|  |
| --- |
| **Sur la modélisation des êtres et de leurs manières d'être dans certaines situations**  Présentation du langage linéaire d'ICEO et de son implantation en Smalltalk (Pharo) |
|  |
|  |
|  |
|  |

Contenu

[Installation d'ICEO dans Pharo et premier exemple 5](#_Toc162639992)

[Exemple 2 Sur la composition d'une essence 9](#_Toc162639993)

[Exemple 3 Sur la notion de multiplicité 10](#_Toc162639994)

[Exemple 4 Sur le principe d'héritage des attributs des essences 11](#_Toc162639995)

[Exemple 5 Sur le principe d'identification des essences 13](#_Toc162639996)

[Exemple 6 Sur le principe d'identification des êtres 15](#_Toc162639997)

[Exemple 7 Sur la notion de méta essence (essence d'une essence). 16](#_Toc162639998)

[Exemple 8 Sur les notions de situation générique et de situation individuelle 19](#_Toc162639999)

[Exemple 9 Sur la notion de manière d'être essentielle et permanente 21](#_Toc162640000)

[Exemple 10 Sur la notion de manière d'être accidentelle 22](#_Toc162640001)

[Exemple 11 Sur les attributs d'une manière d'être 24](#_Toc162640002)

[Exemple 12 Sur les relations entre manières d'être 26](#_Toc162640003)

[Exemple 13 Sur la subsomption des manières d'être 28](#_Toc162640004)

[Exemple 14 Sur la relation entre états 30](#_Toc162640005)

[Exemple 15 Sur la notion d'état dans un état 32](#_Toc162640006)

[Exemple 16 Sur la notion d'être inconnu 34](#_Toc162640007)

[Exemple 17 Sur la notion de contrainte structurelle interne 36](#_Toc162640008)

[Exemple 18 Sur la notion de famille 39](#_Toc162640009)

[Exemple 19 Sur la notion de couleur 42](#_Toc162640010)

[Exemple 20 Sur la notion d'action 46](#_Toc162640011)

[Exemple 21 Sur les changements de situation 48](#_Toc162640012)

[Annexe Sur l'implantation d'ICEO en Pharo 51](#_Toc162640013)

Le langage linéaire d'ICEO est implanté en Smalltalk1.

Le choix de ce langage2 est lié à l'élégance de sa syntaxe mais aussi et surtout au fait que de nombreux concepts d'ICEO sont inspirés ou sont ceux de ce langage. La preuve en est que l'implantation d'ICEO en Smalltalk se contente de quelques centaines de lignes de code.

Nous avons utilisé Pharo qui est, parmi les versions de Smalltalk dérivées de la version de Smalltalk-80 originelle, l'une des plus proches.

Cette construction n'a quasiment exigé aucune modification de l'environnement de base de Pharo.

Nous allons présenter le langage ICEO à partir d'une série d'exemples inspirés de ceux donnés dans la première partie de ce document.

Pour les étudier, nous vous proposons de le faire de manière interactive dans l'environnement intégré de Pharo.

Ceci sera peut-être pour certains l'occasion de découvrir ce merveilleux langage qu'est Smalltalk.

1 Smalltalk est un [langage de programmation](https://fr.wikipedia.org/wiki/Langage_de_programmation) [orienté objet](https://fr.wikipedia.org/wiki/Programmation_orient%C3%A9e_objet) qui a été conçu en 1972 par des génies informatiques : [Alan Kay](https://fr.wikipedia.org/wiki/Alan_Kay), [Dan Ingals](https://fr.wikipedia.org/wiki/Daniel_Henry_Holmes_Ingalls,_Jr.), [Ted Kaehler](https://fr.wikipedia.org/w/index.php?title=Ted_Kaehler&action=edit&redlink=1) et [Adele Goldberg](https://fr.wikipedia.org/wiki/Adele_Goldberg) au [Palo Alto Research Center](https://fr.wikipedia.org/wiki/Palo_Alto_Research_Center) de [Xerox](https://fr.wikipedia.org/wiki/Xerox). Ce langage qui a inspiré une multitude d'autres langages (tels que Java, C++, Python, ...) fut l'un des tout premiers à disposer d'un [environnement de développement intégré](https://fr.wikipedia.org/wiki/Environnement_de_d%C3%A9veloppement) complètement graphique.

### Installation d'ICEO dans Pharo et premier exemple

Pharo est disponible librement en téléchargement sur le site https://pharo.org/download

Pour nos lecteurs qui ne connaissent pas Smalltalk, en particulier dans sa version Pharo, nous conseillons de commencer par la lecture du livre "*Pharo By Example*" de [Stéphane Ducasse](http://stephane.ducasse.free.fr/), Gordana Rakic, Sebastian Kaplar et Quentin Ducasse dont une version libre au format pdf peut être téléchargée à l'adresse

https://books.pharo.org/pharo-by-example9

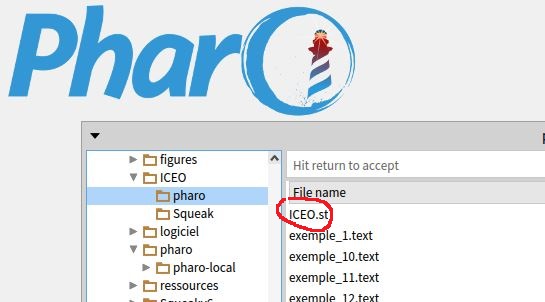
Les fichiers sources d'ICEO sont téléchargeables à l'adresse https://github.com/rodejaphgh/ICEO

ou à l'adresse alternative isnel.net/ICEO.zip

Ils comprennent le code source d'ICEO (fichier ICEO.st) et le code des exemples présentés dans ce document.

Installation d'ICEO

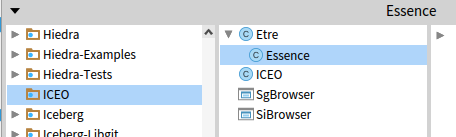
Pour installer ICEO dans l'environnement de Pharo, il convient d'utiliser l'outil "File Browser" accessible sous l'onglet "System" dans la barre d'outils de Pharo. Exécuter l'option "Install into image" du fichier ICEO.st précédemment téléchargé :



Attendre quelques instants (soyez patient !) que la barre de menu principale affiche un nouvel onglet nommé ICEO :

C:\Users\rober\AppData\Local\Microsoft\Windows\INetCache\Content.Word\2024-03-27 12_44_00-D__myosotis_ice_ICE_pharo_Pharo11-SNAPSHOT-64bit-aece1b5.image.jpg

Dans un Browser, la catégorie ICEO s'affiche avec les classes Etre, Essence, et ICEO :

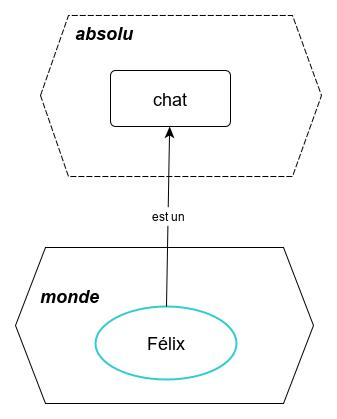


La classe ICEO est l'interprète du langage linéaire d'ICEO.

Une instance de cette classe est créée lors de l'installation, nommée "iceo" (en minuscules).

Avant de continuer, il est conseillé de sauvegarder votre image. Pour retrouver Pharo dans l'état actuel lors du prochain lancement, il suffira ainsi de choisir cette image.

Intéressons-nous maintenant au premier exemple, qui correspond au graphe suivant :



Dans la fenêtre File Browser sélectionner le fichier "exemple\_1.text".

Le texte suivant s'affiche1 :

iceo definition: #chat.

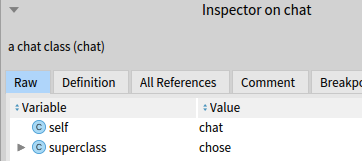
iceo soit: #Félix essence: chat.

Rappelons que dans ICEO le nom commun des essences s'écrit obligatoirement (comme en langage naturel) avec une minuscule et que le nom propre des êtres s'écrit habituellement avec une majuscule.

Dans ce texte, vous pouvez évaluer les lignes une par une ou en bloc, en utilisant l'option "doIt" offerte avec le bouton droit de la souris. Pour une évaluation en bloc, il faut que chaque ligne se termine par un point.

La première ligne définit l'essence chat dans l'absolu, subsumée par l'essence chose.

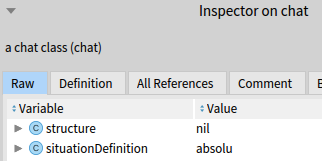
Pour le vérifier, il suffit d'inspecter le mot chat :



On voit que l'essence chat a été créée sous la forme d'une classe Smalltalk. Les différents attributs qui sont listés seront expliqués au moment opportun.

L'attribut "superclass" montre que le genus de l'essence chat est l'essence chose qui a été définie lors de l'initialisation d'ICEO.

L'attribut "situationDefinition" montre que l'essence chat est définie dans l'absolu :

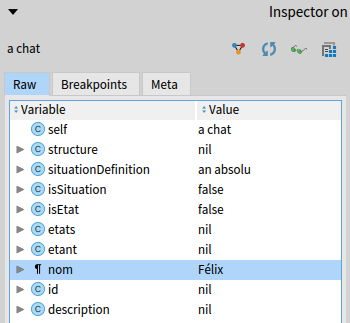


1 Ne pas utiliser une fenêtre de type "Playground" qui n'offre pas tout-à-fait les mêmes possibilités.

L'accès à l'essence chat pourrait se faire avec l'expression " absolu get: #chat " où absolu est la situation générique de définition de chat (situation par défaut), mais ce n'est pas nécessaire car les essences définies dans l'absolu sont accessibles directement par leur nom.

La deuxième ligne crée l'être nommé Félix comme instance de chat dans le monde.

L'expression "monde get: #Félix" montre que Félix est un chat présent dans le monde :

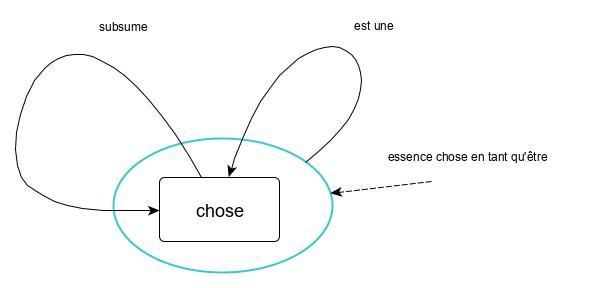


Noter que l'accès à l'être nommé Félix situé dans le monde exige de passer par l'expression "monde get: #Félix", car les êtres ne sont pas connus dans l'absolu mais dans le monde.

L'essence chose est son propre genus et, en tant qu'être, est instance d'elle-même (elle est sa propre méta-essence).

Ainsi les expressions " chose getGenus " et " chose getEssence" retournent chose.

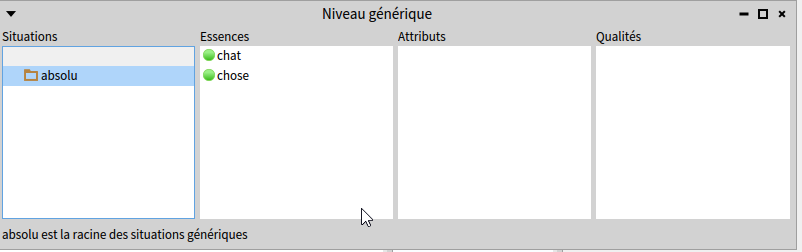
Ceci est conforme au schéma :



Vous êtes maintenant parés pour étudier d'autres exemples.

Services accessibles via l'onglet "ICEO" de la fenêtre principale1

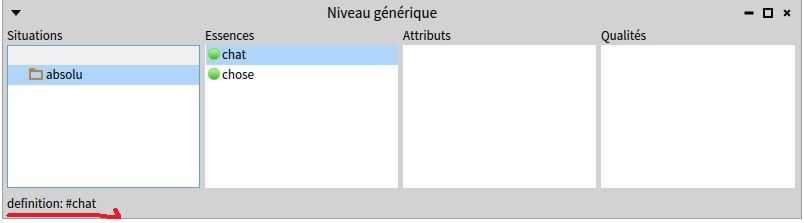
L'option "SgBrowser" ouvre une fenêtre sur les situations génériques et les essences définies dans l'absolu



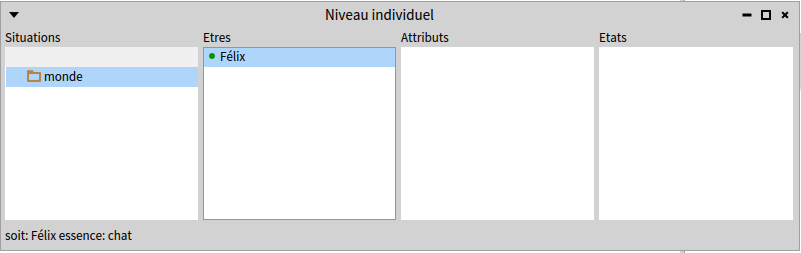
Un double-click sur un item (par exemple l'essence chat) ouvre une fenêtre permettant d'inspecter celle-ci.

Le label affiché en bas de la fenêtre rappelle l'expression qui a donné naissance à l'entité sélectionnée.

Exemple, pour l'essence chat :



L'option "SiBrowser" ouvre une fenêtre sur les situations individuelles et les êtres définis dans le monde :



L'option "ICEO reset" réinitialise le contenu de absolu et du monde. Ceci évite qu'ICEO ne conserve la mémoire des exemples précédemment étudiés.

En principe, cette instruction n'est pas nécessaire ici, car il s'agit de notre premier exemple.

Par contre, il peut être nécessaire de l'exécuter avant de démarrer l'étude d'un autre exemple, pour éviter des interférences avec le précédent.

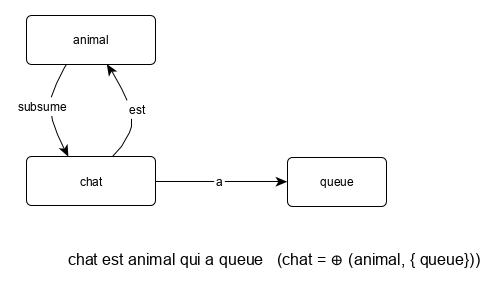
Rappelons que dans ICEO :

* deux essences de même nom ne peuvent être définies dans la même situation.
* deux êtres de même nom ne peuvent coexister dans une situation, sauf s'ils peuvent être distingués par l'un de leurs états (par exemple, deux êtres nommés Pierre dont l'un est fils et l'autre père dans une même famille).

1 Ces services sont également accessibles avec le bouton gauche de la souris dans la fenêtre principale.

### Exemple 2 Sur la composition d'une essence

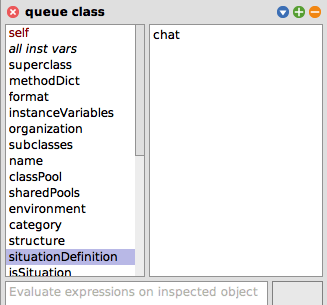
Cet exemple correspond au graphe suivant :



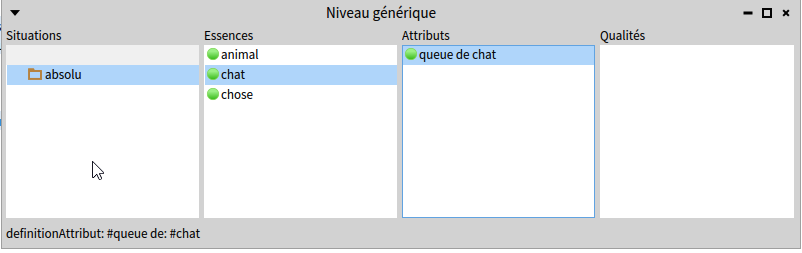
L'essence chat subsumée par animal possède un attribut propre nommé queue :

iceo definition: #animal.iceo definition: #chat genus: animal.iceo definitionAttribut: #queue de: chat.

Les essences animal et chat ont été définies dans l'absolu, tandis que l'essence chat constitue la situation de définition de sa queue, ce qui peut être vérifié par l'expression " chat getEssenceAttribut: #queue " :

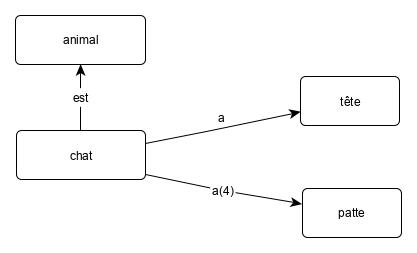


Ceci est confirmé dans un SgBrowser :



### Exemple 3 Sur la notion de multiplicité

Considérons le graphe suivant :



iceo definition: #animal.

iceo definition: #chat genus: animal.

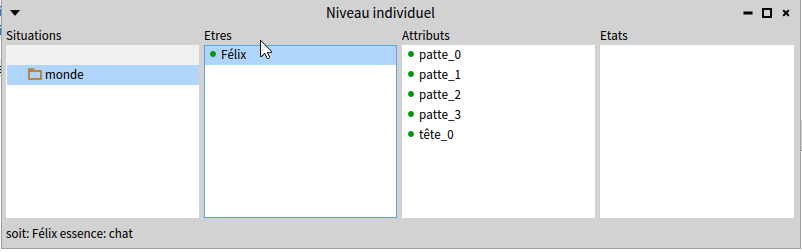
iceo definitionAttribut: #tête de: chat cardinalite: 1.

iceo definitionAttribut: #patte de: chat cardinalite: 4.

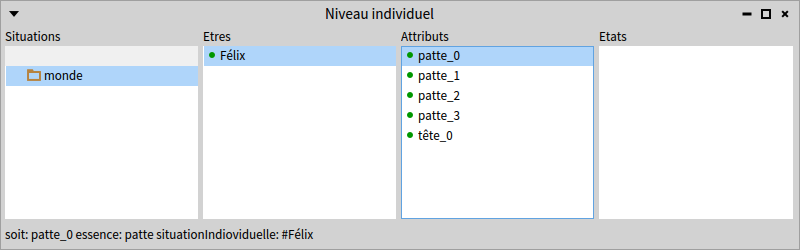
iceo soit: #Félix essence: chat.

