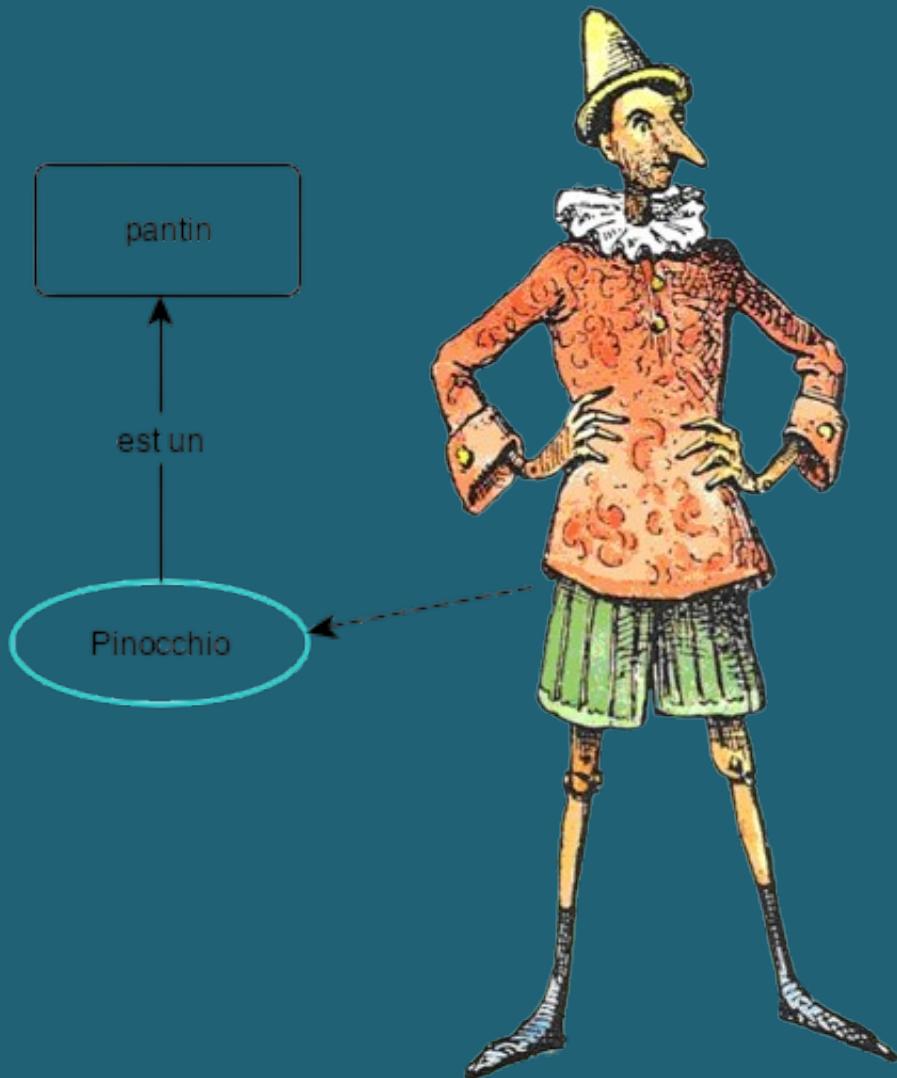


# Sur la modélisation des êtres et de leurs manières d'être dans certaines situations

Robert Bourgeois





Je dédie ce livre à Daniel Kayser  
Spécialiste des sciences cognitives et de l'intelligence artificielle  
Professeur à l'Université de Paris-Nord

## Crédits

L'image de Pinocchio qui apparaît sur la page de couverture a été dessinée par Enrico Mazzanti (1852-1910)

Ce document a été écrit en utilisant l'application TeXstudio

Les graphes présentés dans ce document ont été réalisés avec l'outil Yed de yWorks



# Avant-propos

Le paradigme objet est né en informatique de la notion de *Record Class* de Hoare et des langages comme Simula et Smalltalk qui ont posé les bases des langages orientés objets à classes.

Voici comment Jean François Perrot<sup>1</sup> présente l'essence de l'approche objet : "L'apport des langages objets est de libérer l'esprit et d'écrire comme on pense ...En somme, programmer par objets, c'est tout bonnement programmer de façon naturelle".

Dans ce livre, nous faisons une forte distinction entre les notions d'être (généralisation de la notion d'objet) et de manière d'être.

En français, la phrase "Je sais Paul et je connais qu'il est marié" n'est pas acceptable.

Une tournure correcte serait "Je connais Paul et je sais qu'il est marié".<sup>2</sup>

Le titre de ce livre qui est "Sur la modélisation des êtres et de leurs manières d'être dans certaines situations" aurait pu être : "Sur la représentation des connaissances et des savoirs".

L'objet de ce livre est de modéliser ce qui peut être ou devenir réel, comme les ingénieurs le font par exemple avec UML<sup>3</sup>

Le formalisme que nous présentons, que nous avons nommé ICEO<sup>4</sup>, se présente à la fois sous une forme visuelle graphique<sup>5</sup> et comme un langage textuel directement interprétable par un ordinateur.

La présentation du langage textuel d'ICEO est faite en annexe avec une série d'exemples de complexité croissante<sup>6</sup>

Le code source d'ICEO et les exemples peuvent être librement téléchargés à l'adresse <https://github.com/rodejaphgh/ICEO>

---

<sup>1</sup> "Introduction à la programmation par objets, le système Smalltalk-80"

<sup>2</sup> La connaissance porte essentiellement sur les êtres et le savoir sur les manières d'être.

<sup>3</sup>Unified Modeling Language standardisé par l'OMG.

<sup>4</sup>"iceo" est un mot de l'ancien français signifiant "ceci"

<sup>5</sup>Suivant l'adage "un dessin vaut parfois mieux qu'un long discours".

<sup>6</sup>Pour bâtir ce langage textuel nous avons utilisé Smalltalk; plus précisément, la version  de Smalltalk.



# Introduction

La programmation par objets est essentiellement née de l'osmose entre les langages SIMULA [Dahl et Nygaard, 66] et LISP [McCarthy & al., 62] sous l'impulsion d'Alan Kay (machine FLEX [Kay, 69] et Smalltalk-72) et de Karl Hewitt (langages d'acteurs PLANNER [Hewitt, 72], de CONNIVER [Sussmann & McDermott, 72]. La descendance de Simula et de Smalltalk est nombreuse, elle s'est différenciée dans de nombreux langages comme C++, Java, Python, ...

Alan Kay a construit le langage Smalltalk en partant de six principes :

1. Tout est objet.
2. Les objets communiquent par l'envoi et la réception de messages (eux mêmes des objets).
3. Les objets ont leur propre mémoire (toujours en termes d'objets).
4. Chaque objet est instance d'une classe (qui doit être un objet).
5. La classe détient le comportement partagé par ses instances (sous la forme d'objets dans un programme).
6. Pour l'évaluation du programme, le contrôle est passé au premier objet, le reste est traité par messages.

Il n'est pas question pour nous de remettre en cause ces principes énoncés par Alan Kay il y a plus de 50 ans, dont la pertinence n'est plus à démontrer.

Par contre, on peut légitimement s'interroger sur le choix des mots utilisés.

Suivant le premier principe, en Smalltalk tout est objet : une personne, un animal, une plante, un chapeau, ...

En fait, il s'agit de tout ce qui existe, ce qu'en métaphysique les philosophes désignent depuis plusieurs millénaires par le mot "être".

Un problème plus délicat se situe dans l'utilisation du mot "classe" pour désigner ce qui est tout ... sauf une classe.

Le sens du mot classe (que l'on nous a appris lorsque nous étions en classe !) désigne un ensemble d'entités qui ont une ou plusieurs propriétés caractéristiques en commun. Cette définition est celle utilisée par exemple par Georges Boole dans "Les lois de la pensée", et

qui a été établie par des mathématiciens et logiciens comme Alfred North Whitehead ou Bertrand Russel dans la théorie des classes.

Ce qu'Alan Kay appelle une classe est en réalité un moule qui permet de créer des exemplaires appelés "instances".

Dans ce sens, le mot "essence" défini en philosophie aurait été beaucoup mieux adapté.

En résumé :



Le mot Félix est ici un nom propre, tandis que chat est un nom commun.

Le nom commun d'une essence est généralement appelé un "terme". Pourquoi utiliser ce mot, qui suggère la dernière étape d'un processus ?

Considérez l'image suivante :



Le terme "éléphant" marque la terminaison d'un processus de classification d'animaux qui, selon le Littré, sont de grands et gros mammifères qui se distinguent par leur trompe et leurs longues défenses.

Ce processus de classification aboutit à la création, ce qui peut sembler paradoxal, non pas d'une classe, mais d'une essence.

La classe des éléphants qui sont en Afrique diffère de la classe de ceux qui sont en Asie, et la classe des éléphants sur terre à un instant t est différente de celle à l'instant t'.

Par contre, l'essence éléphant est immuable.

Une faiblesse que l'on observe dans tous les langages objets est de ne pas différencier les notions d'être et de manière d'être.

Une manière d'être peut être essentielle (comme la qualité d'être mortel pour l'homme) ou accidentelle (comme la qualité d'époux ou d'épouse pour une personne).

La notion de "méthode" des langages objets, qui répond au principe d'Alan Kay de communication des objets par l'envoi et la réception de messages, n'est rien d'autre qu'un cas particulier de manière d'être. Ainsi, par exemple, dormir pour un chat correspond à une manière d'être essentielle et intermittente.

Les relations entre les êtres (autres que la relation de composition) sont liées à leurs manières d'être dans certaines situations.

Ainsi, par exemple, un homme et une femme peuvent être liés dans leur situation maritale. Ce lien disparaît s'ils divorcent.

Quel sens donnons-nous au mot situation ? Ce mot désigne à la fois un acte et le résultat de cet acte. En tant que résultat, une situation correspond à un espace où peuvent se situer des êtres, par exemple dans l'espace et le temps. Pour des animaux, une faune correspond à l'ensemble des animaux d'un pays.

En résumé, dans cette étude nous ne parlerons pas d'objets et de classes, mais d'êtres et d'essences.

Nous introduirons une distinction fondamentale entre les notions d'être et de manière d'être. Le comportement d'un être sera considéré comme un cas particulier de manière d'être qui produit un effet.

Comme nous le montrons dans cette étude, cette distinction n'est pas en contradiction avec les principes d'Alan Kay.



# Première partie : sur la notion d'être



# Etres

D'un point de vue étymologique, le mot "être" dénote dans les langues indo-européennes "ce qui se tient en soi-même"<sup>1</sup>.

Les êtres considérés dans cette étude sont des êtres anoméomères (insécables)<sup>2</sup> ou homéomères (dont la division ou l'agrégation produisent des choses de même essence); ils peuvent être concrets (comme un cheval) ou abstraits (comme l'essence d'un être, la joie, le courage, ...), traditionnellement qualifiés d'universels par les philosophes et les logiciens, au sujet desquels ils se divisent encore sur la question de leur existence.

Noter que les êtres considérés comme homéomères ne le sont qu'à un certain degré de division et/ou d'agrégation.

Ainsi, un glaçon n'est ni une poussière de glace ni un glacier.

Un être peut être atomique ou non. Un être non atomique naît de la relation établie entre ses constituants<sup>3</sup>, qui peuvent être communs à d'autres êtres, sauf dans le cas d'individus (nous reviendrons sur cette notion plus loin).

Tout être possède une existence propre.

## Existence d'un être

L'existence d'un être est ce qui résulte de sa création.

Un être se caractérise par l'immuabilité de son existence et la stabilité de sa structure.

Comme le faisait observer E. Kant<sup>4</sup>, dire qu'un être existe n'est pas ajouter l'attribut existence à sa structure.

L'existence d'un être est ce qui subsiste quand on le considère comme un entier, dans son entièreté, comme un tout<sup>5</sup>.

Si on considère l'existence comme étant le fondement de tout être, tout ce qui existe peut être considéré comme un être (les êtres vivants, les objets, les essences, les qualités, les

<sup>1</sup>cf. l'ouvrage "*En guise de contribution à la grammaire et à l'étymologie du mot "être"*" de Martin Heidegger.

<sup>2</sup>En coupant un cheval en deux on obtient deux bouts de cheval, mais pas deux chevaux.

<sup>3</sup>Un cheval est constitué d'une tête, de quatre pattes, ...

<sup>4</sup>cf. "*La critique de la raison pure*" d'Emmanuel Kant

<sup>5</sup>L'axiomatique de Peano qui porte sur les "entiers" naturels ne prend en compte que l'existence des êtres vus dans leur entièreté, en faisant abstraction de leur essence et de leur individualité.

actions, les états, les faits, les événements, les nombres, le faux et le vrai , ...) <sup>1</sup>

L'existence des êtres connus par un sujet est enregistrée dans sa mémoire qui constitue pour lui un référentiel<sup>2</sup>.

Les existences sont discernables et peuvent être dénombrées.

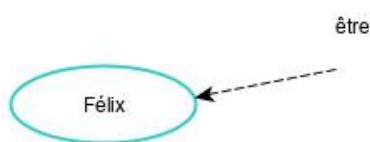
Faire référence à un être ne requiert que la connaissance de son existence.

Un être peut être dénoté par un nom propre permettant de l'identifier dans une situation donnée.

Suivant l'usage en français, nous écrirons le nom propre des êtres avec une majuscule, et

nous utiliserons le symbole graphique  pour représenter l'existence d'un être.

Exemple, pour un être nommé Félix :



## Situation

Le mot situation désigne l'acte de situer des êtres ou le résultat de cet acte.

En tant qu'acte, une situation distingue et rassemble les êtres sur une propriété particulière.

Tous les ensembles considérés dans ICEO sont des ensembles discrets et finis<sup>3</sup>.

Par exemple : les hommes, les hommes célibataires, les objets posés sur une table, ...

Un être peut apparaître dans différentes situations, mais la situation de définition d'un être (celle où il a commencé à exister) est unique.

Une situation peut concerner un ensemble d'êtres qui n'ont pas coexisté dans le temps, comme les rois et les reines d'Angleterre.

Une situation peut évoluer dans l'espace et le temps.

Compter des êtres suppose de les avoir situés préalablement ensemble. Ce comptage ne peut faire abstraction de la propriété caractéristique qui permet de les rassembler, de les situer collectivement.

<sup>1</sup>Le premier postulat en Smalltalk est : tout est objet. Pour nous, l'équivalent de ce postulat (auquel nous adhérons totalement) est : "tout est être".

<sup>2</sup>Constitué dans un ordinateur par un espace ordinal où chaque existence est représentée de manière univoque par une adresse mémoire, un entier naturel.

<sup>3</sup>Leurs éléments sont des points d'un espace topologique où chacun est isolé, éloigné des autres, et qu'il est possible de compter.

Dire par exemple qu'il y a 4 carottes dans un panier suppose qu'elles soient existentiellement discernables et fait référence à leur essence carotte. Cette remarque peut sembler idiote, mais si l'on veut prendre en compte (compter) également les 2 poireaux qui s'y trouvent, on ne pourra dire qu'il contient 6 carottes ou 6 poireaux, mais qu'il contient 6 légumes, en supposant que les carottes et les poireaux soient aussi des légumes.

Le terme qui suit une cardinalité dans une expression dénote toujours une essence ou une qualité qui est commune à tous les êtres comptés.

Comme tout ensemble, une situation peut être identifiée par un nom (par exemple la "famille Gonthier ", ...).

# Essence d'un être

L'idée qui sous-tend la notion d'essence<sup>1</sup> est celui de plan qui permet de créer les êtres, de les classifier et de les reconnaître.

La notion d'essence est proche de la notion de concept générique des réseaux sémantiques, de classe des langages objets dits "de classe", de frame des langages de frames<sup>2</sup>.

La connaissance d'un être ne doit pas être confondue avec la connaissance de l'existence d'un être.

En français<sup>3</sup>, la connaissance de l'existence d'un être s'exprime avec le verbe "savoir".

Un sujet peut savoir l'existence ou l'inexistence d'êtres d'une essence donnée dans une situation donnée.

La classe des hommes dans l'univers est différente de celle des hommes qui existent en France, et la classe des hommes qui existent à l'instant  $t_1$  est différente de celle à  $t_2$ .

Pourtant, n'existe t-il aucun point commun entre les éléments de ces différentes classes ? C'est leur essence.

La classe des hommes qui existent à un instant donné dépend de la situation considérée, alors que l'essence homme est immuable.

Comme le disait Abélard<sup>4</sup>, même s'il n'y avait plus une seule rose au monde, le nom rose aurait une signification pour l'entendement.

Nous verrons plus loin qu'une essence peut être considérée elle-même comme un être ayant sa propre essence (sa "méta essence")

---

<sup>1</sup>Selon le Littré, le mot essence provient du latin *essentia*, de *esse*, être. "L'essence est ce qui fait qu'une chose est ce qu'elle est, ce en l'absence de quoi elle ne serait pas ce qu'elle est". Les auteurs des langages informatiques dit "objets" ont choisi de nommer "classe" ce que nous nommons essence.

<sup>2</sup>Le terme frame (cadre) a d'abord été utilisé par Marvin Minsky comme paradigme de compréhension et de traitement du langage naturel.

<sup>3</sup>En français la phrase "Je connais Paul" est correcte, tandis que "Je sais Paul" ne l'est pas. Par contre "Je sais que Paul existe (ou n'existe pas)" est correcte. Toutes les langues n'ont pas dans leur vocabulaire des mots faisant cette distinction entre savoir et connaître.

<sup>4</sup>Pierre Abélard est un philosophe français né en 1079 et mort en 1142. Sur sa vision de la notion d'essence, on pourra se référer à l'article "*La signification des universaux chez Abélard*" paru dans la Revue Philosophique de Louvain en 1982

## Constitution de l'essence d'un être

Une essence est définie<sup>1</sup> par l'agencement d'un ensemble d'autres essences qui constituent ses attributs.

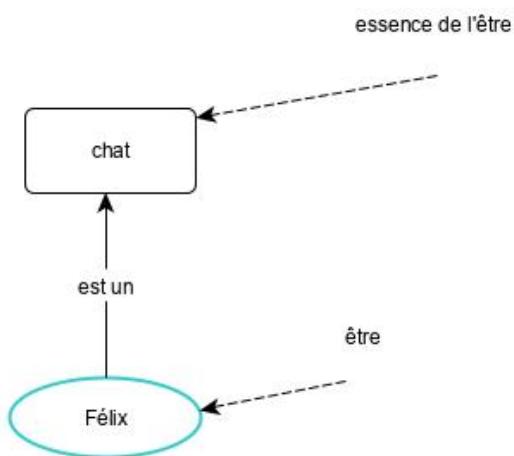
L'essence d'un être prend en compte :

- sa composition en termes d'essences (ses attributs)
- les contraintes structurelles imposées à ses attributs
- ses manières d'être essentielles.

Une essence est dénotée par un terme qui marque la terminaison de l'accomplissement d'un processus de modélisation et de différentiation par rapport aux autres essences. En français, un nom commun est un terme.

Suivant l'usage en français, nous écrirons le nom commun des essences avec une minuscule, et nous utiliserons le symbole graphique  pour représenter l'essence d'un être.

Exemple :



Nous allons nous intéresser tout d'abord à la composition d'une essence.

Le principe exposé est très proche du principe de composition d'Aristote.

Les notions de contrainte structurelle et de manière d'être seront abordées plus loin.

---

<sup>1</sup>Le sens du terme "définition" que nous utilisons est celui donné par Aristote dans les Topiques : "une définition est une phrase indiquant l'essence de quelque chose"

## Composition d'une essence

Pour composer une essence nous définissons une loi de composition externe à droite nommée "attribution" symbolisée par " $\oplus$ " qui permet de définir une essence  $y$  à partir d'une essence  $x$  et de l'ensemble de ses attributs  $a_1, a_2, \dots, a_n$  :

$$y = \oplus (x, \{ a_1, a_2, \dots, a_n \}) \text{ où } a_1, a_2, \dots, a_n \text{ sont des essences}$$

Exemple : chat =  $\oplus$  (animal, { tête, queue, ... })

Une essence ne peut être l'un de ses attributs.

Reprenant la terminologie d'Aristote, nous appellerons  $x$  le *genus* de  $y$  et  $\{ a_1, a_2, \dots, a_n \}$  son *differentia*.

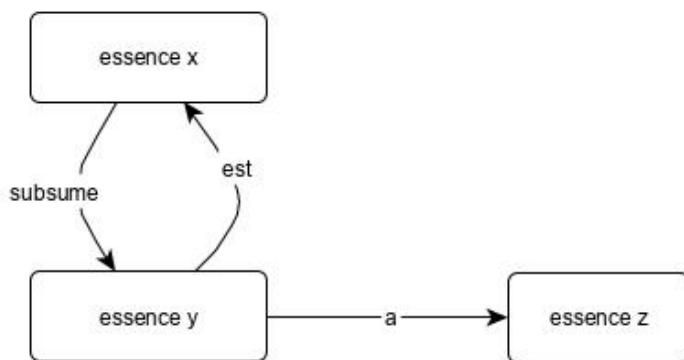
La loi d'attribution permet de définir une relation d'ordre entre essences appelée subsomption :

$$x \text{ subsume } y \text{ si et seulement si il existe un ensemble d'essences } E \text{ tel que } y = \oplus (x, E)$$

Ainsi dans notre exemple, l'essence animal subsume l'essence chat.

Nous nommerons "est" ou "spécialise" la relation converse de "subsume".

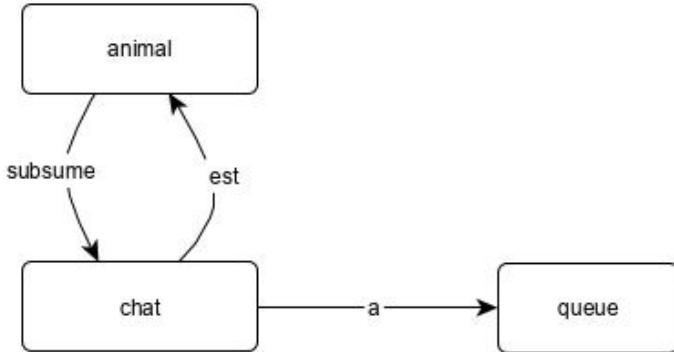
### Représentation graphique



Légende :  $y = \text{attribution}(x, \{ z \})$  ou  $y = \oplus (x, \{ z \})$

Supposons que l'essence chat soit définie par : "chat est animal qui a queue" (ce qui est certes un peu restrictif par rapport à la réalité ... )

La représentation graphique de cette définition est :



**chat est animal qui a queue** ( $\text{chat} = \oplus (\text{animal}, \{ \text{queue} \})$ )

Par définition, toute essence est subsumée par elle-même.

Une essence ne peut avoir plusieurs genus<sup>1</sup>.

Le graphe de subsomption est un graphe hiérarchique dont la racine est l'essence nommée "chose" qui n'est subsumée que par elle-même.

L'ensemble des attributs de chose est vide<sup>2</sup>.

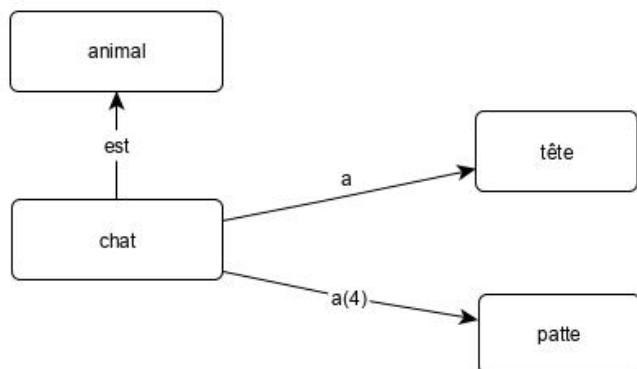
### Prise en compte de la notion de multiplicité

La multiplicité est une contrainte structurelle qui porte sur le nombre d'êtres d'une essence donnée rentrant dans la composition d'un être<sup>3</sup>.

Par exemple, nous pourrions écrire :  $\text{chat} = \oplus (\text{animal}, \{ (\text{tête}, 1), (\text{patte}, 4), \dots \})$

#### Représentation graphique

La cardinalité sera omise si elle est égale à 1 et notée \* si elle est quelconque.



<sup>1</sup>Le graphe de subsomption n'est donc pas un treillis comme c'est le cas pour la hiérarchie de types définie dans les réseaux sémantiques tels que décrits dans l'ouvrage "*Conceptual Structures. Information processing in mind and machine*" de J.F. Sowa.

<sup>2</sup>Le differentia de chose est l'ensemble vide, car l'essence chose est son propre genus. Nous appellerons "rien" l'ensemble d'essences vide. Cet ensemble vide est l'élément neutre à droite de la loi d'attribution.

<sup>3</sup>Rappelons que dans ICEO les ensembles considérés sont discrets et finis.

## Sur la notion d'attribut

ICEO diffère d'autres formalismes pour la définition des attributs.'

Dans les langages comme SMALLTALK, OBJVLISP, FLAVORS, Java, ... un attribut est la valeur d'une variable, comme l'est un membre d'une structure dans le langage C, équivalent à un champ d'un record en Pascal.

Les langages SMALLTALK, RLL, LORE, KRS, CLOS, CLASSTALK, MERING IV, ... ont opté pour une définition récursive des attributs, un attribut devenant une référence d'un objet sur un autre objet.

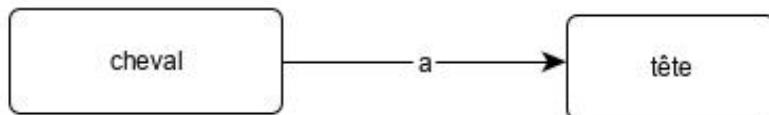
Si cette définition récursive des attributs permet de créer des structures emboîtées d'objets, elle n'uniformise toutefois pas totalement les notions d'objet et d'attribut.

Une distinction subsiste toujours entre la définition des essences et de leurs attributs.

Ainsi dans ces langages l'attribut de nom "tête" de cheval peut avoir aussi bien la classe "tête" que la classe "queue", sans aucun lien sémantique avec elle.

Ceci n'est pas possible dans ICEO.

Ainsi le graphe suivant définit dans ICEO que l'essence cheval a un attribut nommé "tête" qui est une essence :



Les graphes suivants seraient incorrects :

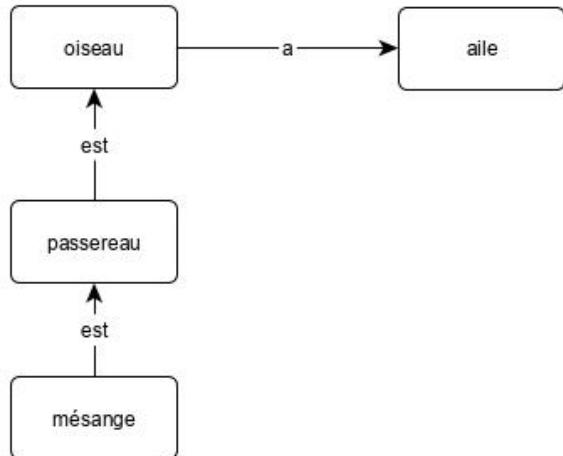


## Principe d'héritage des attributs des essences

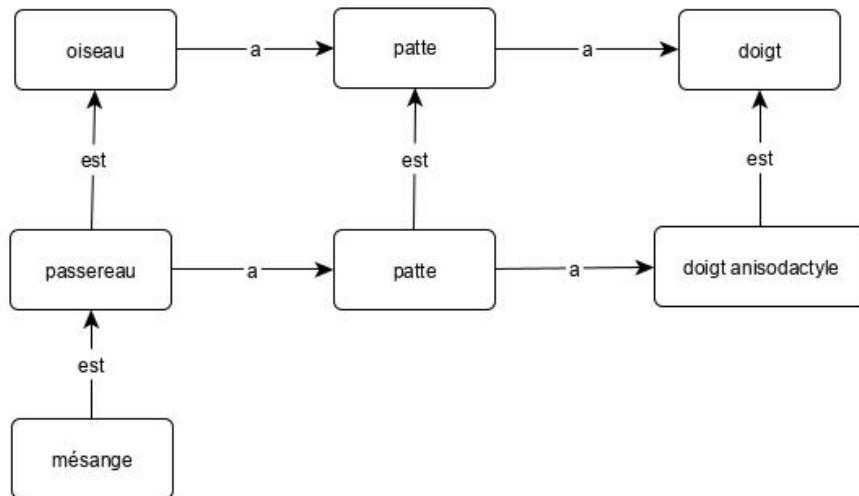
La transitivité de la relation d'ordre de subsumption est exploitée dans un principe communément appelé "héritage".

Une essence hérite des attributs des essences subsumantes.

Ainsi par exemple, si "oiseau a { aile }", "passereau est oiseau" et "mésange est passereau" alors par héritage on en déduit que "passereau a { aile de oiseau}" et que "mésange a { aile de oiseau }" :



Les oiseaux ont des pattes et les passereaux ont des pattes qui se terminent par 4 doigts dont trois sont orientés vers l'avant et un vers l'arrière (ils sont anisodactyles) :



Dans cet exemple, mésange hérite de l'attribut patte de passereau qui spécialise l'attribut patte de oiseau.

Nous verrons plus loin que le principe d'héritage prend également en compte les autres aspects d'une essence que sont ses contraintes structurelles internes et ses manières d'être.

## Création d'une essence

Une essence dépend des essences qui la composent, mais l'inverse n'est pas toujours vrai.

Autrement dit, les essences attributs d'une essence peuvent ou non lui être propres.

Deux principes sont utilisables pour créer une essence :

- soit créer l'essence à partir d'essences attributs qui lui sont propres.
- soit créer l'essence à partir d'essences existantes.

### Définition d'une essence à partir d'attributs qui lui sont propres.

Dans ce cas, l'essence constitue la situation de définition de ses essences attributs.

Ainsi par exemple, voici une définition de l'essence chat :

$$\text{chat} = \oplus (\text{animal}, \{ (\text{tête de chat}, 1), (\text{patte de chat}, 4), \dots \})$$

patte de chat est propre à chat. Elle a par exemple des griffes rétractiles que ne possède pas patte de chien.

Noter qu'un attribut n'est généralement propre à une essence que jusqu'à un certain niveau de décomposition.

Ainsi, les gènes qui constituent l'essence chat ne lui sont pas tous propres, et encore moins les atomes qui constituent ses gènes.

### Définition d'une essence basée sur l'agencement d'essences qui ne lui sont pas propres.

Dans de nombreux cas, les essences qui composent une essence ne lui sont pas propres.

Autrement dit, l'essence ne constitue pas la situation de définition de ses essences attributs. Elle se contente de référencer les essences qui la composent.

C'est le cas par exemple des atomes, des composés chimiques, de la plupart des objets conçus par l'homme.

Ainsi, en électronique, rien ne distingue l'essence "patte de transistor" de l'essence "patte de diode".

En chimie, oxygène-16 est l'isotope le plus abondant dans l'oxygène naturel.

Sa définition est :

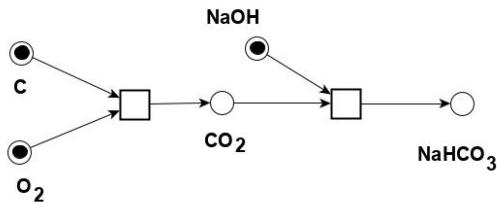
$$\text{oxygène-16} = \oplus \{ \text{chose}, \{ (\text{proton}, 8), (\text{électron}, 8), (\text{neutron}, 8) \} \}$$

La définition des essences proton, électron et neutron est indépendante de celle de l'oxygène.

L'essence oxygène rentre elle-même dans la composition de nombreuses essences (composés chimiques) mais sa définition est indépendante des composés dont elle fait partie.

Le processus de création des composés chimiques est à l'origine des réseaux de Petri<sup>1</sup>.

Exemple:



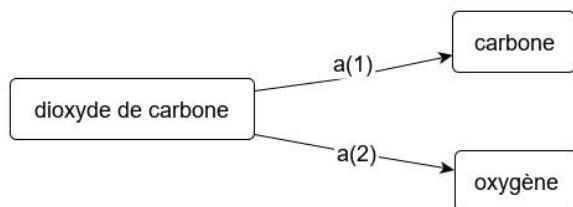
On voit dans cet exemple que l'essence O (oxygène) rentre dans la composition de différentes essences comme CO<sub>2</sub> (oxyde de carbone), NaHCO<sub>3</sub> (bicarbonate de sodium), ...

Noter que le processus permettant de créer (synthétiser) une essence à partir de ses essences attributs n'est pas forcément unique.

Ainsi, le processus le plus utilisé actuellement pour produire le bicarbonate de sodium est plutôt décrit par :



Voici la représentation de la molécule de dioxyde de carbone :



Les figures géométriques sont un exemple d'essence définie à partir d'essences existantes (comme point, ligne, ...).

En aéronautique, on peut retrouver par exemple la même essence "hélice" dans la composition de différents avions à hélice.

Dans l'industrie, le fait de chercher à créer des composants réutilisables est érigé en principe pour des questions de coût de conception et de fabrication.

<sup>1</sup>inventés en août 1939 par l'Allemand Carl Adam Petri, à l'âge de 13 ans

## Situation générique

Une situation générique est un ensemble d'essences (les essences pouvant être considérées comme des êtres, nous reviendrons plus loin sur ce sujet).

Lors de sa création, l'extension d'une situation générique est vide.

Les situations génériques forment un graphe hiérarchique dont la racine est appelée "absolu".

L'essence chose est définie dans l'absolu et les essences définies dans l'absolu sont celles d'individus (nous reviendrons plus loin sur cette notion).

Le critère de rassemblement d'essences dans une situation générique est une qualité abstraite. Une situation générique peut correspondre à un point de vue existentiel, social, esthétique, statique, dynamique, ...

Ainsi, la situation générique "zoologie" définie dans l'absolu regroupe toutes les essences en rapport avec les animaux, leur organisme et leurs modes de vie.

Une essence est définie dans une et une seule situation générique (sa situation de définition) et par principe deux essences de même nom ne pourront apparaître dans la même situation générique de définition (celle-ci formant un espace de nommage).

Par contre, rien n'empêche d'avoir des essences homonymes définies dans des situations génériques différentes (comme l'essence chien définie en zoologie, l'essence chien d'une arme à feu, l'essence chien du jeu de tarot, l'essence chien assis définie en architecture, l'essence chien jaune définie en aéronautique, ...).

Une essence peut être référencée dans d'autres situations génériques que sa situation générique de définition.

