : <PrepN>
--> Verification de l'existence d'un prototype portant la valeur categorielle <PrepN>
Nombre de Prototype ayant un attribut forme :161
--> Verification terminee
--> Il existe un Prototype qui porte cette valeur categorielle, on continue!

Creation des arbres elementaires associes a la contrainte.
-----------------------------------------------------------


/home/info/fleury/Self/source/Self-3.0/Phase13/work/contraintsOn_Corpus.self
/home/info/fleury/Self/source/Self-3.0/Phase13/work/contraintsOn_Corpus.self
reading  /home/info/fleury/Self/source/Self-3.0/Phase13/work/contraintsOn_Corpus.self...
/home/info/fleury/Self/source/Self-3.0/Phase13/work/constituantOnContraints.self
/home/info/fleury/Self/source/Self-3.0/Phase13/work/constituantOnContraints.self
reading  /home/info/fleury/Self/source/Self-3.0/Phase13/work/constituantOnContraints.self...
Arbre elementaire en Cours de Construction :
<list{'lesion', 'Nom', list{catSn, catN1prepN2, catN1prepV, catAdjN, catNAdj, catPrepN}, list{list, list,
list, list, list, list}}>


--> Verification de l'existence d'un prototype portant la valeur categorielle <'Sn'> et des composants
associes <list{'Det', 'Nom'}>
Nombre de Prototype ayant un attribut forme :161
--> Verification terminee
--> Il existe un Prototype qui porte cette valeur categorielle avec ces composants, on continue!
--> Verification de l'existence d'un prototype portant la valeur categorielle <'N1prepN2'> et des
composants associes <list{'Nom', 'Prep', 'Nom'}>
Nombre de Prototype ayant un attribut forme :161
--> Verification terminee
--> Il existe un Prototype qui porte cette valeur categorielle avec ces composants, on continue!
--> Verification de l'existence d'un prototype portant la valeur categorielle <'N1prepV'> et des
composants associes <list{'Nom', 'Prep', 'Verbe'}>
Nombre de Prototype ayant un attribut forme :161
--> Verification terminee
--> Il existe un Prototype qui porte cette valeur categorielle avec ces composants, on continue!
--> Verification de l'existence d'un prototype portant la valeur categorielle <'SAdjN'> et des composants
associes <list{'Adj', 'Nom'}>
Nombre de Prototype ayant un attribut forme :161
--> Verification terminee
--> Il existe un Prototype qui porte cette valeur categorielle avec ces composants, on continue!
--> Verification de l'existence d'un prototype portant la valeur categorielle <'NSAdj'> et des composants
associes <list{'Nom', 'Adj'}>
Nombre de Prototype ayant un attribut forme :161
--> Verification terminee
--> Il existe un Prototype qui porte cette valeur categorielle avec ces composants, on continue!
--> Verification de l'existence d'un prototype portant la valeur categorielle <'PrepN'> et des composants
associes <list{'Prep', 'Nom'}>
Nombre de Prototype ayant un attribut forme :161
--> Verification terminee
--> Il existe un Prototype qui porte cette valeur categorielle avec ces composants, on continue!

Fin de Lecture du Fichier des Contraintes.

      Contraintes Lues :
list{list{'pontage', 'Nom', list, list}, list{'lesion', 'Nom', list, list}}


Nouvelle(s) contraintes sur le mot <'pontage' 'Nom'> :
     <object 3> -> 'Sn'
       Composants:list('Det', 'Nom')
```

```
Nouvelle(s) contraintes sur le mot <'pontage' 'Nom'> :
     <object 4> -> 'N1prepN2'
       Composants:list{'Nom', 'Prep', 'Nom'}


Nouvelle(s) contraintes sur le mot <'pontage' 'Nom'> :
     <object 5> -> 'N1prepV'
       Composants:list{'Nom', 'Prep', 'Verbe'}


Nouvelle(s) contraintes sur le mot <'pontage' 'Nom'> :
     <object 6> -> 'SAdjN'
       Composants:list{'Adj', 'Nom'}


Nouvelle(s) contraintes sur le mot <'pontage' 'Nom'> :
     <object 7> -> 'NSAdj'
       Composants:list{'Nom', 'Adj'}