Comme chat est considéré par défaut comme un individu, son instanciation entraîne la création de ses attributs.

Ainsi, l'inspection de "Félix " montre que Félix possède une tête et quatre pattes :



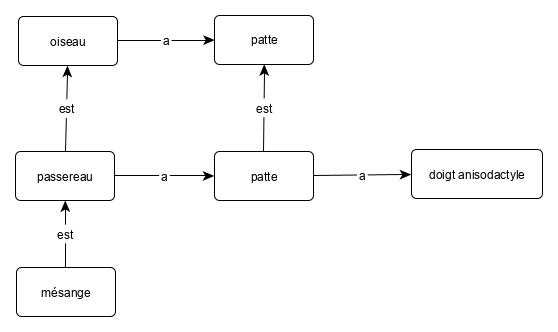
Le nom des membres de Félix a été généré automatiquement en tenant compte du nom de leur essence.



On voit que la situationIndividuelle de patte\_0 est Félix; il s'agit d'un attribut propre de Félix.

### Exemple 4 Sur le principe d'héritage des attributs des essences

Considérons le graphe suivant : :



iceo definition: #oiseau.

iceo definition: #passereau genus: oiseau.

iceo definition: #mésange genus: passereau.

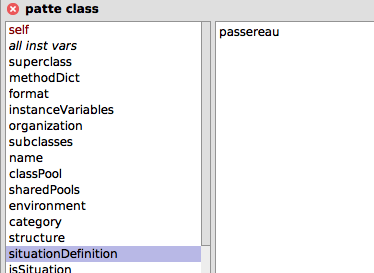
iceo definitionAttribut: #patte de: oiseau.

iceo definitionAttribut: #doigt de: (oiseau getEssenceAttribut: #patte).

iceo definitionAttribut: #patte de: passereau genus: (oiseau getEssenceAttribut: #patte).

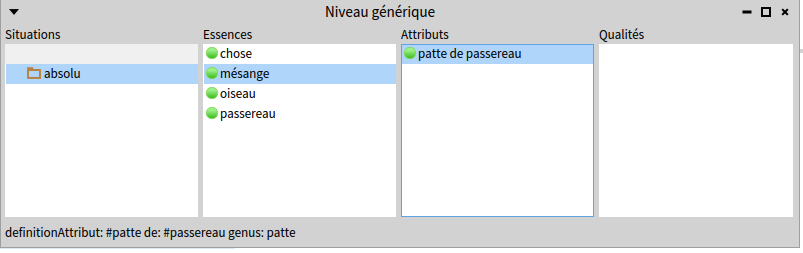
iceo definitionAttribut: 'doigt anisodactyle' de: (passereau getEssenceAttribut: #patte).

L'inspection de " mésange getEssenceAttribut: #patte " donne :

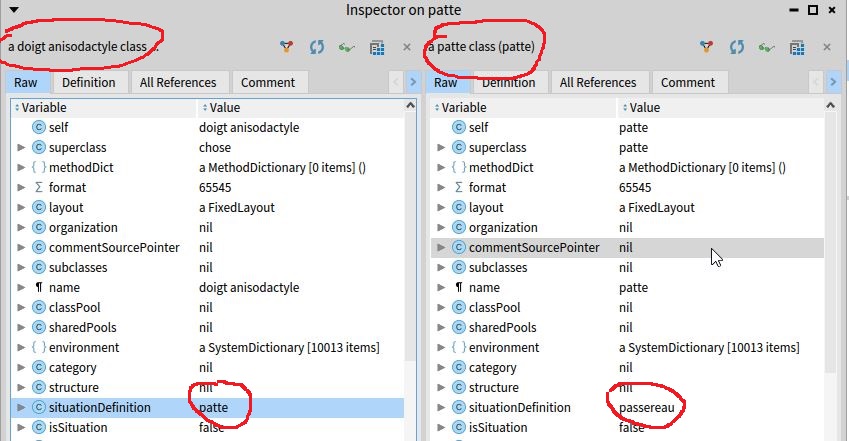


Il s'agit donc de l'attribut patte de passereau

L'inspection de mésange donne :

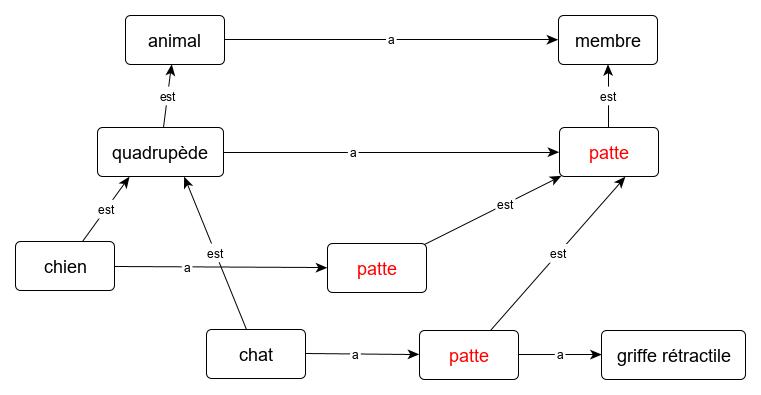


L'essence attribut "doigt anisodactyle" de patte de mésange est héritée de patte de passereau, ce qui confirmé par l'inspection de " (mésange getEssenceAttribut: #patte) getEssencesAttributs at: 1 " :



### Exemple 5 Sur le principe d'identification des essences

Considérons le graphe suivant :



iceo definition: #animal.

iceo definition: #quadrupède genus: animal.

iceo definition: #chien genus: quadrupède.

iceo definition: #chat genus: quadrupède.

iceo definitionAttribut: #membre de: animal.

iceo definitionAttribut: #patte de: quadrupède genus: (animal getEssenceAttribut: #membre).

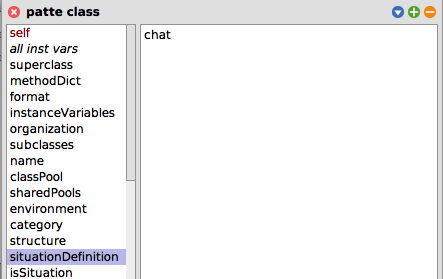
iceo definitionAttribut: #patte de: chien genus: (quadrupède getEssenceAttribut: #patte).

iceo definitionAttribut: #patte de: chat genus: (quadrupède getEssenceAttribut: #patte).

iceo definitionAttribut: 'griffe rétractile' de: (chat getEssenceAttribut: #patte).

On voit dans cet exemple que diverses essences peuvent avoir le même nom. L'identification d'une essence peut se faire à partir de l'essence dont elle est attribut. Ainsi, l'inspection de l'expression

" chat getEssenceAttribut: #patte " donne:



Du fait que différentes essences peuvent avoir le même nom, la génération à la demande d'un identificateur propre à chaque essence peut être est utile, pour nous pauvres lecteurs humains, pour vérifier que l'essence patte de quadrupède est différente de celle de chat et de chien.

Ainsi, les expressions suivantes :

Transcript show: 'patte de quadrupède : ', (quadrupède getEssenceAttribut: #patte) getId; cr. Transcript show: 'patte de chien : ', (chien getEssenceAttribut: #patte) getId; cr. Transcript show: 'patte de chat : ', (chat getEssenceAttribut: #patte) getId; cr

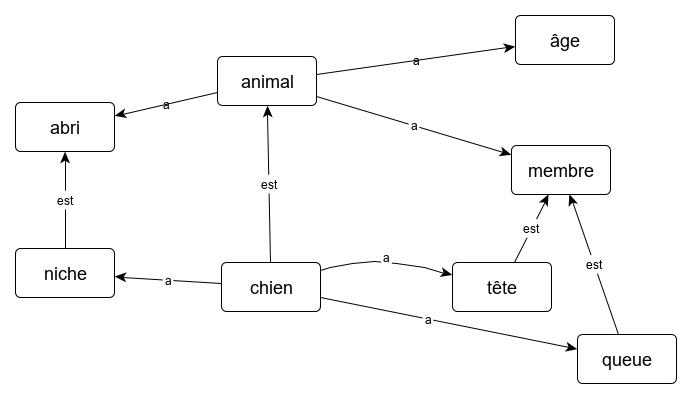
donnent, en supposant qu'une fenêtre de type Transcript soit ouverte :

patte de quadrupède : patte\_0patte de chien : patte\_1patte de chat : patte\_2

Exemple\_5\_bis

L'identification d'une essence peut aussi se faire en utilisant le nom de son genus.

Considérons le graphe suivant :



iceo definition: #animal.

iceo definitionAttribut: #âge de: animal.

iceo definitionAttribut: #membre de: animal.

iceo definitionAttribut: #abri de: animal.

iceo definition: #chien genus: animal.

iceo definitionAttribut: #niche de: chien genus: (animal getEssenceAttribut: #abri).

iceo definitionAttribut: #tête de: chien genus: (animal getEssenceAttribut: #membre).

iceo definitionAttribut: #queue de: chien genus: (animal getEssenceAttribut: #membre).

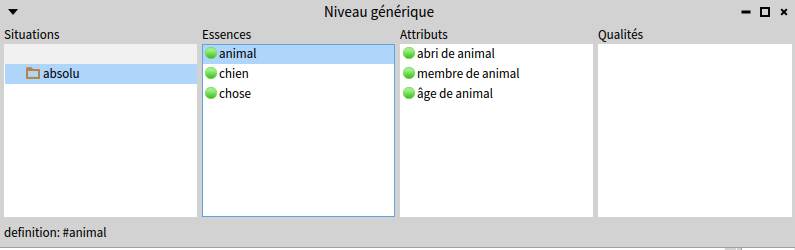
L'expression " chien getEssenceAttribut: #âge " donne : âge

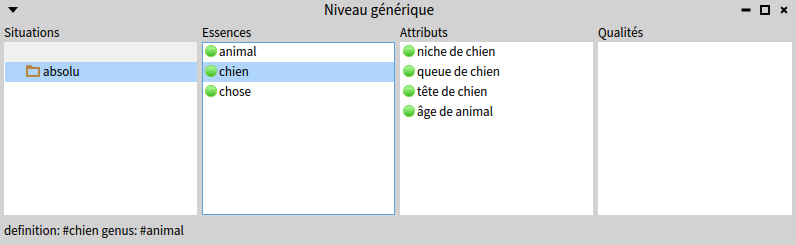
Il s'agit de l'attribut âge hérité de l'essence animal.

L'expression " chien getEssenceAttribut: #abri " donne : niche

et l'expression " chien getEssencesAttributs: #membre " donne : an OrderedCollection(queue tête)

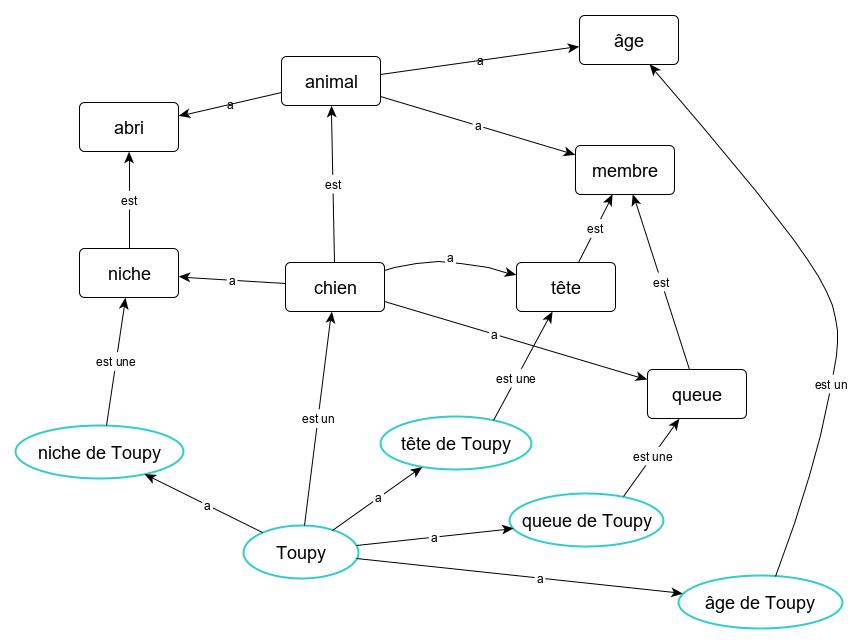
Ce qui se vérifie dans les fenêtres suivantes :





### Exemple 6 Sur le principe d'identification des êtres

Considérons le graphe suivant :



iceo definition: #animal.

iceo definitionAttribut: #âge de: animal cardinalite: 1.

iceo definitionAttribut: #membre de: animal.

iceo definitionAttribut: #abri de: animal.

iceo definition: #chien genus: animal.

iceo definitionAttribut: #niche de: chien genus: (animal getEssenceAttribut: #abri) cardinalite: 1.

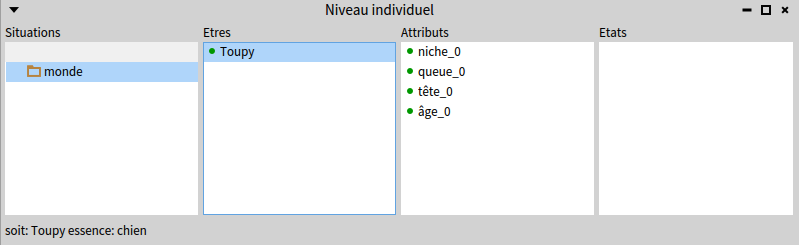
iceo definitionAttribut: #tête de: chien genus: (animal getEssenceAttribut: #membre) cardinalite: 1.

iceo definitionAttribut: #queue de: chien genus: (animal getEssenceAttribut: #membre) cardinalite: 1.

iceo soit: #Toupy essence: chien.

L'expression " (monde get: #Toupy) getEtresAttributs " donne : an OrderedCollection(a niche a queue a tête a âge)

Les attributs de Toupy ont été créés automatiquement car il s'agit par défaut d'un individu. Leur nom a été créé par un générateur de symboles en se basant sur le nom de leur essence.

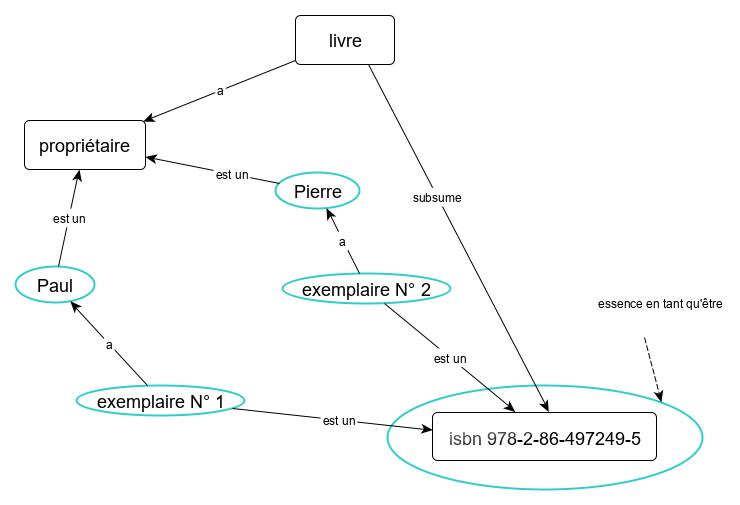


L'expression " ((monde get: #Toupy) getEtreAttribut: #abri) getNom " donne : #'Niche\_0'

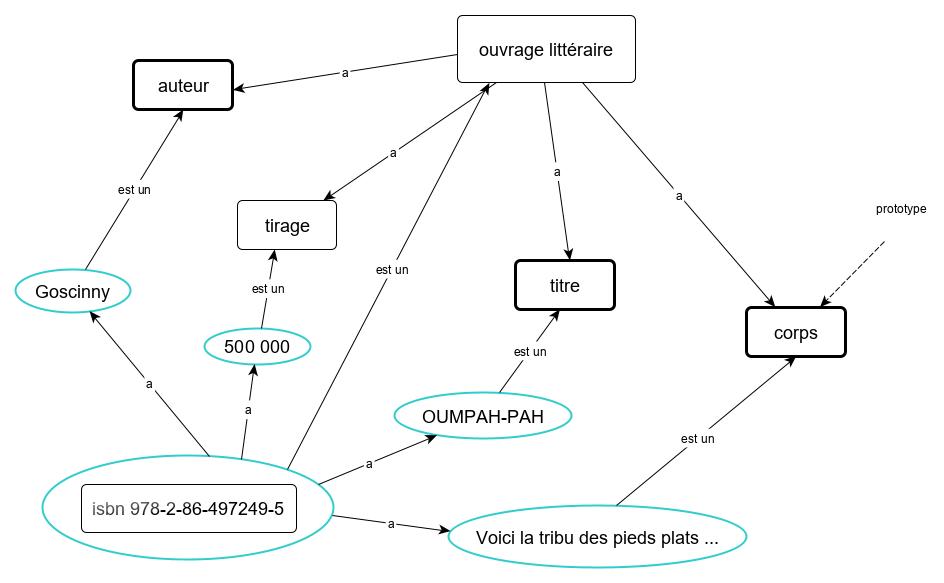
et l'expression " ((monde get: #Toupy) getEtreAttribut: #tête) getEtresAttributs " donne : an OrderedCollection(an oreille an oreille)

### Exemple 7 Sur la notion de méta essence (essence d'une essence).

Le graphe suivant correspond à la définition de l'essence 'isbn 978-2-86-497249-5' subsumée par l'essence livre, avec deux exemplaires dont les propriétaires respectifs sont Paul et Pierre :



En tant qu'être, l'essence "isbn 978-2-86-497249-5" est instance de l'essence "ouvrage littéraire" (sa méta essence" :



Voici la représentation du contenu de ces deux graphes dans ICEO :

iceo definition: 'ouvrage littéraire'.

iceo definitionAttribut: #corps de: (absolu get: 'ouvrage littéraire') isPrototype: true.

iceo definitionAttribut: #tirage de: (absolu get: 'ouvrage littéraire').

iceo definitionAttribut: #auteur de: (absolu get: 'ouvrage littéraire') isPrototype: true.

iceo definitionAttribut: #titre de: (absolu get: 'ouvrage littéraire') isPrototype: true.

iceo definition: #livre.

iceo definitionAttribut: #propriétaire de: livre.

iceo definition: 'isbn 978-2-86-497249-5' genus: livre

metaEssence: (absolu get: 'ouvrage littéraire').