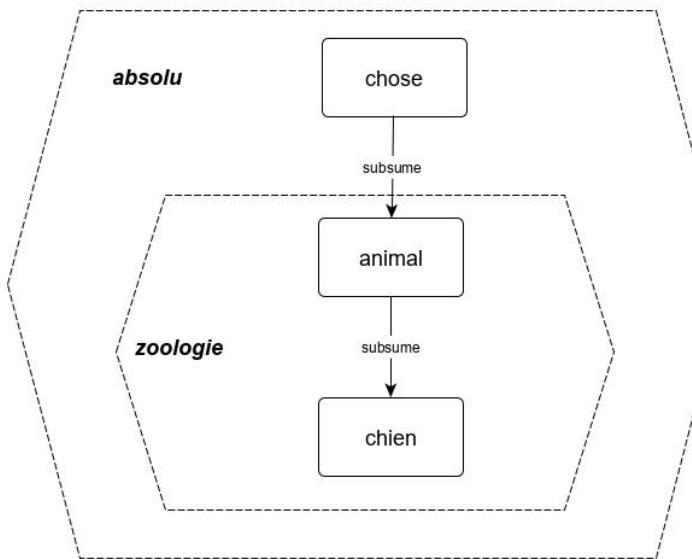
Ainsi, l'essence homme définie en anthropologie peut être référencée dans la situation de couple, de famille, de village, de nation,...

La même essence peut être représentée différemment suivant la perspective utilisée. Nous parlerons de connotations.

Ainsi en est-il de la représentation de l'essence vecteur dans un système de coordonnées cartésiennes ou polaires.

Nous utiliserons le symbole graphique  pour représenter une situation générique

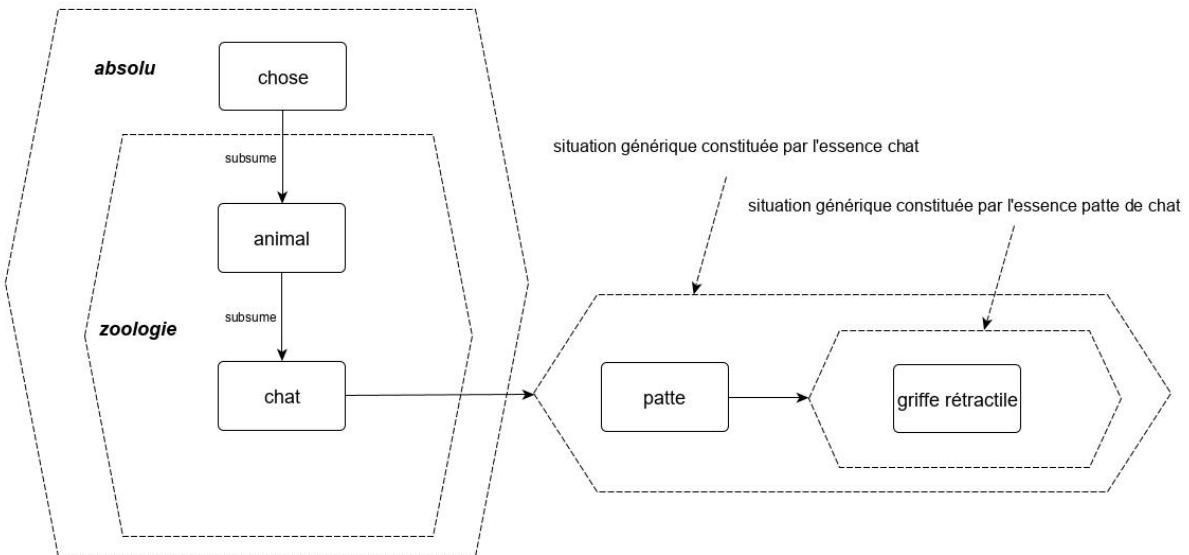
Exemple :



La situation générée zoologie est ici incluse dans l'absolu.

Une essence dont les attributs lui sont propres constitue la situation générée de définition de ses attributs.

Exemple :



L'essence patte n'est pas ici définie en zoologie comme animal et chat, mais dans l'essence chat (le *differentia* de chat). De même l'essence griffe rétractile de patte de chat est définie dans l'essence patte de chat.

Pour éviter une représentation visuelle trop complexe, nous ne présenterons pas systématiquement les situations générées incluses dans l'absolu (comme zoologie) dans nos graphes, ainsi que la situation générée que constitue chaque essence, sauf si cette simplification est source d'ambiguïté.

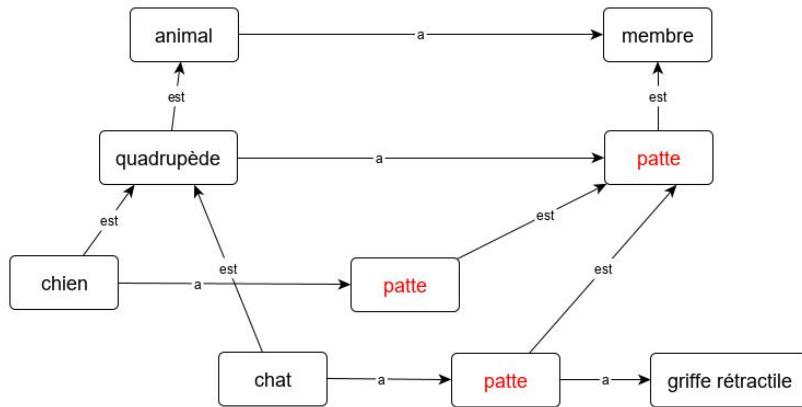
Par défaut, l'essence chose ne sera pas non plus représentée.

# Identification des essences

Une situation générique constitue un espace de nommage pour les essences qui y sont définies, mais des essences de même nom peuvent apparaître dans des situations différentes.

Les essences qui sont attributs propres d'une essence peuvent être identifiées par rapport à l'essence dont elles font partie.

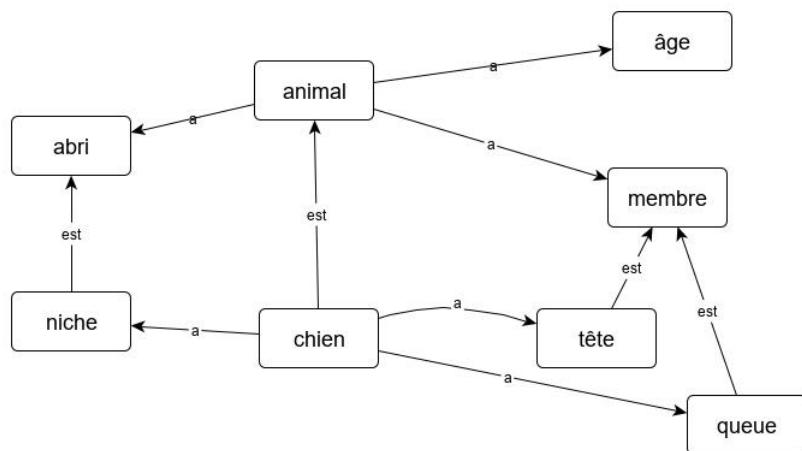
Ainsi, considérons le graphe suivant :



Différentes essences homonymes telles que celles nommées "patte" apparaissent dans ce graphe<sup>1</sup>. De fait, patte de chat n'est pas patte de chien, ne serait-ce que par le fait que patte de chat a griffe rétractile.

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

Considérons l'exemple suivant :



Dans cet exemple, l'expression "âge de chien" retournerait "âge de animal", "abri de chien" retournerait "niche de chien" et "membres de chien" retournerait l'ensemble {tête de chien, queue de chien}

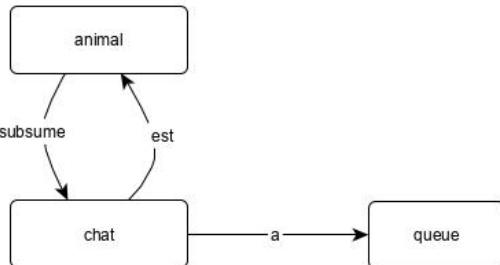
---

<sup>1</sup>En Smalltalk, où l'héritage des variables d'instance est statique, une classe et ses sous-classes ne peuvent avoir des variables d'instance de même nom.

## Essences abstraites

Une essence abstraite est une essence qui a simplement la propriété de subsumer (être le genus) d'une autre essence.

Considérons le graphe suivant :



`chat = attribution (animal, { queue })`

L'expression "chat = attribution (animal, {queue})" est équivalente à "animal = omission(chat, {queue})".

En ce sens, l'essence animal peut être vue comme étant abstraite (soustraite) de l'essence chat en lui enlevant sa queue. Mais rien n'interdit l'instanciation de l'essence animal.

Supposons par exemple que vous entendiez le bruit de quelque chose qui se déplace dans votre grenier.

A priori, vous croyez qu'il s'agit d'un animal. Si vous entendez ensuite cet animal miauler, vous allez en déduire que c'est un chat.

A votre connaissance, l'être perçu a tout d'abord été une instance de l'essence animal, puis est devenu une instance de chat.

La connaissance de l'essence d'un être par un sujet est ainsi révisable.

La possibilité d'instancier toute essence (qu'elle soit abstraite ou non) est primordiale dans une démarche de représentation d'êtres dont l'existence est connue du sujet mais dont la connaissance de l'essence est révisable.

Dans une démarche d'invention d'une nouvelle essence (par exemple un nouveau type d'avion), la situation est différente. La définition de celle-ci passe généralement par divers stades de spécialisation (de conception), mais lors de la création d'un être conformément à une nouvelle essence, la définition de celle-ci doit être aboutie (non abstraite) et la connaissance du sujet sur un être qu'il a créé est totale.

En sachant par exemple que les essences mouton et chien sont subsumées par animal, supposons que vous demandiez à quelqu'un de vous dessiner un animal. La personne va vous dessiner un mouton ou un chien. Elle ne peut pas vous dessiner un animal, car il s'agit d'une essence abstraite qui n'a pas de forme.

## **Essence et normalité**

Savez-vous que la plupart des types de moutons naissent avec une queue ? J'ai longtemps cru le contraire en les voyant dans les champs. En réalité, les bergers leur coupent généralement la queue lorsqu'ils sont agneaux pour des questions d'hygiène, en particulier pour les brebis utilisées pour la traite.



Mais un mouton auquel on a coupé la queue est-il encore un mouton ? Je vous laisse répondre à cette question.

Toute essence définit une normalité qui tolère des écarts, à condition bien sûr de ne pas porter atteinte à des caractéristiques essentielles. Ainsi, une voiture qui a perdu une roue n'est plus une voiture.

# Constitution d'un être

Un être est créé par instantiation<sup>1</sup> de son essence, acte qui lui donne existence<sup>2</sup>.

Tout être possède une et une seule essence.

Un être se caractérise par :

- sa composition en termes d'attributs
- ses contraintes structurelles
- ses états qui correspondent à des manières d'être essentielles

Nous allons tout d'abord nous intéresser à la composition d'un être. Les notions de contrainte structurelle et d'état seront abordées plus loin.

Notons qu'à un instant donné, si une essence x subsume une essence y, la classe des instances de y est incluse dans la classe des instances de x. Ainsi, à un instant donné le nombre d'animaux sur terre est plus grand que celui des chats.

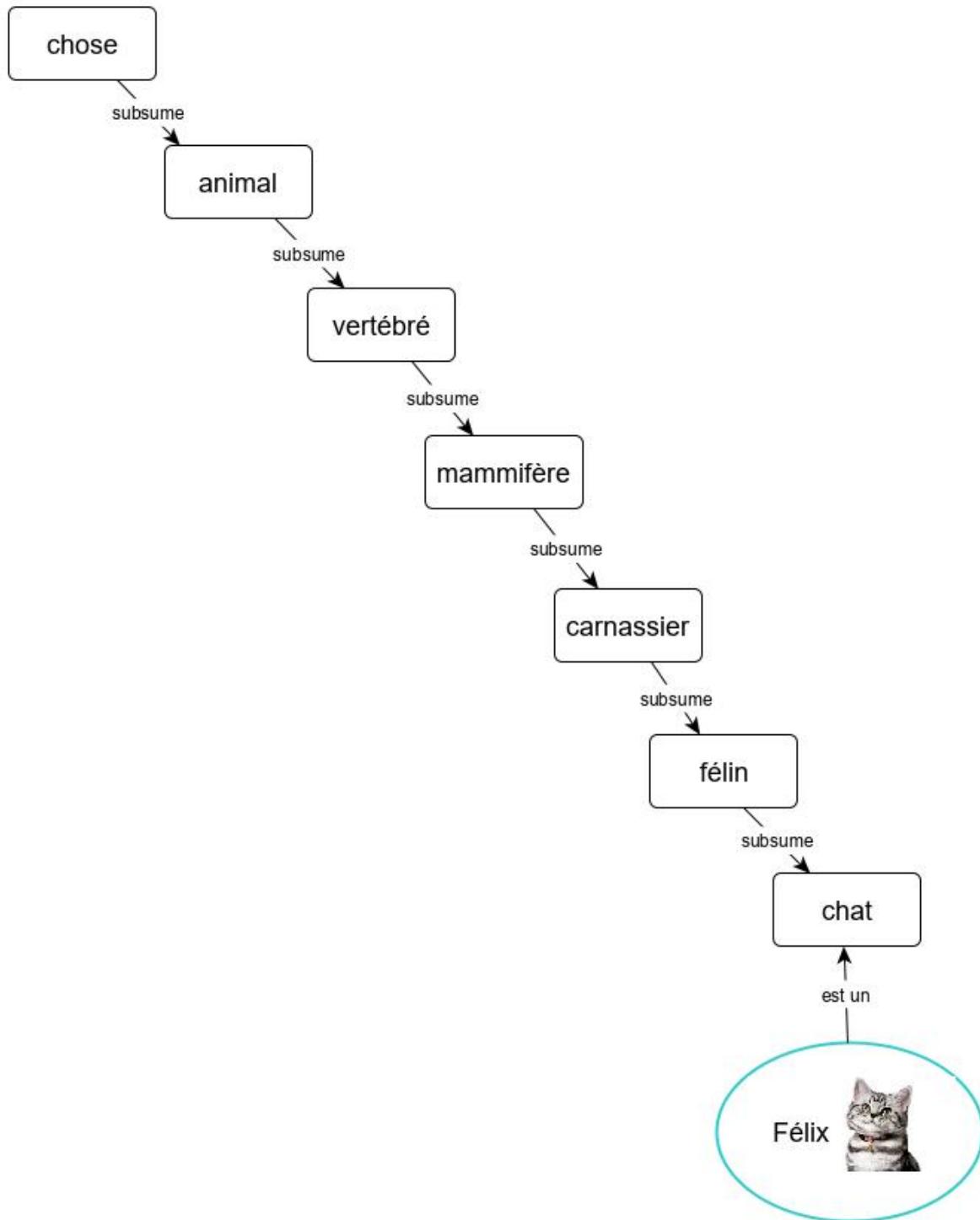
L'essence d'un être est unique, mais tout être est instance des essences subsumant son essence. Par transitivité, tout être est une (quelque) chose.

Ainsi par exemple, le graphe suivant exprime que Félix est un chat mais aussi un félin, un carnassier, un mammifère, un vertébré, un animal et une chose :

---

<sup>1</sup>"Instancier" est en français un néologisme venant du verbe anglais "to instantiate" qui signifie "créer un exemplaire de". L"instanciation" est l'acte de créer (instancier) un être à partir de son essence, et "instance" l'être créé.

<sup>2</sup>Ou par copie d'un être existant (un prototype), la copie créant un être de même essence que l'original.



# Situation individuelle

Une situation individuelle correspond à un ensemble d'êtres.

Une même situation individuelle peut concerner simultanément plusieurs êtres. Par exemple un chien qui dort et un autre qui mange.

Les situations individuelles forment un graphe hiérarchique dont la racine est appelée "monde".

Une situation individuelle est instance d'une situation générique.

Ainsi, la situation individuelle "faune" est instance de la situation générique "zoologie".

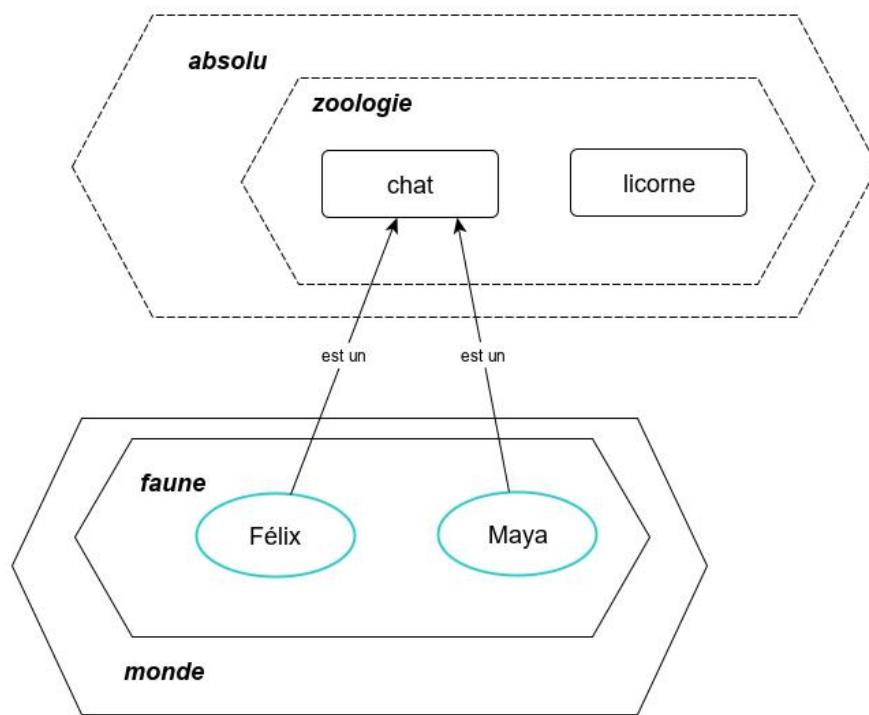
La situation "monde" est instance de la situation générique "absolu".

Lors de sa création, une situation individuelle est vide.



Nous utiliserons le symbole graphique pour représenter une situation individuelle.

Exemple :



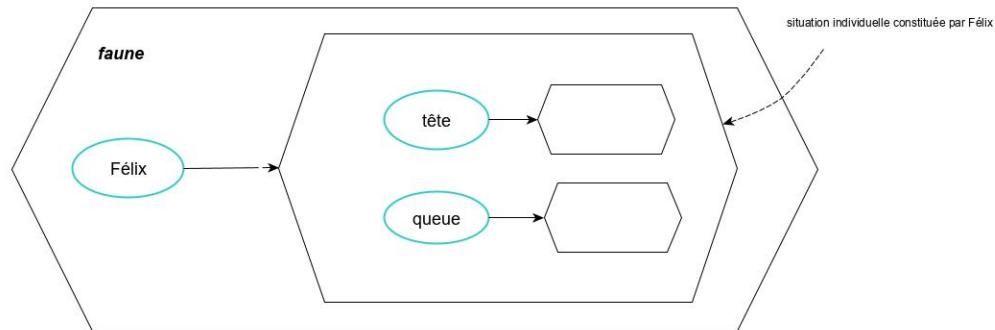
Félix et Maya sont des chats d'une faune située dans le monde. L'essence licorne n'a pas été instanciée (et ne le sera sans doute jamais).

Un être né dans une situation individuelle peut être référencé dans d'autres situations. Ainsi une famille, un village, une nation, ... , sont des situations qui réfèrent des êtres

(membres, habitants, ...).

Un être composé constitue la situation de définition de ses êtres attributs.

Exemple :



Les êtres queue et tête sont situés dans Félix.

## Individus

Un individu<sup>1</sup> est un être indivisible dont la situation de définition n'est pas un autre être et n'est pas une manière d'être (notion que nous étudierons plus loin) .

Exemples :

- Un atome d'oxygène est un individu : son existence n'est pas liée au fait que l'oxygène soit un constituant de composés comme la molécule d'eau.
- Un chien est un individu mais une patte de chien ne l'est pas.
- Une contrainte de structure interne d'un être n'est pas un individu.
- La blancheur de la neige n'est pas un individu.

Noter que le fait d'être élément d'un ensemble pour un être ne lui interdit pas d'être un individu, car son existence n'est généralement pas liée à son appartenance à cet ensemble.

## Création des êtres

Dans ICEO, les êtres qui constituent un être forment un ensemble discret et fini.

Ceci exclut par exemple l'idée de créer un cercle à partir de ses points.

Toutes les instances de l'essence cercle resteront définies en ICEO par leur centre et leur rayon, et éventuellement représentées en pointillés, comme un cercle tracé au compas.

Dans ICEO, un être est créé par agencement des êtres existants qui en font partie.

<sup>1</sup>Selon le dictionnaire de l'Académie française, un individu est un être formant une unité distincte et identifiable, qui ne peut être divisé sans être détruit. Le mot individu vient du mot latin "individuus" qui signifie "indivisible, inséparable".

Ainsi, ça n'aurait pas de sens de créer un chat sans ses pattes ou de créer un avion avant de créer ses ailes, ...

Il existe dans ICEO une sorte de morphisme  $\psi$  entre l'ensemble des êtres et celui des essences qui est tel que la structure d'un être est conforme à la structure générique de son essence.

$$\psi(x) = \text{essence de } x$$

Nous définirons le prédicat "est"<sup>1</sup> qui est tel que si  $x$  est instance de l'essence  $y$  :

$$x \text{ est } y \iff y = \psi(x) \text{ ou } y \text{ subsume } \psi(x)$$

Les êtres qui constituent un être non atomique peuvent lui être propres ou non.

### **Création d'un être à partir d'attributs qui lui sont propres.**

Dans ce cas, l'être constitue la situation de définition de ses êtres attributs.

Ainsi, par exemple, les pattes d'un chat lui sont propres.

### **Création d'un être à partir d'attributs qui ne lui sont pas propres.**

Dans certains cas, les êtres qui constituent un être ne lui sont pas propres.

Autrement dit, l'être ne constitue pas la situation de définition de ses êtres attributs.

C'est le cas par exemple d'une salle dans un bâtiment : les murs qui la composent peuvent être communs à d'autres salles.

Une salle se contente de référencer les murs qui la composent.

C'est également le cas par exemple des lumières reflétées par un objet par rapport aux lumières émises par une lampe, ...

---

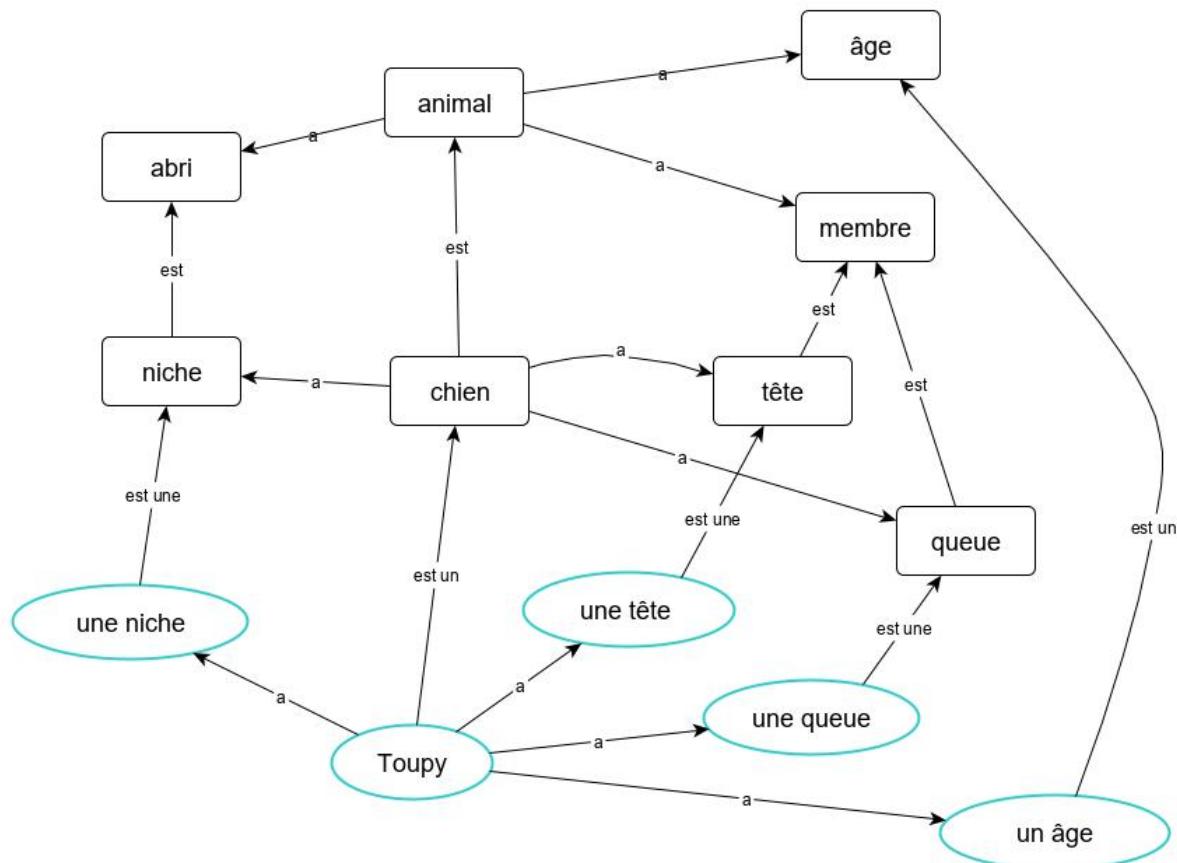
<sup>1</sup>ou "est une" ou "est un" pour le distinguer du prédicat "est" converse de la relation de subsomption

# Identification des êtres

Un être peut avoir un nom propre (comme Félix pour un chat), mais des êtres de même nom peuvent coexister dans la même situation individuelle, à condition de pouvoir être identifiés autrement que par leur nom.

Du fait du morphisme existant entre l'ensemble des êtres et celui des essences, l'identification d'un être peut aussi se faire en utilisant le vocabulaire défini dans l'ensemble des essences.

Ainsi, dans le graphe suivant, "tête de Toupy" identifie également sa tête, car il n'a qu'une seule tête, et "abri de Toupy" identifie sa niche.



Des êtres homonymes peuvent se différencier par leur manière d'être dans une situation qui leur est propre.

Ainsi, deux personnes de nom "Paul" peuvent se différencier par leur (nom de) famille, ou appartenir à la même famille mais avec des qualités distinctes, l'une étant par exemple le père et l'autre un fils (ce qui est assez courant).

Nous reviendrons dans la seconde partie de ce document sur la notion de manière d'être.

## Essence d'une essence

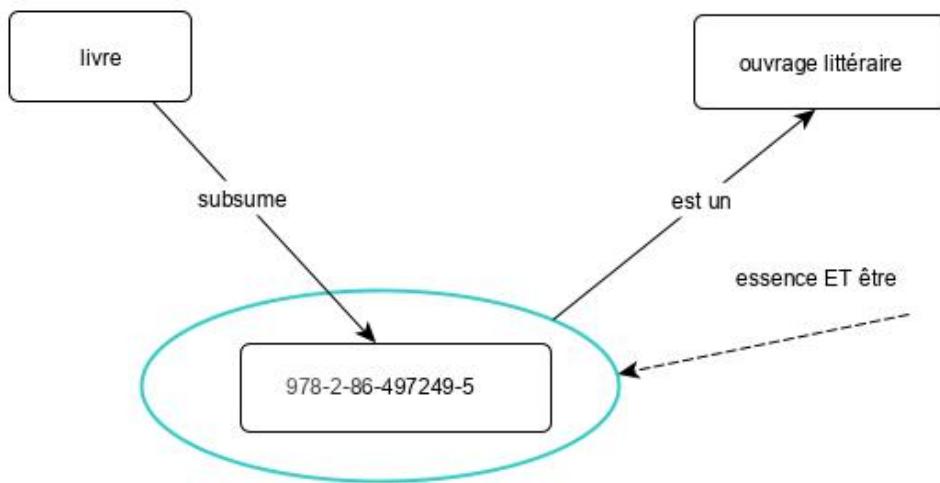
Beaucoup de philosophes et de logiciens se disputent encore sur l'existence ou non des universaux qui correspondent pour nous à des essences.

Dans ICEO, une essence est elle-même un être ayant sa propre essence (sa "méta essence")<sup>1</sup>.

En tant qu'être une essence possède une existence unique et immuable.

Insistons sur le fait qu'une essence n'a pas d'être : elle est un être.

Considérons l'exemple suivant :



L'essence "978-2-86-497249-5" est subsumée par l'essence livre; elle est aussi instance de l'essence ouvrage littéraire (sa méta-essence)<sup>1</sup>.

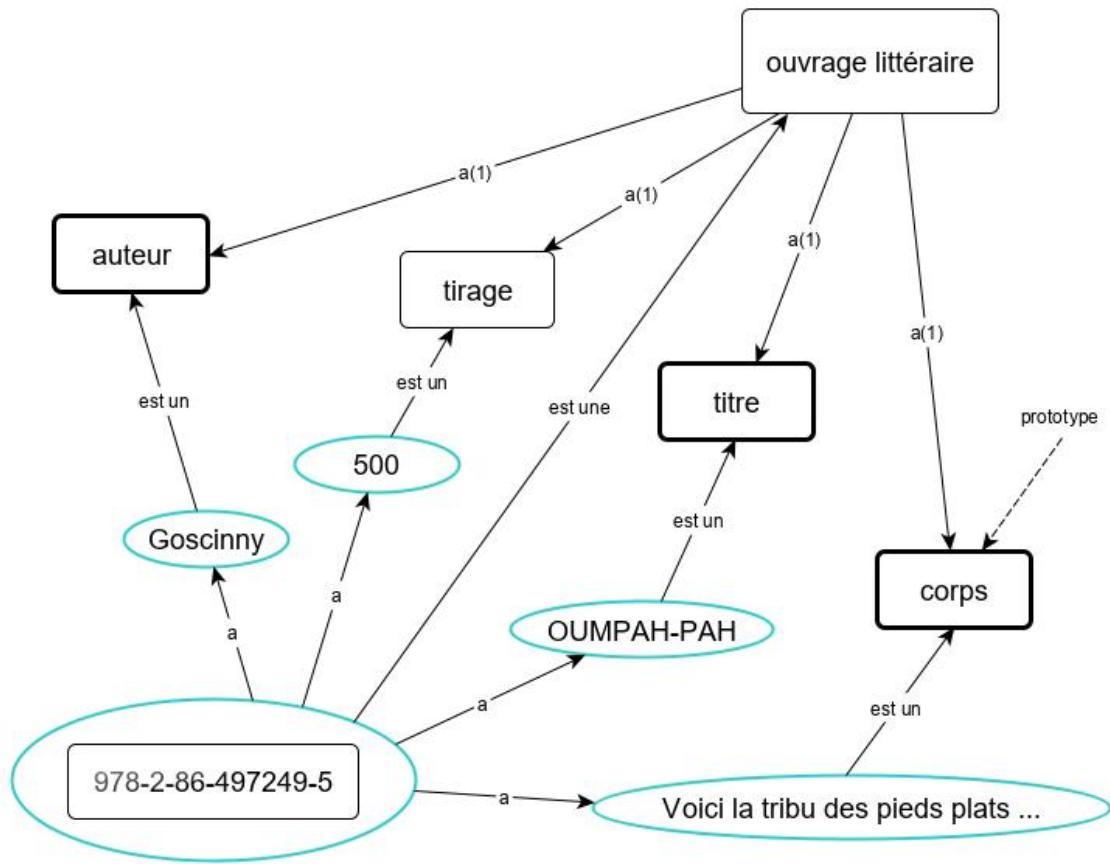
"978-2-86-497249-5" est la valeur d'un attribut ISBN (*International Standard Book Number*) défini au niveau de l'essence ouvrage littéraire, qui est moins ambigu que son titre.

Les essences ouvrage littéraire, livre et 978-2-86-497249-5 sont définies dans une situation générique appelée "littérature"<sup>2</sup>(non représentée).

Voici une représentation de l'essence ouvrage littéraire et de l'essence 978-286-497249-5 avec deux exemplaires :

<sup>1</sup>Ce qui correspond à la vision de langages informatiques comme Smalltalk où une classe est instance de sa métaclassse.

<sup>2</sup>L'essence livre définie en littérature ne doit bien sûr pas être confondue avec celle de livre en physique ou en finance.



Noter qu'en tant qu'être, les attributs d'une essence sont des êtres.

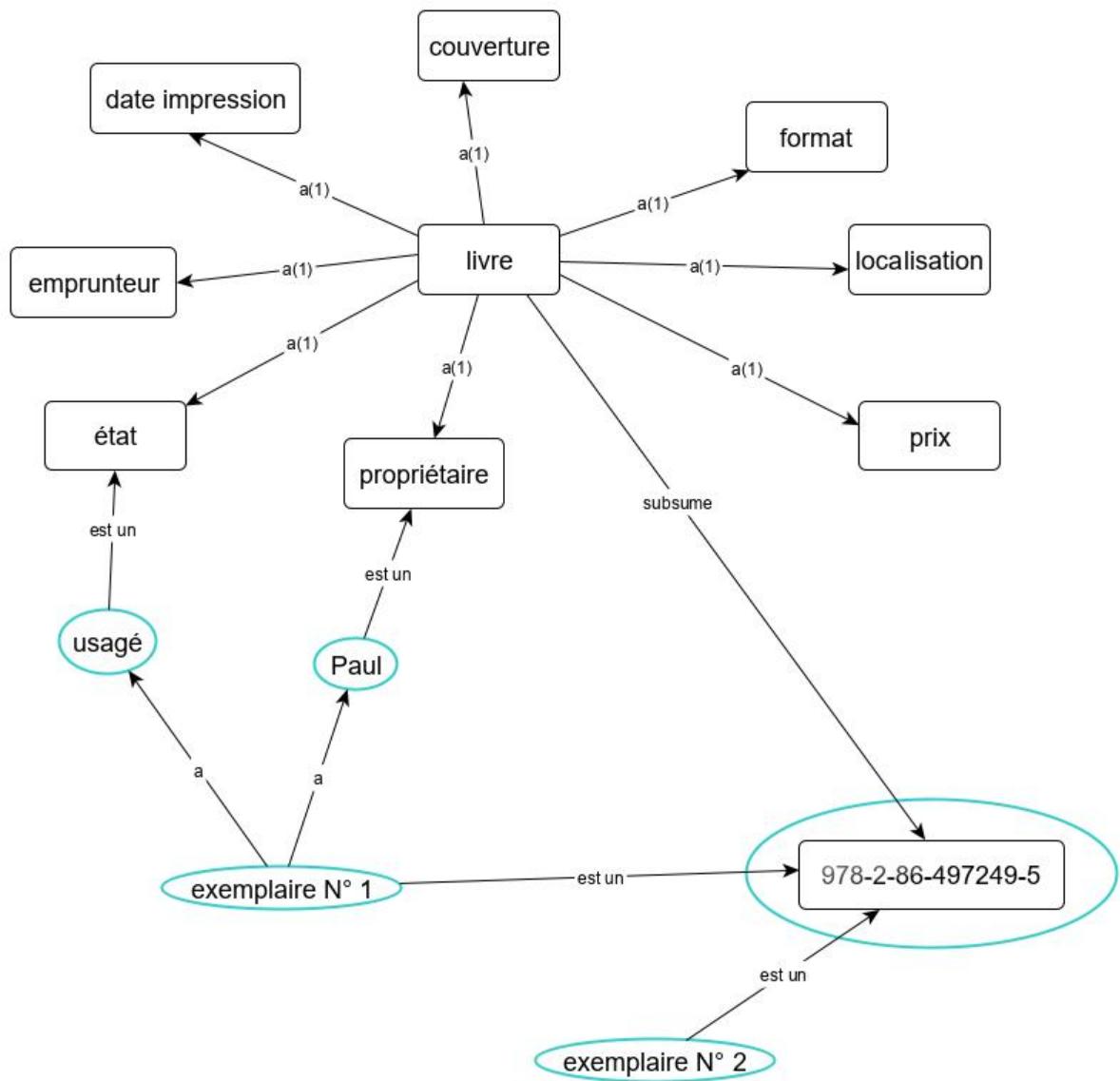
Nous qualifierons les attributs auteur, titre et corps de ouvrage littéraire de "prototypes"<sup>1</sup>, car leur valeur est partagée par toutes les instances de l'essence 978-2-86-497249-5. Ce sont ces attributs qui font l'originalité de l'ouvrage littéraire, qui seraient conservés par copie d'un exemplaire. Ce n'est pas le cas des attributs comme tirage.