Nouvelle(s) contraintes sur le mot <'pontage' 'Nom'> :
     <object 8> -> 'PrepN'
       Composants:list{'Prep', 'Nom'}


Recherche et Mise a jour des prototypes concernes

-------------------------------------------------------------------------------------------------
 Prototype : ['pontage'|'Nom']
--> Mise a jour Du Champ EltTree De Ce Prototype avec les arbres suivants :
list{catSn, catN1prepN2, catN1prepV, catAdjN, catNAdj, catPrepN,}
--> Mise a jour Ok Du Champ EltTree De Ce Prototype
-------------------------------------------------------------------------------------------------

Nouvelle(s) contraintes sur le mot <'lesion' 'Nom'> :
     <object 3> -> 'Sn'
       Composants:list{'Det', 'Nom'}


Nouvelle(s) contraintes sur le mot <'lesion' 'Nom'> :
     <object 4> -> 'N1prepN2'
       Composants:list{'Nom', 'Prep', 'Nom'}


Nouvelle(s) contraintes sur le mot <'lesion' 'Nom'> :
     <object 5> -> 'N1prepV'
       Composants:list{'Nom', 'Prep', 'Verbe'}


Nouvelle(s) contraintes sur le mot <'lesion' 'Nom'> :
     <object 6> -> 'SAdjN'
       Composants:list{'Adj', 'Nom'}


Nouvelle(s) contraintes sur le mot <'lesion' 'Nom'> :
     <object 7> -> 'NSAdj'
       Composants:list{'Nom', 'Adj'}


Nouvelle(s) contraintes sur le mot <'lesion' 'Nom'> :
     <object 8> -> 'PrepN'
       Composants:list{'Prep', 'Nom'}

Recherche et Mise a jour des prototypes concernes

-------------------------------------------------------------------------------------------------
 Prototype : ['lesion'|'Nom']
--> Mise a jour Du Champ EltTree De Ce Prototype avec les arbres suivants :
list{catSn, catN1prepN2, catN1prepV, catAdjN, catNAdj, catPrepN}
--> Mise a jour Ok Du Champ EltTree De Ce Prototype
-------------------------------------------------------------------------------------------------


      Dictionnaire des contraintes disponible


-----------------------------------------------------------
<1: Define Contraints From Corpus>
<2: Display All Defined Word-Contraints From Corpus>
<3: Display One Defined Word-Contraints From Corpus>
<4: Display Pre-Defined Cat-Contraints>

Select a number or q to quit MenuContraints:3
-----------------------------------------------------------
Contraintes sur quel Mot :pontage
Categorie de ce Mot [Cat|*] :Nom
-----------------------------------
Word :pontage
     Cat : <Nom>
      Categories Syntaxiques Associees :
           'Sn'
           'N1prepN2'
           'N1prepV'
           'SAdjN'
           'NSAdj'
           'PrepN'
-----------------------------------------------------------
<1: Define Contraints From Corpus>
<2: Display All Defined Word-Contraints From Corpus>
<3: Display One Defined Word-Contraints From Corpus>
<4: Display Pre-Defined Cat-Contraints>

Select a number or q to quit MenuContraints:q
-----------------------------------------------------------
MenuContraints Shut Down
-----------------------------------------------------------
<1: GasparParser Menu>
<2: GasparContraints Menu>

Select a number or q to quit MenuParser:q
-----------------------------------------------------------
MenuParser Shut Down
-----------------------------------------------------------
```


A-2-7.3. Génération dynamique de prototypes lexicaux (2)

On exécute une réorganisation des savoirs associés aux prototypes représentant la catégorie des `Noms`. On fait tout d'abord une recherche de tous ces prototypes et de leurs contraintes respectives (on retrouve les contraintes précédemment définies); on détermine ensuite les comportements communs: une classe de comportements partagés par tous les prototypes qui disposent de contraintes. Cette classe peut être associée à l'objet `traits` commun à tous ces prototypes. Les prototypes qui ne disposent pas de contraintes à l'issue de la définition de celles-ci disposeront ainsi d'un savoir par défaut. Cela ne veut pas dire que ce savoir sera utilisé pour les entités qui ne disposent pas de contraintes particulières. Il s'agit simplement de marquer que sur une famille donnée, certains prototypes possèdent des contraintes et que ces savoirs sont disponibles mais ne sont pas utilisables pour tous les éléments de la famille.