(absolu get: 'isbn 978-2-86-497249-5') attributionEtre: '500000' .

essence: ((absolu get: 'ouvrage littéraire') getEssenceAttribut: #tirage).

(absolu get: 'isbn 978-2-86-497249-5') attributionEtre: #Goscinny.

essence: ((absolu get: 'ouvrage littéraire') getEssenceAttribut: #auteur).

(absolu get: 'isbn 978-2-86-497249-5') attributionEtre: 'OUMPAH-PAH'.

essence: ((absolu get: 'ouvrage littéraire') getEssenceAttribut: #titre).

(absolu get: 'isbn 978-2-86-497249-5') attributionEtre: 'Voici la tribu des pieds plats...'.

essence: ((absolu get: 'ouvrage littéraire') getEssenceAttribut: #corps).

iceo soit: #Paul essence: (livre getEssenceAttribut: #propriétaire).

iceo soit: #Exemplaire\_1 essence: (absolu get: 'isbn 978-2-86-497249-5') .

(monde get: #Exemplaire\_1) attributionEtre: (monde get: #Paul).

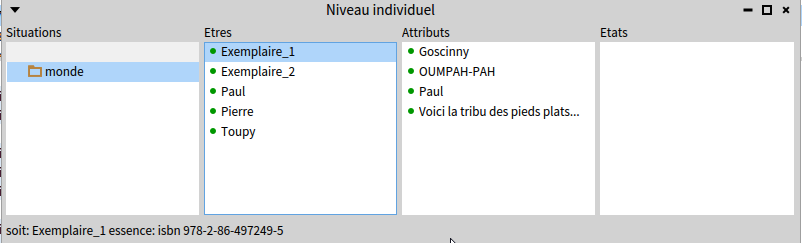
iceo soit: #Pierre essence: (livre getEssenceAttribut: #propriétaire).

iceo soit: #Exemplaire\_2 essence: (absolu get: 'isbn 978-2-86-497249-5').

(monde get: #Exemplaire\_2) attributionEtre: (monde get: #Pierre).

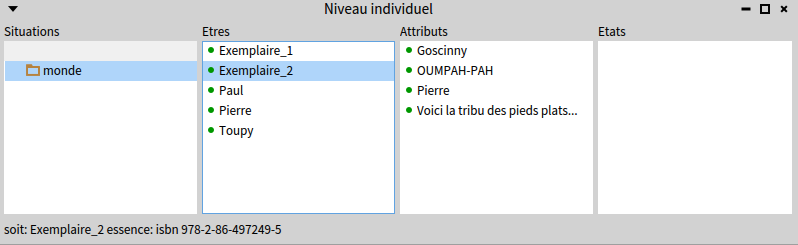
La ligne encadrée définit l'essence "isbn 978-2-86-497249-5" comme subsumée par livre et instance d'ouvrage littéraire (sa méta essence).

Voici le résultat pour l'exemplaire 1 :



On voit que le propriétaire de cet exemplaire est Paul, mais qu'il a aussi le titre, l'auteur et le corps de l'ouvrage littéraire. Ces attributs ont été "hérités" car qualifiés de "prototypes" au niveau de l'ouvrage littéraire.

Et le résultat pour l'exemplaire 2 :



On voit que le propriétaire de ce deuxième exemplaire est Pierre, mais qu'il a les mêmes attributs titre, auteur et corps que l'exemplaire 1 (ce qui est logique, car tous les exemplaires d'un livre ont le même titre, le même auteur et le même contenu).

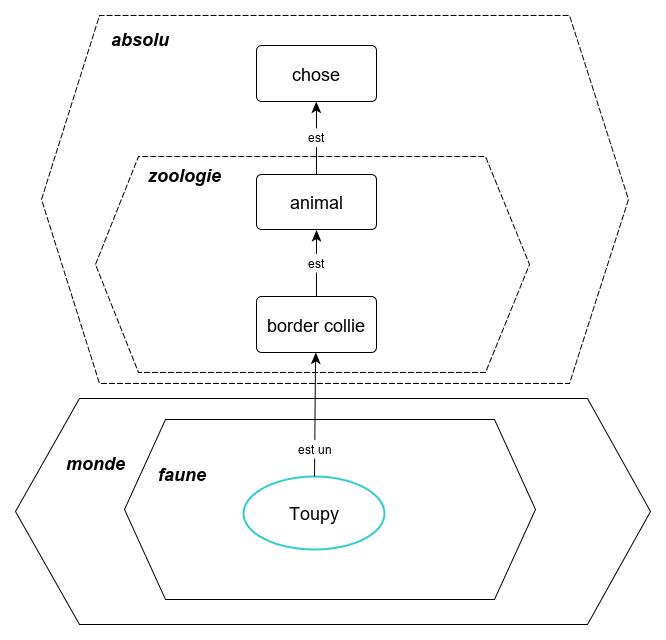
L'évaluation de l'expression "(monde get: #Exemplaire\_1) getEtreAttribut: #tirage" afficherait un message d'erreur signalant que l'attribut tirage n'a pas été trouvé. Ceci s'explique par le fait que l'attribut tirage n'a pas été qualifié de prototype au niveau de ouvrage littéraire.

Par contre, l'expression "((absolu get: 'isbn 978-2-86-497249-5') getEtreAttribut: #tirage) getNom" appliquée à l'essence 'isbn 978-2-86-497249-5' en tant qu'être retourne bien '500000'1.

1 Pour traduire '500000'en nombre entier, il suffirait en Smalltalk de lui envoyer le message "asInteger"

### Exemple 8 Sur les notions de situation générique et de situation individuelle

Considérons le graphe suivant :



iceo definitionSituation: 'zoologie'.

iceo definition: 'animal' situation: zoologie.

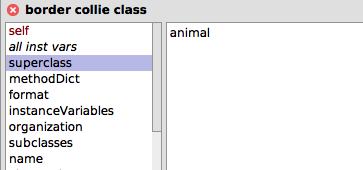
iceo definition: 'border collie' situation: zoologie genus: (zoologie get: #animal).

iceo soit: #faune situationGenerique: zoologie.

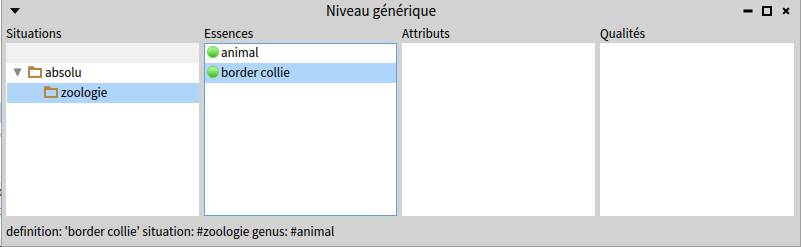
iceo soit: #Toupy essence: (zoologie get: 'border collie') situationIndividuelle: (monde getSituation: #faune).

L'essence nommée "border collie" est subsumée par l'essence animal dans une situation générique nommée "zoologie", et un être "Toupy" est créé dans une situation individuelle nommée "faune". La situation générique zoologie est incluse dans l'absolu et la situation individuelle faune, instance de la situation zoologie, est incluse dans le monde.

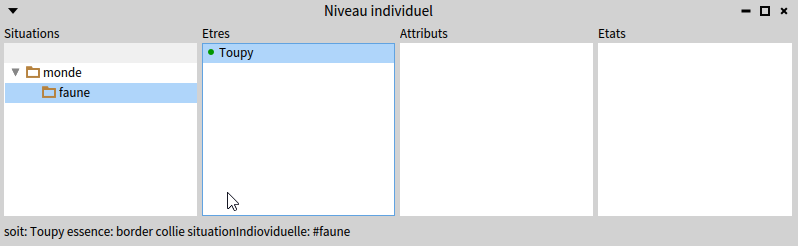
L'inspection de "border collie" en utilisant l'expression "zoologie get: 'border collie' " permet de vérifier qu'elle est bien subsumée par l'essence animal :



et sa situation générique est zoologie :

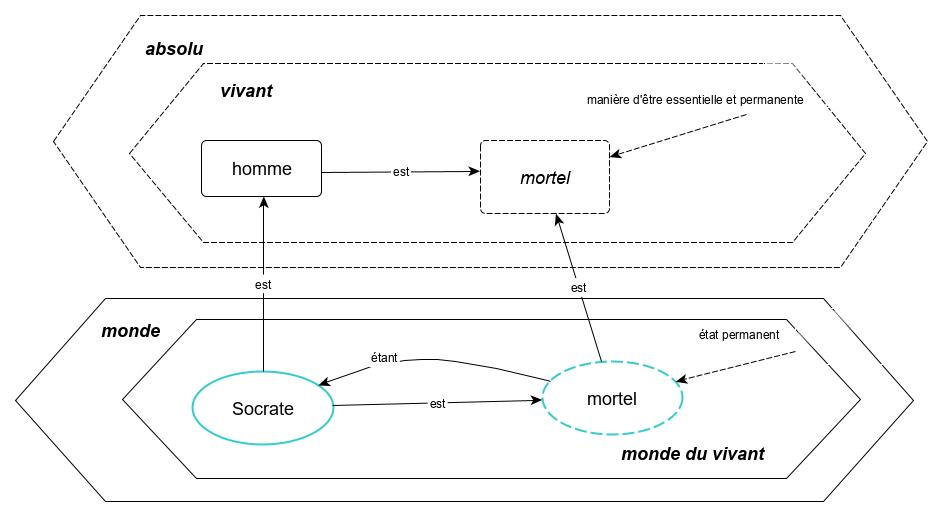


L'expression " (monde getSituation: #faune) get: #Toupy " permet de vérifier que Toupy est bien une instance de border collie présente dans la situation individuelle faune, ce qui se vérifie dans la fenêtre suivante :



### Exemple 9 Sur la notion de manière d'être essentielle et permanente

Considérons le graphe suivant :



iceo definitionSituation: #vivant.

iceo definition: #homme situation: vivant.

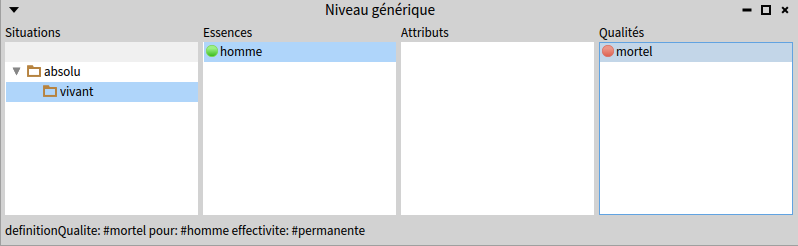
iceo definitionQualiteEssentielle: #mortel pour: (vivant get: #homme) effectivite: #permanente.

iceo soit: 'monde du vivant' situationGenerique: vivant.

iceo soit: #Socrate essence: (vivant get: #homme) situationIndividuelle: (monde get: 'monde du vivant').

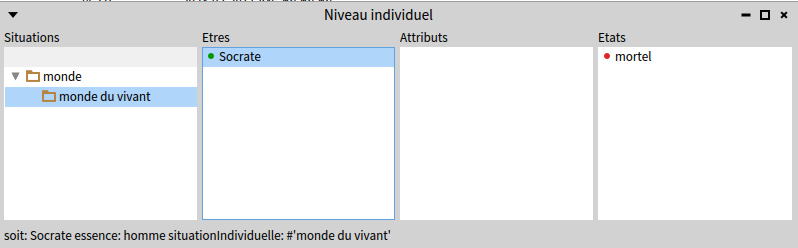
La troisième ligne définit la manière d'être "mortel" comme essentielle et permanente pour homme.

C'est ce qui se vérifie dans la fenêtre suivante :



La petite icône rouge associée à la qualité "mort" indique qu'il s'agit d'une qualité essentielle.

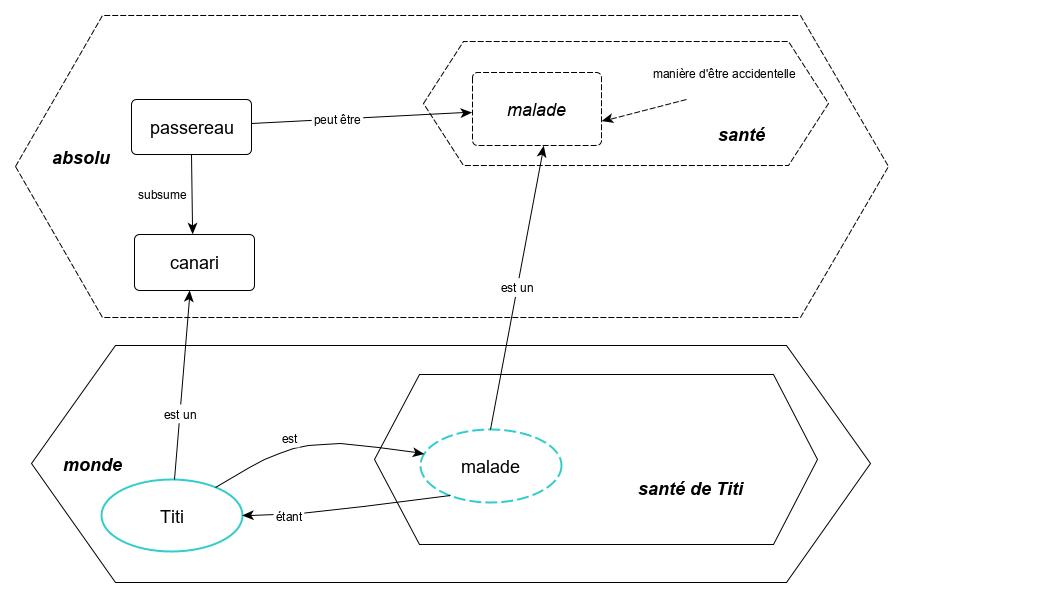
La fenêtre suivante montre que Socrate, en tant qu'homme, est bien mortel :



Ainsi, qu'Aristote se rassure : Socrate en tant qu'homme était bien mortel !

### Exemple 10 Sur la notion de manière d'être accidentelle

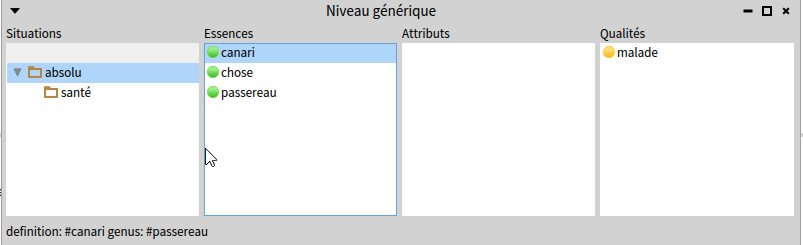
Considérons le graphe suivant :



iceo definition: #passereau. iceo definition: #canari genus: passereau. iceo definitionSituation: #santé. iceo definitionQualite: #malade situation: santé. passereau peutEtre: (santé get: #malade). iceo soit: #Titi essence: canari. iceo soit: 'santé de Titi' situationGenerique: santé. (monde get: #Titi) addEtat: (santé get: #malade) dansSituation: (monde get: 'santé de Titi').

L'essence passereau peut avoir la manière d'être accidentelle "malade" (statut par défaut de la manière d'être "malade" définie à la deuxième ligne).

Cette manière d'être est héritée par l'essence canari qui est subsumée par passereau.



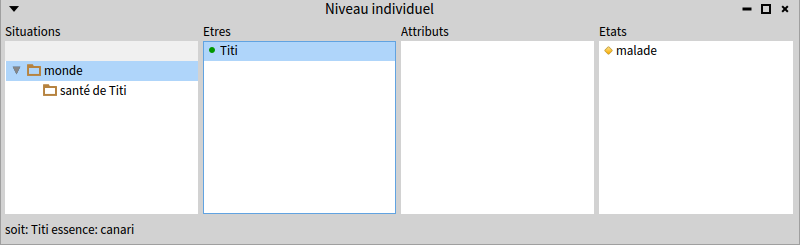
En tant qu'essence, la qualité malade est affichée avec un petit cercle jaune, précisant qu'il s'agit d'une qualité accidentelle.

Pour que Titi soit malade, il faut que cet état lui soit explicitement affecté, car il s'agit d'une manière d'être accidentelle.

C'est ce qui est fait dans la dernière ligne.

L'état malade de Titi est instance de la manière d'être malade.

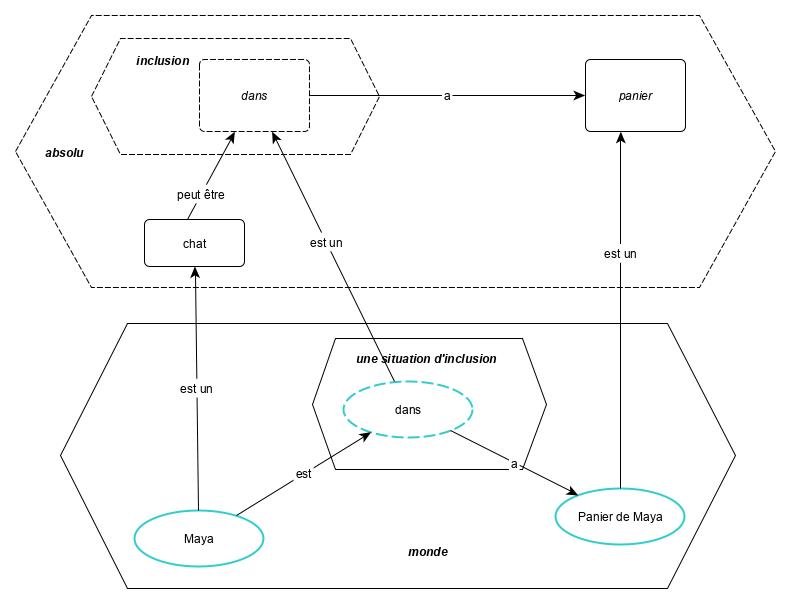
C'est ce que confirme la fenêtre suivante :



Souhaitons à Titi un prompt rétablissement.

### Exemple 11 Sur les attributs d'une manière d'être

Considérons le graphe suivant:



Il est la représentation de la phrase : "La chatte Maya est dans son panier"

iceo definition: #chat.

iceo definition: #panier.

iceo definitionSituation: #inclusion.

iceo definitionQualite: #dans situation: inclusion.