Noter que nous aurions pu prendre le titre "OUMPAH-PAH" de l'ouvrage littéraire comme nom pour l'essence 978-2-86-497249-5, avec le risque d'avoir en littérature un autre ouvrage littéraire de même nom.

Les exemplaires instances de l'essence 978-2-86-497249-5 sont dans une situation individuelle appelée "bibliothèque" (non représentée), instance de la situation générique "littérature".

En tant qu'instance de l'essence livre (qui subsume l'essence 978-2-86-497249-5), chaque exemplaire a une date de création, un propriétaire, un état, ..., qui lui sont propres :

<sup>1</sup>Selon le dictionnaire, un prototype est une oeuvre ou un objet original que l'on imite ou reproduit.

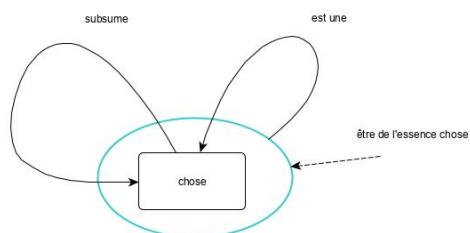


## Essence chose

Toute essence possède une mét-essence.

Par défaut, la mét-essence de toute essence est l'essence chose.

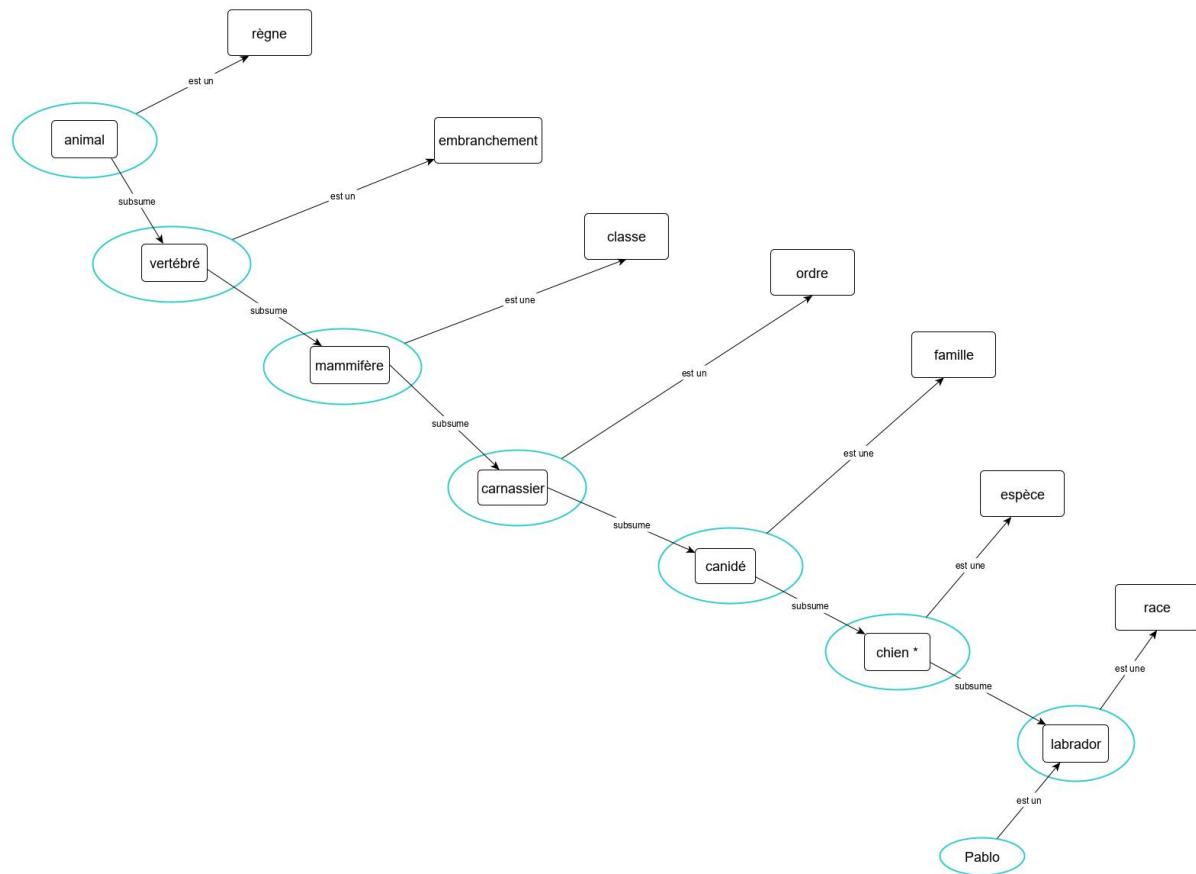
L'essence chose est définie dans l'absolu et son être est dans le monde. Elle est son propre genus et elle est instance d'elle-même (sa mét-essence est chose).



Le *differentia* de l'essence chose est vide.

Toute essence est subsumée directement ou indirectement par l'essence chose.

Voici un exemple emprunté à la zoologie, où la notion de mét-essence est utilisée pour classer des essences :



L'essence de Pablo est labrador, dont la mét-essence est race.

Pablo est un labrador, mais aussi un chien, un canidé, un carnassier, un mammifère, un vertébré et un animal.

## **Seconde partie : sur les relations entre les êtres et leurs manières d'être**



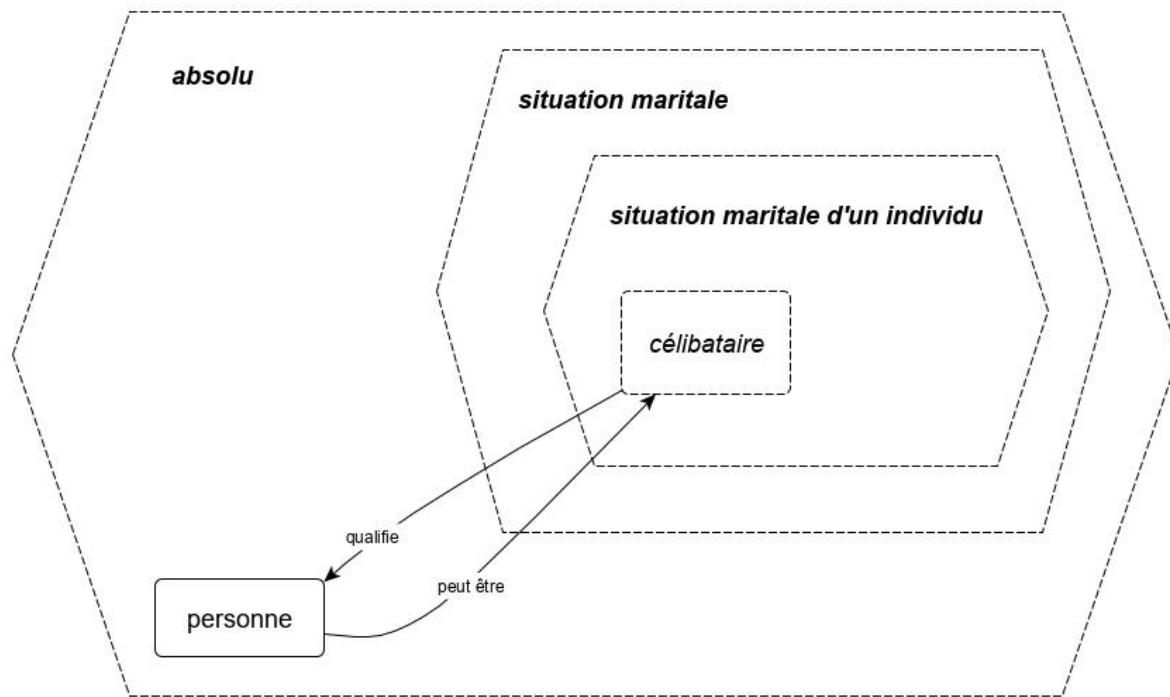
# Manière d'être

Une "manière d'être" ou "qualité"<sup>1</sup> est une essence qui qualifie une autre essence.

En tant qu'essence, une manière d'être est définie dans une situation générique.

Nous utiliserons le symbole graphique  pour représenter une manière d'être.

Exemple :



Dans cet exemple, célibataire est une manière d'être définie dans la situation générique "situation maritale d'un individu" définie sans "situation maritale" elle-même définie dans l'absolu.

Une manière d'être peut être liée à une autre manière d'être, mais n'est jamais un attribut de l'essence d'un être (ne rentre pas dans son differentia). Ainsi dans cet exemple, célibataire ne pourrait être un attribut de personne.

Une manière d'être "qualifie" une essence. La relation converse de "qualifie" est "peut être" ou "est" suivant le caractère accidentel ou essentiel de la manière d'être, que nous

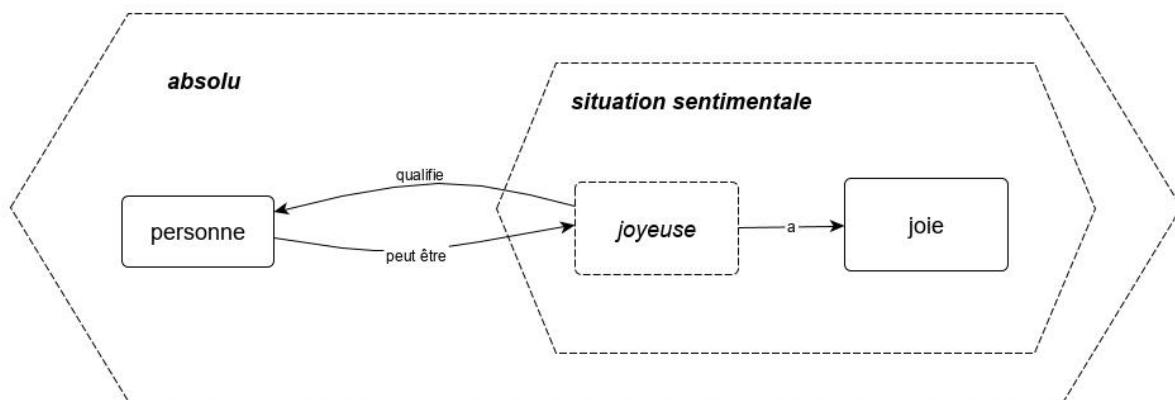
<sup>1</sup>Le mot "qualité" vient du latin "qualitas" qui signifie "manière d'être"

analyserons plus loin.

Il est important de noter qu'une manière d'être ne peut être définie sans préciser l'essence à laquelle elle fait référence (qu'elle qualifie).

Un attribut d'une manière d'être peut être une essence qui n'est pas une manière d'être.

Ainsi, considérons le graphe suivant :



L'essence joie est ici attribut de l'essence joyeux qui est une manière d'être. Nous pourrions exprimer de la même façon qu'être courageux implique d'avoir du courage, qu'être fatigué implique d'avoir de la fatigue, qu'être bleu implique d'avoir de la bleuité, ...

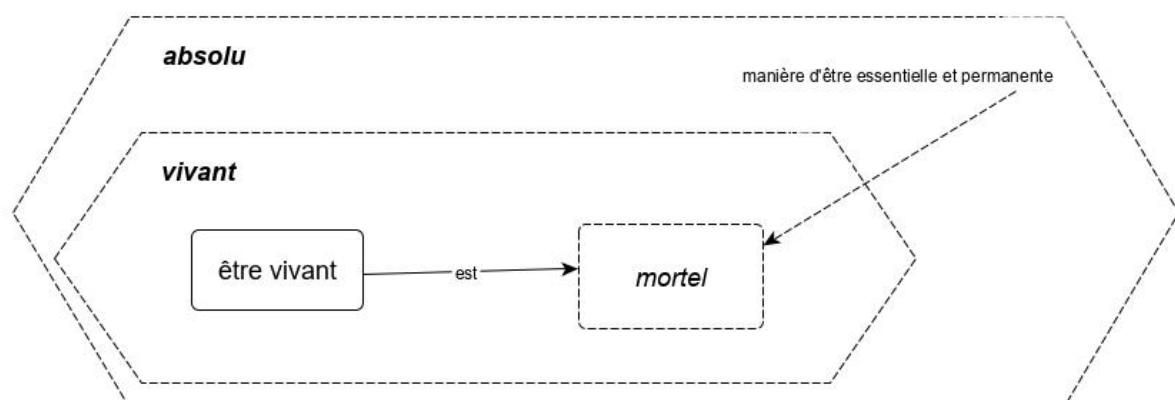
## Manières d'être essentielles ou accidentelles

Une manière d'être peut être essentielle ou accidentelle.

Une manière d'être essentielle rentre dans la définition d'une essence.

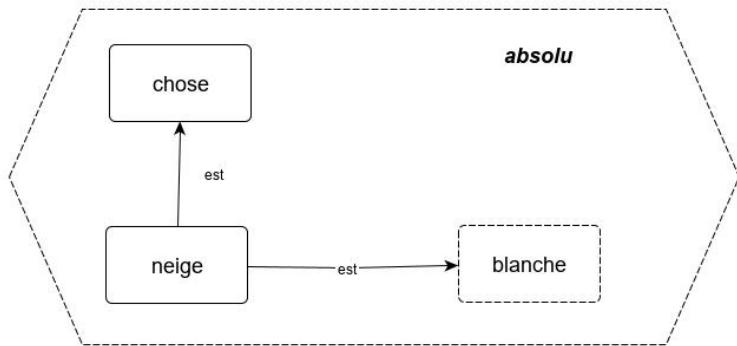
Ainsi en est-il d'être chanteur pour passereau, d'être mortel pour être vivant, ...

Ainsi, la manière d'être "mortel" est essentielle pour "être vivant" :



La relation qui lie une essence à une manière d'être essentielle sera dénotée par le mot "est".

Autre exemple : La manière d'être "blanche" est essentielle pour la neige :



Ce graphe est la représentation d'un exemple de George Boole dans "Les lois de la pensée" : "La neige est blanche est, pour les objectifs de la logique, équivalente à l'expression la neige est une chose blanche".

Contrairement à une manière d'être essentielle, une manière d'être accidentelle ne rentre pas dans la définition d'une essence.

Ainsi, pouvoir être marié ou célibataire ne sont pas des manières d'être essentielles pour personne.

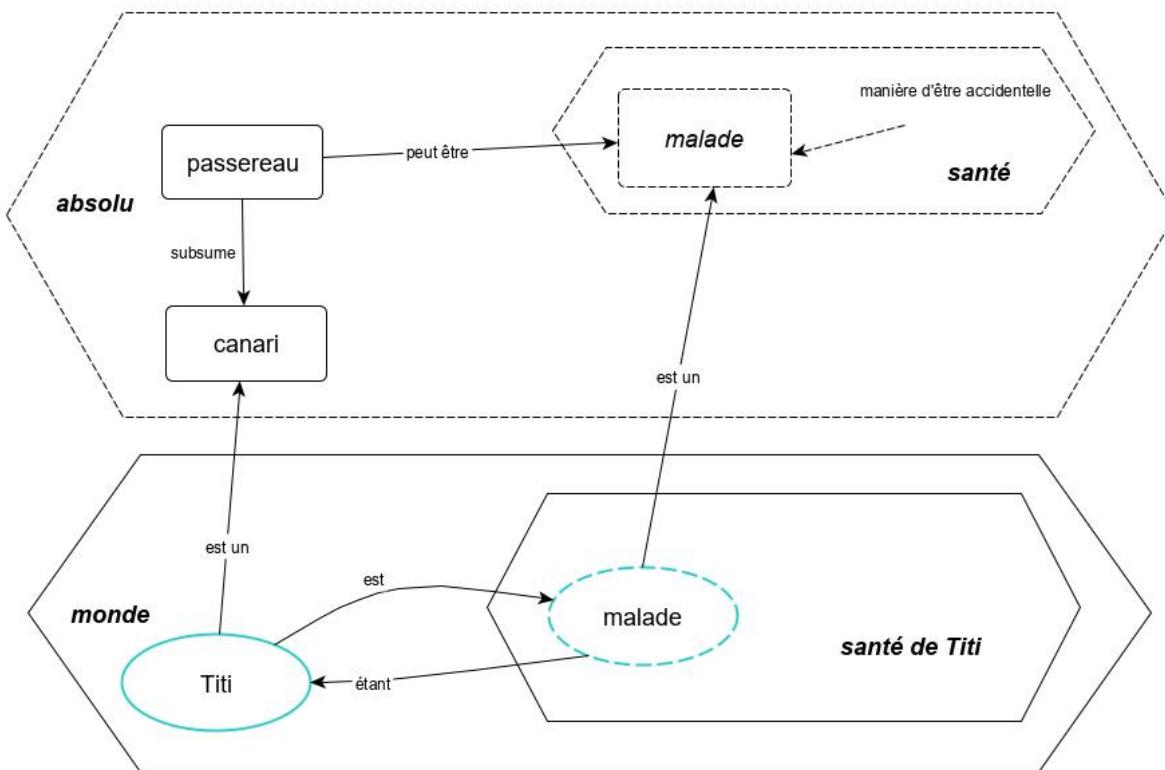
Une manière d'être accidentelle est définie dans une situation générique qui lui est propre.

Une essence hérite des manières d'être de son genus, que celles-ci soient essentielles ou accidentielles.

Ainsi, si passereau est chanteur par définition (il s'agit d'une manière d'être essentielle) alors canari est chanteur (car passereau subsume canari).

Cette règle s'applique également aux manières d'être accidentnelles.

Ainsi, si passereau peut-être malade, alors canari peut être malade.



Le symbole représente un état (ici l'état malade de Titi). La notion d'état est présentée un peu plus loin.

Les essences indénombrables (qualifiées par certains de substances "massières") comme l'eau, la fortune, ... , ont une manière d'être qui est leur quantité. Celle-ci est déterminée à l'aide d'une grandeur mesurable à laquelle est associée une unité, comme la longueur, le volume, l'énergie, la masse, ...

Une essence comme chat qui est de nature nombrière (dénombrable) peut abusivement être considérée comme une substance massière si on la confond avec la substance "viande de chat".

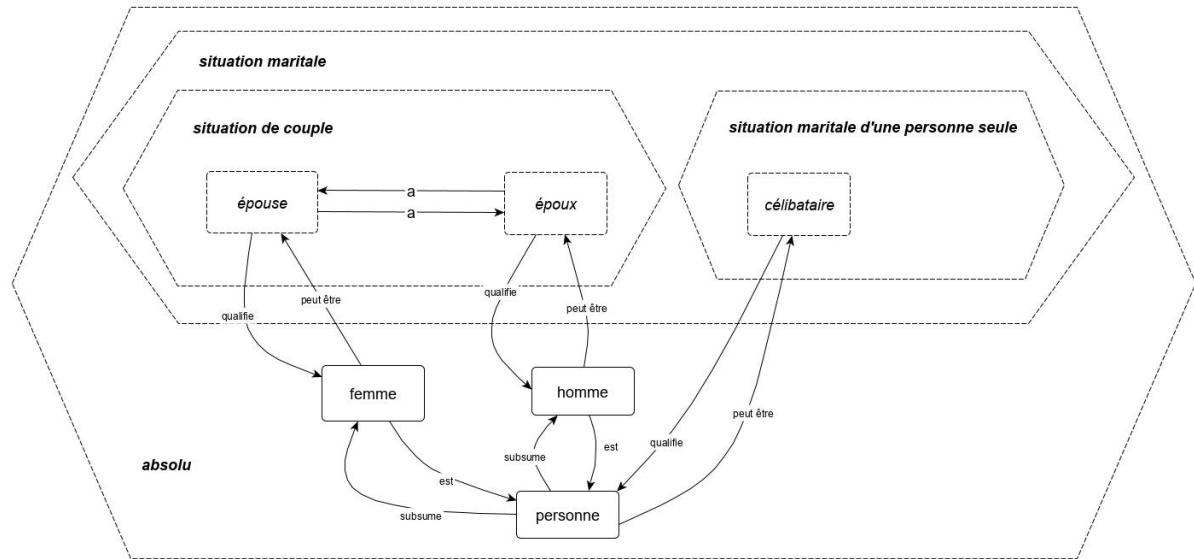
En ce sens, "du chat" + "du chat" donne "du chat" en plus grande quantité, et "du chat" peut avoir du goût (il paraît même que c'est meilleur que "du lapin" !).

# Relations entre manières d'être

Une manière d'être peut être en relation avec une autre manière définie dans la même situation générérique.

Les relations définies entre les manières d'être sont généralement bidirectionnelles (corrélatives).

Le graphe suivant définit une relation entre les manières d'être accidentuelles épouse et époux définies dans la situation de couple :

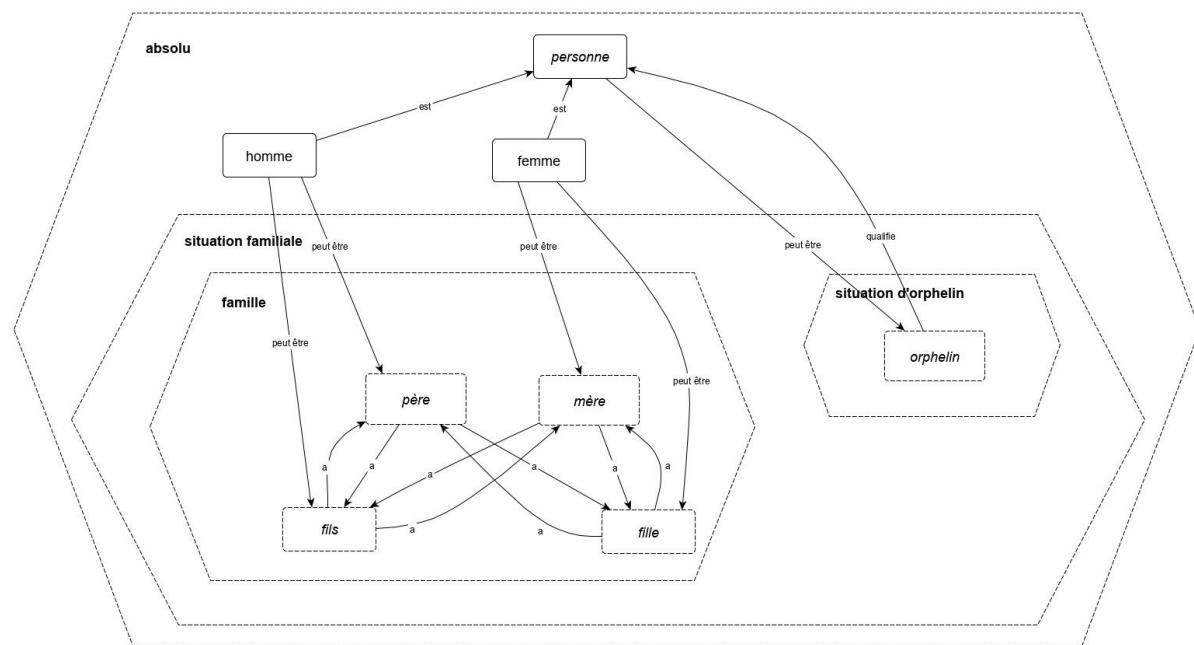


Noter qu'homme et femme ne sont pas ici attributs l'un de l'autre dans l'absolu. Ils sont liés par leur situation de couple.

Dire qu'une femme a un homme pour époux si elle est épouse est un raccourci de langage, car un homme époux n'est pas un attribut de "sa" femme.

Le mot "a" n'est pas utilisé ici pour exprimer une relation de composition, mais une association.

Voici un autre exemple portant sur la situation de famille :



mère et fille sont des manières d'être pour femme, et père et fils sont des manières d'être pour homme.

La manière d'être fille est liée à la manière d'être mère et réciproquement, et la manière d'être fils est liée à la manière d'être père et réciproquement.

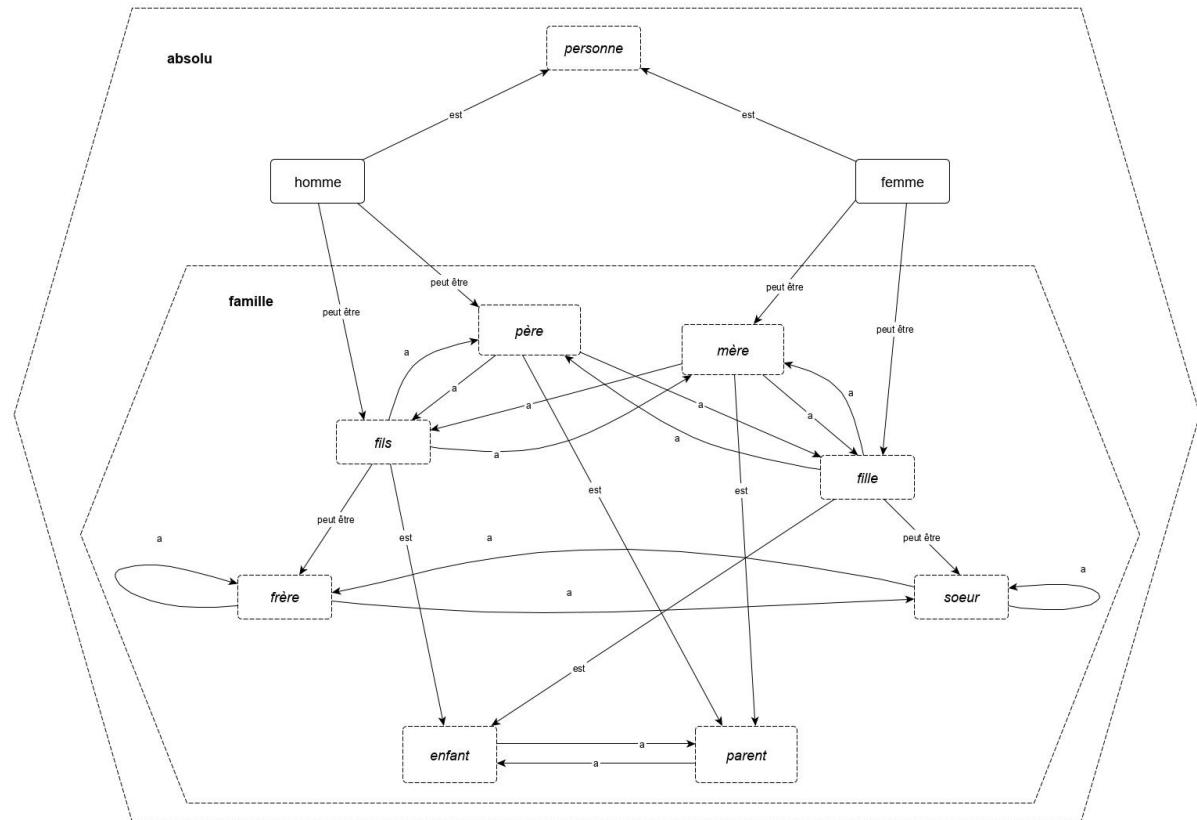
La manière d'être orphelin n'est pas définie dans la situation générique de famille mais dans celle d'orphelin.

# Subsommation des manières d'être

En tant qu'essence, une manière d'être ne peut avoir plusieurs genus mais peut subsumer d'autres essences.

Ainsi, dans la situation générique "famille", la manière d'être "parent" subsume les manières d'être "mère" et "père", et la manière d'être "enfant" subsume les manières d'être "fille" et "fils".

Représentation graphique :



Une essence hérite des manières d'être de son genus qui n'ont pas été spécialisées.

Un être ne peut subsumer une manière d'être ou être subsumé par une manière d'être, et réciproquement.

Les manières d'être forment une hiérarchie indépendante de la hiérarchie des essences.

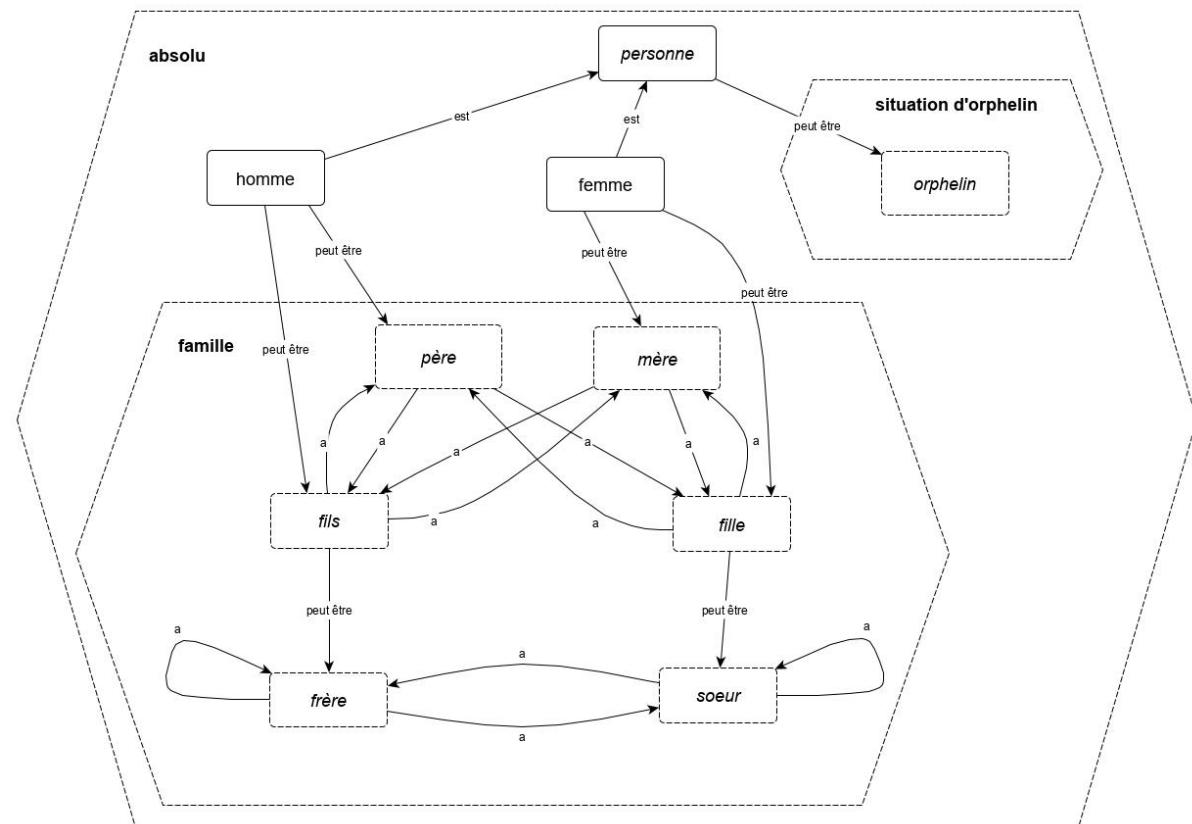
La seule manière d'être essentielle possible de l'essence chose étant d'être elle-même, nous conviendrons que la manière d'être chose est également la racine de la hiérarchie des manières d'être.

Noter que si une essence ne peut être attribut d'elle-même, rien n'empêche une manière d'être d'être attribut d'elle-même (ainsi dans cet exemple soeur est attribut d'elle-même).

# Manière d'être d'une manière d'être

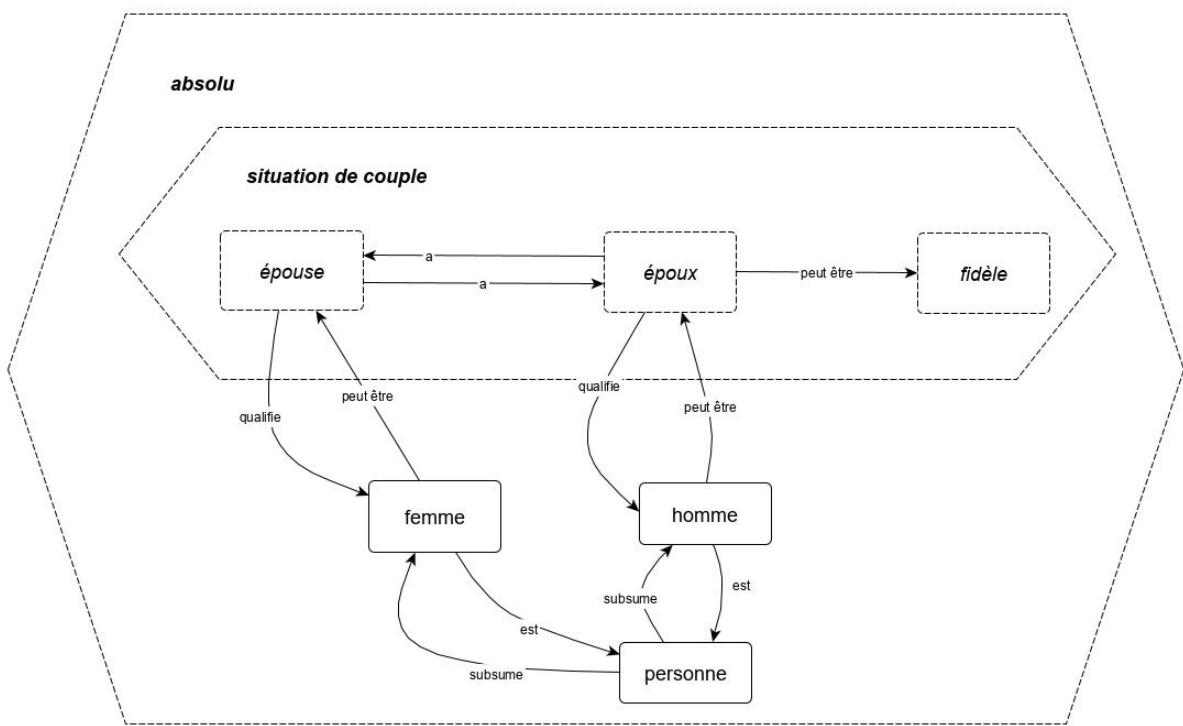
Une manière d'être peut être celle d'une autre manière d'être.

Ainsi par exemple, dans le graphe suivant soeur est une manière d'être de la manière d'être fille, et frère est une manière d'être fils :



Dans cet exemple frère qualifie homme en tant que fils et soeur qualifie femme en tant que fille.

Dans cet autre exemple homme peut être fidèle en tant qu'époux :





# Etat

Un état est une instance de manière d'être d'un autre être qui est son étant. Il est inclus dans une situation individuelle qui est instance de la situation générique de la manière d'être dont il est instance.