```
-----------------------------------------------------------
<1: SearchAndDisplay Parent for all Prototypes>
<2: SearchAndDisplay Parent for one Prototype>
<3: SearchAndDisplay Sons for all Objects Traits>
<4: SearchAndDisplay Sons for one Object Traits>
<5: Modify Links Beetween Selected Prototypes>
<6: Modify Links Beetween All Prototypes>

Select a number or q to quit MenuSearchAndDestroy:5
-----------------------------------------------------------

Modifier Liens sur quelle categorie :Nom

Nombre de Prototype ayant un attribut forme :190

Comportements Communs aux Prototypes  pontage et lésion :

   Sn N1prepN2 N1prepV AdjN NAdj PrepN


Prototype Lexical pontage :

 Ajout du comportement Sn a la liste des comportements individuels

 Ajout du comportement N1PrepN2 a la liste des comportements individuels

 Ajout du comportement N1PrepV a la liste des comportements individuels

 Ajout du comportement SAdjN a la liste des comportements individuels

 Ajout du comportement NSAdj a la liste des comportements individuels

 Ajout du comportement PrepN a la liste des comportements individuels


Prototype Lexical lesion:

 Ajout du comportement Sn a la liste des comportements individuels si celui-ci n'est pas deja recense

 Ajout du comportement N1PrepN2 a la liste des comportements individuels si celui-ci n'est pas deja
recense

 Ajout du comportement N1PrepV a la liste des comportements individuels si celui-ci n'est pas deja
recense

 Ajout du comportement SAdjN a la liste des comportements individuels si celui-ci n'est pas deja recense

 Ajout du comportement NSAdj a la liste des comportements individuels si celui-ci n'est pas deja recense

 Ajout du comportement PrepN a la liste des comportements individuels si celui-ci n'est pas deja recense


--> Comportements Syntaxiques Partages par les 2 Prototypes <pontage|Nom><lesion|Nom> :

  0. Sn

  1. N1PrepN2

  2. N1PrepV

  3. SAdjN

  4. SAdjN

  5. PrepN

--> Ensemble des Comportements Syntaxiques des 2 Prototypes <pontage|Nom><lesion|Nom> :

  0. Sn

  1. N1PrepN2

  2. N1PrepV

  3. SAdjN
```

```
   4. SAdjN

   5. PrepN

------------------------------------------------------------
<1: SearchAndDisplay Parent for all Prototypes>
<2: SearchAndDisplay Parent for one Prototype>
<3: SearchAndDisplay Sons for all Objects Traits>
<4: SearchAndDisplay Sons for one Object Traits>
<5: Modify Links Beetween Selected Prototypes>
<6: Modify Links Beetween All Prototypes>

Select a number or q to quit MenuSearchAndDestroy:q
------------------------------------------------------------
Menu SearchAndDestroy Shut Down
------------------------------------------------------------
```


A-2-7.4. Génération dynamique de prototypes lexicaux (3)

L'exemple qui suit est une illustration de la génération de nouvelles représentations prototypiques de catégories syntaxiques non encore reconnues. Le fichier de contraintes a désormais l'allure suivante:

```
#Dc
pontage
Nom
Sn
(('Det' & 'Nom') asList)
NPivotprepN2
(('NomPivot' & 'Prep' & 'Nom') asList)
N1prepV
(('Nom' & 'Prep' & 'Verbe') asList)
AdjN
(('Adj' & 'Nom') asList)
NAdj
(('Nom' & 'Adj') asList)
PrepN
(('Prep' & 'Nom') asList)
N1NPivot
(('Nom' & 'NomPivot') asList)
Fc#
#Dc
lesion
Nom
Sn
(('Det' & 'Nom') asList)
N1prepN2
(('Nom' & 'Prep' & 'Nom') asList)
N1prepV
(('Nom' & 'Prep' & 'Verbe') asList)
AdjN
(('Adj' & 'Nom') asList)
NAdj
(('Nom' & 'Adj') asList)
PrepN
(('Prep' & 'Nom') asList)
NPivotN2
(('NomPivot' & 'Nom') asList)
Fc#
#EndOfContraintes#
```

Le mot `pontage` peut construire un nouveau comportement du type `N1NPivot {Nom, NomPivot}`, de même le mot `lesion` peut construire un nouveau comportement du type `NPivotN2 {NomPivot, Nom2}`. Il n'existe pas de pré-représentations prototypiques pour ces comportements. Il faut donc définir les objets adéquats pour les représenter. La trace qui suit présente cette création dynamique d'objets et le rattachement de ceux-ci aux prototypes lexicaux concernés.