(inclusion get: #dans) referenceEssence: panier.

chat peutEtre: (inclusion get: #dans).

iceo soit: 'une situation d''inclusion' situationGenerique: inclusion.

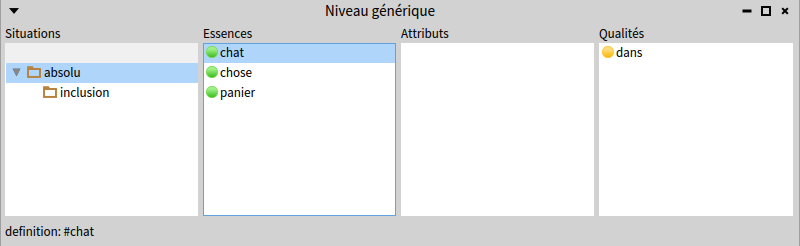
iceo soit: #Maya essence: chat.

iceo soit: 'Panier de Maya' essence: panier.

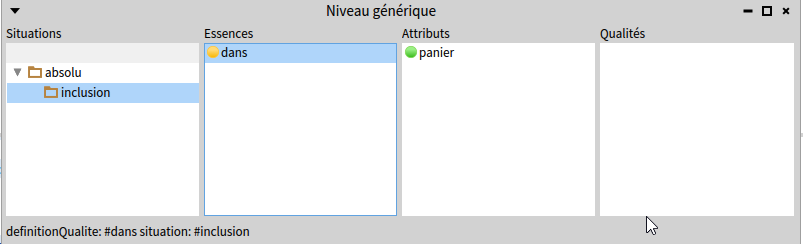
(monde get: #Maya) addEtat: (inclusion get: #dans) dansSituation: (monde get: 'une situation d''inclusion').

((monde get: #Maya) getEtat: #dans) attributionEtre: (monde get: 'Panier de Maya' ).

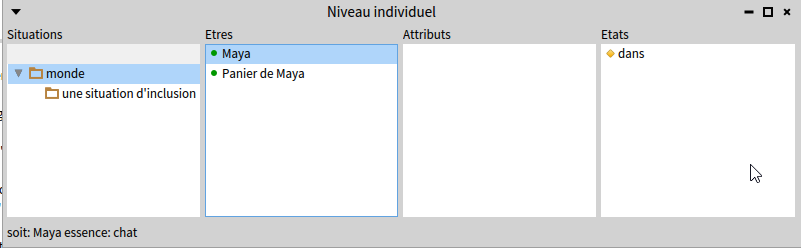
Les fenêtres suivantes montrent que "dans" est une qualité accidentelle de chat :

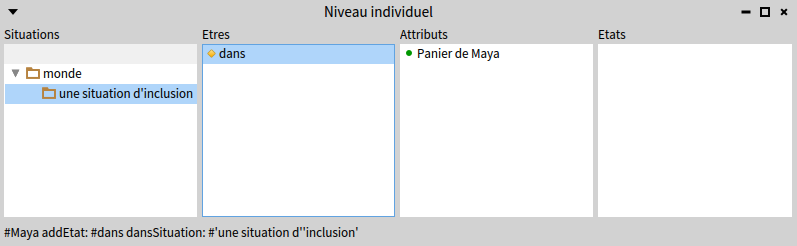


et que l'essence panier est attribut de "dans" :



et celles-ci montrent que Maya est "dans" son panier :





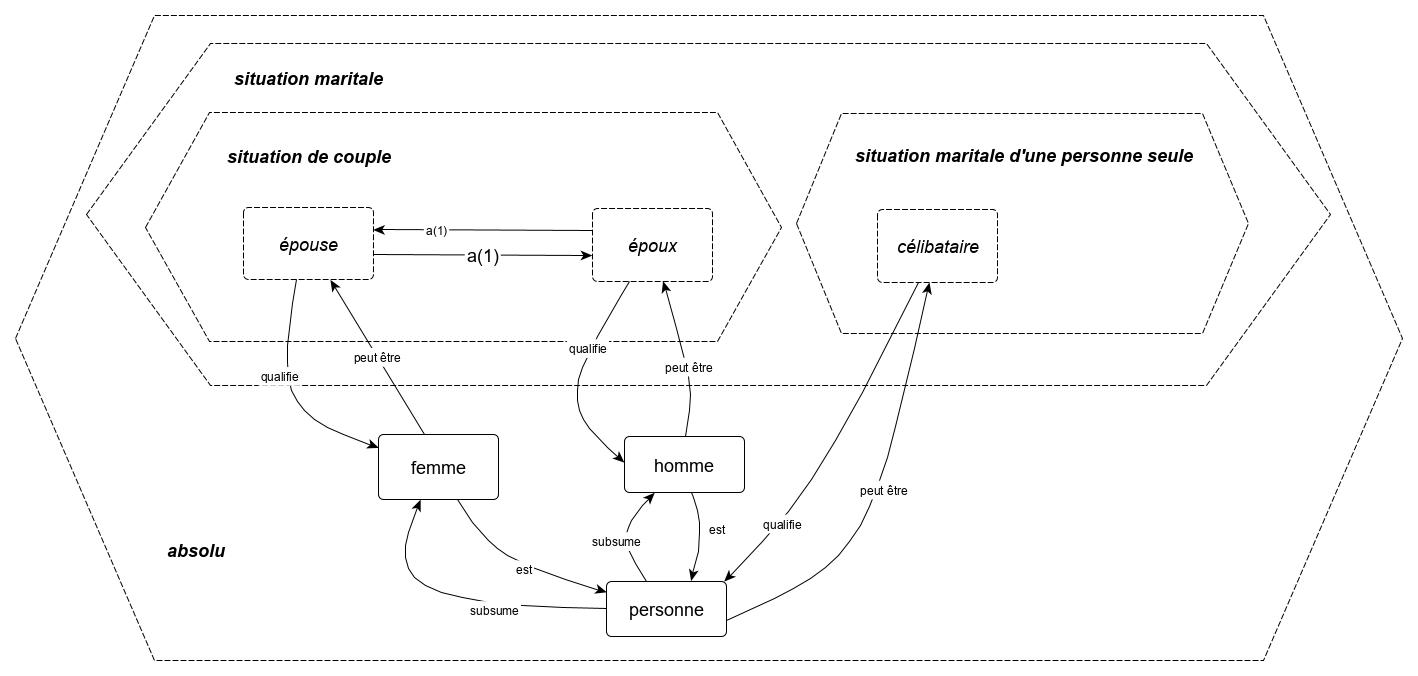
L'expression " (monde get: #Maya) getEtresAttributsEnTantQue: (inclusion get: #dans) "

donne : an OrderedCollection(a panier)

Suivant la même logique, nous pourrions représenter la plupart des prépositions de la langue française (sur, avec, entre, ...)

### Exemple 12 Sur les relations entre manières d'être

Considérons le graphe suivant :



iceo definition: #personne

iceo definition: #femme genus: personne

iceo definition: #homme genus: personne

iceo definition: #maire genus: personne

iceo definitionSituation: 'situation de couple'

iceo definitionSituation: 'situation de personne seule'

iceo definitionQualite: #épouse situation: (absolu getSituation: 'situation de couple')

iceo definitionQualite: #époux situation: (absolu getSituation: 'situation de couple')

iceo definitionQualite: #célibataire situation: (absolu getSituation: 'situation de personne seule')

((absolu getSituation: 'situation de couple') get: #épouse)

associationQualite: ((absolu getSituation: 'situation de couple') get: #époux).

femme peutEtre: ((absolu getSituation: 'situation de couple') get: #épouse).

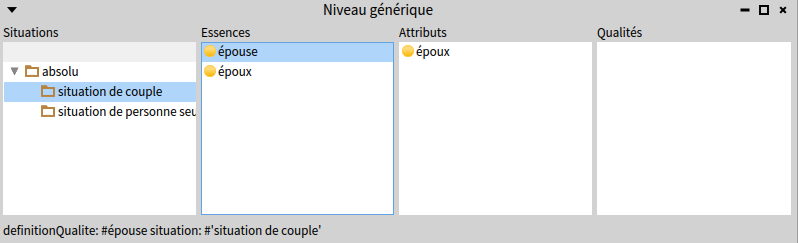
femme peutEtre: ((absolu getSituation: 'situation de personne seule') get: #célibataire).

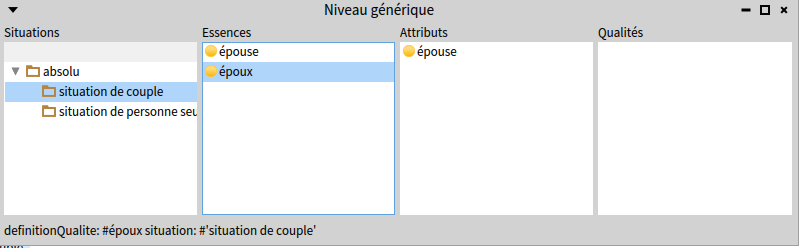
homme peutEtre: ((absolu getSituation: 'situation de couple') get: #époux).

homme peutEtre: ((absolu getSituation: 'situation de personne seule') get: #célibataire).

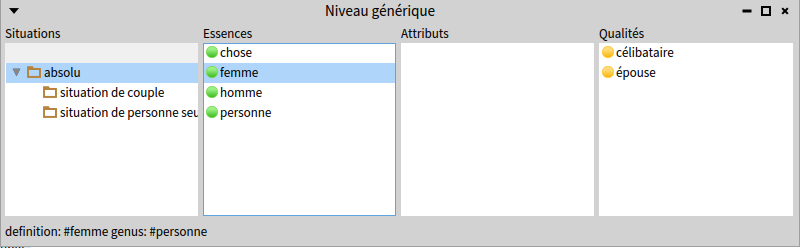
La ligne encadrée définit une relation bidirectionnelle entre les manières d'être épouse et époux.

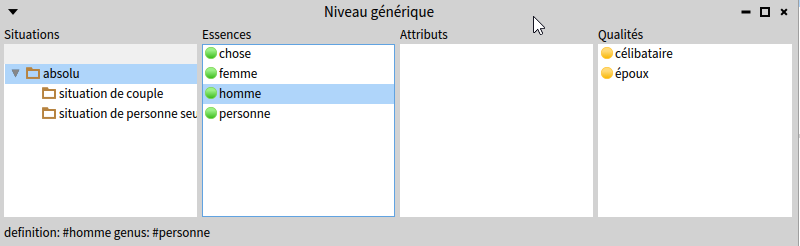
Ceci peut être vérifié dans les fenêtres suivantes :





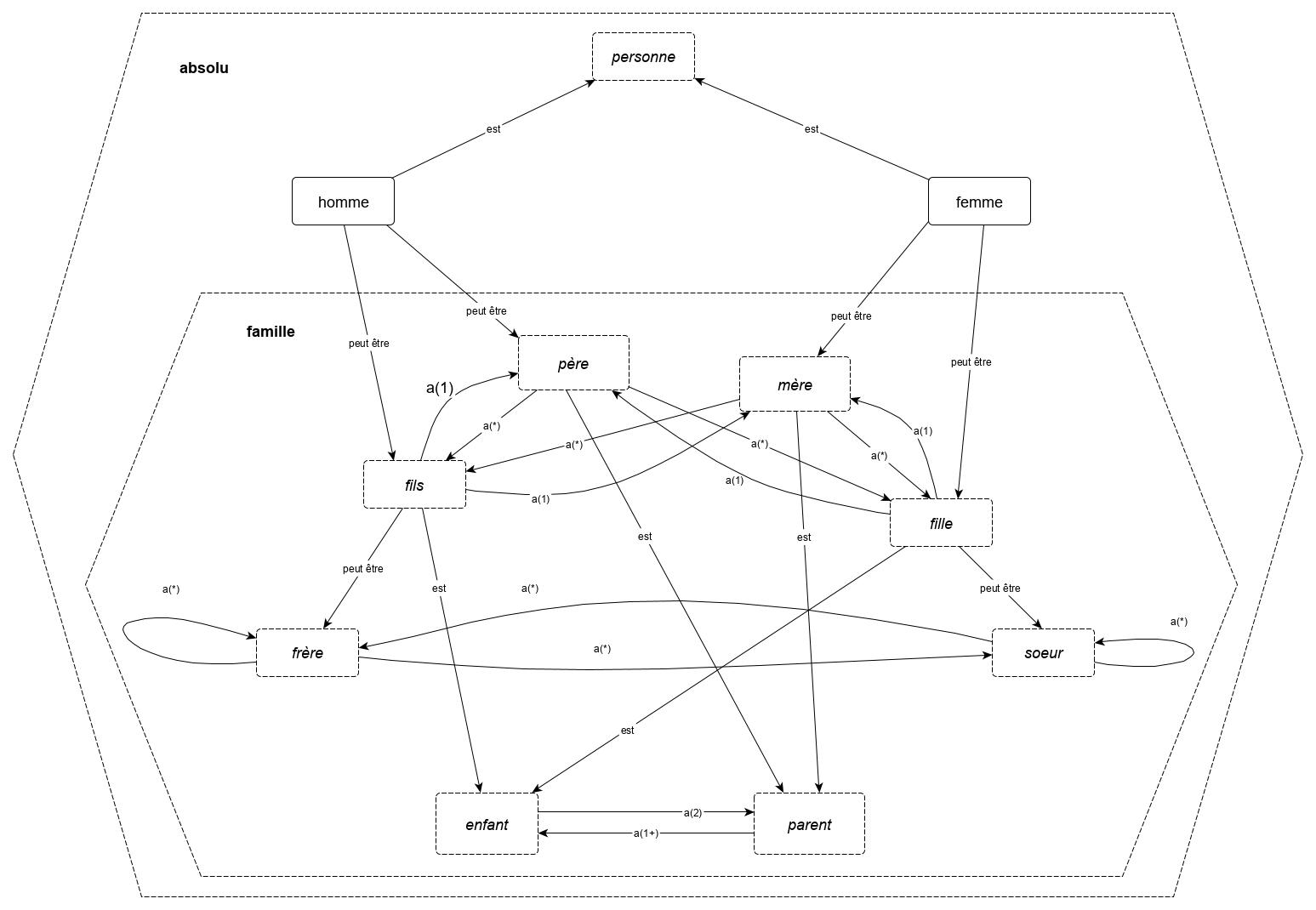
Les manières d'être possibles pour homme et femme apparaissent dans les fenêtres suivantes :





### Exemple 13 Sur la subsomption des manières d'être

Considérons le graphe suivant :



La représentation dans ICEO est la suivante :

iceo definition: #personne

iceo definition: #femme genus: personne

iceo definition: #homme genus: personne

iceo definitionSituation: 'famille'

iceo definitionQualite: #parent situation: (absolu getSituation: #famille)

iceo definitionQualite: #enfant situation: (absolu getSituation: #famille)

iceo definitionQualite: #soeur situation: (absolu getSituation: #famille)

iceo definitionQualite: #frère situation: (absolu getSituation: #famille)

iceo definitionQualite: #fils situation: (absolu getSituation: #famille) genus: ((absolu getSituation: #famille) get: #enfant)

iceo definitionQualite: #père situation: (absolu getSituation: #famille) genus: ((absolu getSituation: #famille) get: #parent)

iceo definitionQualite: #fille situation: (absolu getSituation: #famille) genus: ((absolu getSituation: #famille) get: #enfant)

iceo definitionQualite: #mère situation: (absolu getSituation: #famille) genus: ((absolu getSituation: #famille) get: #parent)

((absolu getSituation: #famille) get: #parent) associationQualite: ((absolu getSituation: #famille) get: #enfant).

((absolu getSituation: #famille) get: #père) associationQualite: ((absolu getSituation: #famille) get: #fils).

((absolu getSituation: #famille) get: #père) associationQualite: ((absolu getSituation: #famille) get: #fille).

((absolu getSituation: #famille) get: #mère) associationQualite: ((absolu getSituation: #famille) get: #fils).

((absolu getSituation: #famille) get: #mère) associationQualite: ((absolu getSituation: #famille) get: #fille).

((absolu getSituation: #famille) get: #frère) associationQualite: ((absolu getSituation: #famille) get: #soeur).

((absolu getSituation: #famille) get: #frère) associationQualite: ((absolu getSituation: #famille) get: #frère).

((absolu getSituation: #famille) get: #soeur) associationQualite: ((absolu getSituation: #famille) get: #soeur).

homme peutEtre: ((absolu getSituation: #famille) get: #père).

homme peutEtre: ((absolu getSituation: #famille) get: #fils).

femme peutEtre: ((absolu getSituation: #famille) get: #mère).

femme peutEtre: ((absolu getSituation: #famille) get: #fille).

((absolu getSituation: #famille) get: #fille) peutEtre: ((absolu getSituation: #famille) get: #soeur).

((absolu getSituation: #famille) get: #fils) peutEtre: ((absolu getSituation: #famille) get: #frère).

Dans cet exemple, certaines manières d'être en subsument d'autres.

Ainsi " ((absolu getSituation: #famille) get: #fils) getGenus " donne : enfant

Notons également qu'une manière d'être peut adopter d'autres manières d'être.

Ainsi " ((absolu getSituation: #famille) get: #fils) getQualites " donne : an OrderedCollection(frère)

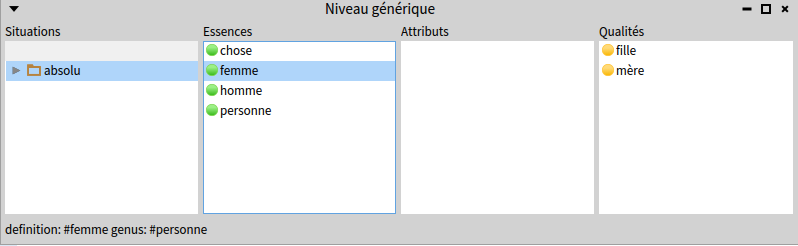
Et, comme nous l'avons déjà vu, une manière d'être peut avoir pour attributs d'autres manières d'être.