Un état peut être un attribut d'un autre état, mais n'est jamais un attribut d'un être.

Habituellement, un état porte par défaut le même nom que son essence. Ainsi, époux est par défaut le nom de tout état instance de la manière d'être époux. Ceci peut être une source d'ambiguïté.

Un état d'un être lui est propre. Ainsi, si deux hommes sont époux, leur état époux est différent.

L'un peut être un époux heureux et l'autre moins.

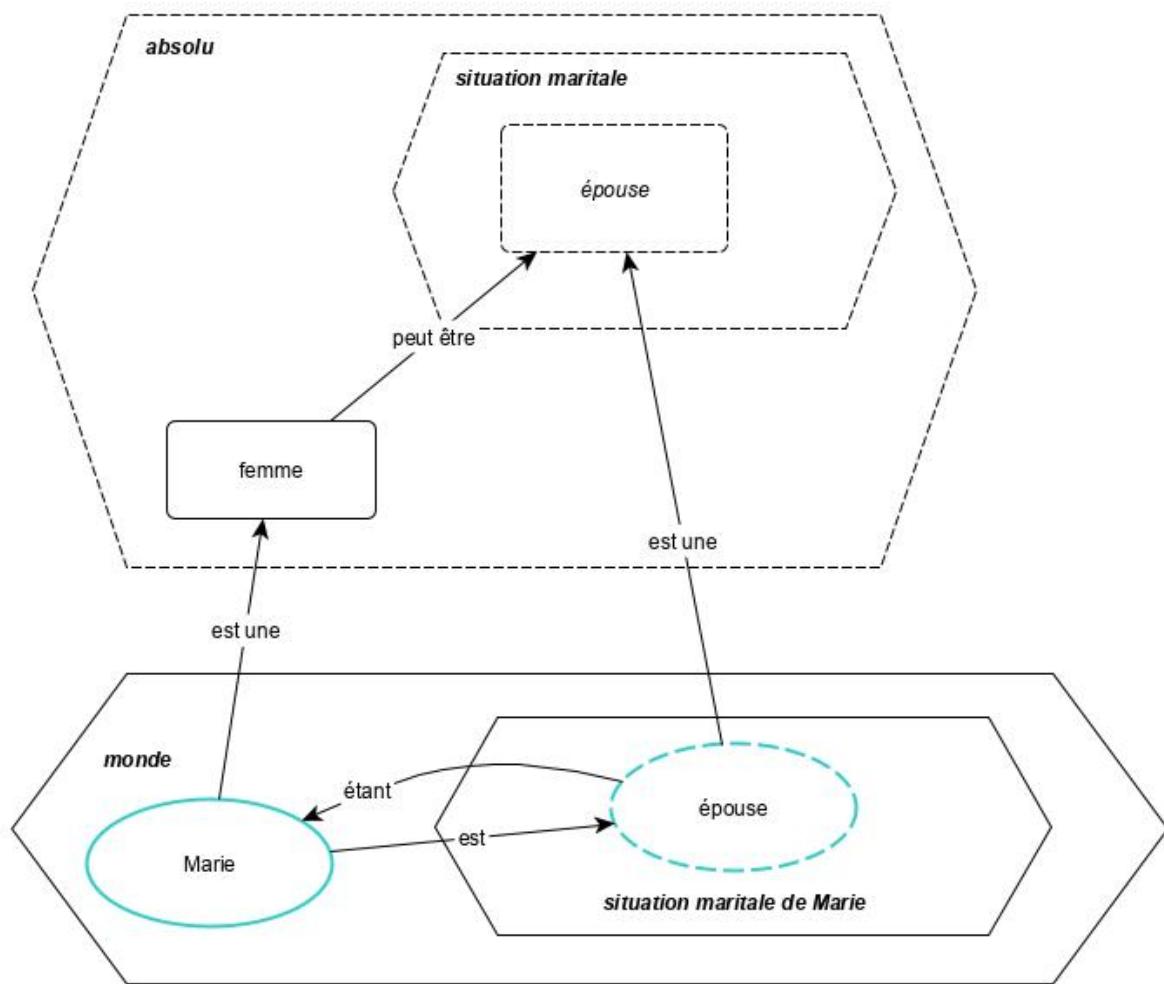
Un état se situe dans un espace spatio-temporel. Le temps d'un état est inclus dans celui de son étant.

L'incompatibilité entre certaines manières d'être se traduit par l'incompatibilité des états instances de ces manières d'être.

Ainsi, les manières d'être célibataire et époux pour l'essence homme sont incompatibles; il en résulte qu'une instance d'homme ne peut être à la fois dans les états célibataire et époux.

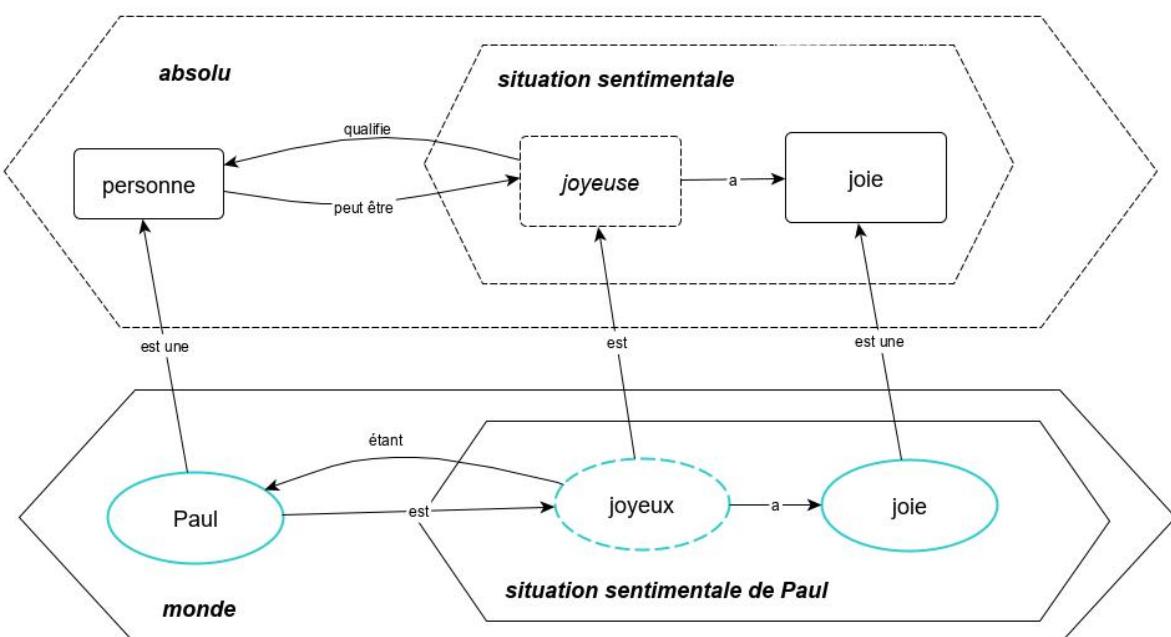
Nous utiliserons le symbole graphique  pour représenter un état.

Le graphe suivant exprime que Marie est dans l'état épouse :



Un attribut d'un état peut être un être.

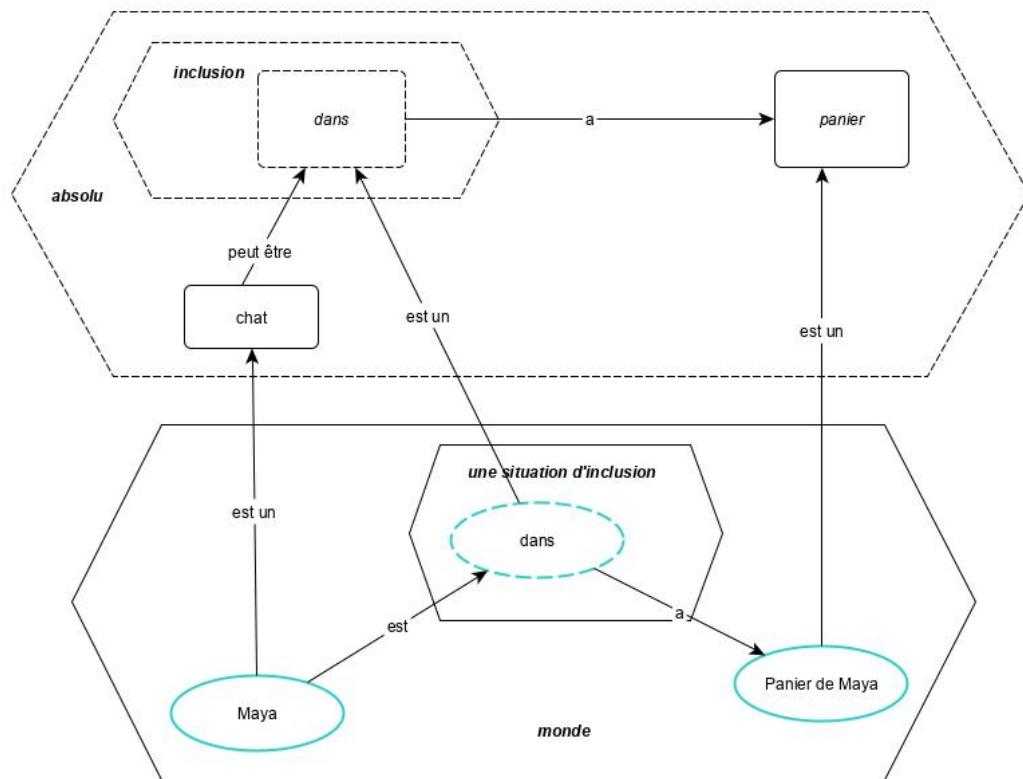
Considérons le graphe suivant :



L'être joie est ici attribut de l'état joyeux.

Paul étant joyeux, il éprouve une joie qui lui est propre, plus ou moins intense.

Considérons le graphe suivant :



Il exprime le fait que Maya est "dans" son panier.

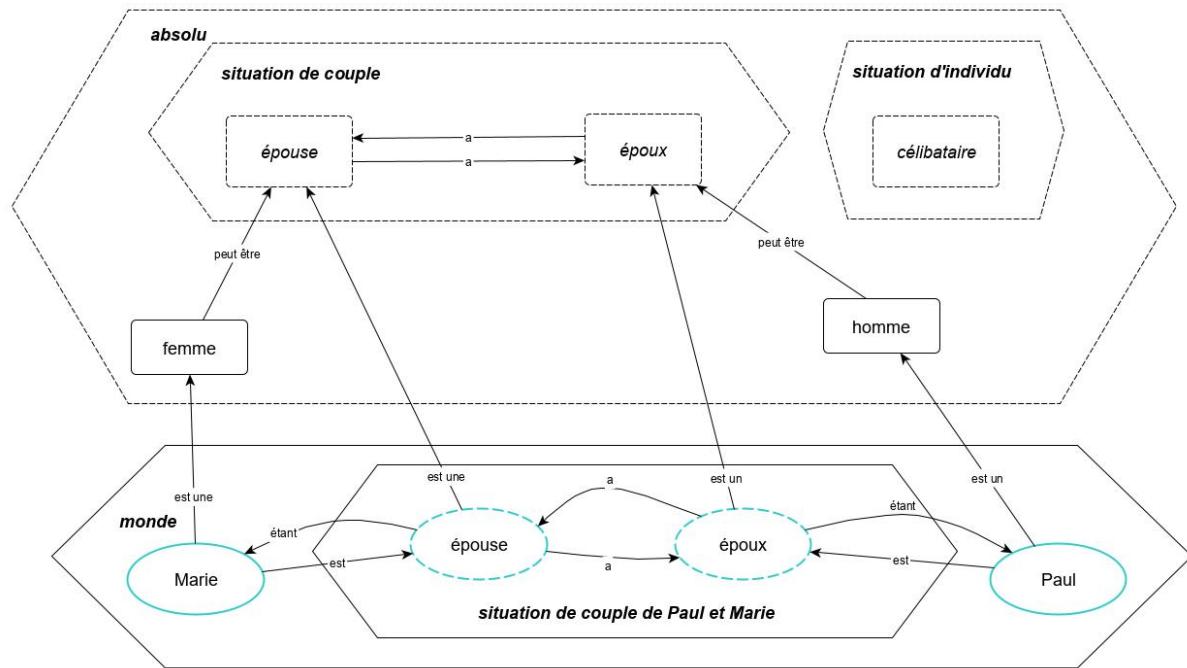
Noter que beaucoup de prépositions (sur, avec, de, ...) pourraient être représentées de la même manière.

# Relations entre états

Un état peut ou non induire une relation entre différents êtres.

Les relations entre états sont binaires, et sont généralement bidirectionnelles (corrélatives).

Considérons le graphe suivant :



Pierre a "acquis" Marie comme épouse en se mariant. Cet attribut lui est apporté par son état d'époux.

Dans cet exemple Marie et Paul ne sont pas attributs l'un de l'autre mais sont liés par l'intermédiaire de leur état. Du fait de leur situation (relation), ils forment un couple dans le monde. Dire que Marie a Paul pour époux est un raccourci de langage.

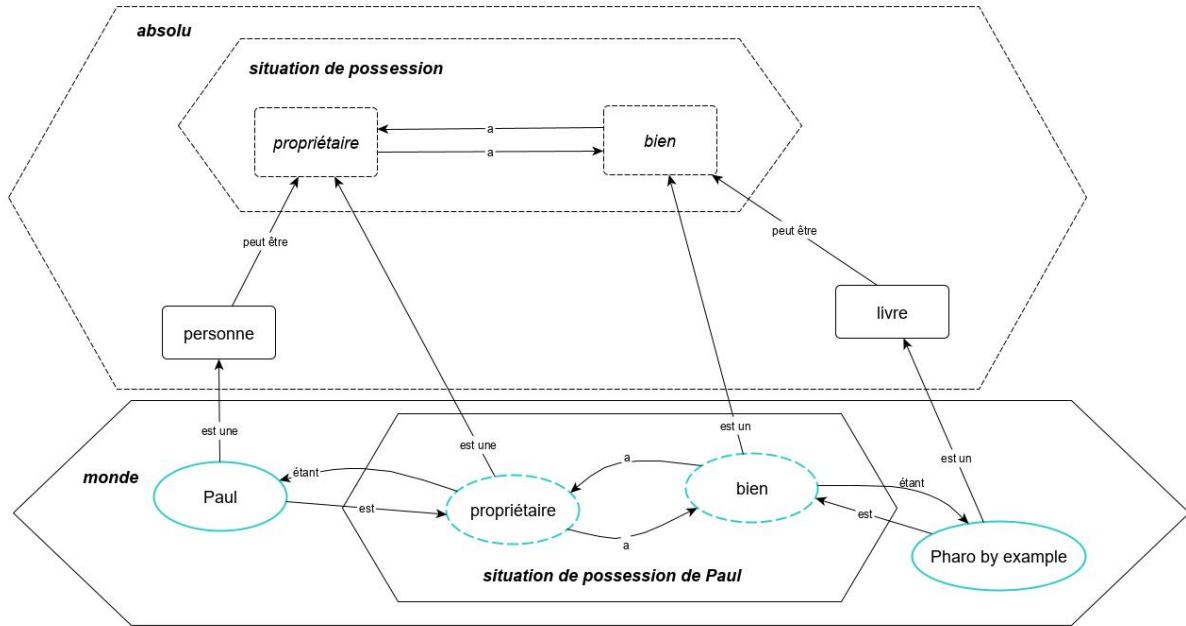
Si Paul était homosexuel, il serait lié à un autre homme par son état d'époux.

Les étants de deux états liés sont forcément distincts, même s'ils sont instances de la même manière d'être. Ainsi, une soeur peut avoir une soeur, mais une fille qui est soeur ne peut être soeur d'elle-même.

Si Paul était polygame, ceci impliquerait qu'il ait un état d'époux distinct par rapport à chacune de ses épouses. Pour l'une, il pourrait être un bon époux tout en étant un mauvais époux pour l'autre.

## Relation entre états différente de la composition

L'exemple suivant illustre la relation possible entre deux êtres qui ne soit pas une relation de composition :



On voit que l'attribut 'Pharo by example' de Paul lui est conféré par son état de propriétaire.

Le fait de dire que le propriétaire d'un livre est une personne est un raccourci de langage.

Bien sûr, un livre ne rentre pas dans la composition d'une personne, et réciproquement !

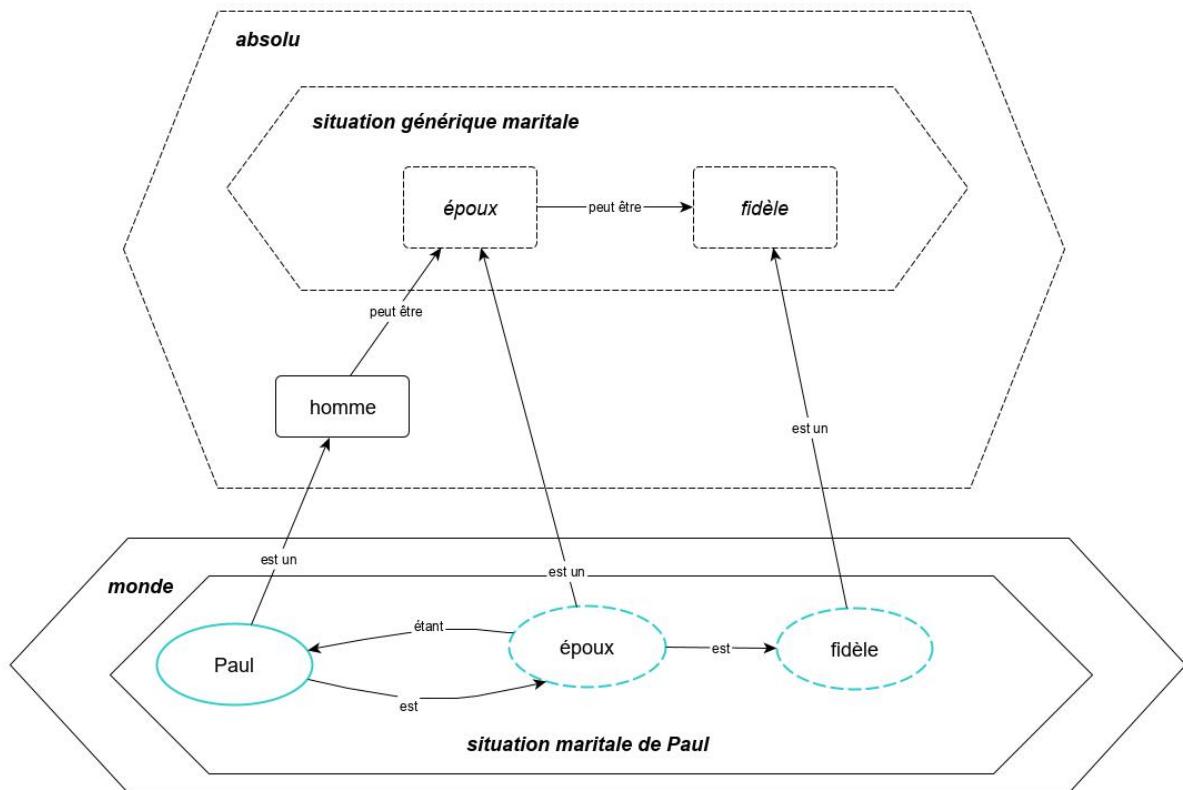
La même logique s'appliquerait pour dire que Paul a un crayon, une veste, ...

## Etat dans un état

Une manière d'être peut être celle d'une autre manière d'être, et un état peut donc être celui d'un autre état.

Par principe, un état d'un état sera supposé concerner le même être (le même étant).

Voici la représentation de l'assertion "Paul est un époux fidèle" :



Ce graphe n'affirme pas que Paul est toujours fidèle, mais qu'il est fidèle en tant qu'époux.

# Effectivité d'une manière d'être

Une manière d'être peut être permanente ou intermittente.

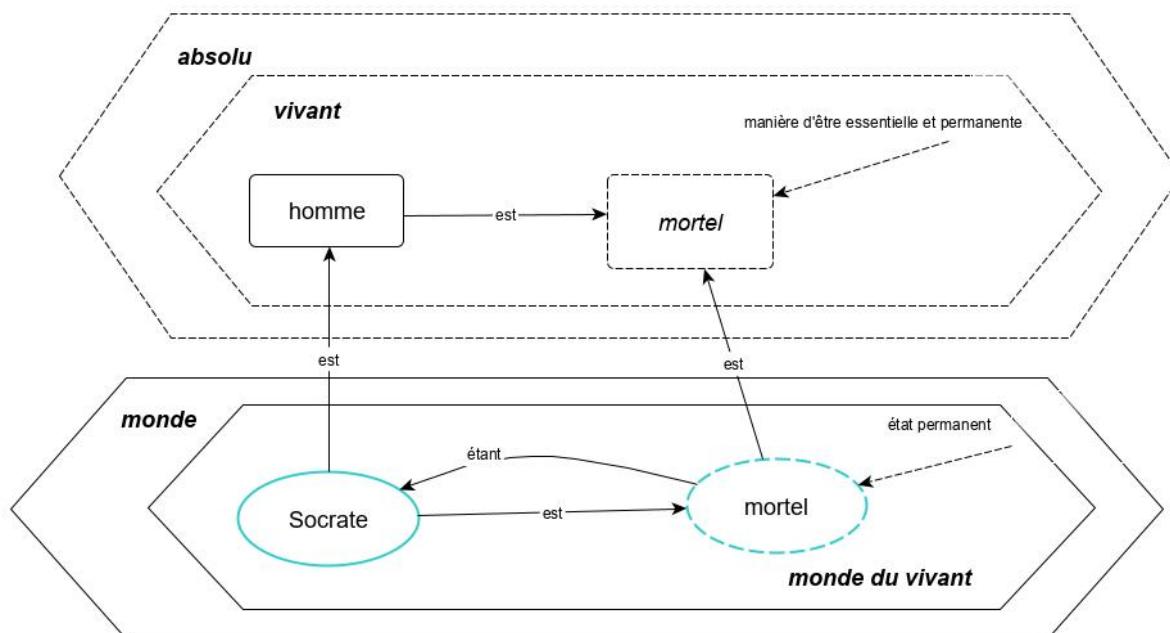
Le temps d'existence d'un état, qui est inclus au sens large dans celui de son étant, dépend de l'effectivité de la manière d'être dont il est instance.

Ainsi, par exemple :

- la manière d'être "chanteur" (ou être chantant) de passereau est essentielle mais intermittente : un passereau ne chante pas en permanence.
- la manière d'être "mortel" de homme est essentielle et permanente : un homme est mortel toute sa vie, de sa naissance jusqu'à sa mort.
- la manière d'être "blanc" de homme est accidentelle et permanente : un homme qui est né blanc le sera toute sa vie.
- la manière d'être "malade" d'être vivant est accidentelle et intermittente : un être vivant n'est pas forcément malade en permanence.

Un état est permanent (intermittent) s'il est instance d'une manière d'être dont l'effectivité est permanente (intermittente).

Considérons le graphe suivant :

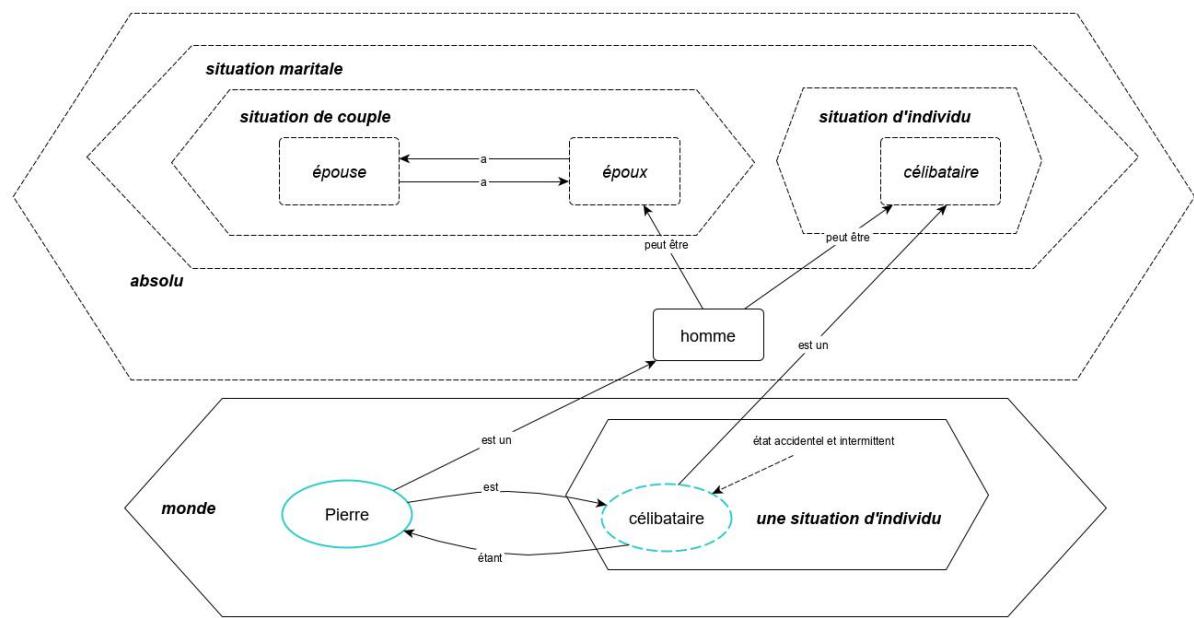


Ce graphe illustre le célèbre syllogisme d'Aristote donné dans l'Organon : "tout homme est mortel, et comme Socrate est un homme, il est mortel".

Noter que ce syllogisme fait abstraction de la notion de temps. Ceci n'est compréhensible que parce que la manière d'être mortel de l'essence homme est permanente.

Autre exemple : l'état célibataire pour un homme correspond à une manière d'être

accidentelle et intermittente.



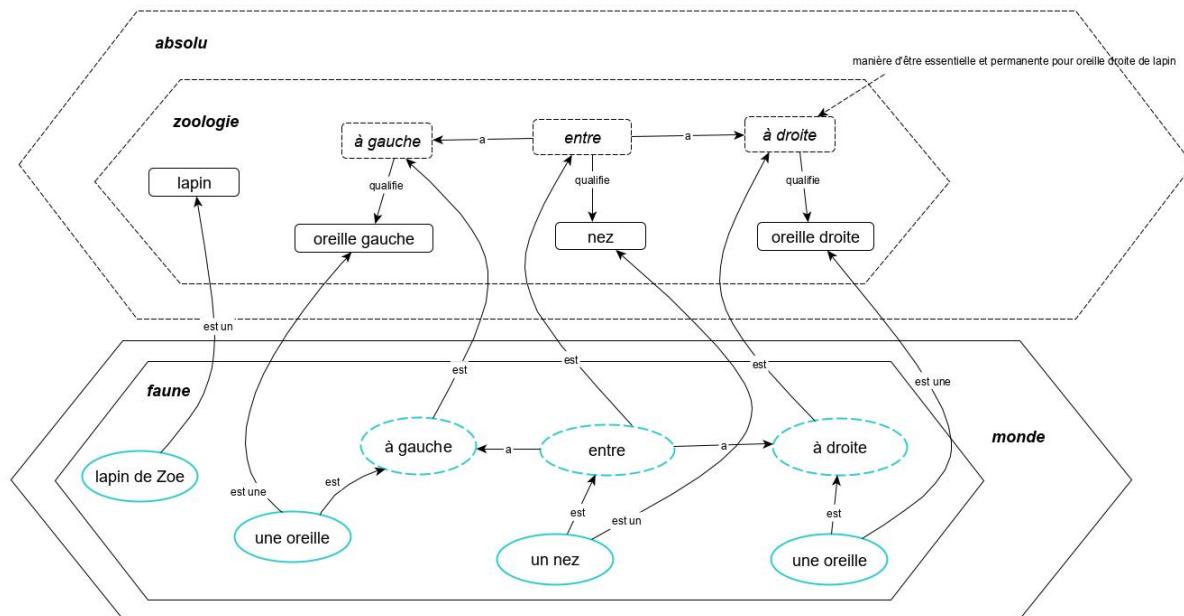
Pierre, en tant qu'homme, est dans l'état célibataire, mais la manière d'être célibataire n'est ni essentielle ni permanente dans la définition de l'essence homme.

# Contraintes structurelles internes

Les contraintes structurelles au sein d'une essence sont des manières d'être essentielles et permanentes imposées à ses attributs. Elles peuvent être des contraintes de cardinalité, d'identité, de position relative, de longueur, de mesure, ..., des attributs.

Ainsi par exemple, les côtés adjacents d'un polygone ont une extrémité en commun.

Le graphe suivant exprime que le nez de lapin doit être "entre" ses deux oreilles :



Les manières d'être "entre", "à gauche" et "à droite" des essences "nez", "oreille gauche" et "oreille droite" de lapin sont essentielles et permanentes. Elles définissent une contrainte structurelle interne de l'essence lapin (sont définies dans la structure de lapin).

Une contrainte structurelle peut permettre de définir une essence à partir d'une essence existante (son genus). Ainsi par exemple, c'est une contrainte structurelle de cardinalité qui permet de définir l'essence quadrilatère à partir de l'essence polygone, en restreignant le nombre de ses côtés à 4. Les contraintes au sein d'une essence peuvent s'appliquer à des manières d'être, par exemple des contraintes de dimension<sup>1</sup> Ainsi, les côtés d'un triangle équilatéral ont la même longueur.

<sup>1</sup>La subsomption exige le respect ou la spécialisation des contraintes. Ainsi par exemple, si l'envergure maximale d'un avion est actuellement de 97,54 m, celle de toute essence subsumée par avion est inférieure ou égale à cette valeur.

# Contraintes externes sur les manières d'être

L'environnement peut exercer des contraintes sur les manières d'être accidentelles d'un être.

Dans le langage courant, il est question de degrés de liberté d'un être.

Pour illustrer ce point, prenons l'exemple de la couleur de la neige.

- dans l'absolu, la neige est blanche. Il s'agit de sa couleur dite "objective", liée aux propriétés physiques de réflexion lumineuse de la neige, qui ne dépend pas de l'éclairage et des capacités de vision du sujet.
- sa couleur apparente dépend de l'éclairage.
- sa couleur subjective dépend des capacités de vision du sujet.

Supposez que le sujet perçoive la neige éclairée par un beau soleil. Elle lui paraît blanche. C'est, par définition, sa couleur objective.

Supposez que le sujet perçoive la neige éclairée par de la lumière rouge. Il verra la neige rouge. Il s'agit d'une couleur apparente.

Supposez que le sujet porte des lunettes qui filtrent la lumière bleue. Il ne verra plus la neige blanche, mais jaune. Il s'agit d'une couleur subjective.

Ces manières d'être sont évidemment liées. Ainsi, un objet de couleur objective magenta (mélange de lumières bleu et rouge) ne pourra jamais apparaître vert ou jaune (mélange de lumières verte et rouge).

Ce qui vient d'être dit pour la couleur d'un être peut s'appliquer à beaucoup de manières d'être, en particulier dans le monde physique.

En informatique, la notion de contrainte est à la base d'un paradigme de programmation<sup>1</sup> permettant de maîtriser la complexité combinatoire de certains problèmes.

Noter que les manières d'être possibles d'un être peuvent aussi être augmentées par sa relation avec d'autres êtres.

C'est ainsi que par exemple une personne peut voler dans un avion.

---

<sup>1</sup>cf. l'ouvrage "Programmation par contraintes" d'Annick Fron – Eyrolles 1994

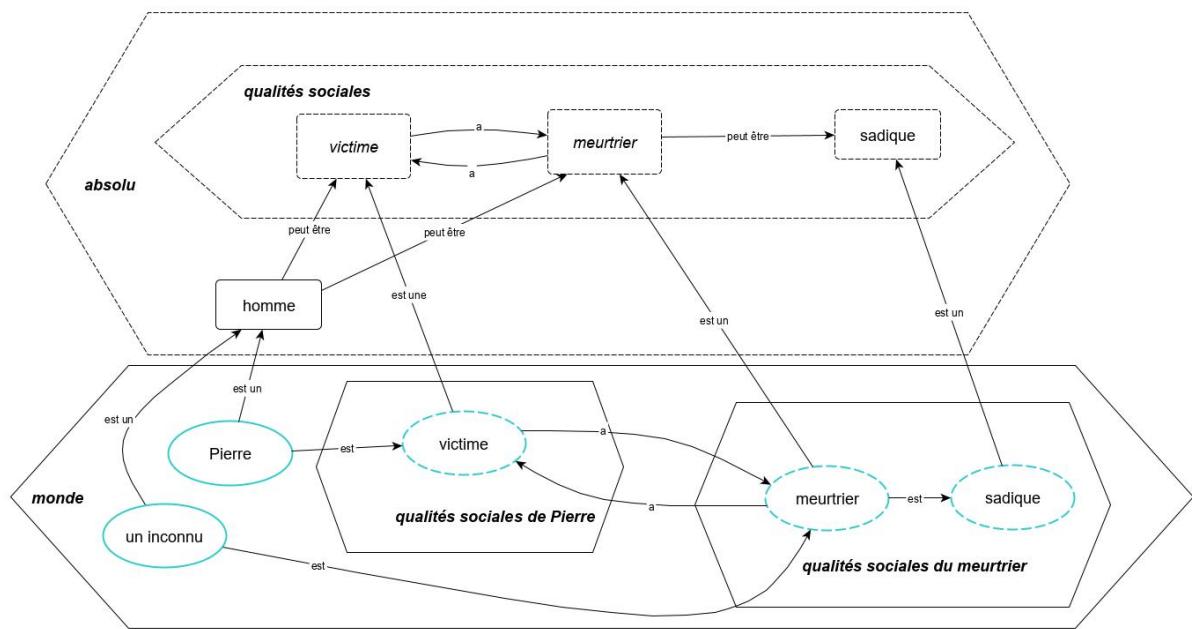
# Êtres inconnus

Considérons 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.

On peut la représenter par le graphe :



L'essence du meurtrier inconnu est ici supposée être l'essence homme. 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 serait simplement identifié comme étant quelque chose.

La recherche de l'identité du meurtrier peut se faire en exploitant les informations connues sur la victime (en particulier ses manières d'être). Si la victime était l'amant d'une femme mariée, la suspicion peut se porter sur le mari de celle-ci.

S'il advenait que cet individu soit identifié, par exemple Paul, alors toutes les informations connues sur le meurtrier seraient attribuées à Paul.



# Actions

Une action est ce qui produit un effet.

Il peut s'agir pour l'acteur de réaliser une fin, ou d'accomplir une action qui est à elle-même sa fin.