```
------------------------------------------------------------
<1: Define Contraints From Corpus>
<2: Display All Defined Word-Contraints From Corpus>
<3: Display One Defined Word-Contraints From Corpus>
<4: Display Pre-Defined Cat-Contraints>

Select a number or q to quit MenuContraints:1
------------------------------------------------------------

       Vidage du dictionnaire des contraintes

       Recreation du dictionnaire des contraintes

       Dictionnaire Vide des contraintes disponible

 Lecture du Fichier Disponible de Contraintes sur les Mots.
/home/info/fleury/Self/source/Self-3.0/Phase13/work/inputContraintes.self


Contrainte[1] dans le fichier des contraintes :
------------------------------------------------------------------------------

Contrainte sur : <pontage>
Valeur categorielle du mot contraint : <Nom>
--> Verification de l'existence d'un prototype portant la valeur categorielle <Nom>
Nombre de Prototype ayant un attribut forme :158
--> Verification terminee
```

```
--> Il existe un Prototype qui porte cette valeur categorielle, on continue!

Constituants de la Contrainte : <(('Det' & 'Nom') asList)>
Categorie de la contrainte lue : <Sn>
--> Verification de l'existence d'un prototype portant la valeur categorielle <Sn>
Nombre de Prototype ayant un attribut forme :158
--> Verification terminee
--> Il existe un Prototype qui porte cette valeur categorielle, on continue!

Constituants de la Contrainte : <(('NomPivot' & 'Prep' & 'Nom') asList)>
Categorie de la contrainte lue : <NPivotPrepN2>
--> Verification de l'existence d'un prototype portant la valeur categorielle <NPivotPrepN2>
Nombre de Prototype ayant un attribut forme :158
--> Verification terminee
--> Il existe un Prototype qui porte cette valeur categorielle, on continue!

Constituants de la Contrainte : <(('Nom' & 'Prep' & 'Verbe') asList)>
Categorie de la contrainte lue : <NPivotPrepV>
--> Verification de l'existence d'un prototype portant la valeur categorielle <NPivotPrepV>
Nombre de Prototype ayant un attribut forme :158
--> Verification terminee
--> Il existe un Prototype qui porte cette valeur categorielle, on continue!

Constituants de la Contrainte : <(('Adj' & 'Nom') asList)>
Categorie de la contrainte lue : <AdjN>
--> Verification de l'existence d'un prototype portant la valeur categorielle <AdjN>
Nombre de Prototype ayant un attribut forme :158
--> Verification terminee
--> Il existe un Prototype qui porte cette valeur categorielle, on continue!

Constituants de la Contrainte : <(('Nom' & 'Adj') asList)>
Categorie de la contrainte lue : <NAdj>
--> Verification de l'existence d'un prototype portant la valeur categorielle <NAdj>
Nombre de Prototype ayant un attribut forme :158
--> Verification terminee
--> Il existe un Prototype qui porte cette valeur categorielle, on continue!

Constituants de la Contrainte : <(('Prep' & 'Nom') asList)>
Categorie de la contrainte lue : <PrepN>
--> Verification de l'existence d'un prototype portant la valeur categorielle <PrepN>
Nombre de Prototype ayant un attribut forme :158
--> Verification terminee
--> Il existe un Prototype qui porte cette valeur categorielle, on continue!