Ainsi, " ((absolu getSituation: #famille) get: #frère) getDifferentia " donne : an OrderedCollection(soeur frère)

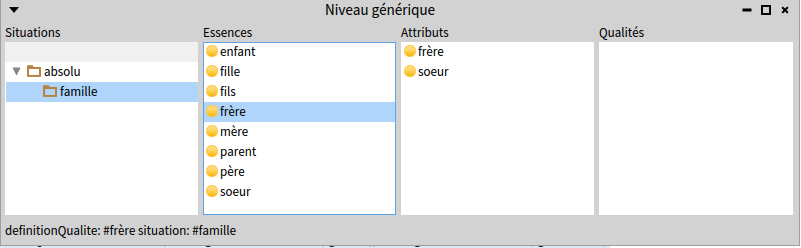
Cet exemple illustre le fait qu'une manière d'être peut être attribut d'elle même.

(Rappelons qu'une essence qui n'est pas une manière d'être ne peut être attribut d'elle même).

Un browser ouvert au niveau générique permet de vérifier par exemple les manières d'être possibles d'une femme :

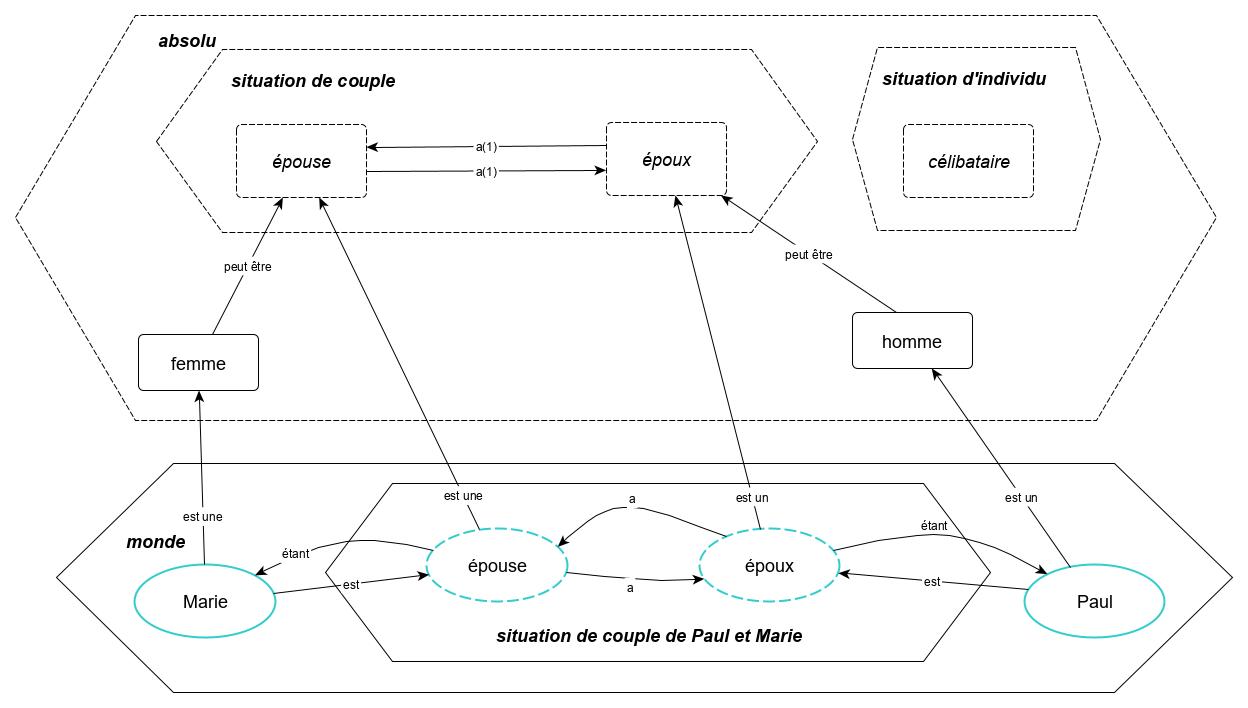


ou les attributs possibles de la manière d'être frère :



### Exemple 14 Sur la relation entre états

Considérons le graphe suivant :



iceo definition: #femme

iceo definition: #homme

iceo definitionSituation: 'situation de couple'

iceo definitionQualite: #épouse situation: (absolu getSituation: 'situation de couple')

iceo definitionQualite: #époux situation: (absolu getSituation: 'situation de couple')

((absolu getSituation: 'situation de couple') get: #épouse)

associationQualite: ((absolu getSituation: 'situation de couple') get: #époux).

femme peutEtre: ((absolu getSituation: 'situation de couple') get: #épouse).

homme peutEtre: ((absolu getSituation: 'situation de couple') get: #époux).

iceo soit: #Marie essence: femme.

iceo soit: #Paul essence: homme.

iceo soit: 'situation de couple de Paul et Marie' situationGenerique: (absolu getSituation: 'situation de couple').

(monde get: #Marie) addEtat: ((absolu getSituation: 'situation de couple') get: #épouse)

dansSituation: (monde getSituation: 'situation de couple de Paul et Marie').

(monde get: #Paul) addEtat: ((absolu getSituation: 'situation de couple') get: #époux)

dansSituation: (monde getSituation: 'situation de couple de Paul et Marie').

(((monde get: #Paul) getEtat: #époux dansSituation: (monde getSituation: 'situation de couple de Paul et Marie'))

associationEtat: ((monde get: #Marie) getEtat: #épouse

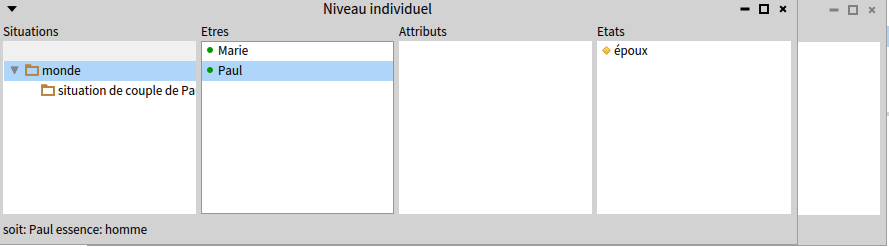
dansSituation: (monde getSituation: 'situation de couple de Paul et Marie'))).

Dans cet exemple, Marie et Paul ne sont pas attributs l'un de l'autre, mais l'expression

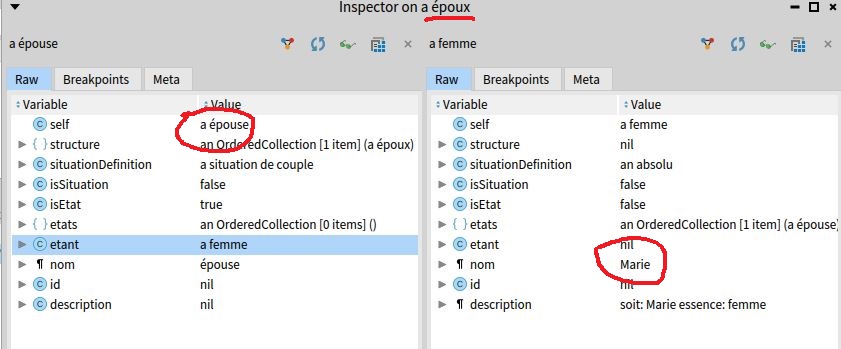
" (monde get: #Paul) getEtresAttributsEnTantQue: ((absolu getSituation: 'situation de couple') get: #époux) "

donne pourtant : an OrderedCollection(a femme).

Cet attribut lui est conféré par son état d'époux de Marie (attribut qu'il perdrait en cas de divorce ...)

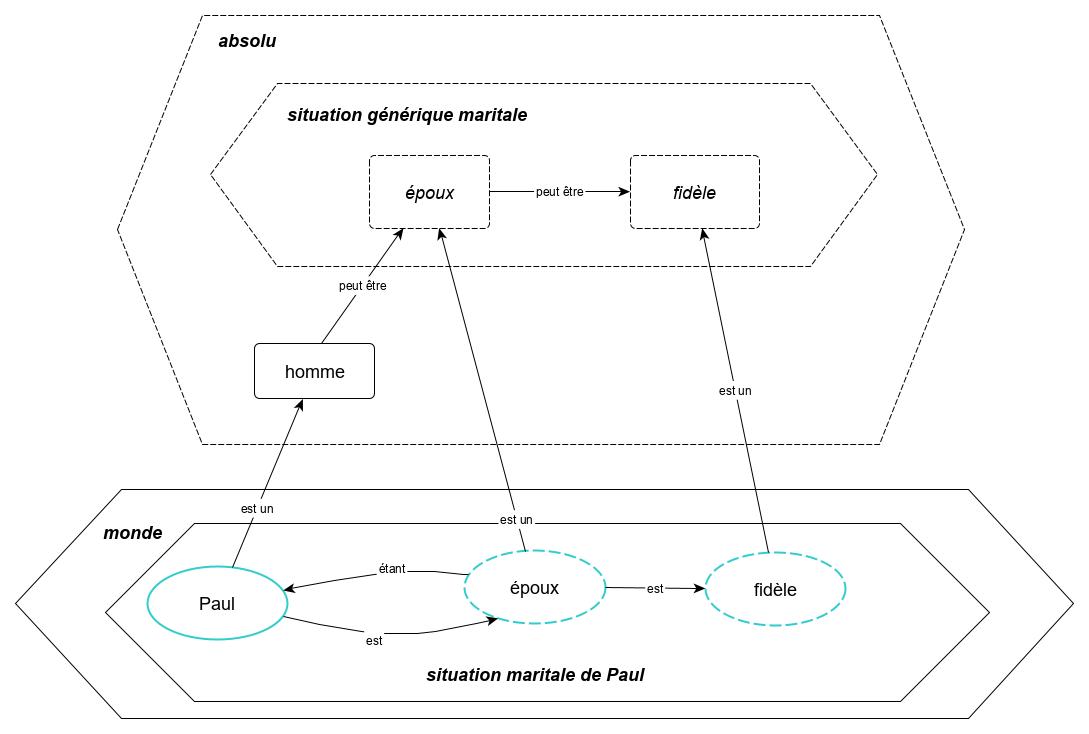


L'inspection de "époux" donne :



### Exemple 15 Sur la notion d'état dans un état

Considérons le graphe suivant :



iceo definition: #homme

iceo definitionSituation: 'situation générique maritale'

iceo definitionQualite: #époux situation:(absolu getSituation: 'situation générique maritale')

iceo definitionQualite: #fidèle situation: (absolu getSituation: 'situation générique maritale')

homme peutEtre: ((absolu getSituation: 'situation générique maritale') get: #époux).

iceo soit: 'situation maritale de Paul' situationGenerique: (absolu getSituation: 'situation générique maritale')

iceo soit: #Paul essence: homme

(monde get: #Paul) addEtat: ((absolu getSituation: 'situation générique maritale') get: #époux)

dansSituation: (monde get: 'situation maritale de Paul')

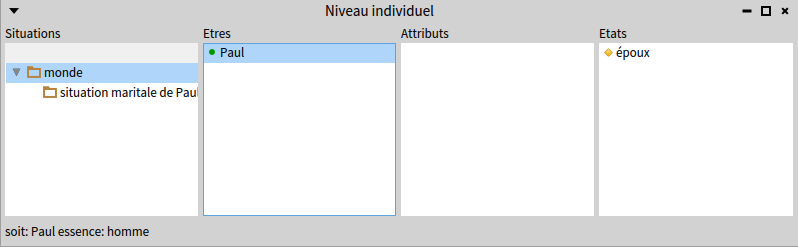
((monde get: #Paul) getEtat: #époux dansSituation: (monde get: 'situation maritale de Paul'))

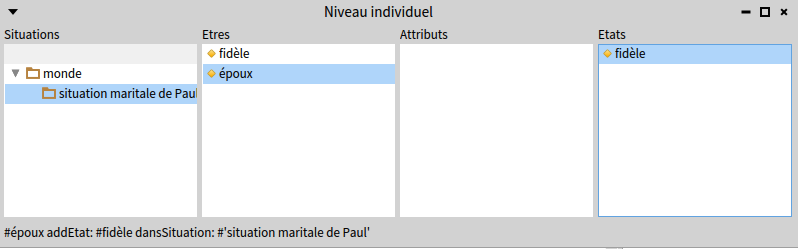
addEtat: ((absolu getSituation: 'situation générique maritale') get: #fidèle)

dansSituation: (monde get: 'situation maritale de Paul')

L'expression " ((monde get: #Paul) getEtat: #époux dansSituation: (monde get: 'situation maritale de Paul')) getEtats " donne : an OrderedCollection(a fidèle)

Ceci est visualisé dans les fenêtres suivantes :

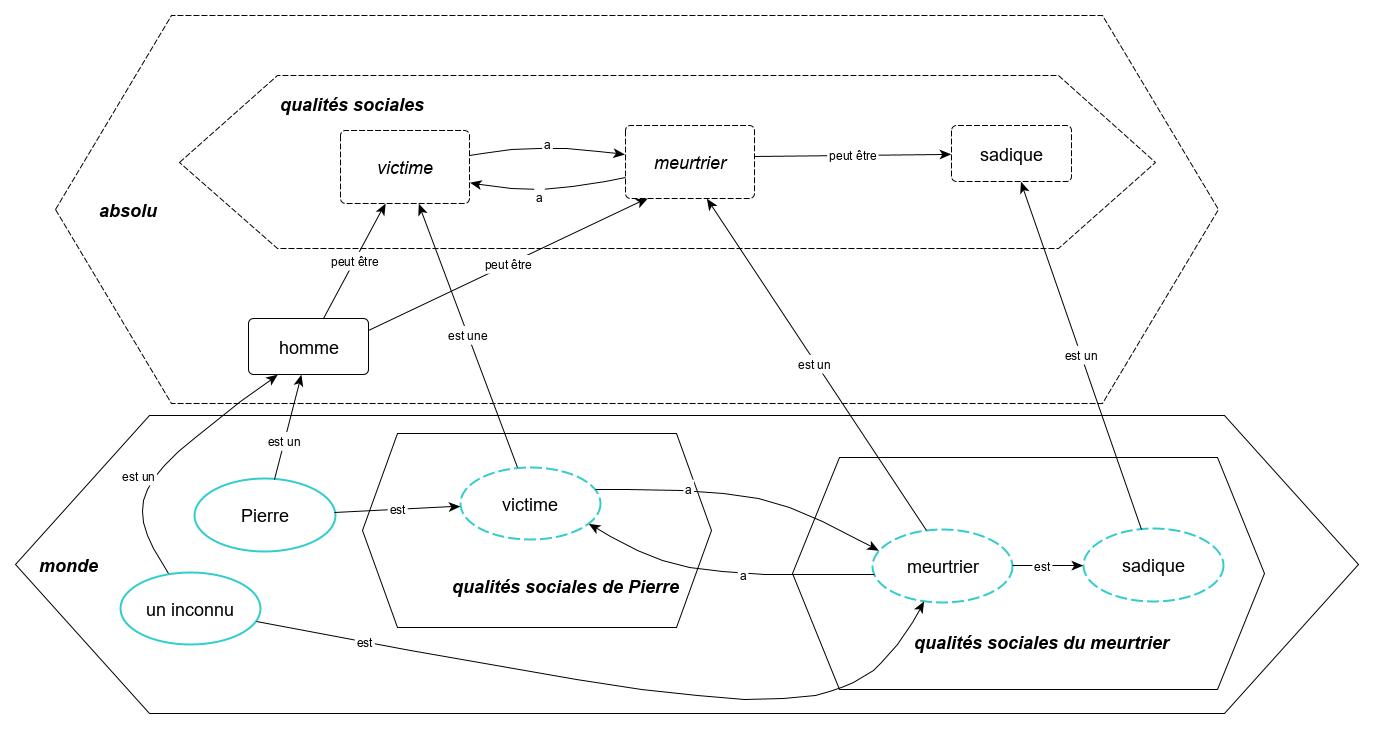




Par contre, L'expression " ((monde get: #Paul) getEtat: #fidèle dansSituation: (monde get: 'situation maritale de Paul')) " ne donne rien, car si Paul est fidèle en tant qu'époux, il ne l'est pas forcément dans toutes les situations !

### Exemple 16 Sur la notion d'être inconnu

Considérons le graphe suivant :



Ce graphe correspond à la représentation de la phrase : "Le meurtrier de Pierre est sadique ".

Cette affirmation n'implique pas la connaissance de l'identité du meurtrier par celui qui la prononce. Nous admettrons que cette phrase a un sens, bien que le meurtrier de Pierre puisse être inconnu; elle peut être formulée en voyant simplement l'état épouvantable de la victime Pierre.

Sa représentation dans ICEo est la suivante :

iceo definition: #personne

iceo definition: #homme genus: personne

iceo definitionSituation: 'qualités sociales'

iceo definitionQualite: #meurtrier situation: (absolu getSituation: #'qualités sociales')

iceo definitionQualite: #victime situation: (absolu getSituation: #'qualités sociales')

iceo definitionQualite: #sadique situation: (absolu getSituation: #'qualités sociales')

homme peutEtre: ((absolu getSituation: #'qualités sociales') get: #meurtrier).

homme peutEtre: ((absolu getSituation: #'qualités sociales') get: #victime).

((absolu getSituation: #'qualités sociales') get: #victime)

associationQualite: ((absolu getSituation: #'qualités sociales') get: #meurtrier).

((absolu getSituation: #'qualités sociales') get: #meurtrier) peutEtre: ((absolu getSituation: #'qualités sociales') get: #sadique).

iceo soit: #Pierre essence: homme.

iceo soit: 'qualités sociales de Pierre' situationGenerique: (absolu getSituation: #'qualités sociales').

(monde get: #Pierre) addEtat: ((absolu getSituation: #'qualités sociales') get: #victime)

dansSituation: (monde get: #'qualités sociales de Pierre' ).

iceo soit: 'qualités sociales du meurtrier' situationGenerique: (absolu getSituation: #'qualités sociales').

" Création d'un état dont l'étant est inconnu"

iceo soitEtat: #meurtrier essence: ((absolu getSituation: #'qualités sociales') get: #meurtrier)

situationIndividuelle: (monde get: #'qualités sociales du meurtrier' ).

((monde get: #'qualités sociales du meurtrier' ) get: #meurtrier)

addEtat: ((absolu getSituation: #'qualités sociales') get: #sadique)

dansSituation: (monde get: #'qualités sociales du meurtrier' ).