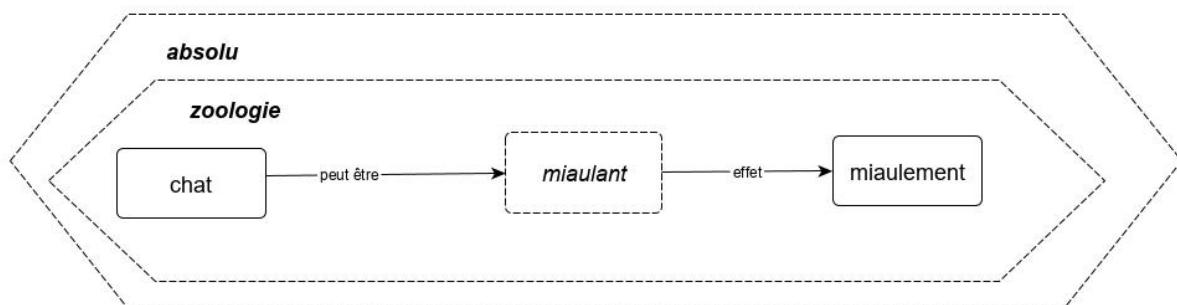
Cette faculté d'agir se transmet lorsqu'un acteur crée un autre acteur, par exemple par l'invention de machines dotées de capacités d'action. Elle se transmet également entre des êtres existants par la communication de savoirs faire.

Toute action peut avoir pour effet la création d'un nouvel être ou d'un nouvel état, ou la destruction d'un être. La destruction d'un être entraîne la destruction de tous ses états et donc de ses relations avec d'autres êtres.

Une action générique correspond à une manière d'être d'une essence, tandis qu'une action individuelle correspond à un état d'un être.

Nous dénoterons une action générique par un adjectif verbal ou un verbe à l'infinitif.

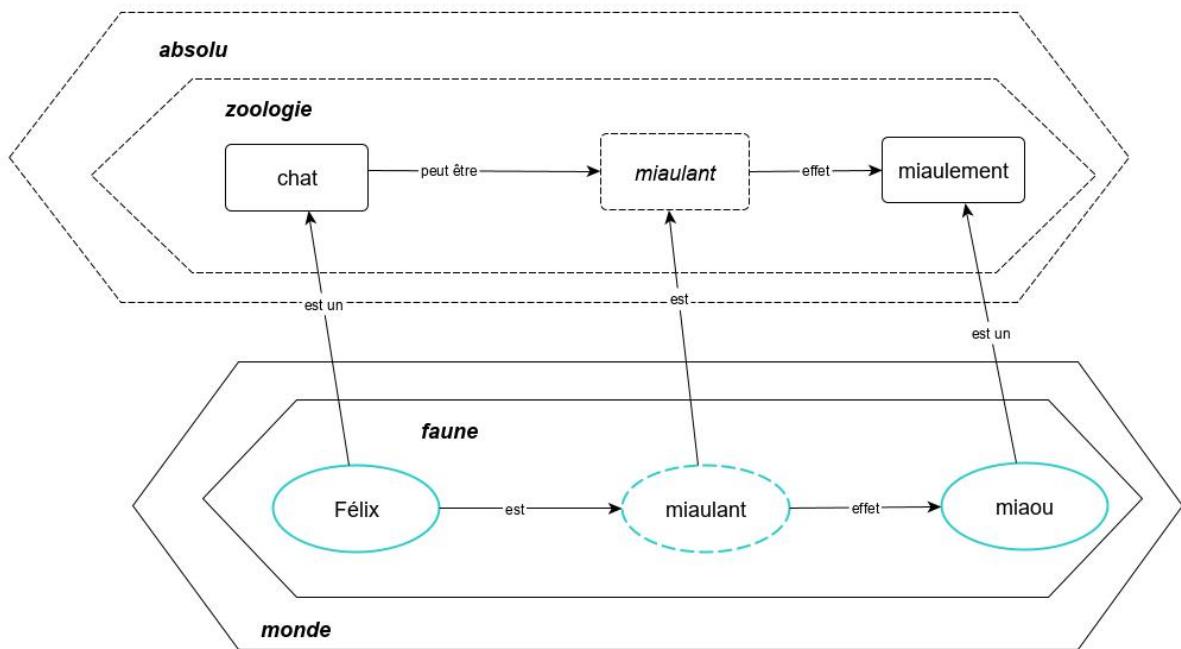
Exemple :



Dans l'expression "peut être miaulant", "miaulant" est un adjectif verbal qui dénote en français une action non contrainte par le temps (ce n'est pas un participe présent). L'expression équivalente "peut miauler" est plus familière.

Si Félix est en train de miauler, nous exprimerons cette action par "Félix est miaulant"<sup>1</sup>.

<sup>1</sup>Ce qui correspond à l'utilisation de la forme progressive en anglais : "*Félix is mewing*". Cette expression s'exprimerait en français par "Félix est en train de miauler" ou "Félix miaule"



L'effet de miauler est un miaulement, qui pour Félix consiste à émettre le cri "miaou".

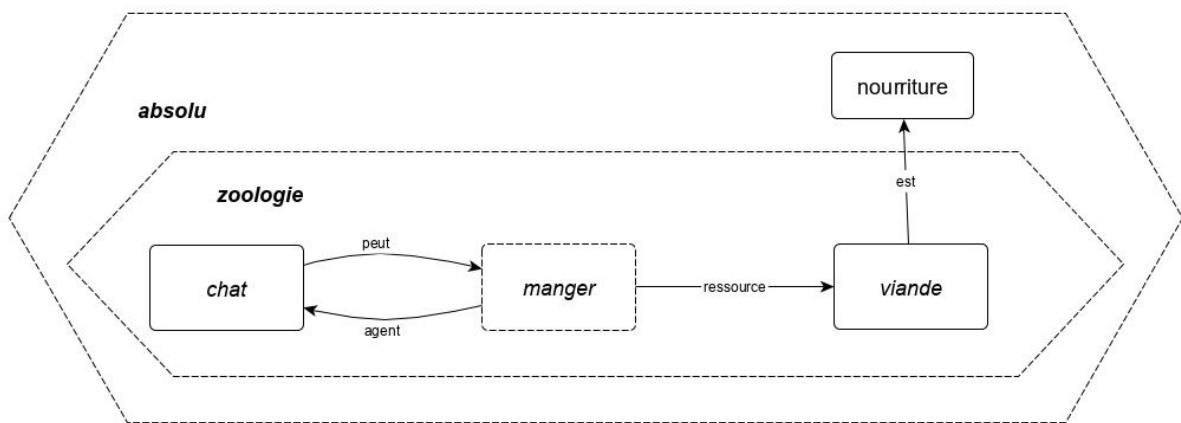
La situation de définition de miaulant est l'essence chat car il s'agit d'une manière d'être essentielle.

En tant que manière d'être, une action générique peut en subsumer d'autres.

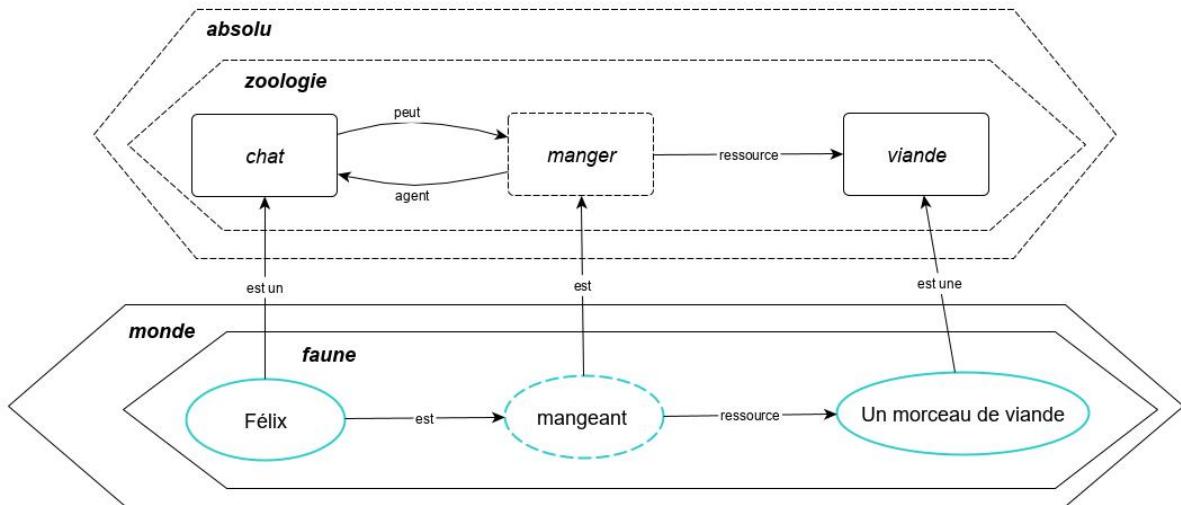
Ainsi, les actions chanter de mésange, hirondelle, ... sont subsumées par l'action chanter de passereau.

Une action peut nécessiter une ressource.

Exemple :



Félix mange un morceau de viande :

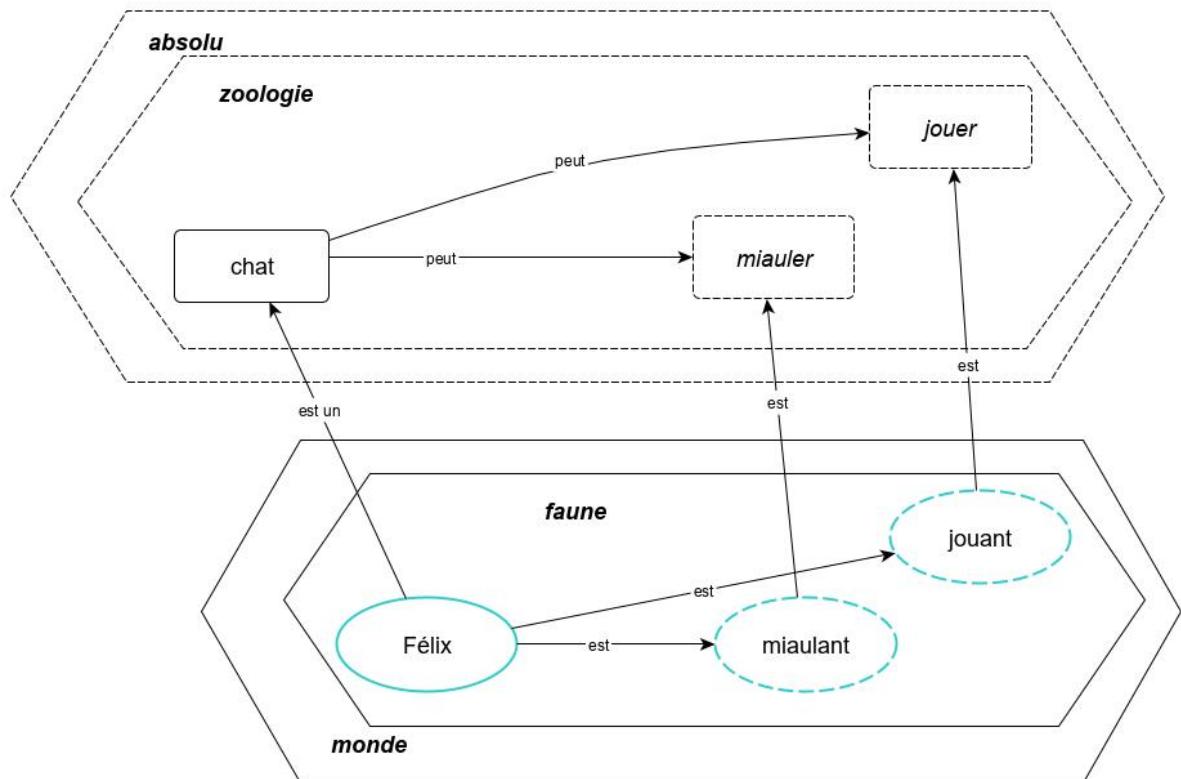


Noter qu'un être utilisé comme ressource n'a pas forcément été conçu pour cela.

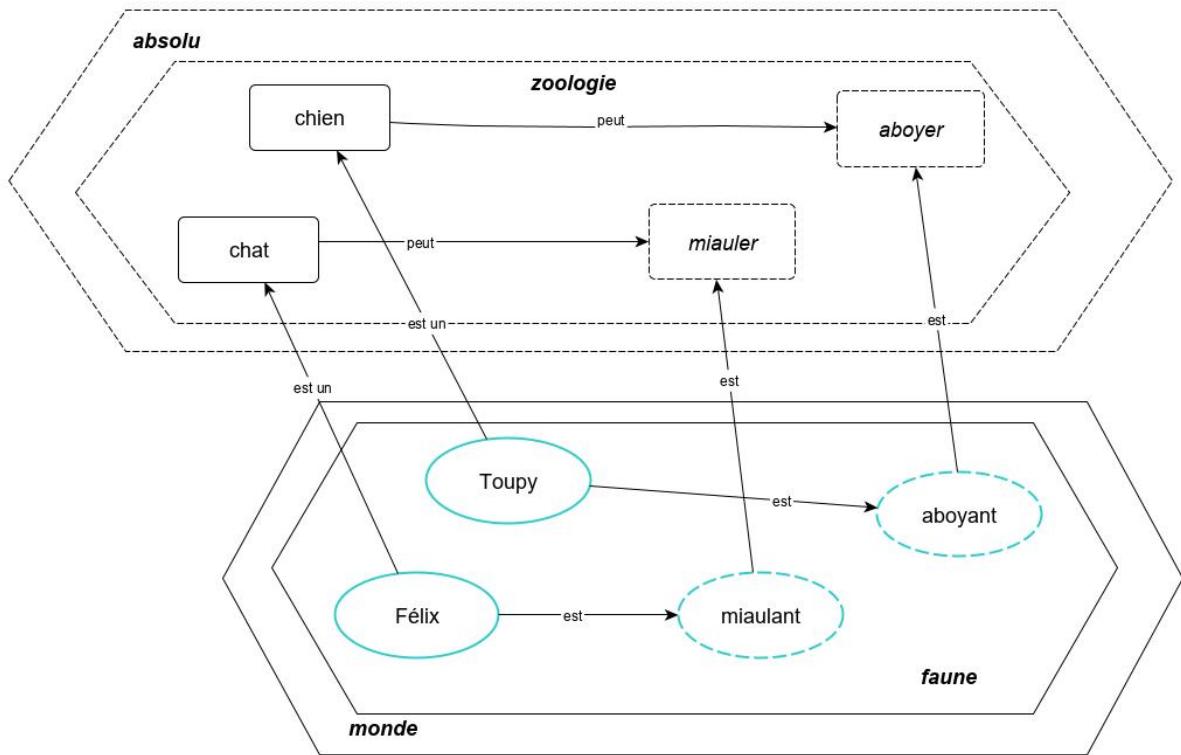
C'est le cas d'un livre utilisé pour coincer une porte, ou d'une baignoire utilisée comme abreuvoir dans un champ.

Dans une même situation un être peut avoir plusieurs actions, et deux êtres peuvent avoir la même action ou des actions différentes.

Dans le graphe suivant les actions génériques miauler et jouer sont essentielles pour chat :



Dans le graphe suivant l'action générique aboyer est également essentielle pour chien :



## Déclenchement d'une action

Un évènement produit (émis) par un être acteur peut déclencher une action d'un autre acteur (ou du même acteur) qui le perçoit. Une action peut aussi déclencher une autre action.

Une action correspond à une manière d'être d'un acteur. Cette action rentre dans la définition de l'essence de l'acteur qui la possède (si elle est essentielle, comme chanter pour passereau) ou peut la posséder (si elle est accidentelle), et non dans la définition de l'acteur qui peut la déclencher.

A noter que l'action d'un être est une manière d'être qui peut être permanente ou intermittente. Une action permanente d'un être est déclenchée par sa création.

## Évènements

Selon le Littré, un évènement est tout ce qui arrive<sup>1</sup>.

En physique, un évènement est tout phénomène se produisant en un point et à un instant donnés.

En calcul de probabilités, un évènement est le résultat éventuel d'un tirage au sort, d'un jeu de hasard, d'un pronostic, etc.

Nous retiendrons de ces définitions que l'avènement de tout ce qui peut se créer, arriver, se produire, est un évènement.

<sup>1</sup>Selon le dictionnaire de l'Académie française, le mot évènement est apparu au XVème siècle, dérivé du mot avènement, du latin evenir, "sortir, se produire", de venire, "venir".

La notion d'évènement est intimement liée à celle d'action mais ne doit pas être confondue avec elle.

Par exemple, le fait d'insérer une pièce dans une machine de vente automatique est une action qui produit l'évènement "pièce insérée".

Pour éviter la confusion entre les deux notions nous dénoterons l'action par "insertion(pièce)" et l'évènement résultant par "(pièce) insérée"<sup>1</sup>.

L'objet pièce est dans cet exemple un attribut (ou paramètre) de l'action et de l'évènement.

A noter que le déclenchement, l'exécution et la fin d'une action constituent en eux-mêmes des évènements, indépendamment de ce que l'action produit.

Une action peut également déclencher une autre action.

Le point commun à tous les évènements est qu'ils correspondent à un changement de situation.

## Changement de situation

Toute action peut avoir pour effet la naissance d'un nouvel être ou d'un nouvel état, ou la disparition d'un être ou d'un état, ce qui a entraîné un changement de situation dans le monde.

Ainsi, la définition d'une action peut faire intervenir des êtres.

Mais comment des êtres ou des états, qui se situent au niveau individuel, peuvent-ils être représentés au niveau générique ?

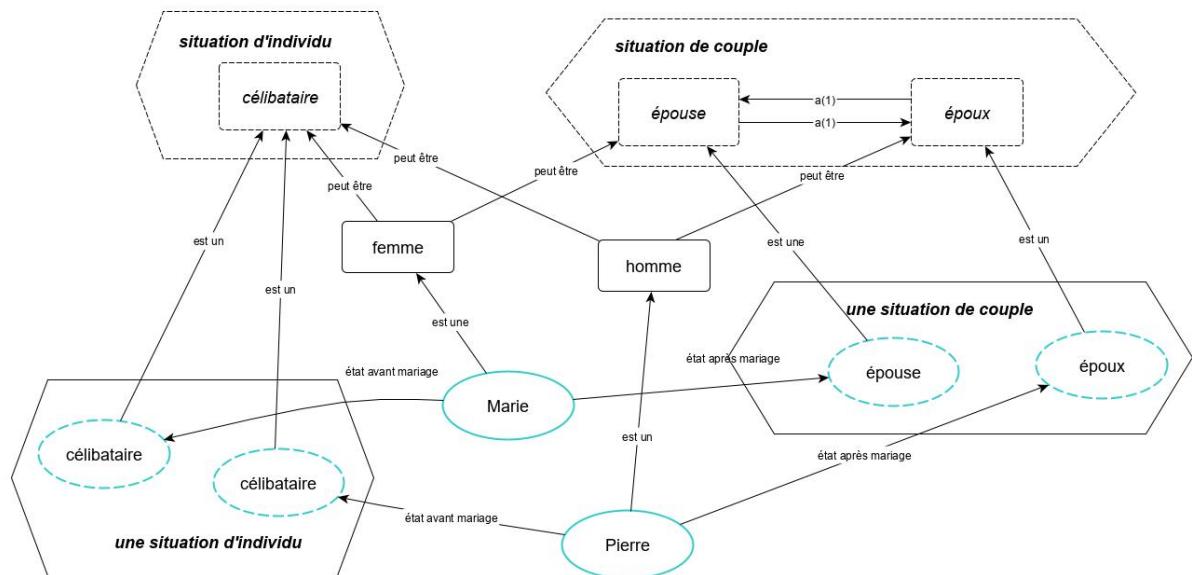
Il s'agit d'êtres que nous qualifierons d'acteurs (ou de rôles) indéterminés.

Pour éclairer ce sujet nous allons repartir d'un exemple précédent portant sur la situation maritale d'un homme et d'une femme.

Le graphe suivant illustre les changements d'états de Marie et de Pierre qui passent respectivement de l'état célibataire à celui d'épouse et d'époux par l'effet de leur mariage :

---

<sup>1</sup>Ce qui correspond à l'utilisation de la voix passive en français.



Comment pouvons nous définir l'essence de l'action marier au niveau générique ?

Le mariage ne correspond pas à une transition entre manières d'être d'une essence mais à un changement d'état d'êtres.

Considérons les cas d'utilisation<sup>1</sup> suivants, qui décrivent le déroulement logique du mariage d'un homme et d'une femme par un maire :

### Cas d'utilisation : mariage

- acteurs : un homme, une femme, un maire
  - préconditions : bans publiés depuis au moins 10 jours, sans que quiconque n'ait fait opposition au mariage.
  - description
1. Le maire demande à l'homme et à la femme d'échanger leur consentement.
  2. Ils eurent beaucoup d'enfants (hors sujet)

La publication des bans est un prérequis pour vérifier que rien ne s'oppose au mariage; en particulier que l'homme et la femme sont célibataires et sont majeurs.

### Cas d'utilisation : publication des bans

- acteurs : un homme, une femme, une mairie
- préconditions : l'homme ou la femme doivent avoir des liens durables avec la commune où aura lieu le mariage.
- description

<sup>1</sup>Le formalisme des cas d'utilisation a été proposé par Yvar Jacobson (l'un des auteurs d'UML) en 1992 dans un langage de modélisation nommé OOSE (*Object Oriented Software Engineering*) pour décrire les acteurs et le comportement d'un système réactif.

1. L'homme et la femme rédigent une annonce de mariage et la transmettent à la mairie
2. La mairie publie l'annonce

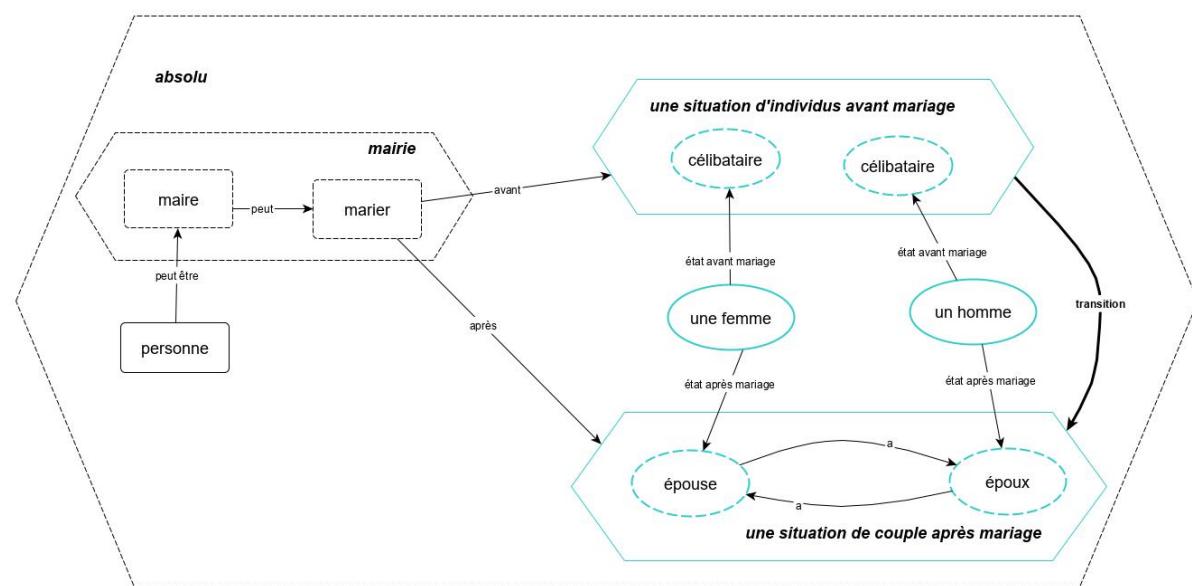
Qui sont la femme, l'homme et le maire qui apparaissent dans ces cas d'utilisation ? Ils sont indéterminés.

Noter que le lieu et la date de mariage, bien qu'indéterminés dans ces cas d'utilisation, sont toutefois contraints dans l'espace-temps par la nécessité du respect d'un délai pour la publication des bans.

Les changements d'états de l'homme et de la femme correspondent à des éventualités définies au niveau générique, où les manières d'être époux et épouses sont compatibles entre elles mais sont incompatibles avec la manière d'être célibataire.

L'action du maire a pour effet de détruire la situation de célibataire de l'homme et de la femme avant leur mariage pour les faire passer dans une situation de couple. Elle enlève à la femme son état célibataire pour lui affecter son nouvel état d'épouse, enlève à l'homme son état célibataire pour lui affecter son nouvel état d'époux et lie les nouveaux états de manière corrélative.

Le mariage d'une femme et d'un homme est représenté dans le graphe suivant :



L'action générique marier définit une transition entre les deux situations avant et après mariage d'un homme et d'une femme.

Nous aurions pu désigner les êtres indéterminés un homme et une femme par des symboles comme xy et xx.

Les logiciens appellent ces symboles des variables. Ceci ne se justifie pas ici dans la mesure où l'action marier ne met en jeu qu'une seule femme et qu'un seul homme. Ceci aurait pu être utile si nous avions décrit le mariage de deux hommes ou de deux femmes.

Les deux états célibataires n'ont pas besoin d'être identifiés par des noms propres, car ils se distinguent par leur étant respectif : "un homme" ou "une femme".

La localisation spatio-temporelle du mariage n'est pas représentée dans la mesure où elle n'intervient pas dans la logique du mariage.

Les actions s'exécutent dans l'espace-temps, mais le temps n'intervient pas forcément dans la logique des situations qui est généralement causale et non temporelle.

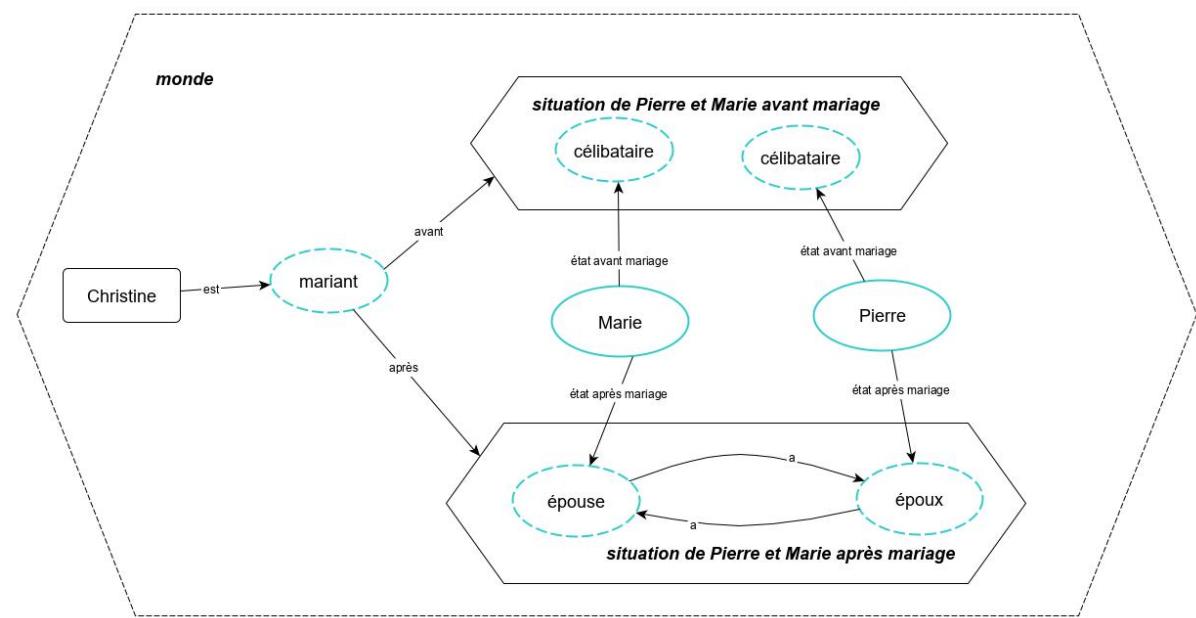
Il est possible que l'action marier ait un "effet de bord" qui porte sur un contrat de mariage.

Cet exemple est évidemment traité de manière très simpliste. Il suppose en particulier que la naissance d'un couple par l'action marier se fait à partir d'une paire de personnes comprenant un homme et une femme célibataires, mais sans préciser ce qui a eu pour effet de constituer cette paire de personnes candidates au mariage !

Pour que le mariage puisse avoir lieu, il faut que une femme et un homme soient au rendez-vous pour celui-ci.

D'une manière générale, beaucoup d'actions nécessitent la constitution préalable d'un ensemble d'êtres (leur mise en situation) concernés par cette action.

Voici une interprétation (un ancrage) possible d'un mariage dans le monde :



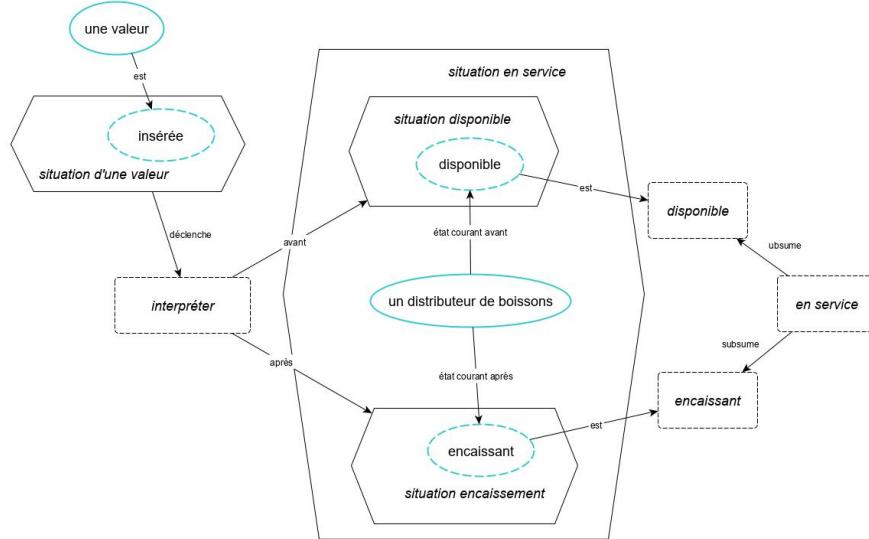
## Interactions entre acteurs

Le comportement d'un acteur peut être actif ou réactif.

Nous dirons qu'il est actif lorsque c'est l'acteur qui déclenche une action, et réactif lorsque son action est une réaction à un évènement produit par un autre acteur (ou lui-même).

L'action d'un acteur peut dépendre ou non de l'effet de ses actions antérieures. Ainsi, l'action d'une machine à laver est en principe identique à chaque lessive, tandis que l'essorage, qui n'est qu'une étape de chaque lessive, est conditionné par le fait que l'action de rinçage soit terminée.

Voici la représentation de la réaction possible d'un distributeur automatique de boissons (système réactif que tout le monde connaît) lorsqu'une première pièce est insérée par un client :



La réaction est un changement de situation du distributeur de boissons, de la situation disponible à la situation encaissement où il attend l'insertion d'une nouvelle pièce ou le choix du type de boisson à distribuer.

A noter que dans cet exemple chaque situation ne concerne qu'un même être (un distributeur de boissons). Nous pourrions représenter le comportement d'un distributeur de boissons sous la forme d'un statechart<sup>1</sup>. Ce formalisme a été conçu par David Harel pour modéliser le comportement des systèmes réactifs à des événements discrets.

Pour un lecteur qui connaît ce formalisme, voici quelques points sur lesquels le formalisme ICEO diffère.

Dans ICEO :

- Les transitions ne se font pas uniquement entre états mais entre situations qui peuvent inclure différents états d'un même être ou les états de différents êtres (comme dans notre exemple du mariage).
- Les états sont instances d'actions qui sont des manières d'être.
- A l'imbrication d'états dans les statecharts correspond la subsomption de manières d'être dans ICEO.

## Scénarios

La logique d'enchaînement des situations décrite dans les cas d'utilisation n'est généralement pas temporelle mais causale.

<sup>1</sup>Le formalisme des statecharts est un mode de représentation visuelle qui étend celui des machines à états finis, en particulier pour modéliser la concurrence.

UML a adopté, avec celui des statecharts, le formalisme des diagrammes d'activités qui sont en réalité une forme simplifiée des réseaux de Petri

Un scénario correspond à un déroulement possible d'un cas d'utilisation dans le temps :

- Les évènements se différencient suivant l'instant où ils se produisent.
- Un scénario ne comporte plus d'alternatives ni d'itérations.

Noter que le monde où s'exécutent les scénarios reste fictif<sup>1</sup> car les acteurs restent indéterminés.

Un scénario se présente comme une pièce de théâtre où l'écrivain décrit des rôles qui dans la réalité seront joués par des acteurs réels.

---

<sup>1</sup>fictif est un adjectif qui qualifie ce qui n'est pas réel

# Pour conclure sur cette présentation des concepts

Aristote considérait déjà la question "qu'est-ce que l'être ?" comme devant être le souci constant des philosophes. De fait, de nombreux philosophes et logiciens se sont penchés sur cette question métaphysique de l'être.

C'est une banalité de dire qu'un être ne peut pas se créer lui-même. La notion d'être est donc indissociable de celle de sujet créateur.

Dans notre formalisme, une essence, qui répond à la notion d'universel des philosophes, est ce qui permet de créer un être; elle possède une existence propre. En tant qu'être, elle est instance de son essence.

**Les relations entre les êtres sont induites par leurs manières d'être dans certaines situations.**

Un point qui n'aura pas échappé à nos lecteurs est l'amphibologie du mot "est" dans le formalisme ICEO, que l'on constate également en français<sup>1</sup>.

Le sens premier du verbe être en français est celui d'exister. C'est la forme intransitive du verbe être, utilisée si l'on affirme que "Paul est".

Mais considérons par exemple les phrases suivantes :

- Paul **est** un homme dans le monde
- être homme c'**est** être vivant dans l'absolu
- l'homme **est** mortel dans le temps
- Paul **est** grand dans l'espace
- Paul **est** époux dans son couple
- Paul **est** dans sa maison
- Paul **est** en train de dormir dans son lit

---

<sup>1</sup>Les langues indo-européennes ont toutes un terme équivalent au verbe être français, ce qui n'est pas le cas de toutes les langues (en particulier les langues austronésiennes, le chinois, le japonais, ...) où les différents sens du verbe être s'expriment avec d'autres tournures de phrase.

Ces exemples illustrent quelque peu l'amphibologie du mot "est" dans la langue française.

Si certains concepts d'ICEO se situent dans le domaine du TAL (traitement automatique des langues), avec la possibilité de modéliser le sens des noms, des adjectifs, des prépositions, des verbes, ... ils pourraient aussi être pris en compte dans les langages destinés à la modélisation et la conception des systèmes, comme UML.

Ils sont en lien avec la logique des situations telle que décrite par Jon Barwise et John Perry<sup>1</sup> où les "*basic building blocks*" sont les individus, leurs propriétés, leurs relations et leur localisation.

ICEO est avant tout un langage déclaratif. Toutefois, le formalisme ICEO a été implanté en Smalltalk qu'il épouse parfaitement; il bénéficie ainsi de la puissance de ce langage pour écrire des algorithmes.

Le code source d'ICEO et une série d'exemples de complexité croissante sont téléchargeables à l'adresse <https://github.com/rodejaphgh/ICEO><sup>2</sup>

Certains concepts d'ICEO pourraient inspirer une évolution de Smalltalk, en particulier :

- le morphisme existant entre la structure des essences et celle des êtres.
- la notion de situation
- la distinction fondamentale entre les notions d'être et de manière d'être

En Smalltalk, une classe définit les attributs de ses instances (par des variables d'instance) mais la structure elle-même des classes est sans relation avec celle de ses instances. La construction des instances repose sur la notion de méthode de classe (méthode d'instanciation définie au niveau métaclassse).