Constituants de la Contrainte : <(('Nom' & 'NomPivot') asList)>
Categorie de la contrainte lue : <N1NPivot>
--> Verification de l'existence d'un prototype portant la valeur categorielle <N1NPivot>
Nombre de Prototype ayant un attribut forme :158
--> Verification terminee
--> Il n'existe pas de Prototype qui porte cette valeur categorielle, on va en creer un!
Creation du Prototype <'N1NPivot'>
/home/info/fleury/Self/source/Self-3.0/Phase13/work/defineProto1Output.self
---> Definition de l' objet Traits associe : En cours
---> Definition de l' objet Traits associe Ok.
---> Definition de l' objet Prototype : En Cours
---> Definition de l' objet Prototype : Ok
---> Lecture et Creation  de l' objet Prototype : En Cours
/home/info/fleury/Self/source/Self-3.0/Phase13/work/defineProto1Output.self
reading  /home/info/fleury/Self/source/Self-3.0/Phase13/work/defineProto1Output.self...
---> Lecture et Creation  de l' objet Prototype Ok.


Creation des arbres elementaires associes a la contrainte.

------------------------------------------------------------

/home/info/fleury/Self/source/Self-3.0/Phase13/work/contraintsOn_Corpus.self
/home/info/fleury/Self/source/Self-3.0/Phase13/work/contraintsOn_Corpus.self
reading  /home/info/fleury/Self/source/Self-3.0/Phase13/work/contraintsOn_Corpus.self...
/home/info/fleury/Self/source/Self-3.0/Phase13/work/constituantOnContraints.self
/home/info/fleury/Self/source/Self-3.0/Phase13/work/constituantOnContraints.self
reading  /home/info/fleury/Self/source/Self-3.0/Phase13/work/constituantOnContraints.self...

Arbre elementaire en Cours de Construction :
<list{'pontage',  'Nom',  list{catSn,  catNPivotPrepN2,  catNPivotPrepV,  catAdjN,  catNAdj,  catPrepN,
catN1NPivot}, list{list, list, list, list, list, list, list}|>

--> Pas de pivot detecte
Position du Pivot : 1.
--> Verification de l'existence d'un prototype portant la valeur categorielle <'Sn'> et des composants
associes <list{'Det', 'Nom'}>
Nombre de Prototype ayant un attribut forme :160
--> Verification terminee
--> Il existe un Prototype qui porte cette valeur categorielle avec ces composants, on continue!

--> Presence d'un pivot
Position du Pivot : 0.
--> Verification de l'existence d'un prototype portant la valeur categorielle <'NPivotPrepN2'> et des
composants associes <list{'NomPivot', 'Prep', 'Nom'}>
Nombre de Prototype ayant un attribut forme :160
--> Verification terminee
--> Il existe un Prototype qui porte cette valeur categorielle avec ces composants, on continue!

--> Pas de pivot detecte
Position du Pivot : 0.
--> Verification de l'existence d'un prototype portant la valeur categorielle <'NPivotPrepV'> et des
composants associes <list{'Nom', 'Prep', 'Verbe'}>
Nombre de Prototype ayant un attribut forme :160
--> Verification terminee
--> Il existe un Prototype qui porte cette valeur categorielle avec ces composants, on continue!

--> Pas de pivot detecte
Position du Pivot : 1.
--> Verification de l'existence d'un prototype portant la valeur categorielle <'SAdjN'> et des composants
associes <list{'Adj', 'Nom'}>
Nombre de Prototype ayant un attribut forme :160
--> Verification terminee
--> Il existe un Prototype qui porte cette valeur categorielle avec ces composants, on continue!

--> Pas de pivot detecte
Position du Pivot : 0.
--> Verification de l'existence d'un prototype portant la valeur categorielle <'NSAdj'> et des composants
associes <list{'Nom', 'Adj'}>
```



```
Nombre de Prototype ayant un attribut forme :160
--> Verification terminee
--> Il existe un Prototype qui porte cette valeur categorielle avec ces composants, on continue!

--> Pas de pivot detecte
Position du Pivot : 1.
--> Verification de l'existence d'un prototype portant la valeur categorielle <'PrepN'> et des composants
associes <list{'Prep', 'Nom'}>
Nombre de Prototype ayant un attribut forme :160
--> Verification terminee
--> Il existe un Prototype qui porte cette valeur categorielle avec ces composants, on continue!