((monde get: #'qualités sociales du meurtrier' ) get: #meurtrier)

associationEtat: ((monde get: #'qualités sociales de Pierre' ) get: #victime).

L'expression " ((monde get: #'qualités sociales du meurtrier' ) get: #meurtrier) getEtats "

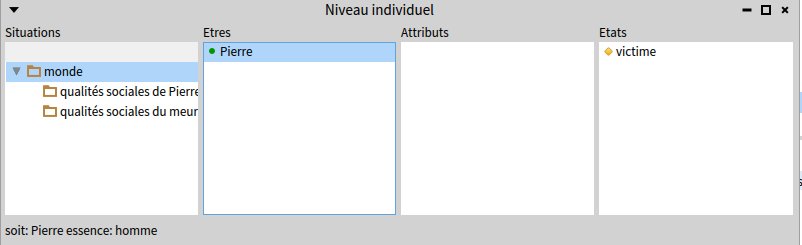
donne : an OrderedCollection(a sadique)

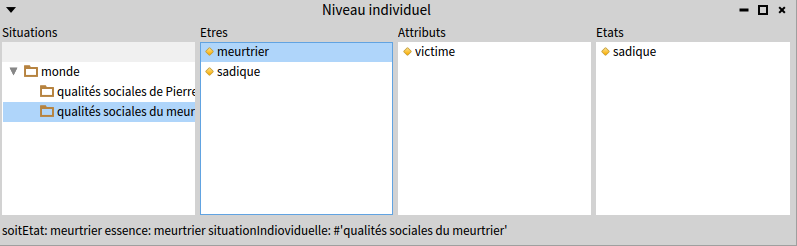
L'expression " (monde get: #Pierre) getEtats " donne : an OrderedCollection(a victime)

Enfin, l'expression " ((monde get: #'qualités sociales du meurtrier' ) get: #meurtrier) getEtant getNom "

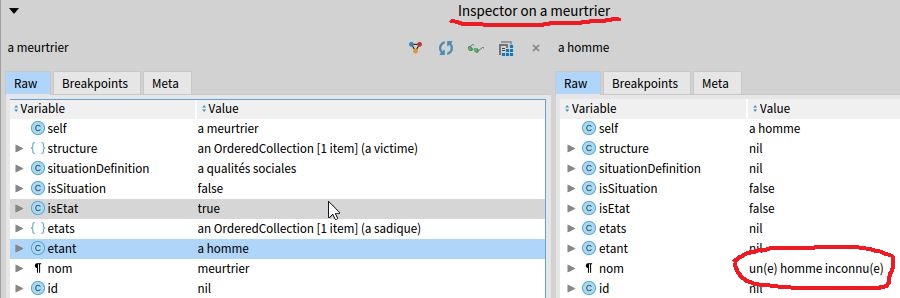
donne : #'Homme inconnu(e)'

Ceci est visualisé dans les fenêtres suivantes :





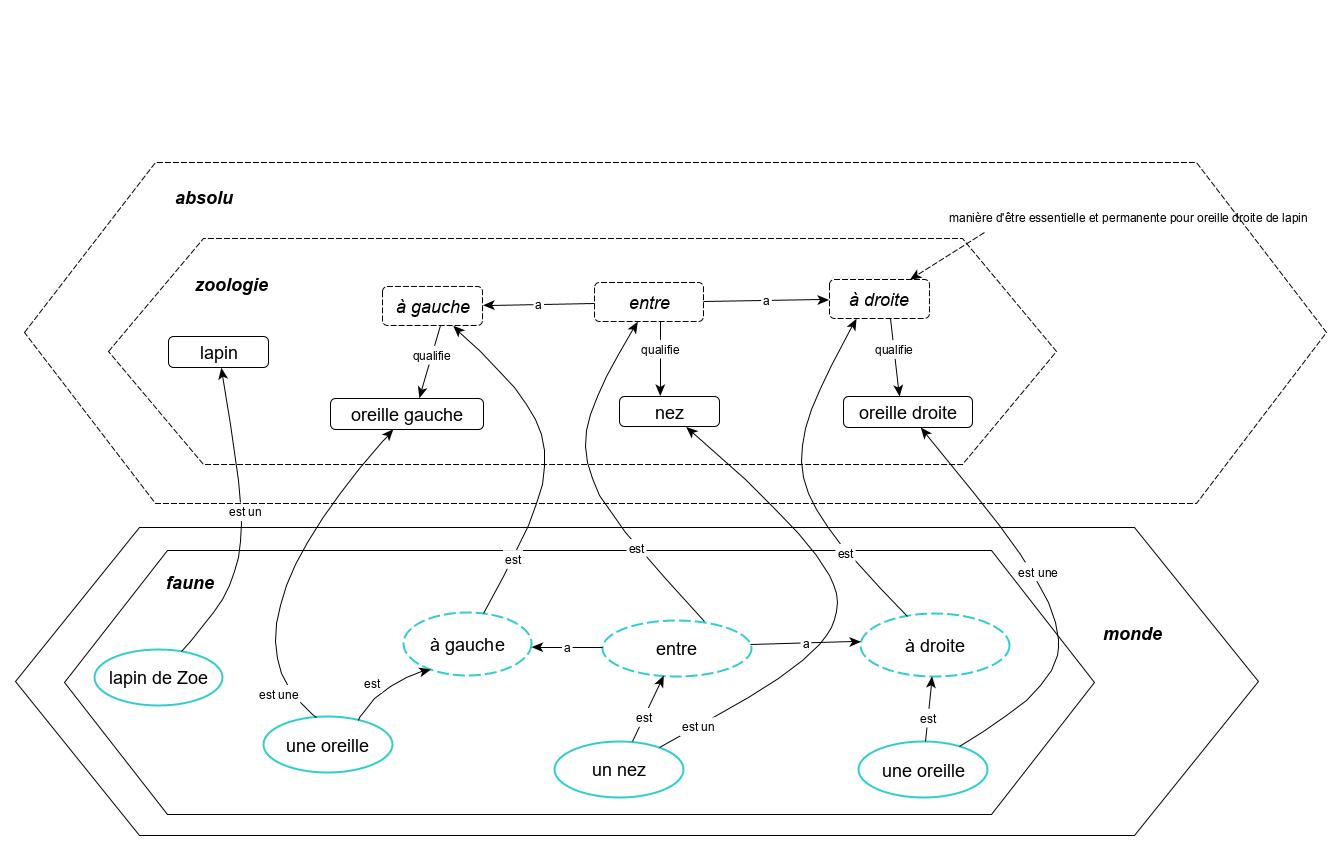
L'inspection du meurtrier de Pierre montre qu'il s'agit d'un homme inconnu :



Si plusieurs essences avaient la qualité de meurtrier, c'est leur premier genus commun qui aurait été choisi comme essence du meurtrier. Si ce genus était l'essence chose, le meurtrier aurait été simplement identifié comme étant "quelque chose".

### Exemple 17 Sur la notion de contrainte structurelle interne

Le graphe suivant exprime que le nez d'un lapin doit se situer entre ses deux oreilles :



(Pour ne pas complexifier l'aspect visuel de ce diagramme, le fait que le nez et les oreilles sont des attributs du lapin n'a pas été représenté).

Ce graphe exprime que les manières d'être "à gauche", "entre" et "à droite" sont essentielles et permanentes pour les essences oreille gauche, nez et oreille droite de lapin.

La représentation dans ICEO est la suivante :

iceo definitionSituation: 'zoologie'

iceo definition: #lapin situation: zoologie

iceo definitionAttribut: #nez de: (zoologie get: #lapin) cardinalite: 1

iceo definitionAttribut: 'oreille gauche' de: (zoologie get: #lapin) cardinalite: 1

iceo definitionAttribut: 'oreille droite' de: (zoologie get: #lapin) cardinalite: 1

iceo definitionQualiteEssentielle: 'entre' genus: ((absolu getSituation: 'contrainte de position relative') get: 'entre')

pour: ((zoologie get: #lapin) getEssenceAttribut: #nez) effectivite: #permanente

iceo definitionQualiteEssentielle: 'à gauche' genus: ((absolu getSituation: 'contrainte de position relative') get: 'à gauche')

pour: ((zoologie get: #lapin) getEssenceAttribut: 'oreille gauche') effectivite: #permanente

iceo definitionQualiteEssentielle: 'à droite' genus: ((absolu getSituation: 'contrainte de position relative') get: 'à droite')

pour: ((zoologie get: #lapin) getEssenceAttribut: 'oreille droite') effectivite: #permanente

(((zoologie get: #lapin) getEssenceAttribut: #nez) getQualite: 'entre')

associationQualite: (((zoologie get: #lapin) getEssenceAttribut: 'oreille gauche') getQualite: 'à gauche').

(((zoologie get: #lapin) getEssenceAttribut: #nez) getQualite: 'entre')

associationQualite: (((zoologie get: #lapin) getEssenceAttribut: 'oreille droite') getQualite: 'à droite').

iceo soit: #faune situationGenerique: zoologie

iceo soit: 'Lapin de Zoé' essence: (zoologie get: #lapin) situationIndividuelle: (monde get: #faune)

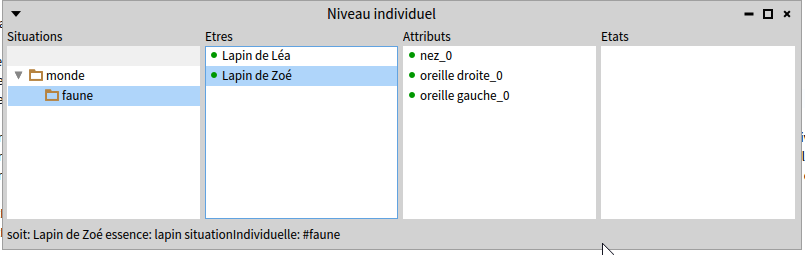
(((monde get: 'Lapin de Zoé') getEtreAttribut: #nez) getEtat: #entre)

associationEtat: (((monde get: 'Lapin de Zoé') getEtreAttribut: 'oreille gauche') getEtat: #'à gauche')

(((monde get: 'Lapin de Zoé') getEtreAttribut: #nez) getEtat: #entre)

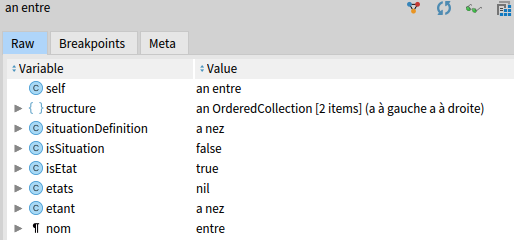
associationEtat: (((monde get: 'Lapin de Zoé') getEtreAttribut: 'oreille droite') getEtat: #'à droite')

Voici une visualisation du lapin de Zoé :

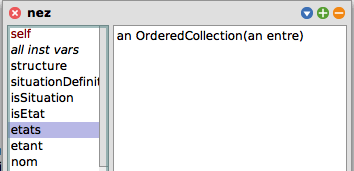


Le nez du lapin de Zoé à été automatiquement instancié, comme ses oreilles, car leur cardinalité au niveau de l'essence lapin est de 1.

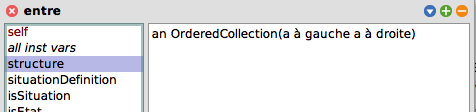
L'inspection de " ((monde get: 'Lapin de Zoé') getEtreAttribut: #nez) getEtats at: 1 " donne :



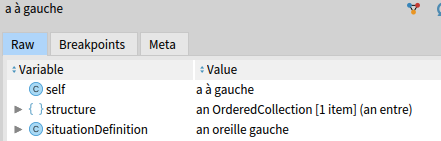
Faisons l'inspection du nez :



L'inspection de l'état "entre" donne :



L'inspection de l'état "à gauche" donne :



On constate qu'il s'agit bien de l'état de l'oreille gauche du lapin de Zoé.

Considérons maintenant un lapin nommé Léa :

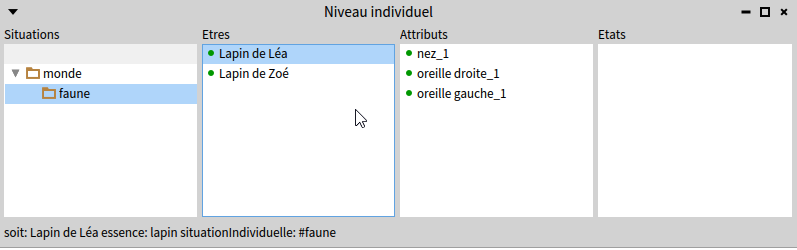
iceo soit: 'Lapin de Léa' essence: (zoologie get: #lapin) situationIndividuelle: (monde get: #faune)

(((monde get: 'Lapin de Léa') getEtreAttribut: #nez) getEtat: #entre)

associationEtat: (((monde get: 'Lapin de Léa') getEtreAttribut: 'oreille gauche') getEtat: #'à gauche')

(((monde get: 'Lapin de Léa') getEtreAttribut: #nez) getEtat: #entre)

associationEtat: (((monde get: 'Lapin de Léa') getEtreAttribut: 'oreille droite') getEtat: #'à droite')



On constate que les oreilles des deux lapins sont différentes, ce qui est logique.

Mais qu'en est-il de leurs états ? Faisons un test en évaluant l'expression

"(((monde get: 'Lapin de Zoé') getEtreAttribut: #'oreille gauche') getEtats at: 1) == (((monde get: 'Lapin de Léa') getEtreAttribut: #'oreille gauche') getEtats at: 1)"

Le résultat est : false

Ce résultat est tout-à-fait correct dans la logique d'ICEO.

En effet, il est important de distinguer l'état "à gauche" de l'oreille gauche du lapin de Léa de l'état "à gauche" de l'oreille gauche du lapin de Zoé.

Sachant que, comme nous l'avons vu, un état peut lui même avoir des états, il serait possible par exemple que l'un soit "un peu plus à gauche" que l'autre.

### Exemple 18 Sur la notion de famille

Cet exemple repart de l'exemple 13 (évaluer cet exemple "en bloc") sur la subsomption des manières d'être pour représenter le famille Gonthier où les parents Charles et Hermeline possèdent deux filles Capucine et Juliette et deux fils Martin et Jean :

iceo soit: #Charles essence: homme.

iceo soit: #Hermeline essence: femme.

iceo soit: #Martin essence: homme.

iceo soit: #Jean essence: homme.

iceo soit: #Capucine essence: femme.

iceo soit: #Juliette essence: femme.

(monde get: #Charles) addEtat: ((absolu getSituation: 'famille') get: #père) dansSituation: (monde get: #Gonthier).

(monde get: #Hermeline) addEtat: ((absolu getSituation: 'famille') get: #mère) dansSituation: (monde get: #Gonthier).

(monde get: #Martin) addEtat: ((absolu getSituation: 'famille') get: #fils) dansSituation: (monde get: #Gonthier).

((monde get: #Martin) getEtat: #fils dansSituation: (monde get: #Gonthier))

addEtat: ((absolu getSituation: #famille) get: #frère) dansSituation: (monde get: #Gonthier).

(monde get: #Jean) addEtat: ((absolu getSituation: 'famille') get: #fils) dansSituation: (monde get: #Gonthier).

((monde get: #Jean) getEtat: #fils dansSituation: (monde get: #Gonthier))

addEtat: ((absolu getSituation: #famille) get: #frère) dansSituation: (monde get: #Gonthier).

(monde get: #Capucine) addEtat: ((absolu getSituation: 'famille') get: #fille) dansSituation: (monde get: #Gonthier).

((monde get: #Capucine) getEtat: #fille dansSituation: (monde get: #Gonthier))

addEtat: ((absolu getSituation: #famille) get: #soeur) dansSituation: (monde get: #Gonthier).

(monde get: #Juliette) addEtat: ((absolu getSituation: 'famille') get: #fille) dansSituation: (monde get: #Gonthier).

((monde get: #Juliette) getEtat: #fille dansSituation: (monde get: #Gonthier))

addEtat: ((absolu getSituation: #famille) get: #soeur) dansSituation: (monde get: #Gonthier).

((monde get: #Charles) getEtat: #père dansSituation: (monde get: #Gonthier))

associationEtat: ((monde get: #Capucine) getEtat: #fille dansSituation: (monde get: #Gonthier)).

((monde get: #Charles) getEtat: #père dansSituation: (monde get: #Gonthier))

associationEtat: ((monde get: #Martin) getEtat: #fils dansSituation: (monde get: #Gonthier)). ((monde get: #Charles) getEtat: #père dansSituation: (monde get: #Gonthier))

associationEtat: ((monde get: #Jean) getEtat: #fils dansSituation: (monde get: #Gonthier)).

((monde get: #Charles) getEtat: #père dansSituation: (monde get: #Gonthier))

associationEtat: ((monde get: #Juliette) getEtat: #fille dansSituation: (monde get: #Gonthier)) .

((monde get: #Hermeline) getEtat: #mère dansSituation: (monde get: #Gonthier))

associationEtat: ((monde get: #Capucine) getEtat: #fille dansSituation: (monde get: #Gonthier)).

((monde get: #Hermeline) getEtat: #mère dansSituation: (monde get: #Gonthier))

associationEtat: ((monde get: #Martin) getEtat: #fils dansSituation: (monde get: #Gonthier)).

((monde get: #Hermeline) getEtat: #mère dansSituation: (monde get: #Gonthier))

associationEtat: ((monde get: #Jean) getEtat: #fils dansSituation: (monde get: #Gonthier)).

((monde get: #Hermeline) getEtat: #mère dansSituation: (monde get: #Gonthier))

associationEtat: ((monde get: #Juliette) getEtat: #fille dansSituation: (monde get: #Gonthier)) .

(((monde get: #Capucine) getEtat: #fille dansSituation: (monde get: #Gonthier)) getEtat: #soeur

dansSituation: (monde get: #Gonthier)) associationEtat: (((monde get: #Martin) getEtat: #fils

dansSituation: (monde get: #Gonthier)) getEtat: #frère dansSituation: (monde get: #Gonthier)) .