Cette approche est plus procédurale que déclarative. Si dans ICEO une essence comme celle de chat possède les essences attributs tête, patte et queue, ses instances auront automatiquement les êtres attributs tête, patte et queue avec la cardinalité et les contraintes de structure définies au niveau essence.

La notion de variable d'instance n'existe pas dans ICEO : **les attributs des essences sont des essences et les attributs des êtres sont des êtres.**

La notion de manière d'être tend à rendre la notion d'héritage multiple superflue.

Ainsi, par exemple, pour exprimer qu'un hydravion possède les caractéristiques à la fois d'un avion et d'un bateau, il convient tout d'abord dans ICEO de décider s'il s'agit par essence d'un objet volant ou d'un objet flottant. Le choix du premier cas est le plus logique, car un hydravion est fait avant tout pour voler.

Il semble donc logique de définir hydravion comme ayant pour genus avion et comme manière d'être le fait de pouvoir flotter.

---

<sup>1</sup>cf. les ouvrages "*Situations and attitudes*" de Jon Barwise et John Perry publié par le MIT en 1983 et "*Logic and information*" de Keith Devlin publié par Cambridge University Press en 1991

<sup>2</sup>Pour bâtir ce langage textuel nous avons utilisé Smalltalk; plus précisément, la version  de Smalltalk.

# Bibliographie

[Fron, 94] Annick Fron : "Programmation par contraintes", Eyrolles

[Barwise, 88] Jon Barwise : "The situation in logic", Center For the Study of Language an Information

[Barwise & Perry, 83] Jon Barwise & John Perry : "Situations and Attitudes", CSLI

[Bobrow & Winograd, 77] Daniel G. Bobrow et Terry Winograd : "An overview of KRL, a knowledge representation language", Cognitive Sciences

[Böhm & Jacopini, 66] Corrado Böhm, Giuseppe Jacopini : "Flow diagrams, Turing Machines and Languages with only two formation rules", ICS, Rome

[Borning, 79] A. Borning et T. O'Shea "THINGLAB : A constraint-oriented simulation laboratory, PhD Thesis, Standford University

[Brachman, 77] Ronald J. Brachman : "What is a concept : Structural Foundations for Semantics Networks", International Journal of Man-Machine Studies 9 (2), pp.127-152

[Briot, 85] Jean-Pierre Briot : "Instanciation et héritage dans les langages objets". Thèse de 3<sup>ème</sup> cycle, Paris

[Cointe & Briot, 89] Pierre Cointe, Jean-Pierre Briot "ClassTalk : une transposition des métaclasses ObjVLisp à Smalltalk-80". RFIA, Paris pp. 127-146

[Dahl & Nygaard, 66] O.J. Dahl, K. Nygaard "Simula : an algol-based simulation language". Communications of the ACM. 9: 671:678

[Dahl & al., 70] O.J. Dahl, K. Nygaard, B. Myhraug "Simula-67 Common Base Language". SIMULA information, S22, Norwegian Computing Center, Oslo, Norway

[Desclaux & Bourgeois, 87] C. Desclaux, R. Bourgeois "ECCAO : Etablissement des Cahiers des Charges Assistée par Ordinateur" Premières journées Internationales : le Génie Logiciel & ses applications. pp. 1345-1364. Toulouse

[Devlin, 91] Keith Devlin : "Logic and Information", Cambridge University Press

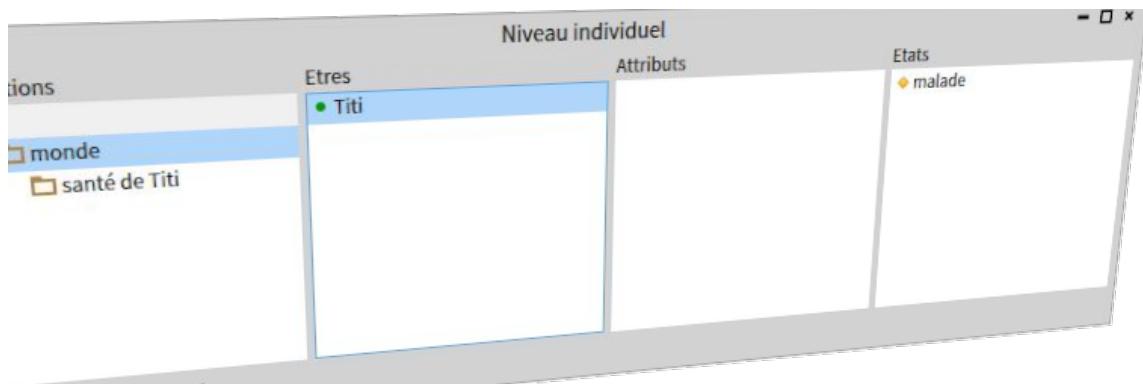
[Ducasse & al., 2022] Stéphane Ducasse, Gordana Rakic, Sebastian Kaplar, Quentin Ducasse : "Pharo by Example 9 – 2022" téléchargeable à l'adresse <https://github.com/SquareBracketAssociates/PharoByExample9/>

- [Dyer, 83] Michael G. Dyer "In-depth understanding" MIT
- [Einstein, 44] Albert Einstein : "Remarks on Bertrand Russell's theory of knowledge", Vol. V of "The Library of Living Philosophers," édité par Paul Arthur Schilpp
- [Ferber, 89] Jacques Ferber "Objets et agents : une étude des structures de représentation et de communication en intelligence artificielle". Thèse d'état, LITP Université Paris VI
- [Gardies, 2004] Jean-Louis Gardies : "Du mode d'existence des êtres de la mathématique", Librairie philosophique J. Vrin
- [Gilson, 2018] Etienne Gilson : "L'être et l'essence", Librairie philosophique J. Vrin
- [Harel, 84] David Harel, "Statecharts: A Visual Formalism for Complex Systems " Science of Computer Programming, The Weizmann Institute of Science
- [Hewitt, 69] C. Hewitt, "PLANNER : A language for manipulating models and proving theorems in a robot" Int. Joint Conf. Artificial Intelligence, Washington D.C.
- [Hewitt, 71] C. Hewitt, "Procedural Embedding of Knowledge in PLANNER" Int. Joint Conf. Artificial Intelligence, London
- [Hewitt, 77] C. Hewitt, "Viewing control structures as patterns of passing messages" Artificial Intelligence, 8: 323-364
- [Jacobson, 92] Ivar Jacobson : "OOSE : Object-Oriented Software engineering", ACM Press
- [Kant, 1787] Emmanuel Kant : "Critique de la raison pure" Traduction par François Picavet, Librairie Félix Alcan
- [Kay, 68] Alan Kay "FLEX : A flexible extendible language". Computer Science Dept. Technical report, University of Utah
- [Kay, 69] Alan Kay "The reactive machine" Doctoral dissertation, University of Utah
- [Kayser, 88] Daniel Kayser : "Le raisonnement à profondeur variable", actes des 2èmes journées internationales du GRECO-P.R.C. d'intelligence artificielle, éditions Teknéa
- [Kayser, 97] Daniel Kayser : "La représentation des connaissances", Hermès
- [Liebermann, 86] Henry Liebermann " Using prototypical objects to implements shared behavior in object oriented systems" OOPSLA Proceedings
- [Masini & al., 89] G. Masini, A. Napoli, D. Colnet, D. Léonard, K. Tombre : " Les langages à objets ", InterEditions
- [McCarthy & al., 62] J. McCarthy, J. P. W. Abrahams, D.J. Edwards, T.P. Hart, M. I. Levin "LISP 1.5 Programmer's Manual". MIT Press, Cambridge, MA
- [McCarthy, 63] J. McCarthy "A basis for a mathematical theory of computation". P. Bradford and D. Hirschberg (Eds), Computer programming and formals systems. Amsterdam, North-Holland

- [Minsky, 68] Marvin Minsky : "Semantic information processing", MIT press
- [Minsky, 86] Marvin Minsky : "The Society of Mind", New York, Simon & Schuster
- [Pachet, 90] François Pachet "Mixing rules and objects : an experiment in the world of Euclidean geometry" Rapport LAFORIA
- [Parry & Hacker, 91] William T. Parry & Edward A. Hacker : "Aristotelian logic", State University of New York Press
- [Quilian, 68] M. Ross Quilian "Semantic memory" dans [Minsky, 68]
- [Quine, 86] William Van Orman Quine : "Le mot et la chose", Flammarion
- [Riesbeck, 75] Christopher K. Riesbeck "Conceptual analysis" dans [Schank, 75]
- [Robinson, 50] Richard Robinson : "Definition", Oxford University Press
- [Russel, 1903, 1937] Bertrand Russel : "The principles of Mathematics"
- [Russel, 1905] Bertrand Russel : "On denoting" revue Mind
- [Schank, 75] R.G. Schank " Scripts Plans Goals and understanding – an inquiry to human knowledge structures" LEA
- [Sowa, 84] John F. Sowa : "Conceptual Structures. Information Processing In Mind and Machine", Addison Wesley
- [Strawson, 50] P.F. Strawson "On referring" revue Mind
- [Sussman & McDermott, 72] G. Sussman & D. McDermott "From planner to CONNIVER" AFIPS
- [Tarsky, 69] Alfred Tarsky "Introduction à la logique", Paris – Gauthier-Villars
- [Wertz, 2015] Harald Wertz "Programmation orientée objet avec Smalltalk. Classes, instances, messages et héritage" Editions ISTE
- []Woods, 75] W.A. Woods "What's in a link : foundations for semantic networks" Cognitive Science



# Présentation du langage textuel d'ICEO





# Implantation en Smalltalk

Le langage textuel d'ICEO est implanté en Smalltalk<sup>1</sup>

Le choix de ce langage 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 (version 12) qui est l'une des variantes de Smalltalk dérivées de la version de Smalltalk-80 originelle.

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

Pour étudier ces exemples, 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.

---

<sup>1</sup>Smalltalk est un langage de programmation orienté objet qui a été conçu en 1972 par des génies informatiques : Alan Kay, Dan Ingals, Ted Kaehler et Adele Goldberg au Palo Alto de 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 complètement graphique.



# Installation d'ICEO dans Pharo et exemples

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

La version utilisée est la version 12 qui peut être installée en utilisant le launcher de Pharo ou en mode standalone (VM + image).

Pour nos lecteurs qui ne connaissent pas Smalltalk, nous conseillons de commencer par la lecture du livre "*Programmation orientée objet avec Smalltalk*" d'Harald Wertz ou du livre "*Pharo By Example*" de Stéphane Ducasse, 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>

## Installation d'ICEO

En supposant Pharo installé, il convient, pour intégrer ICEO dans l'environnement de Pharo, de récupérer les fichiers sources d'ICEO.

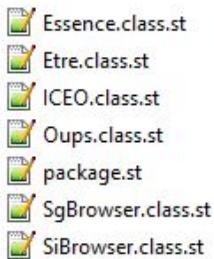
Tous les fichiers sont téléchargeables à l'adresse <https://github.com/rodejaphgh/ICEO>; ils comprennent le code source d'ICEO et le code des exemples présentés dans ce document.

Pour récupérer les fichiers situés sous github, il est conseillé d'utiliser l'outil Metacello intégré dans Pharo. Cet outil est utilisé dans Pharo pour gérer les projets, leurs dépendances et leurs métadonnées.

Ceci peut être fait en ouvrant une fenêtre de type Playground et en évaluant le code :

```
Metacello new baseline: 'ICEO';
repository: 'github:// rodejaphgh/ICEO:main/src';
load .
```

Ces instructions vont récupérer localement, dans un sous-dossier nommé "pharo-local/iceberg/rodejaphgh/ICEO/src/ICEO" situé dans le dossier où est installé Pharo, tous les fichiers sources situés dans la repository github nommée ICEO :

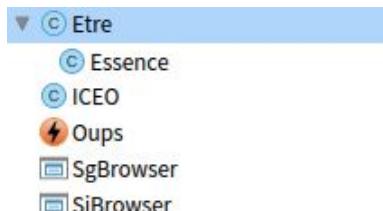


La dernière instruction " load " installe ICEO dans l'environnement de Pharo.

Au bout de quelques instants la barre de menu principale affiche un nouvel onglet nommé ICEO :



Dans un Browser, la catégorie (le "package") ICEO s'affiche avec en particulier les classes Etre, Essence, et ICEO<sup>1</sup> :

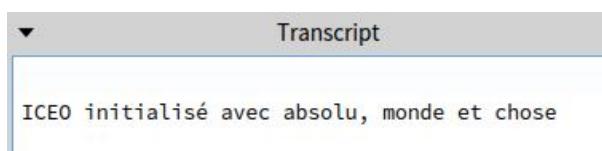


La classe ICEO est l'interprète du langage textuel d'ICEO.

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

Avant de commencer l'étude d'un premier exemple, il est impératif d'exécuter l'instruction " ICEO start ".

Dans une fenêtre Transcript, le texte suivant s'affiche alors :

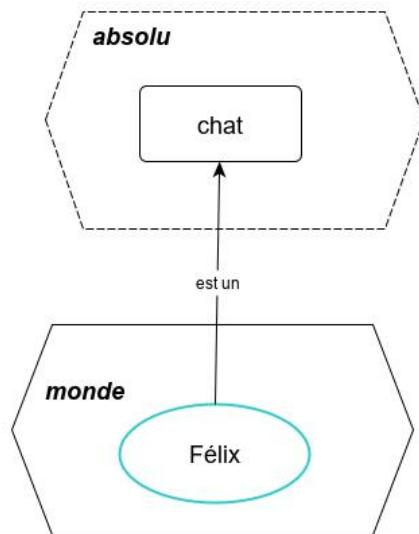


A ce stade, il est conseillé de sauvegarder votre image. Pour retrouver Pharo dans l'état actuel lors du prochain lancement, il suffira ainsi de choisir cette image.

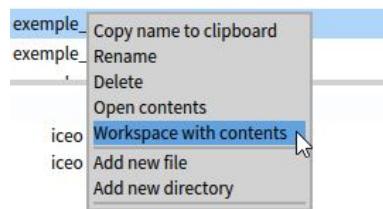
---

<sup>1</sup>Le browser est un outil essentiel en Smalltalk pour accéder à l'API (ensemble des méthodes) des classes. Dans le code d'une méthode, les commentaires sont écrits entre doubles quotes "".

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



Dans une fenêtre File Browser sélectionner le fichier "exemple\_1.text" et l'option Workspace with contents :



Une fenêtre de type Playground s'affiche avec le code de l'exemple :

```
1 iceo definition: #chat.
2 iceo soit: #Félix essence: chat.
```

Dans le texte de l'exemple, vous pouvez sélectionner et interpréter les instructions (ou expressions) une par une ou en bloc, en utilisant l'option "Do it" offerte avec le bouton droit de la souris.

Pour l'interprétation d'un bloc d'instructions, il faut que chacune se termine par un point.

Si le premier objet de plusieurs instructions consécutives est le même, il est possible de les écrire en cascade, séparées par un point-virgule.

Exemple :

```
iceo definition: #chat; soit: #Félix essence: chat.
```

Une longue instruction peut s'étendre sur plusieurs lignes.

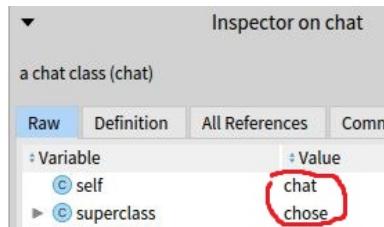
Chaque exemple peut être exécuté en totalité en une seule passe en utilisant une combinaison des touches Ctrl a (pour sélectionner l'ensemble du texte) suivie d'un DoIt ou de Ctrl d<sup>1</sup>

Un conseil : ayez toujours une fenêtre de type Transcript ouverte lors de l'exécution d'un bloc d'instructions. En cas d'erreur, l'instruction fautive y sera affichée.

La première instruction du premier exemple définit l'essence chat dans la situation générique absolu, subsumée par l'essence chose.

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.

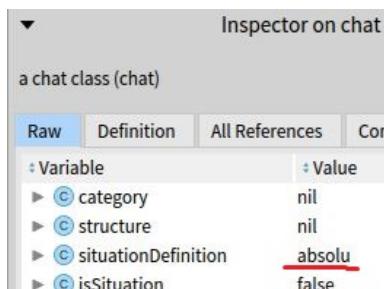
Selectionnez le mot chat et utilisez l'option "Inspect it" offerte avec le bouton droit de la souris :



Vous voyez 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.

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



L'accès à l'essence chat pourrait se faire avec l'expression " `absolu get: #chat` " mais ce n'est pas nécessaire car les essences définies dans l'absolu sont accessibles directement par leur nom.

La deuxième instruction de l'exemple 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 :

---

<sup>1</sup>ou en cliquant sur le petit bouton  , si vous souhaitez exécuter l'exemple en totalité et ouvrir un inspecteur sur le résultat de la dernière instruction exécutée

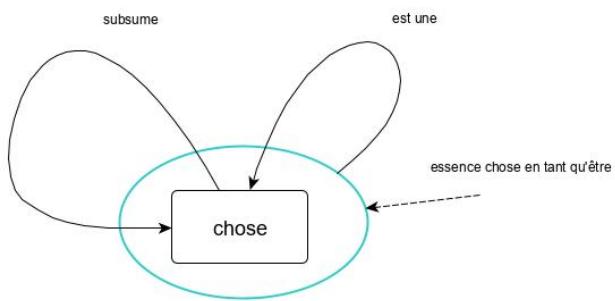
Inspector on a chat	
a chat	an absolu
Raw	Breakpoints
Variable	Value
self	a chat
structure	nil
situationDefinition	an absolu
isSituation	false
isEstat	false
etats	nil
etant	nil
nom	Félix
id	nil
Raw	Breakpoints
Variable	Value
self	an absolu
structure	an OrderedCollecti
situationDefinition	nil
isSituation	true
isEstat	false
etats	nil
etant	nil
nom	monde
id	nil

Noter que l'accès à l'être nommé Félix situé dans le monde exige de passer par l'expression "monde get: #Félix ", car Félix ne se situe pas dans l'absolu comme l'essence chat mais dans le monde.

L'essence chose définie dans l'absolu est son propre genus et, en tant qu'être, est instance d'elle-même (elle est sa propre mét-essence).

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

Ceci est conforme au schéma :



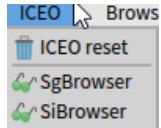
Par défaut, l'essence d'une essence (sa mét-essence) est l'essence chose.

Ainsi, l'expression " chat getEssence " retourne chose.

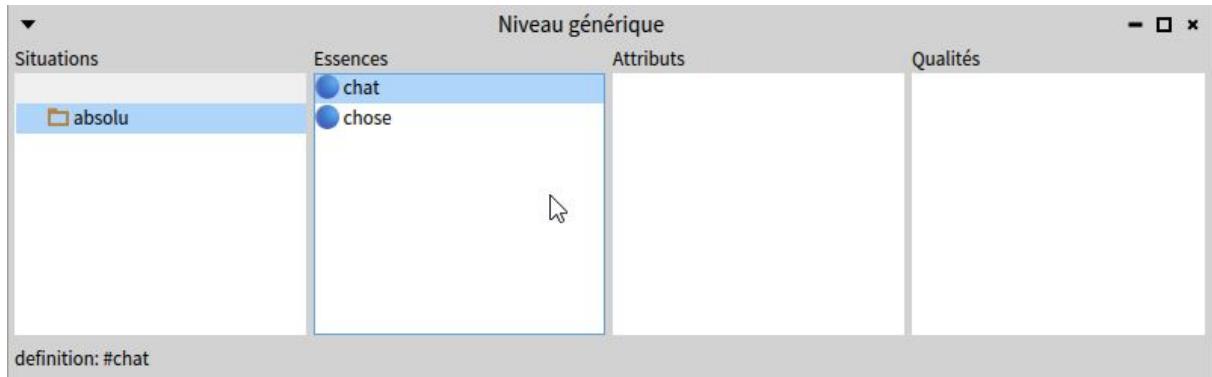
Vous êtes maintenant parés pour étudier d'autres exemples.

# Services accessibles via l'onglet "ICEO" de la fenêtre principale

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



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

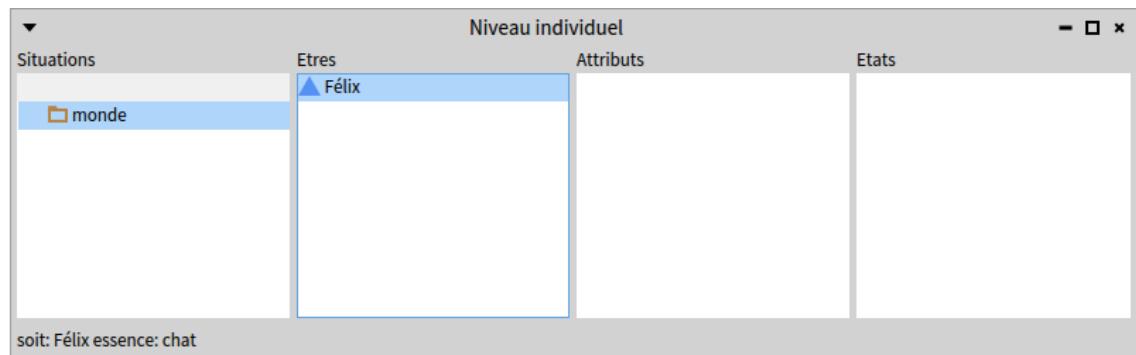


Le nom des essences est précédé d'un petit cercle bleu.

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'instruction qui a donné naissance à l'entité sélectionnée.

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



Le nom des êtres est précédé d'un petit triangle bleu.

L'option "ICEO reset" réinitialise le contenu de absolu et de 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.

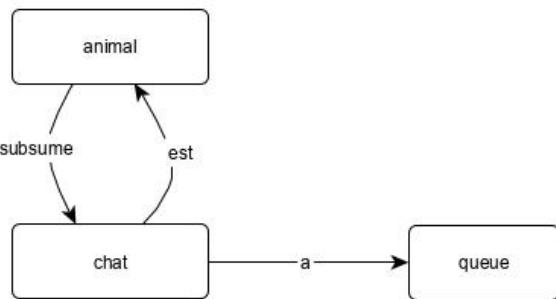
Par contre, il pourra être nécessaire de l'exécuter avant de démarrer l'étude d'un autre exemple, pour éviter des interférences avec le précédent.

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).

## Exemple 2 : sur la composition d'une essence

Cet exemple correspond au graphe suivant :



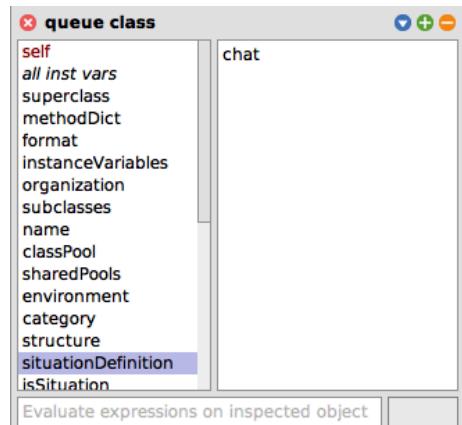
chat est animal qui a queue (chat =  $\oplus$  (animal, { queue}))

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

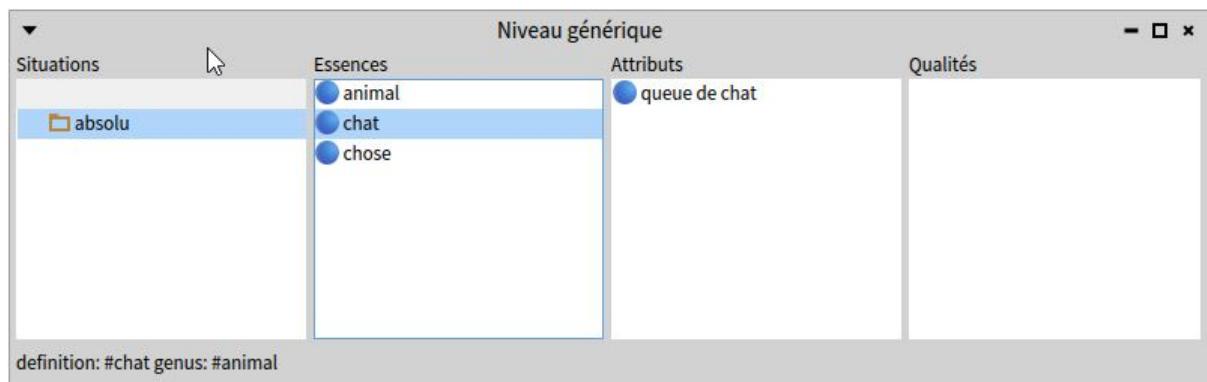
```
iceo definition: #animal.  
iceo definition: #chat genus: animal.  
iceo definitionAttribut: #queue de: chat.
```

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 en faisant un "Inspect it" sur l'expression donnée en commentaire entre doubles quotes :

"chat getEssenceAttribut: #queue " :

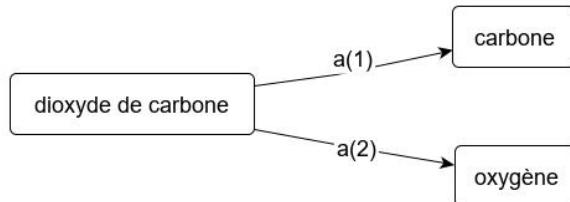


Ceci est confirmé dans un SgBrowser :



## Exemple 3 : sur la composition d'une essence à partir d'essences existantes

Cet exemple correspond au graphe suivant :



L'essence 'dioxyde de carbone' comporte un atome de carbone et deux atomes d'oxygène qui ne lui sont pas propres :

```
iceo definition: #oxygène.  
iceo definition: #carbone.  
iceo definition: 'dioxyde de carbone'.  
(absolu get: 'dioxyde de carbone' referenceEssence: oxygène cardinalite: 2.  
(absolu get: 'dioxyde de carbone' referenceEssence: carbone cardinalite: 1.  
iceo soit: 'une molécule de dioxyde de carbone' essence: (absolu get: '  
dioxyde de carbone' ).
```

La quatrième instruction indique que l'essence 'dioxyde de carbone' référence l'essence existante oxygène (définie ici dans l'absolu) avec une cardinalité de 2.

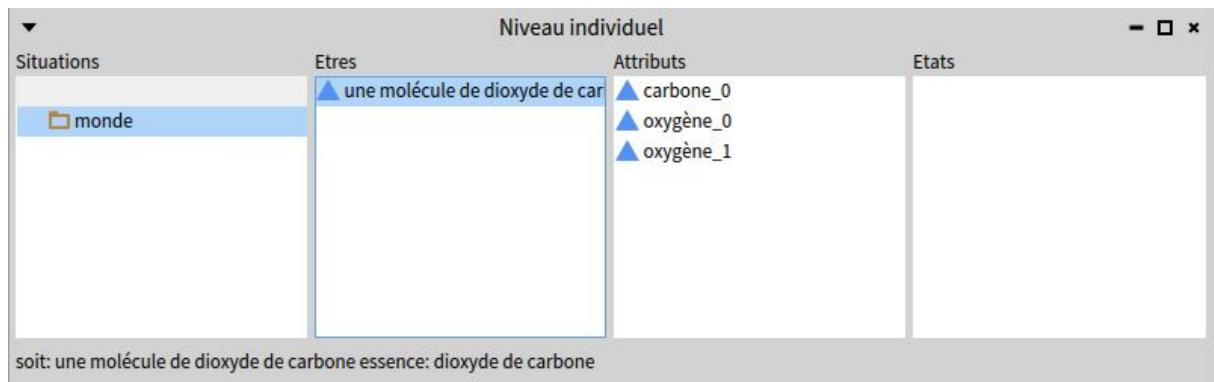
Les attributs carbone et oxygène sont affichés avec un petit cercle violet dans un SgBrowser pour indiquer qu'il ne s'agit pas d'attributs propres de dioxyde de carbone :

Niveau générique

Situations	Essences	Attributs	Qualités
absolu	carbone chose dioxyde de carbone oxygène	carbone oxygène	

definition: "dioxyde de carbone"

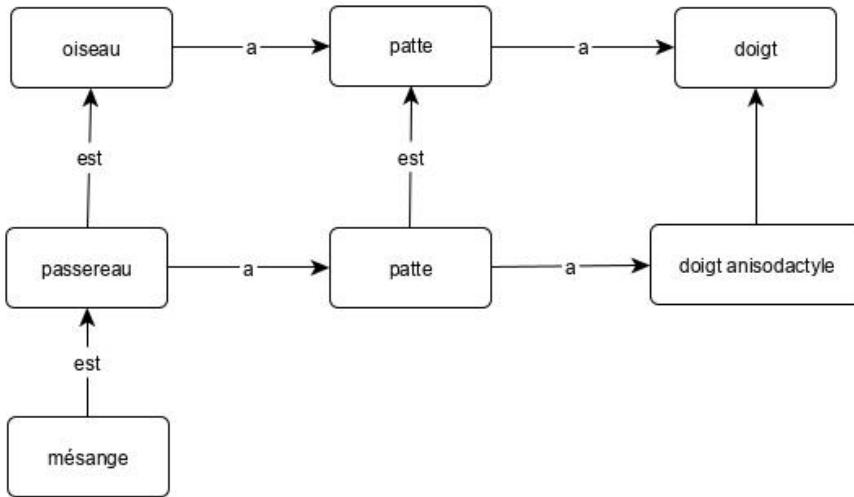
Dans un SiBrowser nous voyons que les êtres attributs de 'une molécule de dioxyde de carbone' ont été créés avec la cardinalité définie au niveau de son essence :



Comme le dioxyde de carbone est un individu, son instantiation a entraîné la création de ses attributs. Le nom des attributs a été généré automatiquement en tenant compte du nom de leur essence.

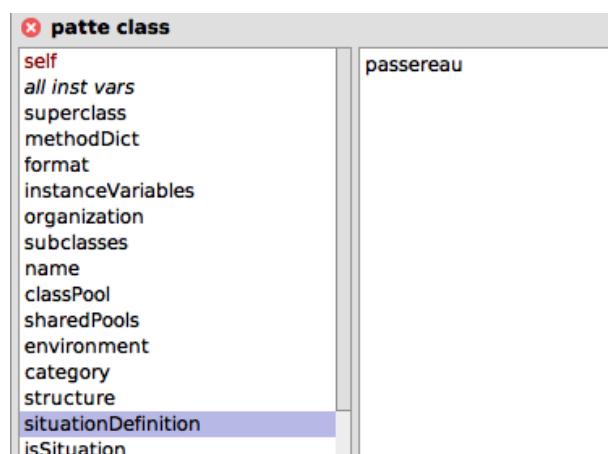
## Exemple 4 : sur le principe d'héritage des attributs des essences

Considérons le graphe suivant :



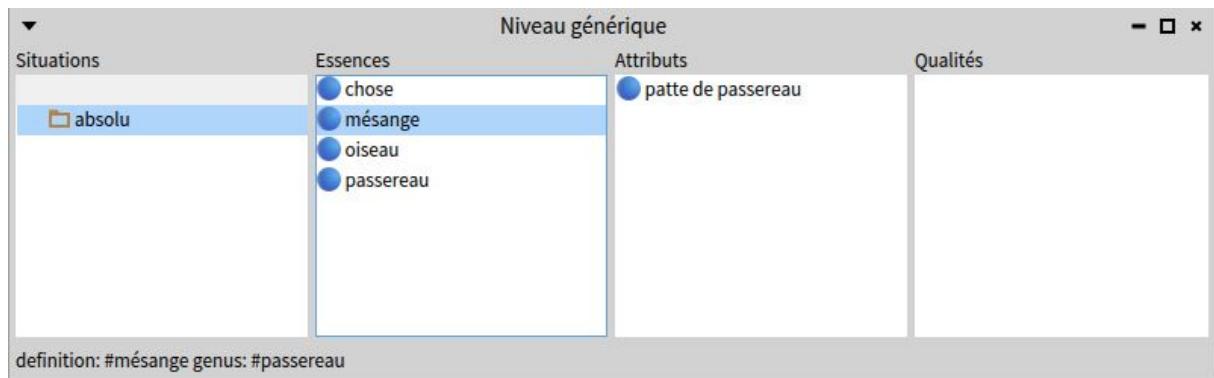
```
iceo definition: #oiseau.
iceo definition: #passereau genus: oiseau.
iceo definition: #mésange genus: passereau.
iceo definitionAttribut: #patte de: oiseau.
iceo definitionAttribut: #doigt de: (oiseau getEssenceAttribut: #patte).
iceo definitionAttribut: #patte de: passereau
    genus: (oiseau getEssenceAttribut: #patte).
iceo definitionAttribut: 'doigt anisodactyle' de: (passereau
    getEssenceAttribut: #patte) genus: ((oiseau getEssenceAttribut: #patte)
    getEssenceAttribut: #doigt) .
```

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



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

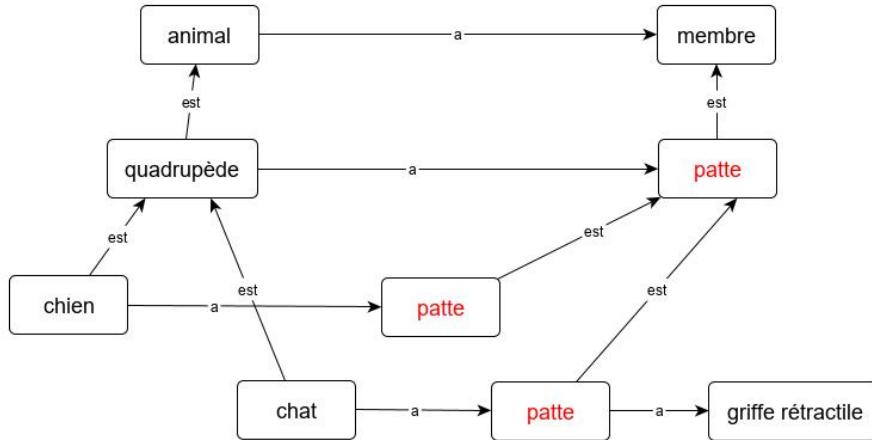
Ceci est confirmé dans un SgBrowser :



L'essence attribut "doigt anisodactyle" de patte de mésange est héritée de patte de passereau, ce qui confirmé par un "Inspect it" de " (mésange getEssenceAttribut: #patte) getEssenceAttribut: 'doigt anisodactyle' "

## 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 dans des situations différentes.

Du fait que différentes essences peuvent avoir le même nom, il est possible de demander (pour nous, lecteur humain) la génération d'un nom propre à chaque essence, pour vérifier par exemple que l'essence patte de quadrupède est différente de celle de chat et de chien.