--> Presence d'un pivot
Position du Pivot : 1.
--> Verification de l'existence d'un prototype portant la valeur categorielle <'N1NPivot'> et des
composants associes <list{'Nom', 'NomPivot'}>
Nombre de Prototype ayant un attribut forme :160
--> Verification terminee
--> Il n'existe pas de Prototype qui porte cette valeur categorielle avec ces composants, on va en creer
un!
Creation du Prototype <'N1NPivot'> --> <list{'Nom', 'Nom'}>
/home/info/fleury/Self/source/Self-3.0/Phase13/work/defineProto2Output.self
---> Definition de l' objet Traits associe : En cours
---> Definition de l' objet Traits associe Ok.
---> Definition de l' objet Prototype : En Cours
---> Definition de l' objet Prototype : Ok
---> Lecture et Creation  de l' objet Prototype : En Cours
/home/info/fleury/Self/source/Self-3.0/Phase13/work/defineProto2Output.self
reading  /home/info/fleury/Self/source/Self-3.0/Phase13/work/defineProto2Output.self...
---> Lecture et Creation  de l' objet Prototype Ok.
--> Definition et Creation du prototype Ok.
--> Mise a jour de l'arbre elementaire : En cours.
--> Mise a jour de l'arbre elementaire Ok.
```

(... idem avec lesion)

```
Fin de Lecture du Fichier des Contraintes.


      Contraintes Lues :

list{list{'pontage', 'Nom', list, list}, list{'lesion', 'Nom', list, list}}


Nouvelle contrainte sur le mot <'pontage' 'Nom'> :

     <object 3> -> 'Sn'
       Composants:list{'Det', 'Nom'}

Nouvelle contrainte sur le mot <'pontage' 'Nom'> :

     <object 4> -> 'NPivotPrepN2'
       Composants:list{'NomPivot', 'Prep', 'Nom'}

Nouvelle contrainte sur le mot <'pontage' 'Nom'> :

     <object 5> -> 'NPivotPrepV'
       Composants:list{'Nom', 'Prep', 'Verbe'}

Nouvelle contrainte sur le mot <'pontage' 'Nom'> :

     <object 6> -> 'SAdjN'
       Composants:list{'Adj', 'Nom'}

Nouvelle contrainte sur le mot <'pontage' 'Nom'> :

     <object 7> -> 'NSAdj'
       Composants:list{'Nom', 'Adj'}

Nouvelle contrainte sur le mot <'pontage' 'Nom'> :

     <object 8> -> 'PrepN'
       Composants:list{'Prep', 'Nom'}

Nouvelle contrainte sur le mot <'pontage' 'Nom'> :

     <object 9> -> 'N1NPivot'
       Composants:list{'Nom', 'NomPivot'}


Recherche et Mise a jour des prototypes concernes

------------------------------------------------------------------------------------------------------
 Prototype : ['pontage'|'Nom']
--> Mise a jour Du Champ EltTree De Ce Prototype avec les arbres suivants :
list{catSn, catNPivotPrepN2, catNPivotPrepV, catAdjN, catNAdj, catPrepN, catN1NPivot}
--> Mise a jour Ok Du Champ EltTree De Ce Prototype
------------------------------------------------------------------------------------------------------
```

(idem avec lesion)

```
      Dictionnaire des contraintes disponible

------------------------------------------------------
<1: Define Contraints From Corpus>
<2: Display All Defined Word-Contraints From Corpus>
<3: Display One Defined Word-Contraints From Corpus>
<4: Display Pre-Defined Cat-Contraints>