(((monde get: #Capucine) getEtat: #fille dansSituation: (monde get: #Gonthier)) getEtat: #soeur

dansSituation: (monde get: #Gonthier)) associationEtat: (((monde get: #Jean) getEtat: #fils

dansSituation: (monde get: #Gonthier)) getEtat: #frère dansSituation: (monde get: #Gonthier)). .

(((monde get: #Capucine) getEtat: #fille dansSituation: (monde get: #Gonthier)) getEtat: #soeur

dansSituation: (monde get: #Gonthier)) associationEtat: (((monde get: #Juliette) getEtat: #fille

dansSituation: (monde get: #Gonthier)) getEtat: #soeur dansSituation: (monde get: #Gonthier)).

(((monde get: #Juliette) getEtat: #fille dansSituation: (monde get: #Gonthier)) getEtat: #soeur

dansSituation: (monde get: #Gonthier)) associationEtat: (((monde get: #Jean) getEtat: #fils

dansSituation: (monde get: #Gonthier)) getEtat: #frère dansSituation: (monde get: #Gonthier)).

(((monde get: #Juliette) getEtat: #fille dansSituation: (monde get: #Gonthier)) getEtat: #soeur

dansSituation: (monde get: #Gonthier)) ,associationEtat: (((monde get: #Martin) getEtat: #fils

dansSituation: (monde get: #Gonthier)) getEtat: #frère dansSituation: (monde get: #Gonthier)).

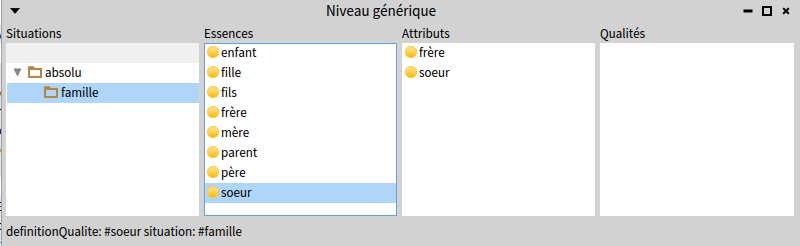
(((monde get: #Martin) getEtat: #fils dansSituation: (monde get: #Gonthier)) getEtat: #frère

dansSituation: (monde get: #Gonthier)) associationEtat: (((monde get: #Jean) getEtat: #fils

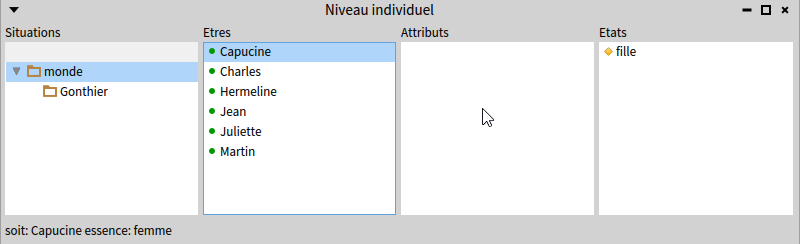
dansSituation: (monde get: #Gonthier)) getEtat: #frère dansSituation: (monde get: #Gonthier)).

Au niveau générique, rien n'a changé par rapport à l'exemple 13.

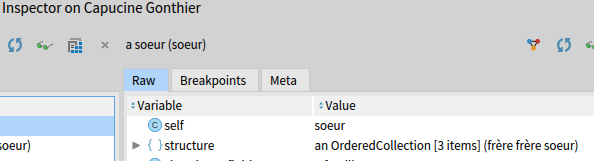
Par exemple :



Au niveau individuel, Capucine est une fille :



En tant que fille, elle est soeur de Juliette, Jean et Martin :



La méthode "printOn: aStream" de la classe Object a été surchargée dans cet exemple au niveau de l'essence chose pour rendre les noms des membres d'une famille plus lisibles :

chose compile: ' printOn: aStream

self isEtat

ifTrue: [self getEtats

detect: [:e | e getSituationDefinition class getNom == #famille]

ifFound: [:x | aStream nextPutAll: self getEtant getNom , '' '' , x getSituationDefinition getNom]

ifNone: [aStream nextPutAll: self getNom]]

ifFalse: [ self getEtats

detect: [:e | e getSituationDefinition class getNom == #famille]

ifFound: [:x | aStream nextPutAll: self getNom , '' '' , x getSituationDefinition getNom]

ifNone: [aStream nextPutAll: self getNom]]'

Ainsi, voici les réponses données par diverses expressions :

" ((monde get: #Charles) getEtresAttributsQuiSont: ((absolu getSituation: #famille) get: #enfant)) "

donne : an OrderedCollection(Capucine Gonthier Martin Gonthier Jean Gonthier Juliette Gonthier)

" ((monde get: #Charles) getEtresAttributsQuiSont: ((absolu getSituation: #famille) get: #fille)) "

donne : an OrderedCollection(Capucine Gonthier Juliette Gonthier)

" ((monde get: #Capucine) getEtresAttributsQuiSont: ((absolu getSituation: #famille) get: #frère)) "

donne : an OrderedCollection(Martin Gonthier Jean Gonthier)

" ((monde get: #Martin) getEtresAttributsQuiSont: ((absolu getSituation: #famille) get: #soeur)) "

donne : an OrderedCollection(Capucine Gonthier Juliette Gonthier)

" ((monde get: #Martin) getEtresAttributsQuiSont: ((absolu getSituation: #famille) get: #frère)) "

donne : an OrderedCollection(Jean Gonthier)

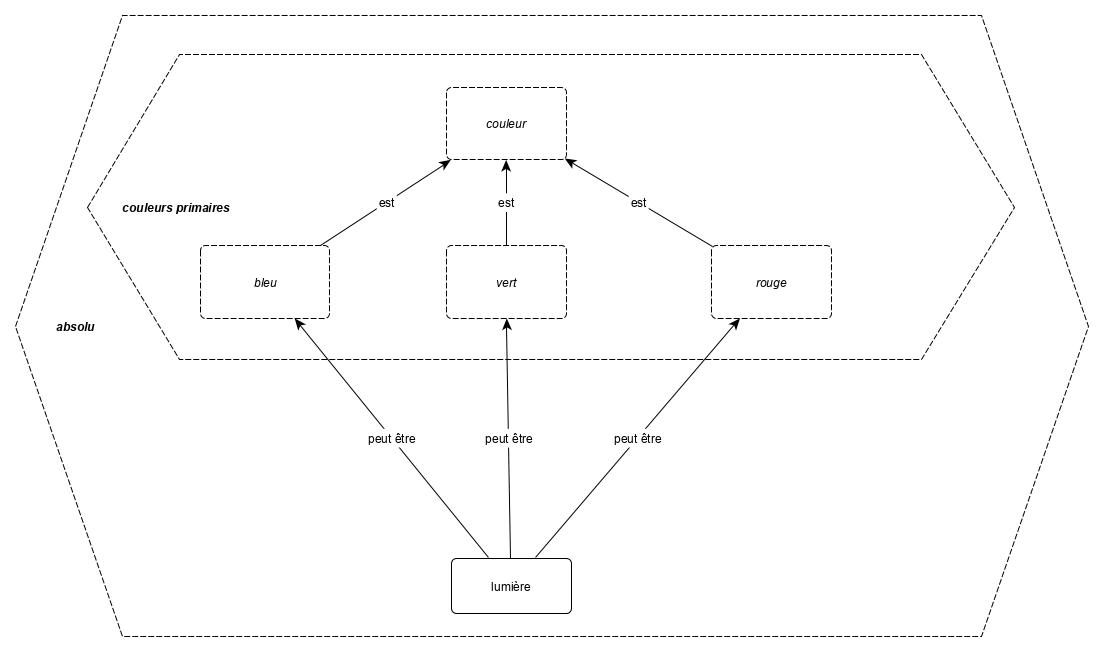
### Exemple 19 Sur la notion de couleur

En soi, la lumière se présente comme de l’eau : elle n’a pas de forme, sa substance est massière.

Une lumière peut avoir différentes qualités. Ainsi une lumière peut être chaude (quand sa couleur tire vers l'orange) ou froide (lorsque sa couleur tire vers le bleu), …, de la même manière que l'eau peut nous paraître chaude, tiède, froide, …

Une couleur est une qualité de la lumière perçue par un sujet.

Voici la représentation des trois couleurs dites "primaires" :



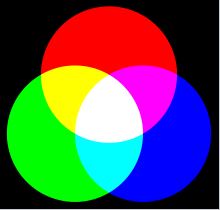
Une lumière perçue peut être bleue, rouge ou verte, c'est-à-dire avoir l'état bleu, rouge ou vert (ou encore avoir la couleur bleue, rouge ou verte).

Les couleurs primaires bleu, vert où rouge sont celles des trois (spectres de) lumières perçues et distinguées par l'œil humain.

Une lumière peut aussi être composée de lumières.

Les différentes compositions de lumières font apparaître d'autres couleurs possibles, reconnaissables par celui qui a appris à reconnaître ces compositions. Ainsi "blanc" est la couleur d'une lumière composée à parts égales de lumière rouge, verte et bleue.

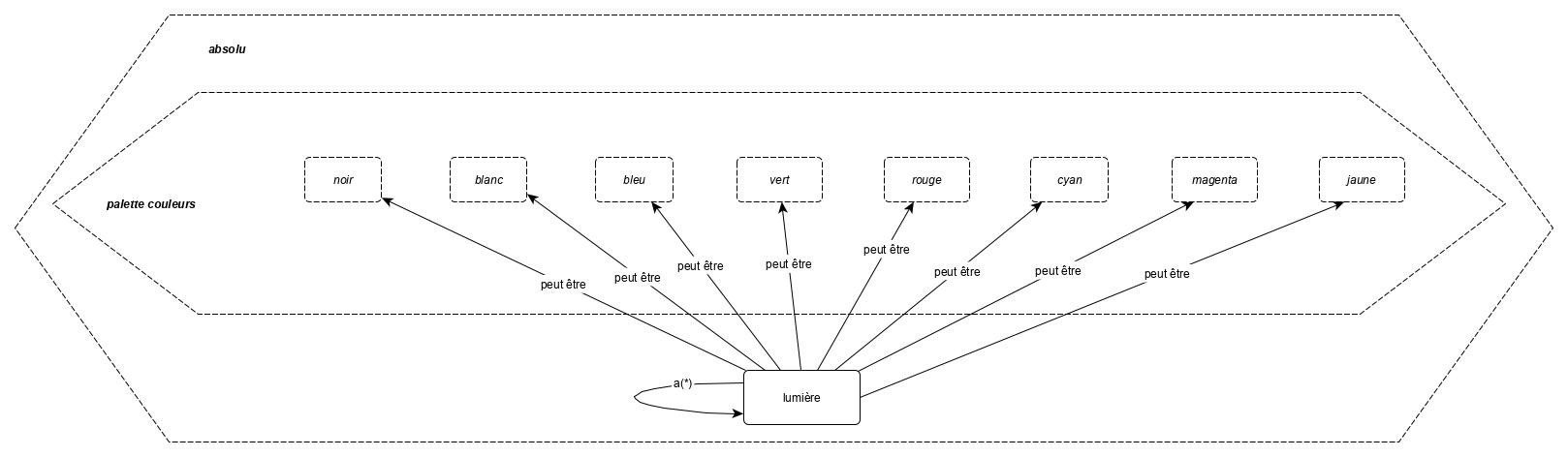
Les couleurs des lumières composées à parts égales de deux lumières ayant une couleur primaire sont appelées secondaires. Ce sont les couleurs cyan, magenta et jaune.



Les couleurs tertiaires viennent de la composition à parts égales d'une lumière ayant une couleur secondaire et de l’une des deux lumières de couleur primaire de sa couleur secondaire. Ainsi la couleur vermillon est composée à partir du jaune et du rouge.

noir est la couleur d'une lumière dont la composition est vide.

Ceci enrichit notre palette de couleurs :



L'expression de ce graphe dans le langage ICEO est la suivante :

iceo definitionQualite: #couleur situation: (absolu getSituation: #colorée)

iceo definitionSituation: 'palette de couleurs'

iceo definitionQualite: #bleu situation: (absolu getSituation: 'palette de couleurs')

genus: ((absolu getSituation: #colorée) get: #couleur)

iceo definitionQualite: #vert situation: (absolu getSituation: 'palette de couleurs')

genus: ((absolu getSituation: #colorée) get: #couleur)

iceo definitionQualite: #rouge situation: (absolu getSituation: 'palette de couleurs')

genus: ((absolu getSituation: #colorée) get: #couleur)

iceo definitionQualite: #cyan situation: (absolu getSituation: 'palette de couleurs')

genus: ((absolu getSituation: #colorée) get: #couleur)

iceo definitionQualite: #magenta situation: (absolu getSituation: 'palette de couleurs')

genus: ((absolu getSituation: #colorée) get: #couleur)

iceo definitionQualite: #jaune situation: (absolu getSituation: 'palette de couleurs')

genus: ((absolu getSituation: #colorée) get: #couleur)

iceo definitionQualite: #blanc situation: (absolu getSituation: 'palette de couleurs')

genus: ((absolu getSituation: #colorée) get: #couleur)

iceo definitionQualite: #noir situation: (absolu getSituation: 'palette de couleurs')

genus: ((absolu getSituation: #colorée) get: #couleur)

La palette de couleurs ainsi définie est :

palette de couleurs : [magenta, jaune, noir, rouge, bleu, vert, blanc, cyan]

Voici comment peuvent se définir les différentes lumières avec les couleurs primaires et secondaires, le blanc et le noir :

iceo definition: #lumière.

lumière referenceEssence: lumière.

iceo soit: 'lumière bleu' essence: lumière.

iceo soit: 'lumière vert' essence: lumière.

iceo soit: 'lumière rouge' essence: lumière.

iceo soit: 'lumière cyan' essence: lumière.

iceo soit: 'lumière magenta' essence: lumière.

iceo soit: 'lumière jaune' essence: lumière.

iceo soit: 'lumière blanc' essence: lumière.

iceo soit: 'lumière noir' essence: lumière.

(monde get: 'lumière bleu') addEtat: ((absolu getSituation: 'palette de couleurs') getElement: #bleu) dansSituation: (monde get: 'une palette de couleurs').

(monde get: 'lumière bleu') referenceEtre: (monde get: 'lumière bleu').

(monde get: 'lumière vert') addEtat: ((absolu getSituation: 'palette de couleurs') getElement: #vert) dansSituation: (monde get: 'une palette de couleurs').

(monde get: 'lumière vert') referenceEtre: (monde get: 'lumière vert').

(monde get: 'lumière rouge') addEtat: ((absolu getSituation: 'palette de couleurs') getElement: #rouge) dansSituation: (monde get: 'une palette de couleurs').

(monde get: 'lumière rouge') referenceEtre: (monde get: 'lumière rouge').

(monde get: 'lumière cyan') addEtat: ((absolu getSituation: 'palette de couleurs') getElement: #cyan) dansSituation: (monde get: 'une palette de couleurs').

(monde get: 'lumière cyan') referenceEtre: (monde get: 'lumière bleu').

(monde get: 'lumière cyan') referenceEtre: (monde get: 'lumière vert').

(monde get: 'lumière magenta') addEtat: ((absolu getSituation: 'palette de couleurs') getElement: #magenta) dansSituation: (monde get: 'une palette de couleurs').

(monde get: 'lumière magenta') referenceEtre: (monde get: 'lumière bleu').

(monde get: 'lumière magenta') referenceEtre: (monde get: 'lumière rouge') .

(monde get: 'lumière jaune') addEtat: ((absolu getSituation: 'palette de couleurs') getElement: #jaune) dansSituation: (monde get: 'une palette de couleurs').

(monde get: 'lumière jaune') referenceEtre: (monde get: 'lumière rouge').

(monde get: 'lumière jaune') referenceEtre: (monde get: 'lumière vert').

(monde get: 'lumière blanc') addEtat: ((absolu getSituation: 'palette de couleurs') getElement: #blanc) dansSituation: (monde get: 'une palette de couleurs').

(monde get: 'lumière blanc') referenceEtre: (monde get: 'lumière bleu').

(monde get: 'lumière blanc') referenceEtre: (monde get: 'lumière rouge').

(monde get: 'lumière blanc') referenceEtre: (monde get: 'lumière vert').

(monde get: 'lumière noir') addEtat: ((absolu getSituation: 'palette de couleurs') getElement: #noir) dansSituation: (monde get: 'une palette de couleurs')

L'ensemble de lumières ainsi défini est :

[lumière jaune, lumière magenta, lumière blanc, lumière noir, lumière bleu, lumière rouge, lumière vert, lumière cyan]

Couleur des objets

La couleur d'un objet est celle de la lumière qu'il refléte ou diffuse lorsqu'il est éclairé.

un objet éclairé avec une lumière peut être coloré comme il peut être mouillé avec un liquide.

Les capacités de diffusion de la lumière d'un objet sont liées à sa composition physique; elles ne dépendent pas de l'éclairage mais déterminent sa couleur une fois éclairé. Elles sont bien sûr les mêmes en l'absence de lumière (dans le noir).

En peinture, la composition des couleurs est basée sur les capacités de diffusion de la lumière par la peinture. La composition est soustractive. Les trois couleurs fondamentales en peinture sont cyan, magenta et jaune. La peinture verte est obtenue par mélange de pigments jaune et cyan :



Par convention, la couleur d'un objet est celle qu'il possède éclairé par de la lumière blanche. C'est sa couleur prise conventionnellement par défaut, que nous qualifierons de couleur "objective". Avec cette convention, éclairé par de la lumière rouge, un objet jaune paraîtra rouge (car la couleur objective jaune signifie que l'objet diffuse les lumières rouge et vert).

Quel que soit l'éclairage, un objet noir apparaitra toujours noir, car il absorbe toutes les lumières.

Nous qualifierons de "couleur apparente" la couleur que prend un objet avec un éclairage particulier.

Autrement dit, c'est le couple (couleur objective, éclairage) qui détermine la couleur apparente de l'objet. Changer l'éclairage entraîne un changement de couleur apparente des objets éclairés. En l'absence d'éclairage, c'est-à-dire dans le noir, un objet parait noir.

L'éclairage est le contexte où la couleur d'un objet devient perceptible.