Ainsi, un "Print it" des expressions suivantes :

```

" 'patte de quadrupède : ', (quadrupède getEssenceAttribut: #patte) getId "
" 'patte de chien : ', (chien getEssenceAttribut: #patte) getId "
" 'patte de chat : ', (chat getEssenceAttribut: #patte) getId "
  
```

donne :

patte de quadrupède : patte\_0

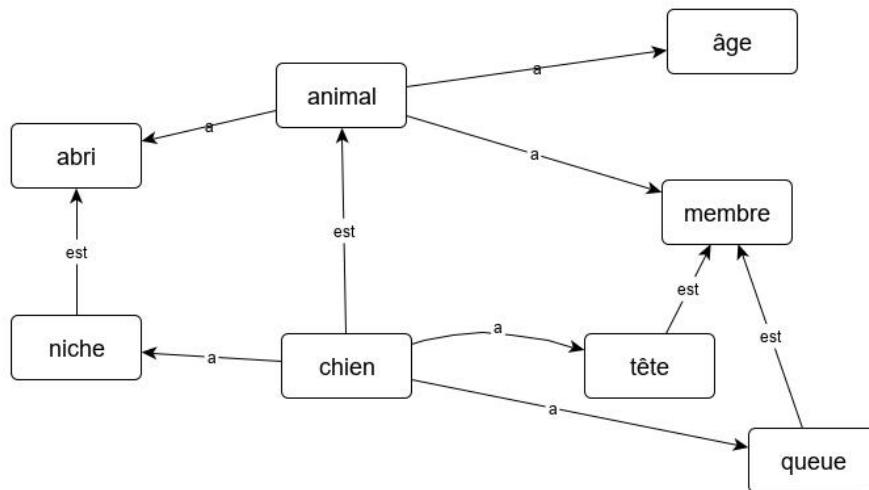
patte de chien : patte\_1

patte de chat : patte\_2

## Exemple 5\_2

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).
```

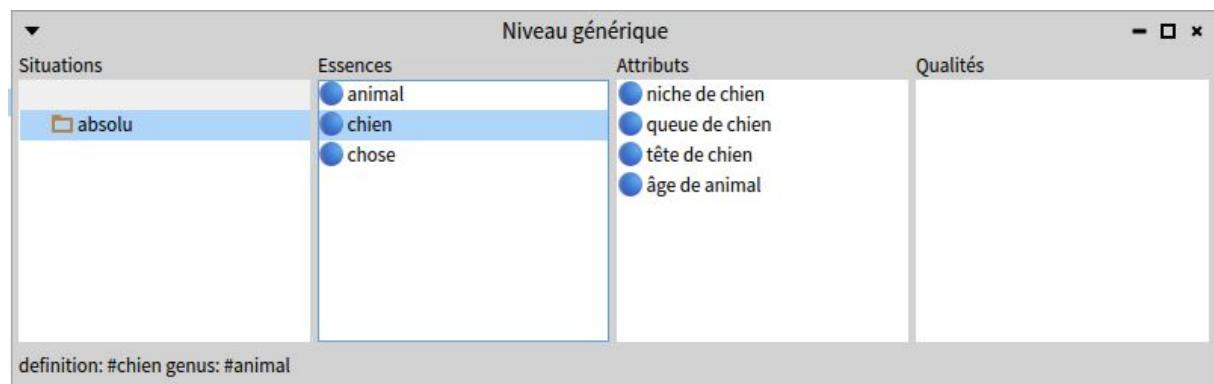
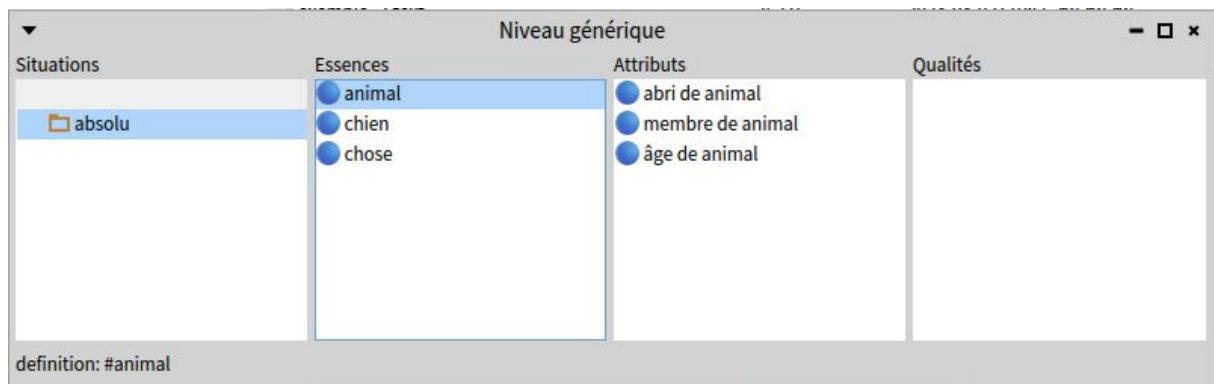
L'expression " chien getEssenceAttribut: #âge " donne : âge

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

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

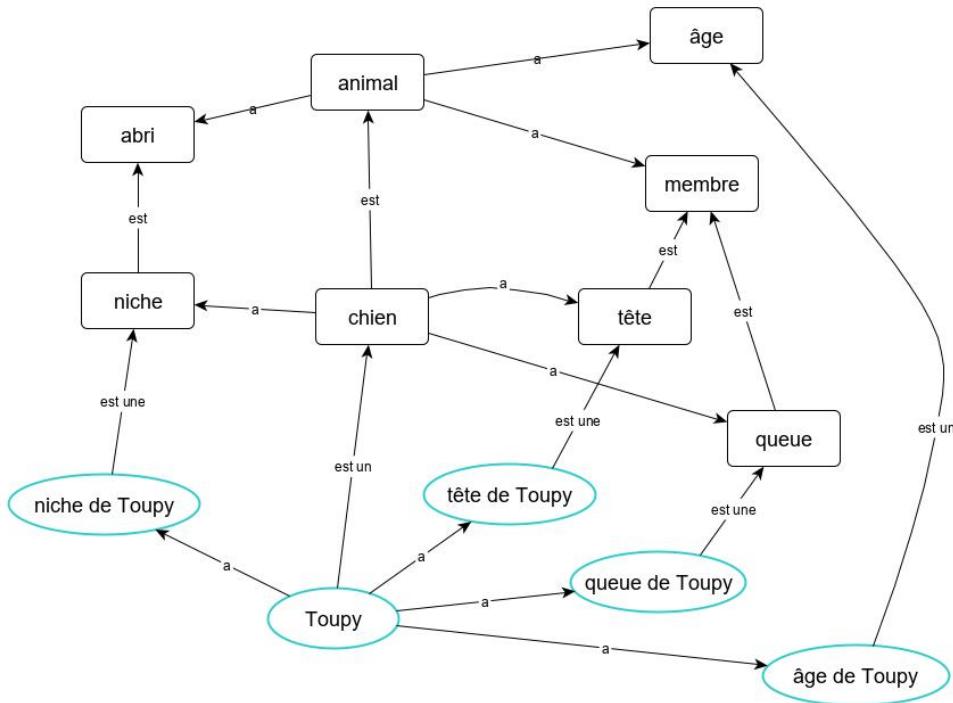
et l'expression " chien getEssencesAttributs: #membre " donne : an OrderedCollection(queue tête)

Ce qui se vérifie dans les fenêtres suivantes :



## Exemple 6 : sur le principe d'identification des êtres

Considérons le graphe suivant :



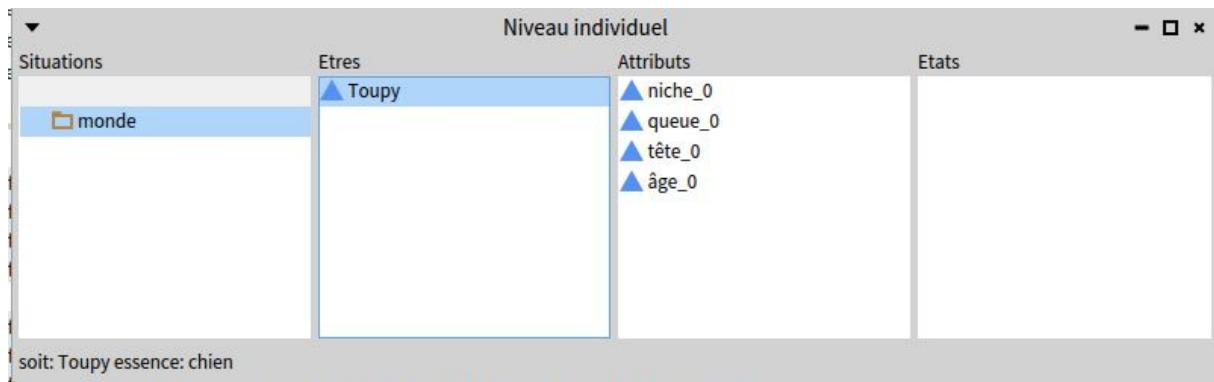
```

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`(âge\_0 niche\_0 tête\_0 queue\_0)

Les attributs de Toupy ont été créés automatiquement car il s'agit par défaut d'un individu.

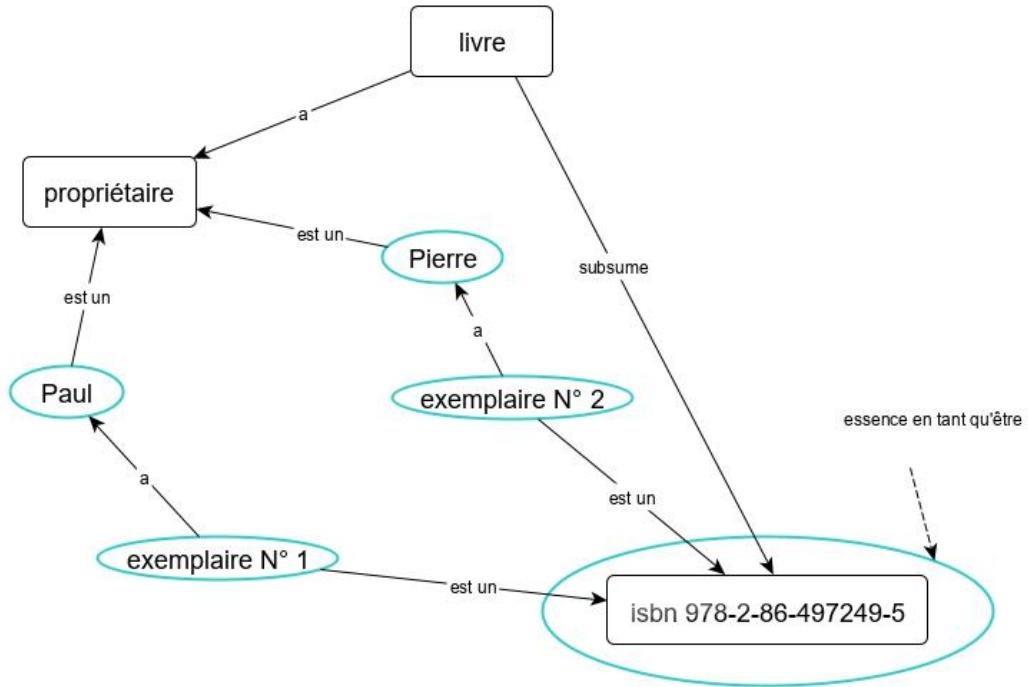
Leur nom a été créé par un générateur de symboles en se basant sur le nom de leur essence.



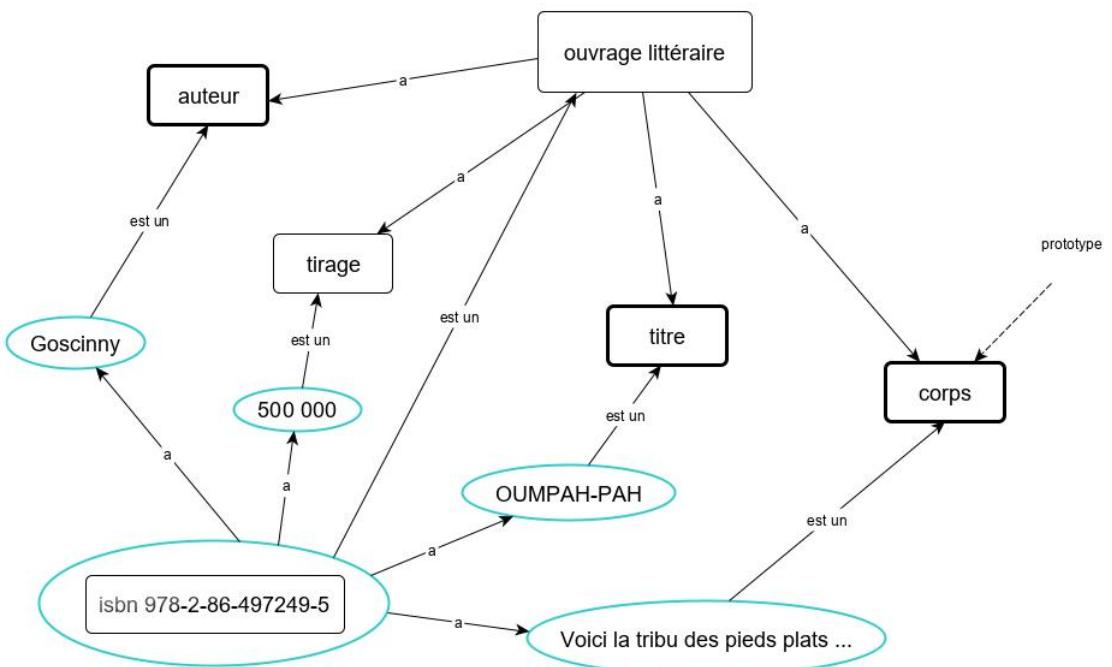
L'expression " ((monde get: #Toupy) getEtreAttribut: #abri) getNom " donne :  
`#niche_0`

## Exemple 7 : sur la notion de métá 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 métá 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).

```

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

Dans un SgBrowser nous pouvons visualiser les attributs des essences ouvrage littéraire et isbn 978-2-86-497249-5 :

Niveau générique			
Situations	Essences	Attributs	Qualités
absolu	chose isbn 978-2-86-497249-5 livre <b>ouvrage littéraire</b>	auteur de ouvrage littéraire corps de ouvrage littéraire tirage de ouvrage littéraire titre de ouvrage littéraire	
definition: 'ouvrage littéraire'			

Niveau générique			
Situations	Essences	Attributs	Qualités
absolu	chose isbn 978-2-86-497249-5 livre ouvrage littéraire	propriétaire de livre	
definition: 'isbn 978-2-86-497249-5' genus: #livre metaEssence #'ouvrage littéraire'			

et dans un SiBrowser les attributs de l'essence isbn 978-2-86-497249-5 en tant qu'être :

Niveau individuel			
Situations	Etres	Attributs	Etats
monde	Exemplaire_1 Exemplaire_2 Paul Pierre isbn 978-2-86-497249-5	50 000 Goscinny OUMPAH-PAH Voici la tribu des pieds plats...	
definition: 'isbn 978-2-86-497249-5' genus: #livre metaEssence #'ouvrage littéraire'			

et les attributs des exemplaire 1 et 2 :

Niveau individuel			
Situations	Etres	Attributs	Etats
monde	Exemplaire_1 Exemplaire_2 Paul Pierre Toupy	Goscinny OUMPAH-PAH Paul Voici la tribu des pieds plats...	
soit: Exemplaire_1 essence: isbn 978-2-86-497249-5			

Niveau individuel			
Situations	Etres	Attributs	Etats
monde	Exemplaire_1 Exemplaire_2 Paul Pierre Toupy	Goscinny OUMPAH-PAH Pierre Voici la tribu des pieds plats...	
soit: Exemplaire_2 essence: isbn 978-2-86-497249-5			

On voit que les deux exemplaires ont les mêmes attributs titre, auteur et corps (qualifiés de prototypes au niveau de ouvrage littéraire) mais que leur attribut propriétaire, que ne possède pas isbn 978-2-86-497249-5, est différent.

Par contre, il n'ont pas l'attribut tirage que possède isbn 978-2-86-497249-5.

Notons qu'une petite incorrection apparait dans cet exemple, car il n'est pas vraiment correct de considérer Pierre ou Paul comme attributs d'un exemplaire du livre.

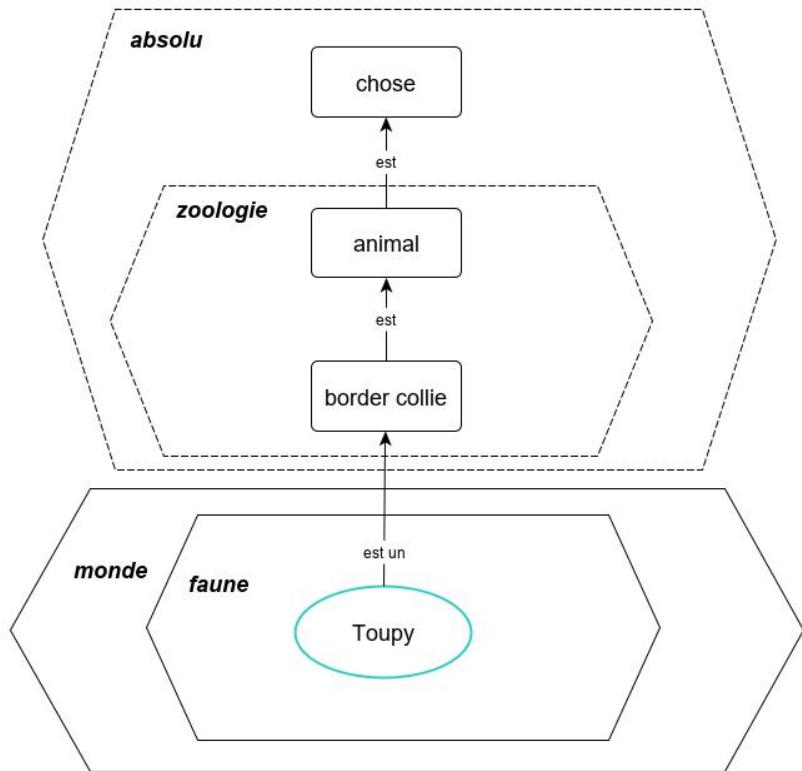
En fait, ce sont Pierre et Paul qui sont chacun propriétaire d'un exemplaire du livre.

Dire qu'un livre a un propriétaire est en fait un raccourci de langage pour exprimer qu'un exemplaire du livre appartient à une personne qui en est propriétaire.

Nous verrons comment corriger ce point dans l'exemple 14.

## 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) .
    
```

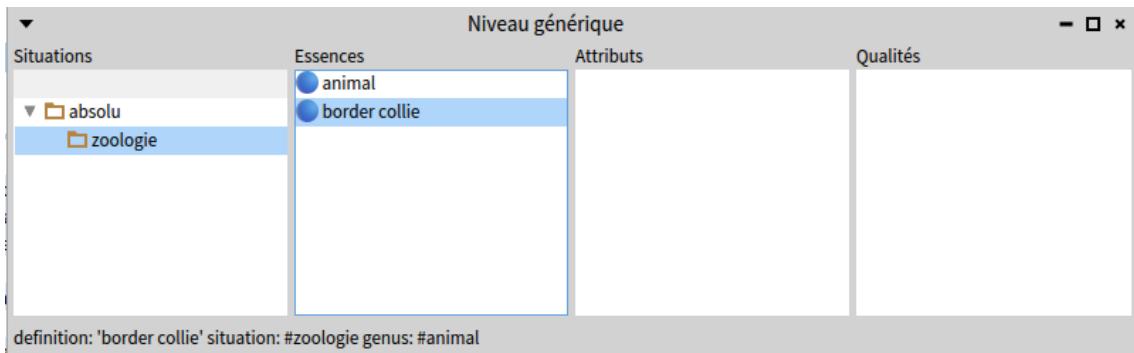
L'essence nommée "border collie" est subsumée par l'essence animal dans une situation générique nommée "zoologie", et un être "Toupy" est créé dans une situation individuelle nommée "faune". La situation générique zoologie est incluse dans l'absolu et la situation individuelle faune, instance de la situation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

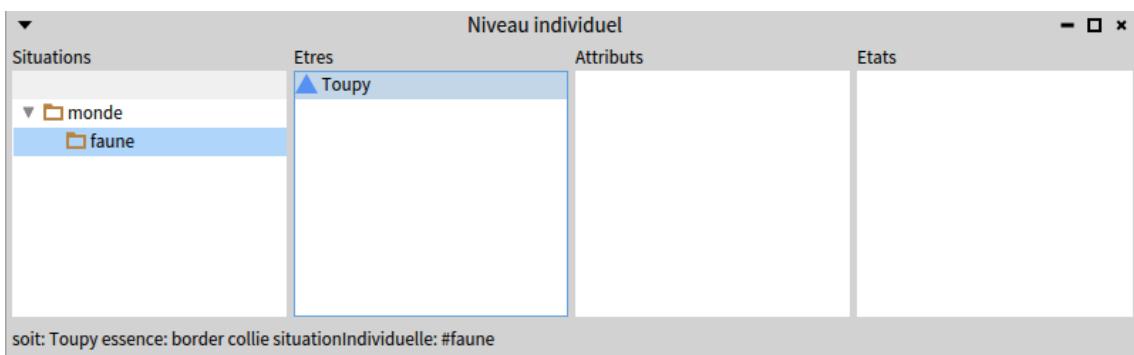
**border collie class**

<b>self</b>	animal
<b>all inst vars</b>	
<b>superclass</b>	
<b>methodDict</b>	
<b>format</b>	
<b>instanceVariables</b>	
<b>organization</b>	
<b>subclasses</b>	
<b>name</b>	

et sa situation générique est zoologie :

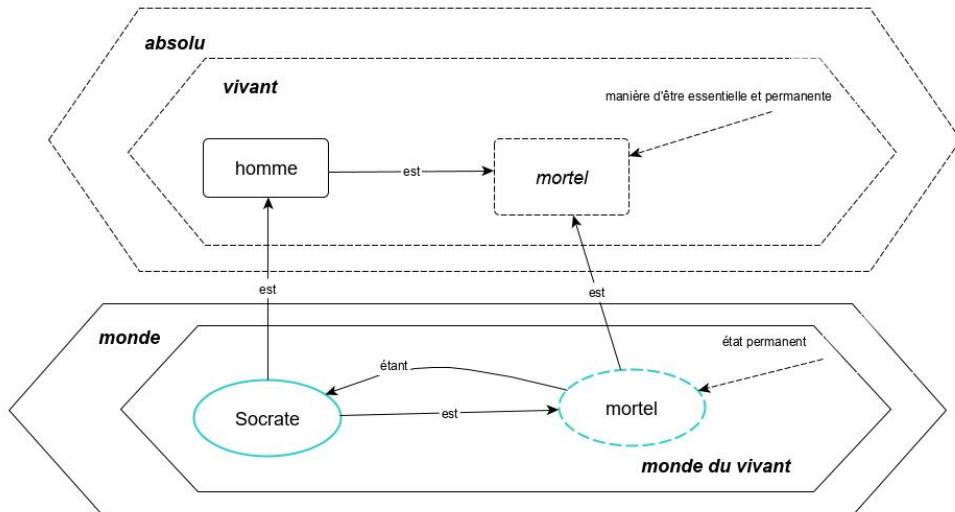


L'expression "`(monde getSituation: #faune) get: #Toupy`" permet de vérifier que Toupy est bien une instance de border collie présente dans la situation individuelle faune, ce qui se vérifie dans la fenêtre suivante :



## 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') .

```

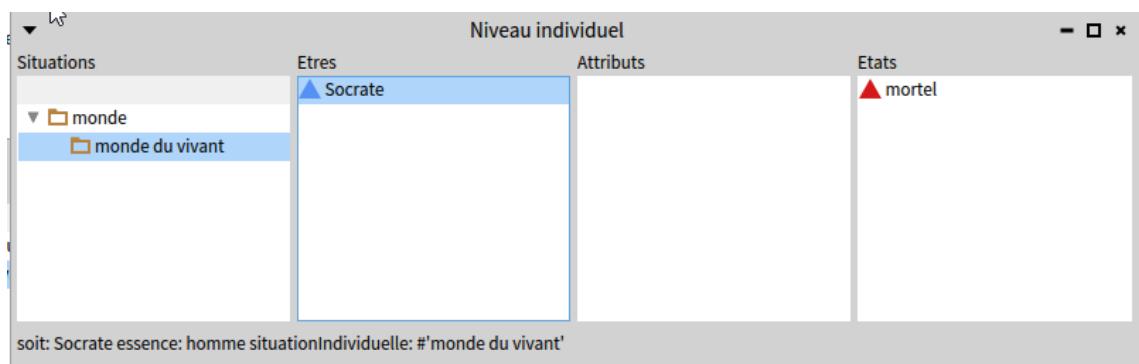
La troisième instruction définit la manière d'être "mortel" comme essentielle et permanente pour homme.

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

Niveau générique			
Situations	Essences	Attributs	Qualités
absolu vivant	homme		mortel
definition: #homme situation: #vivant			

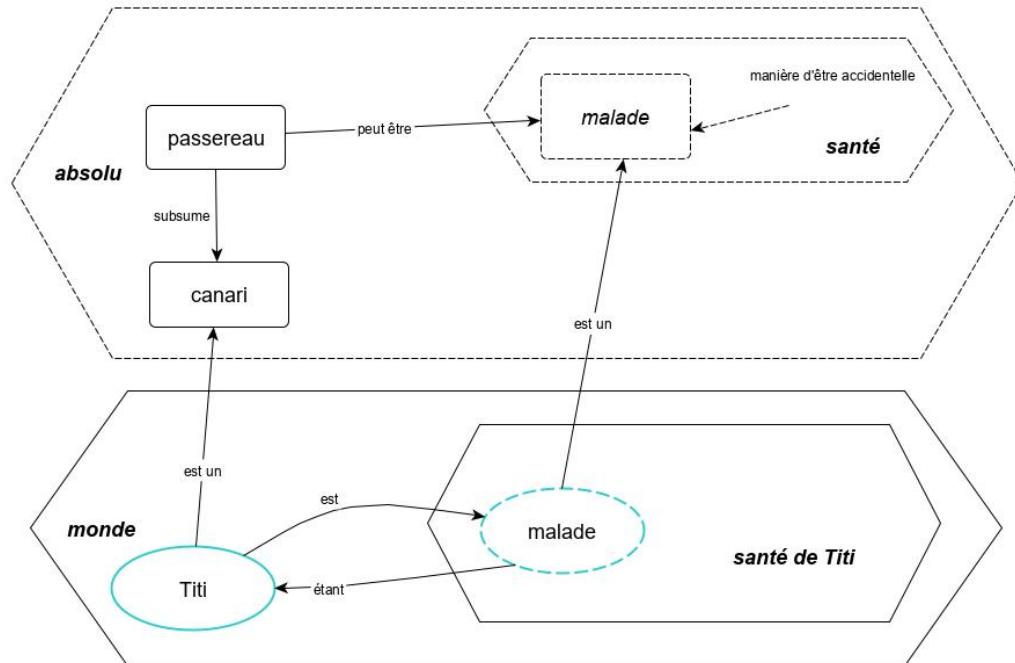
La petite icône rouge associée à la qualité "mort" indique qu'il s'agit d'une qualité essentielle.

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



## Exemple 10 : sur la notion de manière d'être accidentelle

Considérons le graphe suivant :

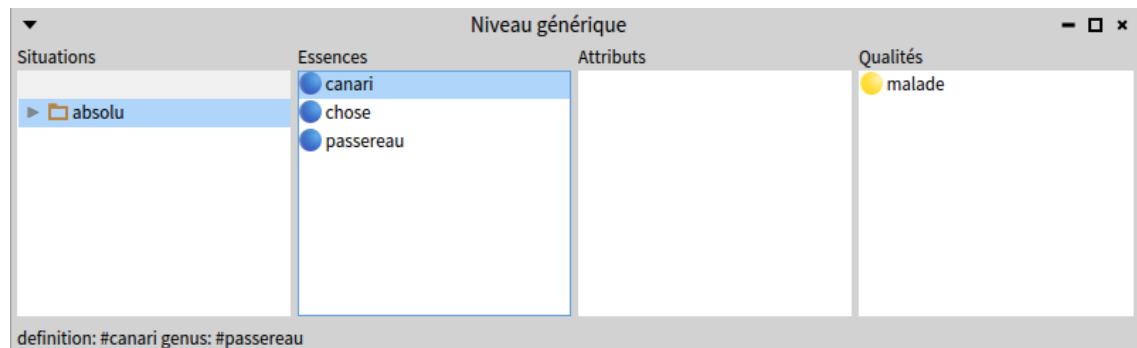


```

iceo definition: #passereau .
iceo definition: #canari genus: passereau .
iceo definitionSituation: #santé .
iceo definitionQualité: #malade situation: santé . passereau peutEtre:
    (santé get: #malade) .
iceo soit: #Titi essence: canari .
iceo soit: 'santé de Titi' situationGénérique: santé .
(monde get: #Titi) affecteEtat: (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 quatrième instruction).

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



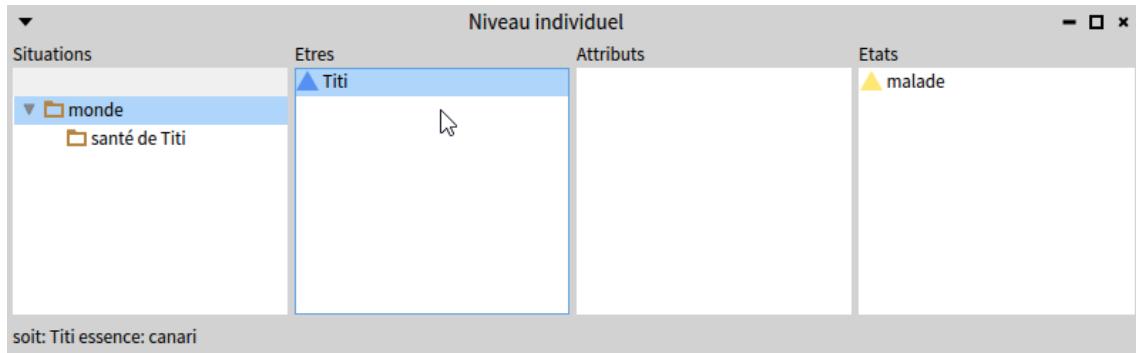
La petite icône jaune associée à la qualité "malade" indique qu'il s'agit d'une qualité accidentelle.

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 instruction.

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

C'est ce que confirme la fenêtre suivante :

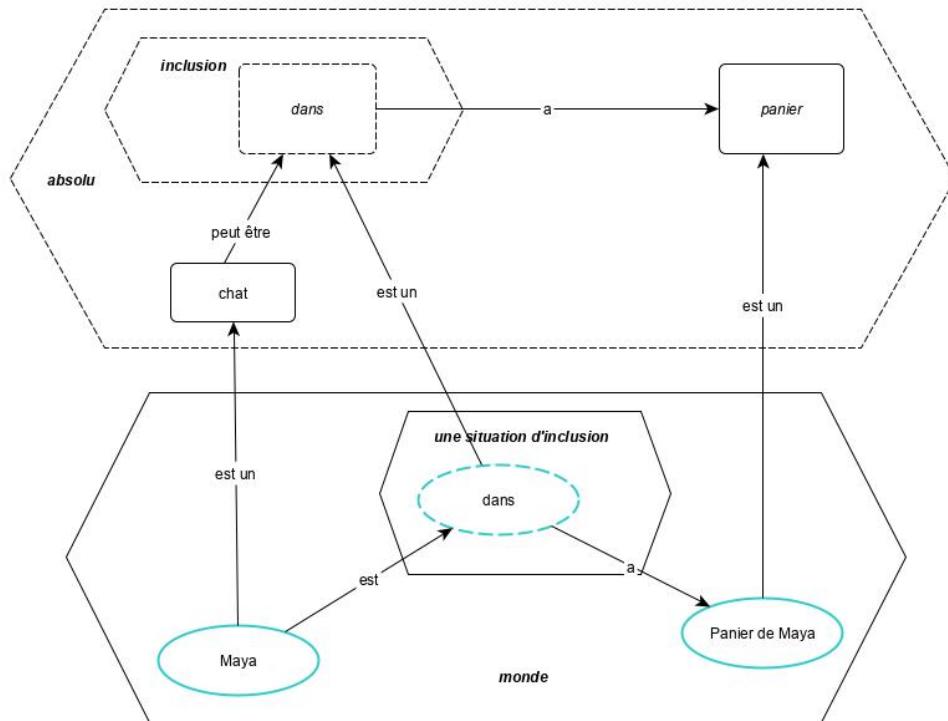


Le petit triangle jaune associé à l'état "malade" de Titi indique qu'il s'agit d'un état accidentel.

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"

```

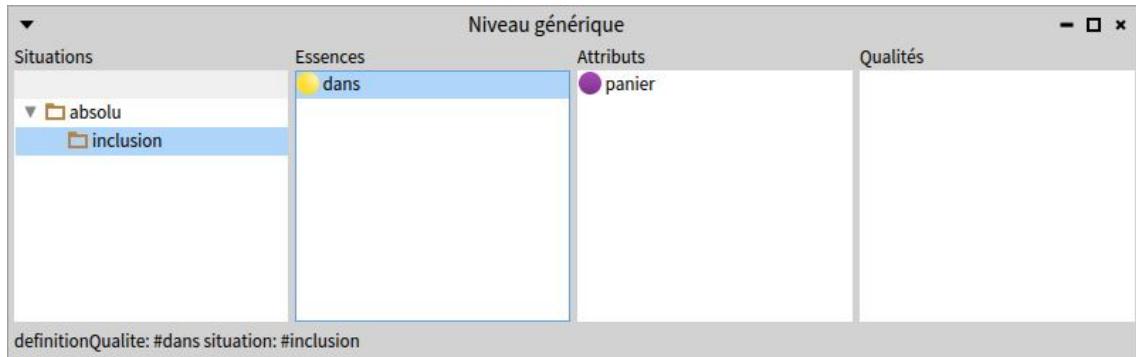
iceo definition: #chat.
iceo definition: #panier.
iceo definitionSituation: #inclusion.
iceo definitionQualite: #dans situation: inclusion.
(inclusion get: #dans) referenceEssence: panier.
chat peutEtre: (inclusion get: #dans).
iceo soit: 'une situation d'inclusion' situationGenerique: inclusion.
iceo soit: #Maya essence: chat.
iceo soit: 'Panier de Maya' essence: panier.
(monde get: #Maya) affecteEtat: (inclusion get: #dans) dansSituation: (
    monde get: 'une situation d'inclusion').
((monde get: #Maya) getEtat: #dans) attributionEtre: (monde get: 'Panier de
    Maya').