Select a number or q to quit MenuContraints:
------------------------------------------------------
```

On donne ci-dessous des exemples de prototypes construits; ces derniers sont présentés via l'interface graphique de Self:

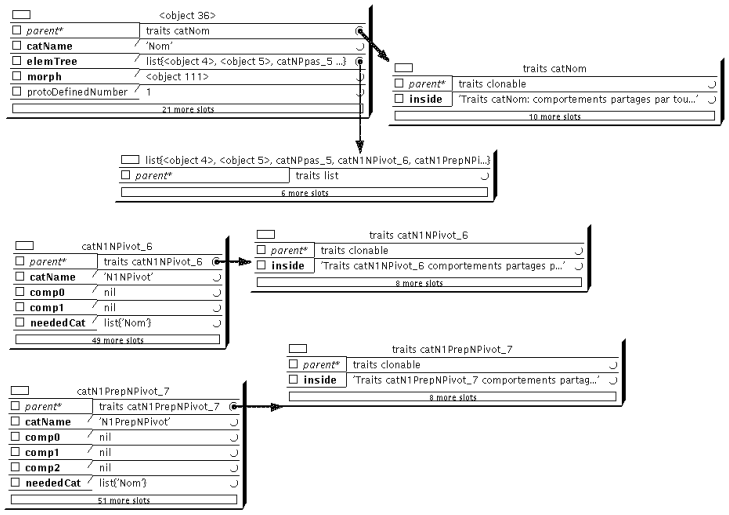

Figure a-2-7.2: Génération du prototype `pontage`

Cette première figure présente le prototype `pontage` construit ainsi que les arbres associés. La figure qui suit présente le prototype lexical `effort` construit par clonage et les arbres associés.

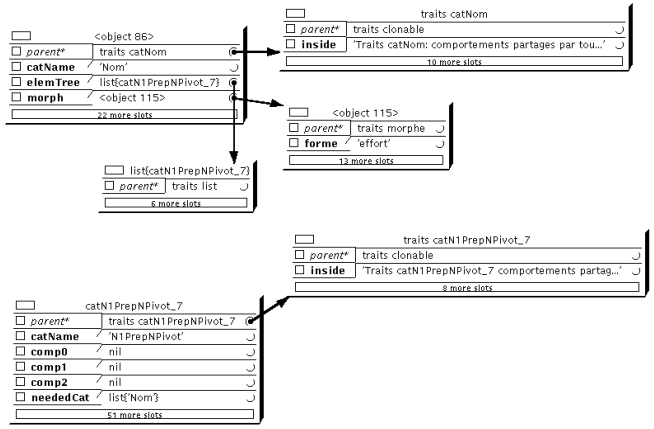

Figure a-2-7.3: Génération du prototype `effort`

A-2-7.5. Détermination des comportements partagés à une classe catégorielle donnée

Si on exécute de nouveau le classement des comportements associés aux noms, on obtient les résultats suivants.

```
------------------------------------------------------------
<1: SearchAndDisplay Parent for all Prototypes>
<2: SearchAndDisplay Parent for one Prototype>
<3: SearchAndDisplay Sons for all Objects Traits>
<4: SearchAndDisplay Sons for one Object Traits>
<5: Modify Links Beetween Selected Prototypes>
<6: Modify Links Beetween All Prototypes>

Select a number or q to quit MenuSearchAndDestroy:5
------------------------------------------------------------


Modifier Liens sur quelle categorie :Nom
Nombre de Prototype ayant un attribut forme :195
Comportements Communs aux Prototypes  pontage et lésion :
   Sn N1prepN2 N1prepV AdjN NAdj PrepN

Prototype Lexical pontage :
 Ajout du comportement Sn a la liste des comportements individuels
 Ajout du comportement N1PrepN2 a la liste des comportements individuels
 Ajout du comportement N1PrepV a la liste des comportements individuels
 Ajout du comportement SAdjN a la liste des comportements individuels
 Ajout du comportement NSAdj a la liste des comportements individuels
 Ajout du comportement PrepN a la liste des comportements individuels
 Ajout du comportement N1NPivot a la liste des comportements individuels

Prototype Lexical lesion:
 Ajout du comportement Sn a la liste des comportements individuels si celui-ci n'est pas deja recense
 Ajout du comportement N1PrepN2 a la liste des comportements individuels si celui-ci n'est pas deja
recense
 Ajout du comportement N1PrepV a la liste des comportements individuels si celui-ci n'est pas deja
recense
 Ajout du comportement SAdjN a la liste des comportements individuels si celui-ci n'est pas deja recense
 Ajout du comportement NSAdj a la liste des comportements individuels si celui-ci n'est pas deja recense
 Ajout du comportement PrepN a la liste des comportements individuels si celui-ci n'est pas deja recense
 Ajout du comportement NPivotN2 a la liste des comportements individuels si celui-ci n'est pas deja
recense