Voici la définition d'un objet de couleur jaune :

iceo definition: #objet

objet peutEtre: ((absolu getSituation: #colorée) get: #couleur)

iceo soit: 'un objet' essence: objet

(monde get: 'un objet')

addEtat: ((absolu getSituation: 'palette de couleurs') getElement: #**jaune**)

dansSituation: (monde get: 'une palette de couleurs')

Et voici la définition d'une torche émettant une lumière magenta :

iceo definition: #torche

iceo soit: 'une torche' essence: torche.

(monde get: 'une torche') referenceEtre: (monde get: 'lumière magenta')

Le codesuivant définit une méthode de l'essence objet qui infère la couleur apparente d'un objet éclairé par une torche, en faisant l'intersection de l'ensemble de lumières émises par la torche avec l'ensemble des lumières reflétées par l'objet :

objet compile: 'couleurApparenteEclairePar: uneTorche dans: uneSituation

^ ((uneSituation getElementsEssence: lumière)

detect: [:each | each getEtresAttributs

= ((uneSituation get: ''lumière '' , ((uneSituation get: ''un objet'') getEtats at: 1) getNom) getEtresAttributs

intersection: (uneTorche getEtresAttributs at: 1) getEtresAttributs)]) getEtats at: 1'

Ainsi par exemple, pour un objet de couleur jaune éclairé par de la lumière magenta, l'expression :

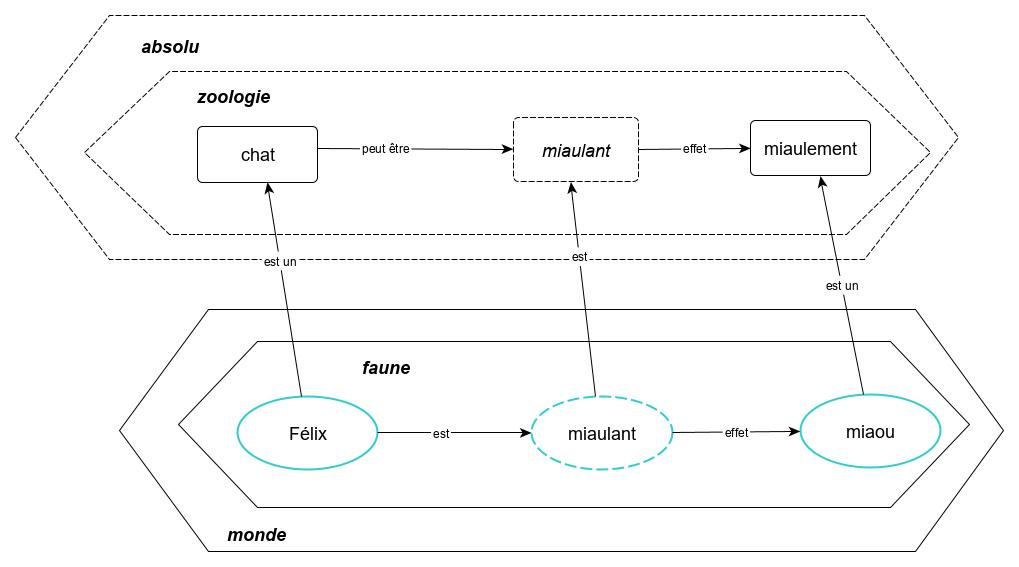
" (monde get: 'un objet') couleurApparenteEclairePar: (monde get: 'une torche') dans: monde " retourne : rouge

Pour un objet de couleur jaune éclairé par de la lumière rouge, la couleur apparente est la même.

Pour un objet de couleur jaune éclairé par de la lumière bleue, la couleur apparente est le noir, ...

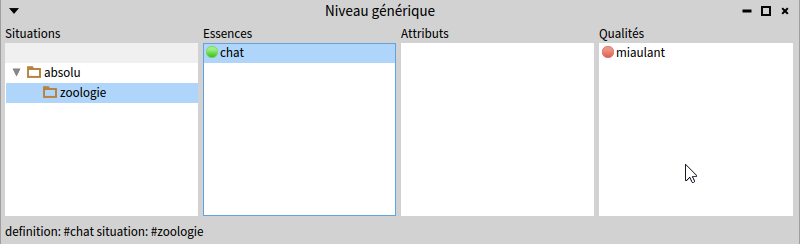
### Exemple 20 Sur la notion d'action

Considérons le graphe suivant :



iceo definitionSituation: #zoologie .iceo definition: #chat situation: zoologie.iceo definitionQualiteEssentielle: #miaulant pour: (zoologie get: #chat) effectivite: #intermittente.iceo definitionAttribut: #miaulement de: ((zoologie get: #chat) getQualite: #miaulant).

Dans cet exemple, la qualité "miaulant" de chat est essentielle (le chat peut miauler par essence)



mais son effectivité est intermittente (un chat ne miaule pas sans arrêt).

Il est donc nécessaire de préciser que Félix est ici en train de miauler.

iceo soit: #faune situationGenerique: zoologie

iceo soit: #Félix essence: (zoologie get: #chat)

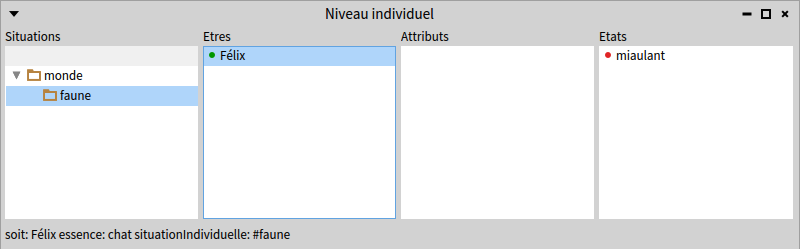
((monde get: #faune) get: #Félix) addEtatEssentiel:

((zoologie get: #chat) getQualite: #miaulant)

(((monde get: #faune) get: #Félix) getEtat: #miaulant) attributionEtre: #miaou

essence: ((((zoologie get: #chat) getQualite: #miaulant))

getEssenceAttribut: #miaulement)



Supposons que la méthode suivante soit associée à la manière d'être miaulant :

((zoologie get: #chat) getQualite: #miaulant) compile: 'miaule . ^(self getEtresAttributs asOrderedCollection at: 1) getNom'.

L'envoi du message "miaule" à l'état miaulant de Félix par :

(((monde get: #faune) get: #Félix) getEtat: #miaulant) miaule

donne : Miaou

Supposons que Maya soit une chatte dont le miaulement produit "meow" :

iceo soit: #Maya essence: (zoologie get: #chat)

((monde get: #faune) get: #Maya) addEtatEssentiel:

((zoologie get: #chat) getQualite: #miaulant)

(((monde get: #faune) get: #Maya) getEtat: #miaulant) attributionEtre: #meow

essence: ((((zoologie get: #chat) getQualite: #miaulant))

getEssenceAttribut: #miaulement)

L'envoi du message miaule à l'état miaulant de Maya par :

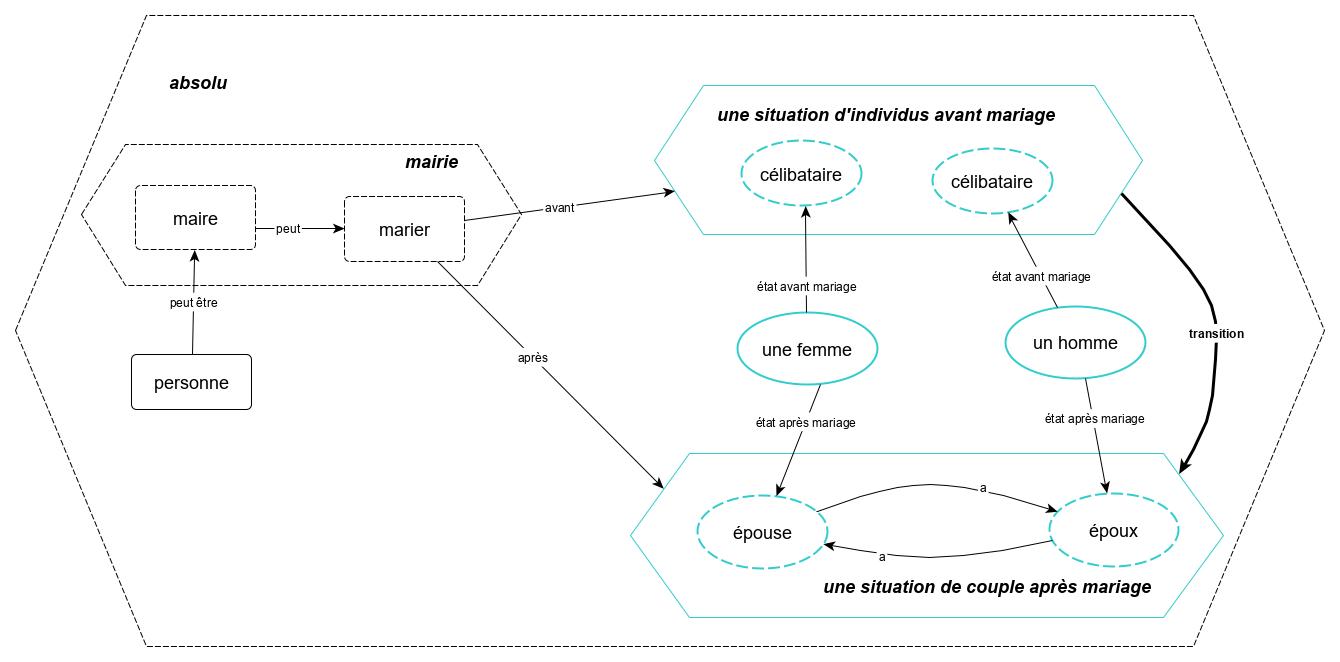
(((monde get: #faune) get: #Maya) getEtat: #miaulant) miaule

donne : Meow

### Exemple 21 Sur les changements de situation

Pour cet exemple, nous allons repartir de l'exemple sur les relations entre manières d'être.

Le graphe suivant décrit le changement de situation d'une femme et d'un homme consécutif à leur mariage par une personne ayant la qualité de maire :



La représentation dans ICEO est la suivante :

iceo definition: #personne

iceo definition: #femme genus: personne

iceo definition: #homme genus: personne

iceo definitionSituation: 'situation de couple'

iceo definitionSituation: 'situation d''individu'

iceo definitionSituation: #mairie

iceo definitionQualite: #épouse situation: (absolu getSituation: 'situation de couple')

iceo definitionQualite: #époux situation: (absolu getSituation: 'situation de couple')

iceo definitionQualite: #célibataire situation: (absolu getSituation: 'situation d''individu')

iceo definitionQualite: #maire situation: (absolu getSituation: #mairie)

((absolu getSituation: 'situation de couple') get: #épouse) associationQualite: ((absolu getSituation: 'situation de couple') get: #époux).

femme peutEtre: ((absolu getSituation: 'situation de couple') get: #épouse).

femme peutEtre: ((absolu getSituation: 'situation d''individu') get: #célibataire).

homme peutEtre: ((absolu getSituation: 'situation de couple') get: #époux).

homme peutEtre: ((absolu getSituation: 'situation d''individu') get: #célibataire).

personne peutEtre: ((absolu getSituation: #mairie) get: #maire).

iceo soit: #Marie essence: femme.

iceo soit: #Pierre essence: homme.

iceo soit: #Christine essence: personne.

iceo soit: 'une mairie' situationGenerique: (absolu getSituation: #mairie).

(monde get: #Christine) addEtat: ((absolu getSituation: #mairie) get: #maire) dansSituation: (monde getSituation: 'une mairie').

iceo soit: 'situation de Pierre et Marie avant mariage' situationGenerique: (absolu getSituation: 'situation d''individu').

iceo soit: 'situation de Pierre et Marie après mariage' situationGenerique: (absolu getSituation: 'situation de couple').

(monde get: #Marie) addEtat: ((absolu getSituation: 'situation d''individu') get: #célibataire) dansSituation: (monde getSituation: 'situation de Pierre et Marie avant mariage').

(monde get: #Pierre) addEtat: ((absolu getSituation: 'situation d''individu') get: #célibataire) dansSituation: (monde getSituation: 'situation de Pierre et Marie avant mariage').

Supposons que la méthode suivante soit associée à la manière d'être maire :

((absolu getSituation: #mairie) get: #maire) compile: 'marier: s1 to: s2

| unHomme uneFemme |

s1 getElements

do: [:etat |

etat getEssence getNom == #célibataire

ifTrue: [etat getEtant removeEtat: etat].

etat getEtant getEssence getNom == #homme

ifTrue: [unHomme := etat getEtant.

unHomme

addEtat: (s2 class get: #époux)

dansSituation: s2]

ifFalse: [uneFemme := etat getEtant.

uneFemme

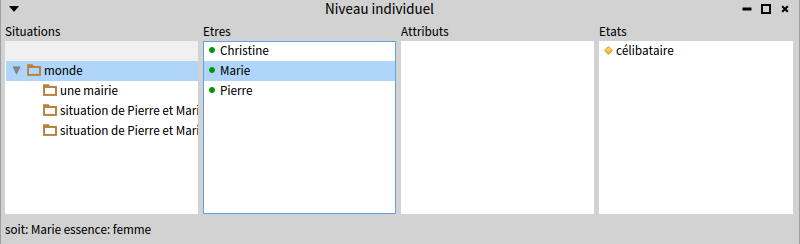
addEtat: (s2 class get: #épouse)

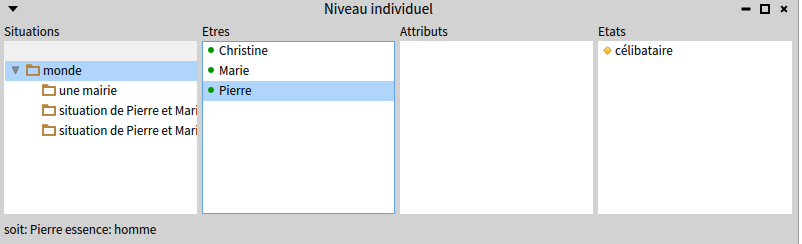
dansSituation: s2]].

(unHomme getEtat: #époux dansSituation: s2)

associationEtat: (uneFemme getEtat: #épouse dansSituation: s2)'.

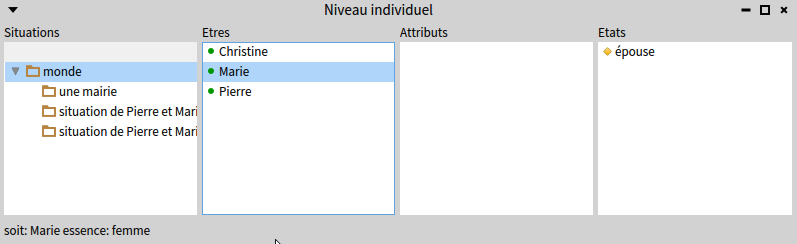
Avant leur mariage, Pierre et Marie sont célibataires :

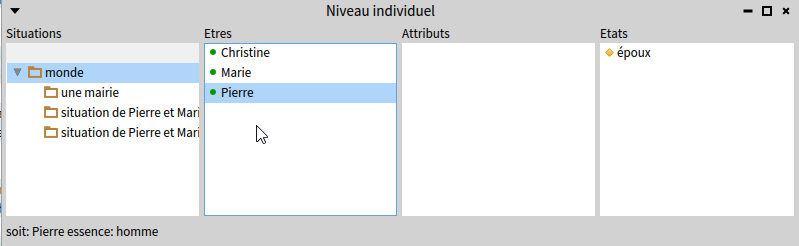




L'action de marier de Christine en tant que maire va les faire passer dans une situation de couple, avec les états d'épouse et d'époux :

((monde get: #Christine) getEtat: #maire) marier: (monde getSituation: 'situation de Pierre et Marie avant mariage') to: (monde getSituation: 'situation de Pierre et Marie après mariage').





Après mariage :

"(((monde get: #Pierre) getEtresAttributsEnTantQue: ((absolu getSituation: 'situation de couple')

get: #époux)) at: 1) getNom" retourne : #Marie

"(((monde get: #Marie) getEtresAttributsEnTantQue: ((absolu getSituation: 'situation de couple')

get: #épouse)) at: 1) getNom" retourne : #Pierre

Ce qui confirme qu'ils ont bien été mariés par Christine en tant que maire et que leur nouvelle situation est une situation de couple.

# Annexe Sur l'implantation d'ICEO en Pharo

Il y a peu de choses à dire sur cette implantation, tant ICEO s'intègre naturellement dans le langage Smalltalk.

Le code source d'ICEO ne pèse que 11 Ko sous forme compressée !

Aucune méthode de l'environnement de base de Pharo n'a été modifiée, à part la méthode evaluateSelection de RubSmalltalkEditor, ce qui évite que le compilateur n'évalue une ligne avant d'avoir terminé d'évaluer la précédente ! (ne s'applique pas dans une fenêtre de type Playground)

RubSmalltalkEditor compile: 'evaluateSelection | node | node := RBParser parseExpression: self selectionAsStream contents. node class == RBMessageNode ifTrue: [ OpalCompiler new evaluate: node newSource ]. node class == RBSequenceNode ifTrue: [ node statements do: [ :each | Semaphore new wait: (Duration milliSeconds: 10) onCompletion: [] onTimeout: [ OpalCompiler new evaluate: each newSource]]]' classified: 'do-it'

Pour permettre l'homonymie des essences, leur création met à profit la possibilité de créer des classes de type "newAnonymousSubclass" :

createFreeSubclass: aSymbol newClass | "not installed in the system" newClass := self newAnonymousSubclass setName: aSymbol. subclasses isNil ifTrue: [ self subclasses: (Array with: newClass) ] ifFalse: [ self subclasses: (subclasses copyWith: newClass) ]. ^ newClass

La classe Essence a comme sous-classes directes les essences "chose" (racine de la hiérarchie des essences dans ICEO) et "absolu" qui est la racine des situations génériques.

La classe ICEO correspond à l'interprète d'ICEO, utilisée pour la définition des situations, des essences et de leurs qualité et l'instanciation des êtres.

La classe Etre est superclasse de la classe Essence. Les attributs et les méthodes de classe de la classe Etre sont les mêmes que ceux des instances de la classe Essence, ce qui permet de voir toute essence comme un être instance de sa propre essence (sa méta-essence).

Par défaut, la méta-essence d'une essence est l'essence chose.

L'essence chose est son propre genus et sa propre méta-essence.