--> Comportements Syntaxiques partages par les 2 Prototypes <pontage|Nom><lesion|Nom> :
  0. Sn
  1. N1PrepN2
  2. N1PrepV
  3. SAdjN
  4. SAdjN
  5. PrepN

--> Ensemble des Comportements Syntaxiques des 2 Prototypes <pontage|Nom><lesion|Nom> :
  0. Sn
  1. N1PrepN2
  2. N1PrepV
  3. SAdjN
  4. SAdjN
  5. PrepN
  6. NPivotN2
  7. N1NPivot

------------------------------------------------------------
<1: SearchAndDisplay Parent for all Prototypes>
<2: SearchAndDisplay Parent for one Prototype>
<3: SearchAndDisplay Sons for all Objects Traits>
<4: SearchAndDisplay Sons for one Object Traits>
<5: Modify Links Beetween Selected Prototypes>
<6: Modify Links Beetween All Prototypes>

Select a number or q to quit MenuSearchAndDestroy:q
------------------------------------------------------------
Menu SearchAndDestroy Shut Down
------------------------------------------------------------
```

Le savoir attaché aux prototypes représentant `pontage` et `lésion` est plus précis que celui attaché à l'objet portant les comportements partagés par tous les noms. Avant la détermination des comportements partagés, les prototypes lexicaux construits ne partagent pas d'arbre en commun:

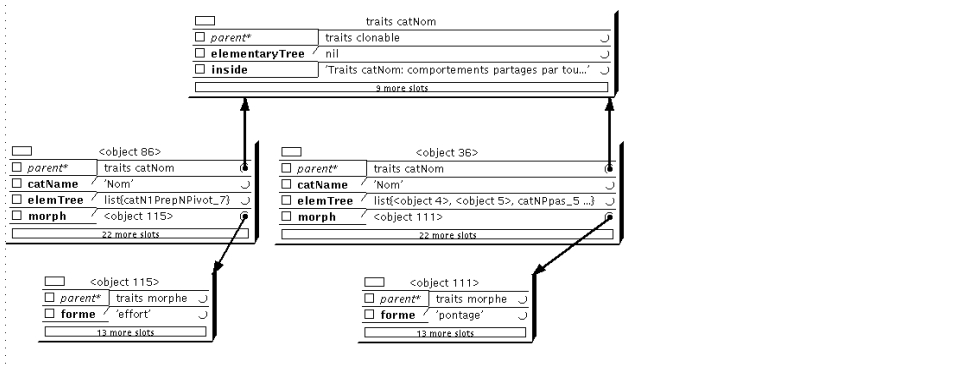

Figure a-2-7.4 : Pas de partage de comportement après génération

Après la détermination des comportements partagés, les prototypes lexicaux délèguent un comportement partagé :

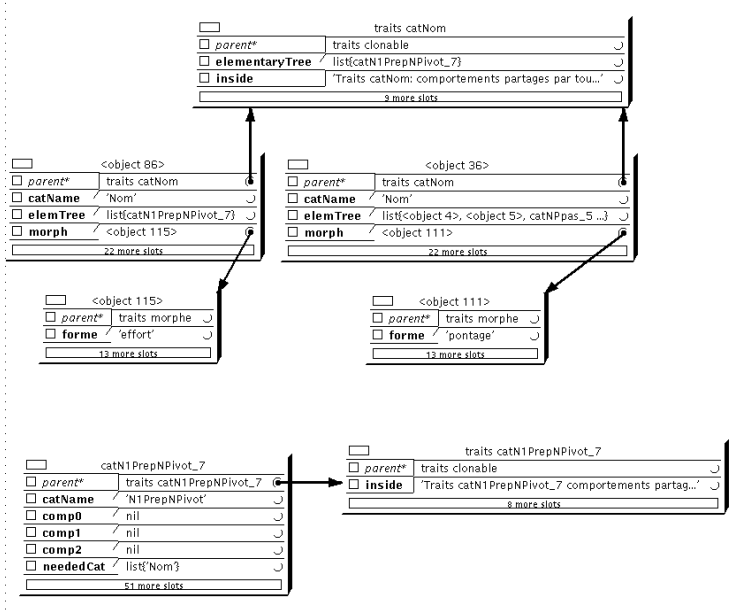

Figure a-2-7.5 : Le classement met en place un partage de comportement