```

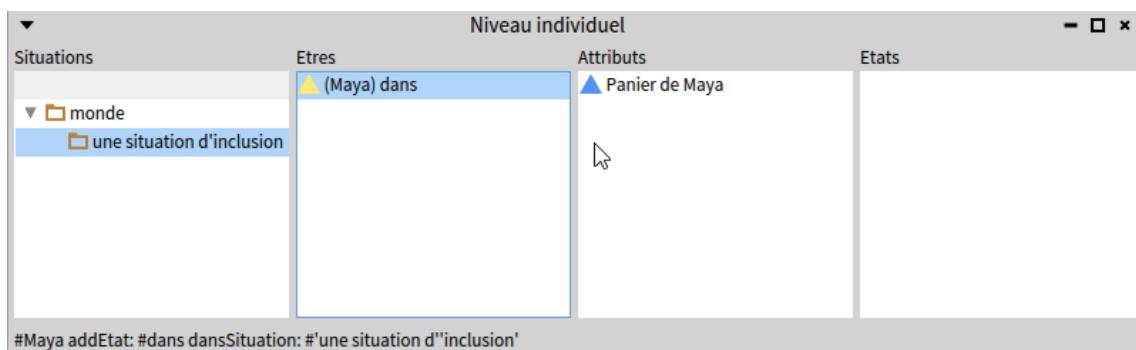
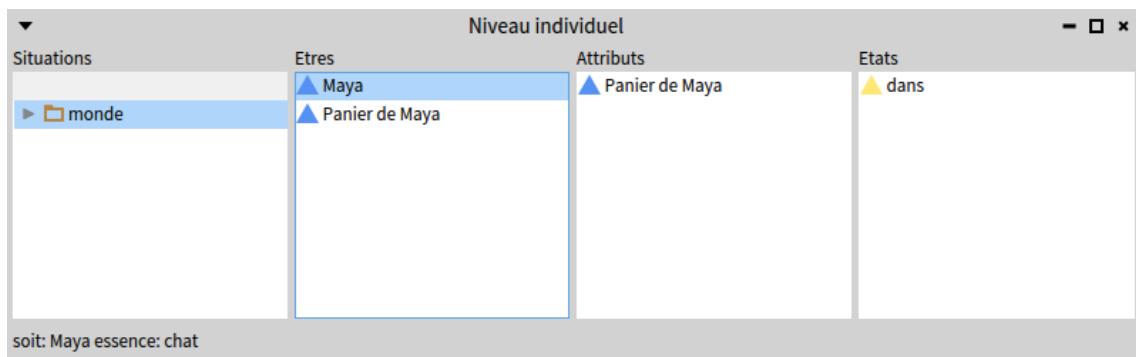
La fenêtre suivante montre que "dans" est une qualité accidentelle de chat :

Niveau générique			
Situations	Essences	Attributs	Qualités
absolu	chat chose panier		dans
definition: #chat			

et celle-ci que l'essence panier est attribut de la manière d'être "dans" :



Celles-ci montrent que Maya est "dans" son panier :



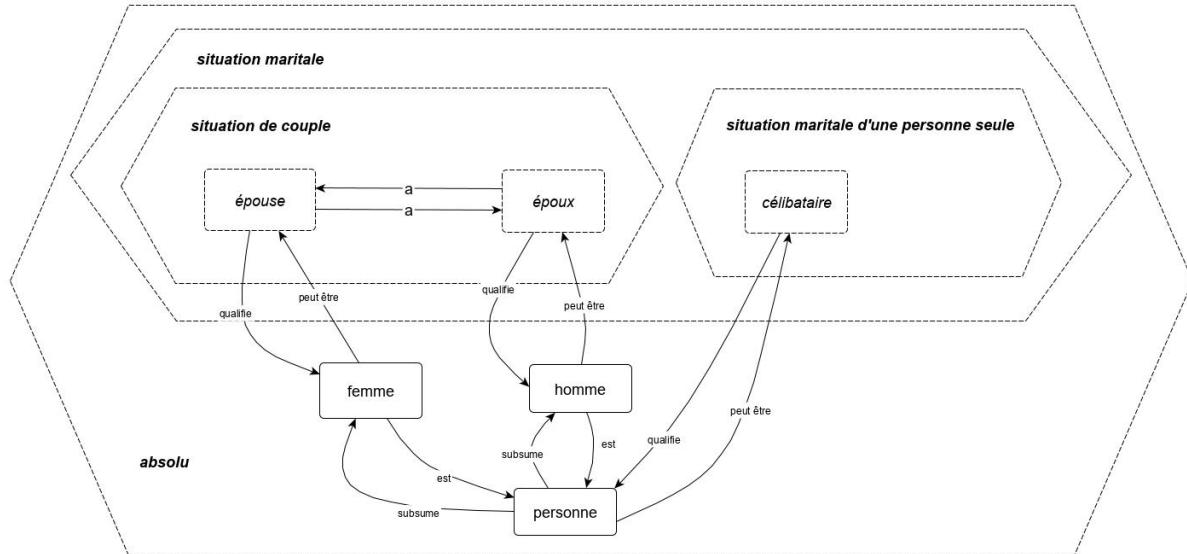
L'état "dans" qui s'affiche en tant qu'être (ce qui est normal, car un état est un être !) est précédé du nom de son étant (l'être qui est dans cet état) placé entre parenthèses.

L'expression "(monde get: #Maya) getEtresAttributsEnTantQue: (inclusion get: #dans)" donne : an OrderedCollection(panier de Maya)

Suivant la même logique, nous pourrions représenter toutes les 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).

```

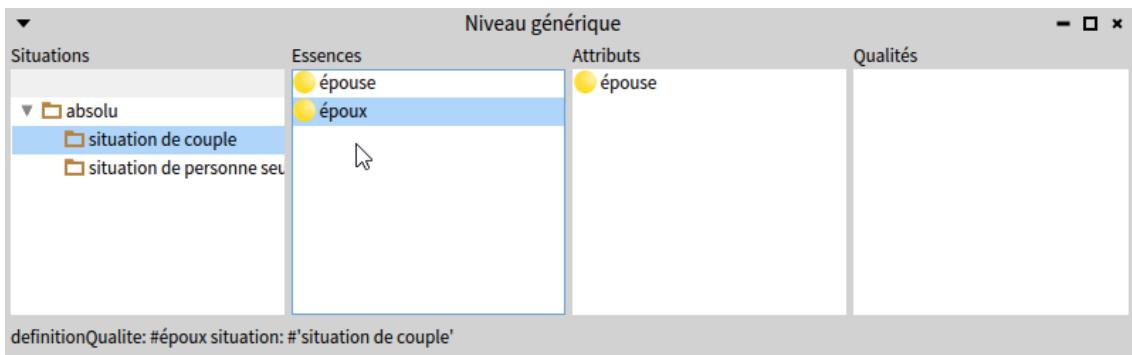
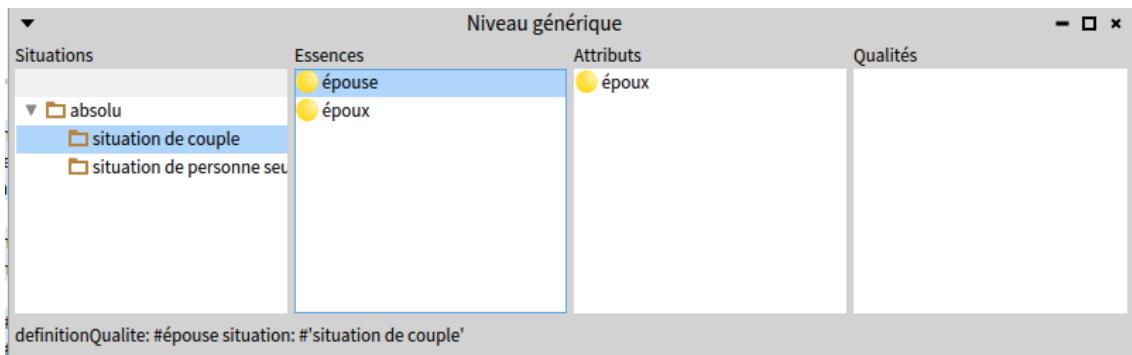
```

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).

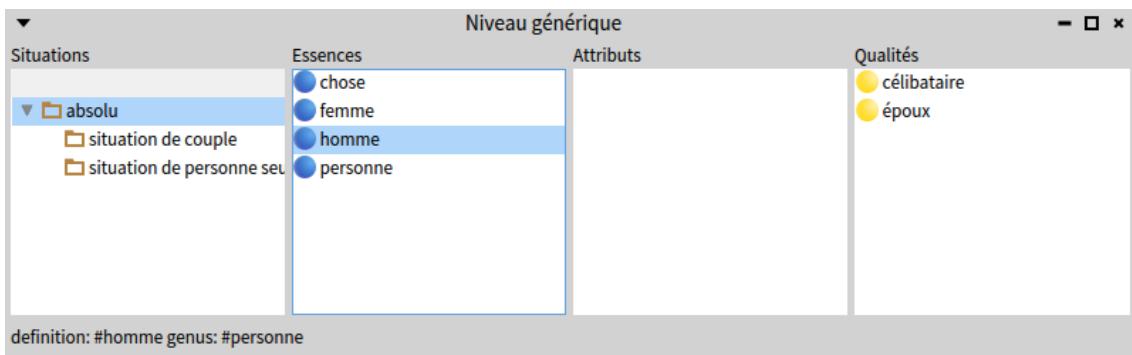
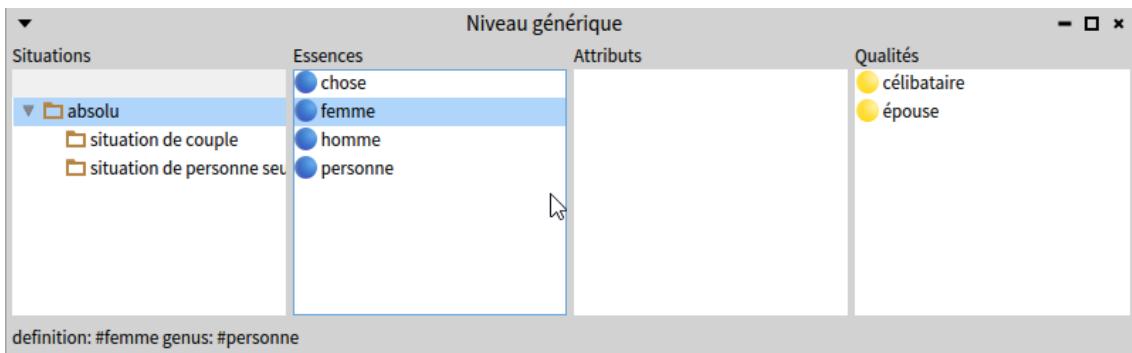
```

L'instruction encadrée définit une relation bidirectionnelle entre les manières d'être épouse et époux.

Ceci peut être vérifié dans les fenêtres suivantes :

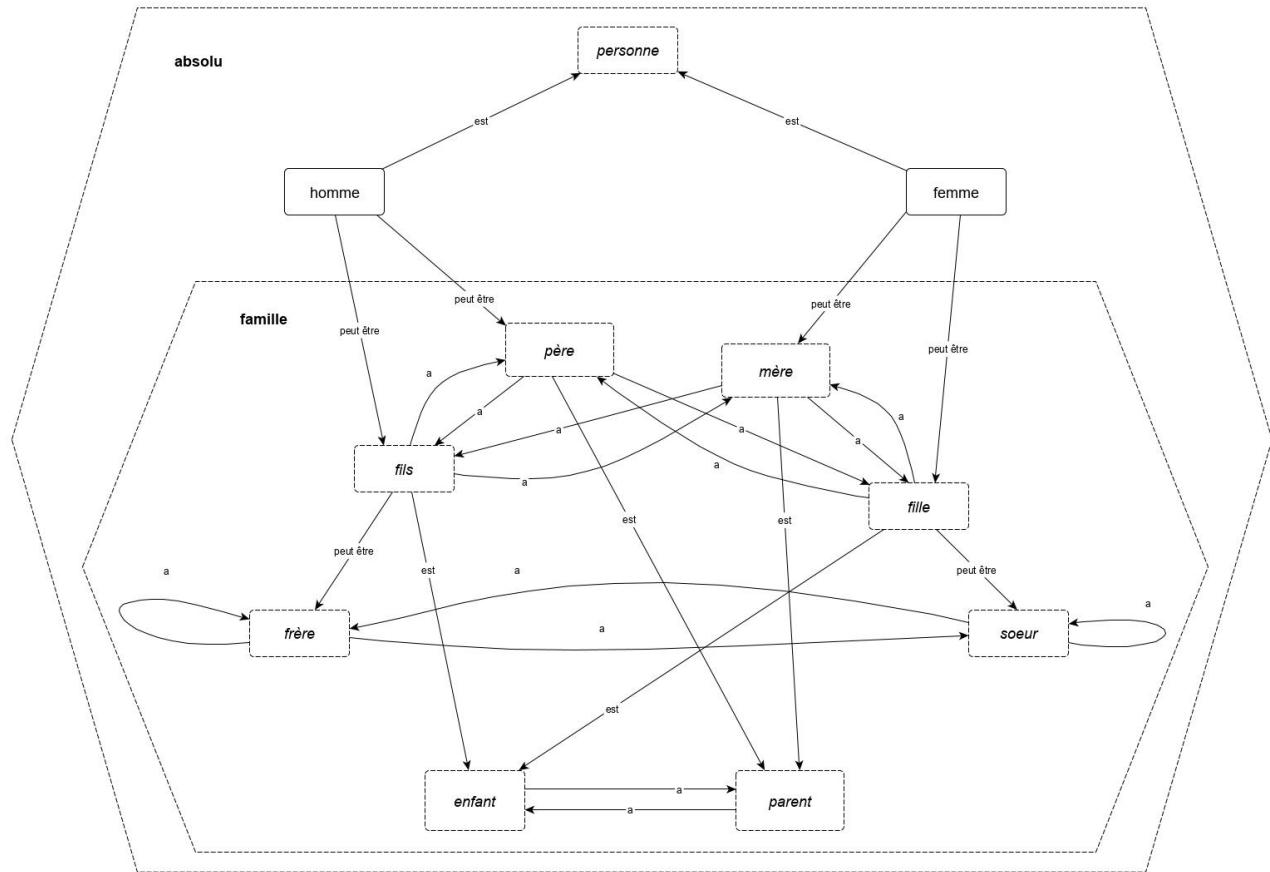


Les manières d'être possibles pour homme et femme apparaissentent dans les fenêtres suivantes :



## 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)

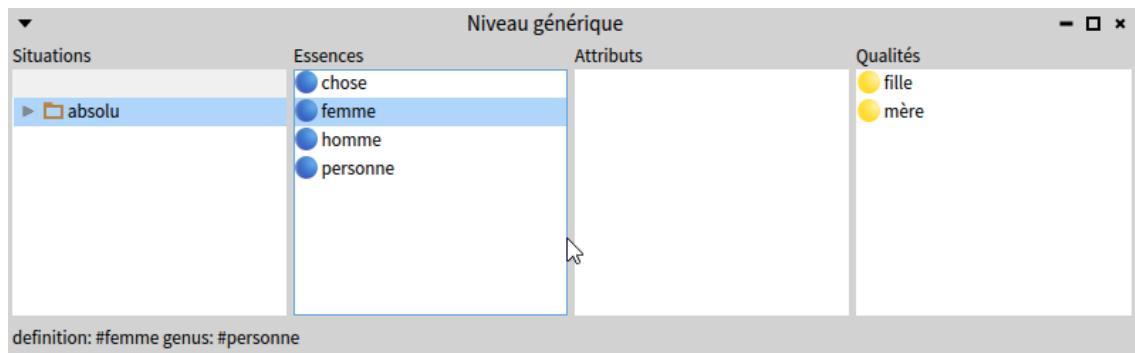
Et, comme nous l'avons déjà vu, une manière d'être peut avoir pour attributs d'autres manières d'être.

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

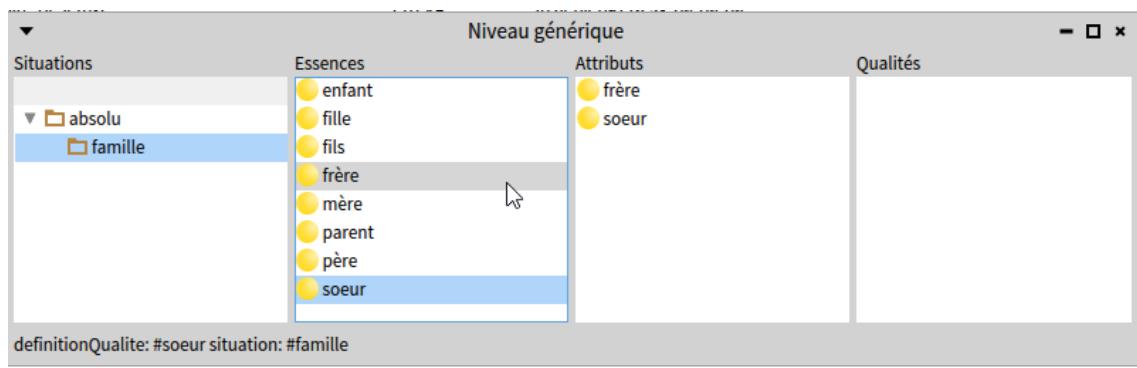
Cet exemple illustre le fait qu'une manière d'être peut être attribut d'elle-même.

(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 :

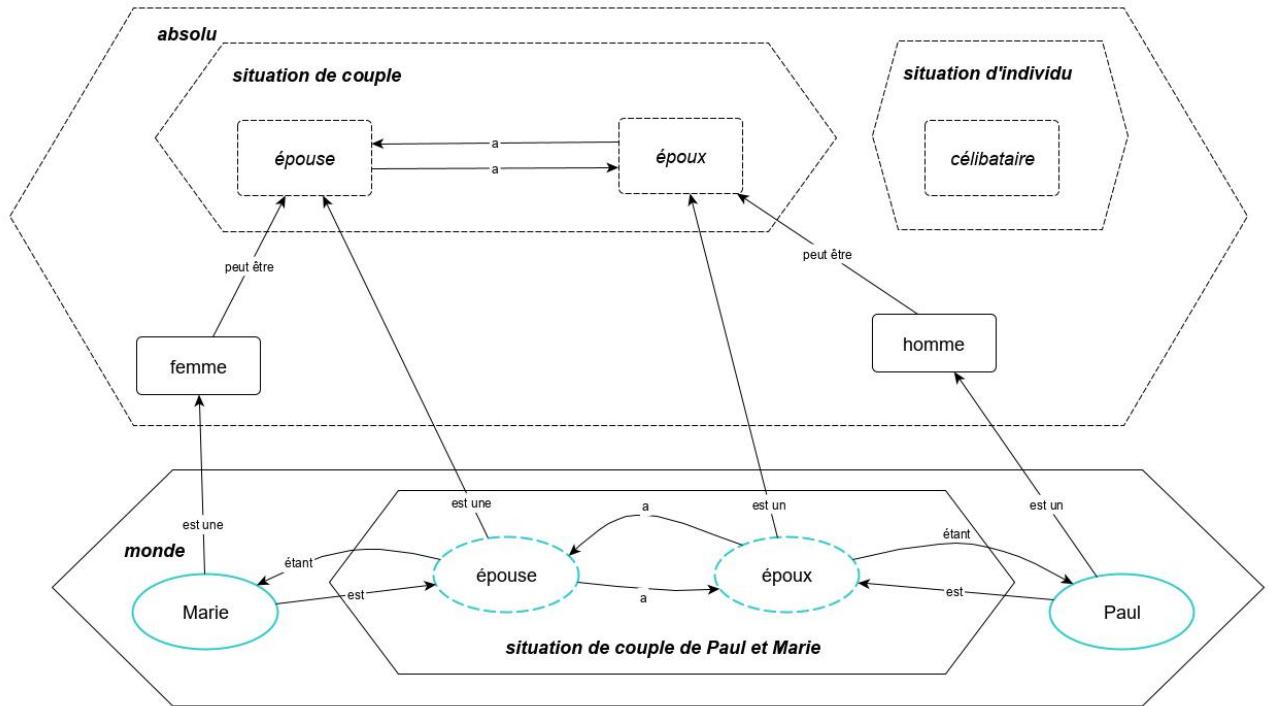


ou les attributs possibles de la manière d'être frère :



## 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) affecteEtat: ((absolu getSituation: 'situation de
couple') get: #épouse) dansSituation: (monde getSituation: 'situation de
couple de Paul et Marie').
(monde get: #Paul) affecteEtat: ((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'))).

```

L'instruction encadrée définit une relation bidirectionnelle entre les états épouse et époux

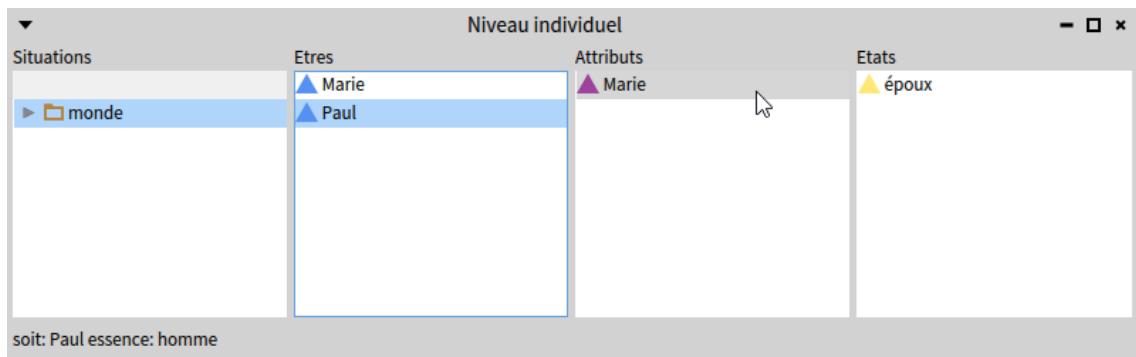
qui est rendue possible par la relation bidirectionnelle entre les manières d'être épouse et époux au niveau générique.

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(Marie).

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



La petite icône violette devant l'attribut Marie indique qu'il ne s'agit pas d'un attribut propre, mais acquis par Paul en sa qualité d'époux.

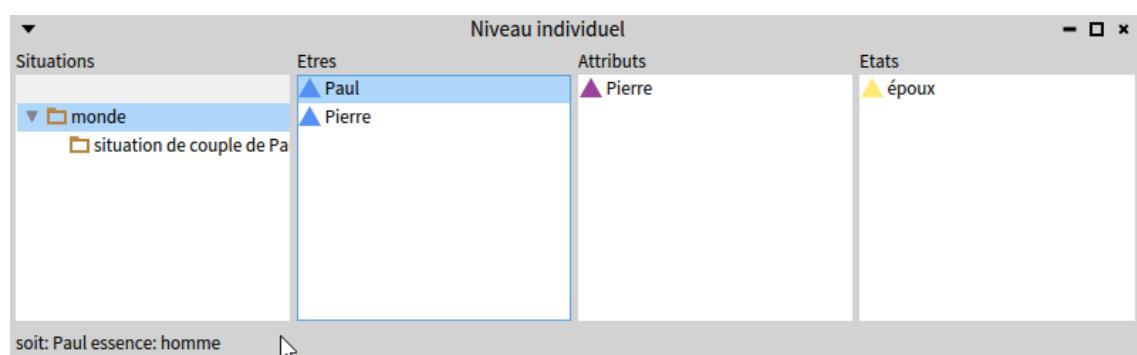
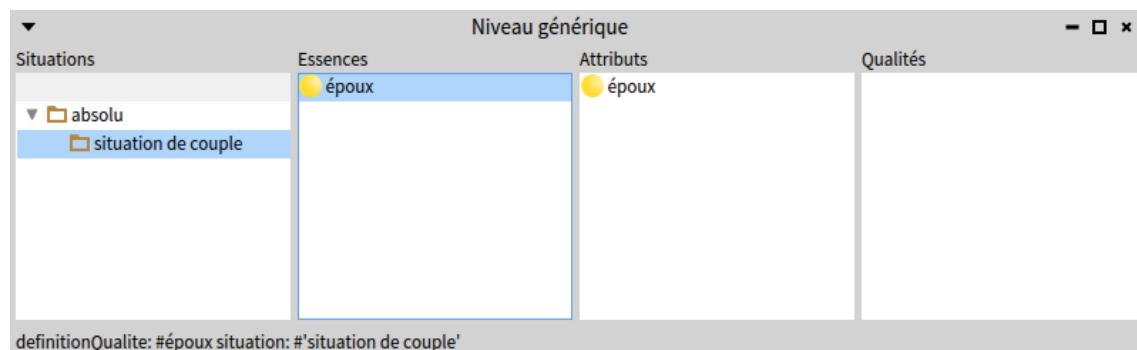
## Exemple 14\_2

Cet exemple est une variante du précédent qui représente un couple homosexuel :

```

iceo definition: #homme.
iceo definitionSituation: 'situation de couple'.
iceo definitionQualite: #époux situation: (absolu getSituation: 'situation de couple').
((absolu getSituation: 'situation de couple') get: #époux)
    associationQualite: ((absolu getSituation: 'situation de couple') get: #époux).
homme peutEtre: ((absolu getSituation: 'situation de couple') get: #époux).
iceo soit: #Pierre essence: homme.iceo soit: #Paul essence: homme.
iceo soit: 'situation de couple de Paul et Pierre' situationGenerique: (
    absolu getSituation: 'situation de couple').
(monde get: #Pierre) affecteEtat: ((absolu getSituation: 'situation de couple') get: #époux) dansSituation: (monde getSituation: 'situation de couple de Paul et Pierre').
(monde get: #Paul) affecteEtat: ((absolu getSituation: 'situation de couple') get: #époux) dansSituation: (monde getSituation: 'situation de couple de Paul et Pierre').
(((monde get: #Paul) getEtat: #époux dansSituation: (monde getSituation: 'situation de couple de Paul et Pierre')) associationEtat: ((monde get: #Pierre) getEtat: #époux dansSituation: (monde getSituation: 'situation de couple de Paul et Pierre'))).

```



## Exemple 14\_3

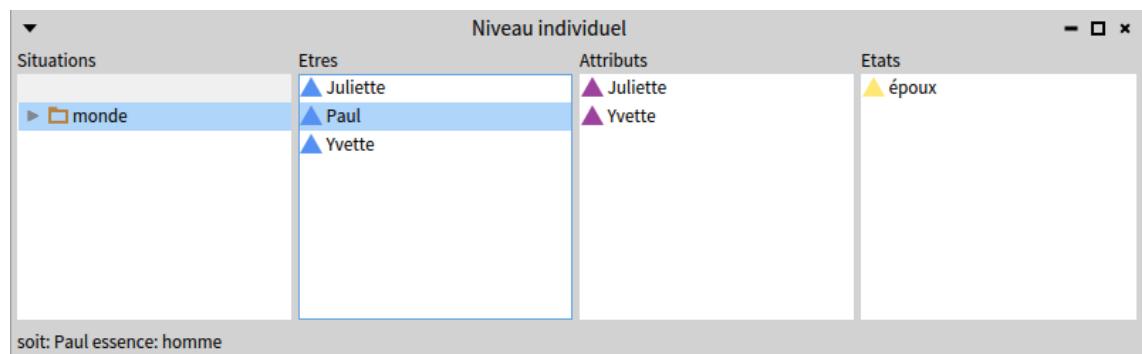
Cet exemple représente le cas d'un homme polygame :

```

iceo definition: #femme.
iceo definition: #homme.
iceo definitionSituation: 'situation de polycouple'.
iceo definitionQualite: #épouse situation: (absolu getSituation: 'situation
de polycouple').
iceo definitionQualite: #époux situation: (absolu getSituation: 'situation
de polycouple').
((absolu getSituation: 'situation de polycouple') get: #épouse)
    associationQualite: ((absolu getSituation: 'situation de polycouple')
get: #époux).
femme peutEtre: ((absolu getSituation: 'situation de polycouple')
get: #épouse).
homme peutEtre: ((absolu getSituation: 'situation de polycouple')
get: #époux).
iceo soit: #Yvette essence: femme.
iceo soit: #Juliette essence: femme.
iceo soit: #Paul essence: homme.
iceo soit: 'situation de polycouple de Paul' situationGenerique: (absolu
    getSituation: 'situation de polycouple').
(monde get: #Yvette) affecteEtat: ((absolu getSituation: 'situation de
    polycouple') get: #épouse) dansSituation: (monde getSituation: '
    situation de polycouple de Paul').
(monde get: #Juliette) affecteEtat: ((absolu getSituation: 'situation de
    polycouple') get: #épouse) dansSituation: (monde getSituation: '
    situation de polycouple de Paul').
(monde get: #Paul) affecteEtat: ((absolu getSituation: 'situation de
    polycouple') get: #époux) dansSituation: (monde getSituation: 'situation
    de polycouple de Paul').

(((monde get: #Paul) getEtat: #époux dansSituation: (monde getSituation:
    'situation de polycouple de Paul')) associationEtat: ((monde get: #
    Yvette) getEtat: #épouse dansSituation: (monde getSituation: '
    situation de polycouple de Paul))).  

(((monde get: #Paul) getEtat: #époux dansSituation: (monde getSituation:
    'situation de polycouple de Paul')) associationEtat: ((monde get: #
    Juliette) getEtat: #épouse dansSituation: (monde getSituation: '
    situation de polycouple de Paul))).
```



On constate que Paul en tant qu'époux "possède" deux épouses.

## Exemple 14\_4

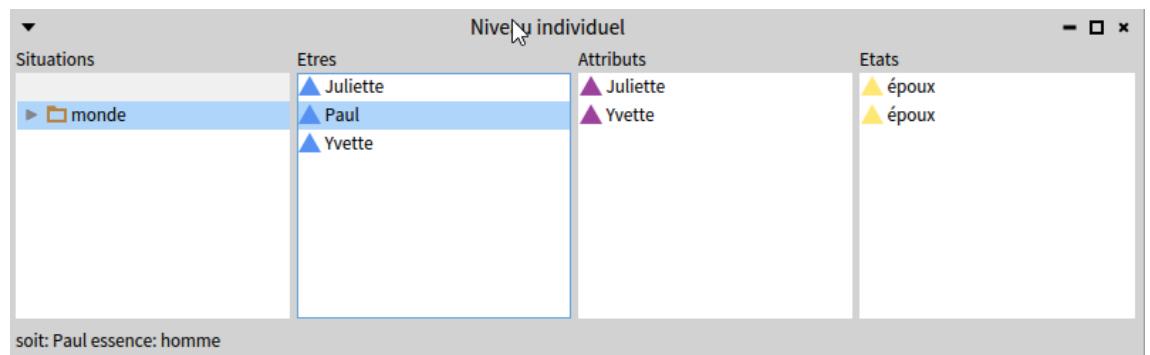
Cet exemple est une variante de l'exemple précédent où l'état d'époux de Paul est différent dans sa relation par rapport à Juliette ou Yvette :

```

iceo definition: #femme.
iceo definition: #homme.
iceo definitionSituation: 'situation de polycouple'.
iceo definitionQualite: #épouse situation: (absolu getSituation: 'situation de polycouple').
iceo definitionQualite: #époux situation: (absolu getSituation: 'situation de polycouple').
((absolu getSituation: 'situation de polycouple') get: #épouse)
    associationQualite: ((absolu getSituation: 'situation de polycouple')
        get: #époux).
femme peutEtre: ((absolu getSituation: 'situation de polycouple')
    get: #épouse).
homme peutEtre: ((absolu getSituation: 'situation de polycouple')
    get: #époux).
iceo soit: #Yvette essence: femme.
iceo soit: #Juliette essence: femme.
iceo soit: #Paul essence: homme.
iceo soit: 'situation de polycouple de Paul' situationGenerique: (absolu
    getSituation: 'situation de polycouple').
(monde get: #Yvette) affecteEtat: ((absolu getSituation: 'situation de polycouple')
    get: #épouse) dansSituation: (monde getSituation: 'situation de polycouple de Paul').
(monde get: #Juliette) affecteEtat: ((absolu getSituation: 'situation de polycouple')
    get: #épouse) dansSituation: (monde getSituation: 'situation de polycouple de Paul').

(monde get: #Paul) affecteEtat: ((absolu getSituation: 'situation de polycouple')
    get: #époux) dansSituation: (monde getSituation: 'situation de polycouple de Paul').
(monde get: #Paul) affecteEtat: ((absolu getSituation: 'situation de polycouple')
    get: #époux) dansSituation: (monde getSituation: 'situation de polycouple de Paul').
(((monde get: #Paul) getEtats: #époux dansSituation: (monde
    getSituation: 'situation de polycouple de Paul')) at: 1)
    associationEtat: ((monde get: #Yvette) getEtat: #épouse
        dansSituation: (monde getSituation: 'situation de polycouple de Paul'))).
(((monde get: #Paul) getEtats: #époux dansSituation: (monde
    getSituation: 'situation de polycouple de Paul')) at: 2)
    associationEtat: ((monde get: #Juliette) getEtat: #épouse
        dansSituation: (monde getSituation: 'situation de polycouple de Paul'))).

```

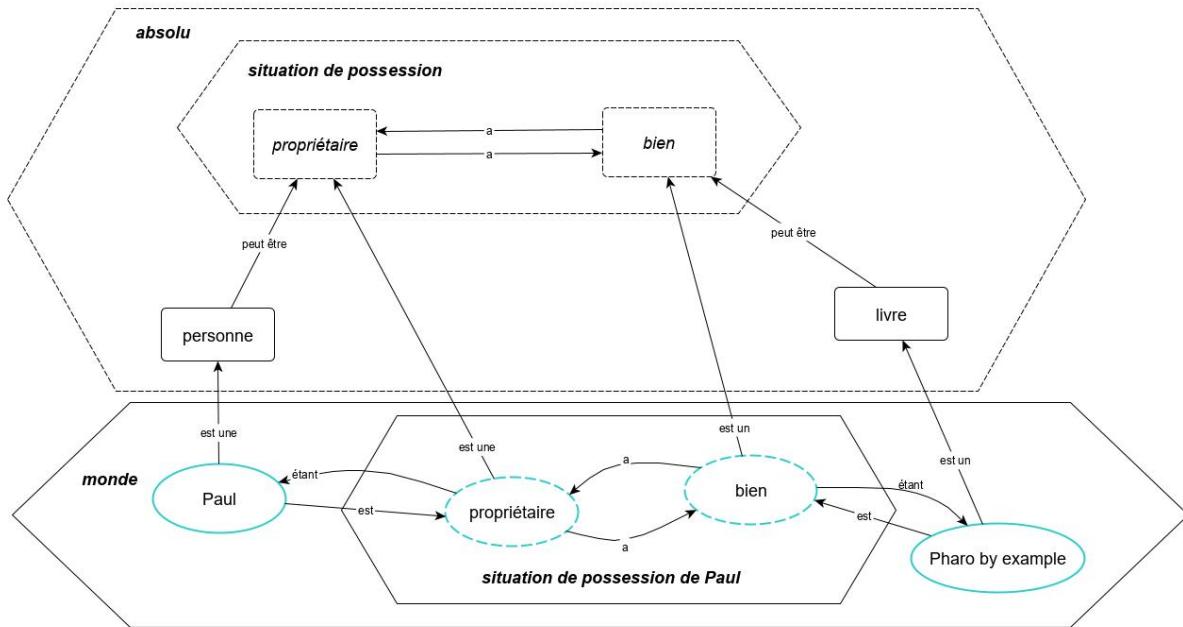


Cet exemple est plus informatif que le précédent car, étant donné qu'un état peut être dans un état, il permettrait de préciser que Paul est un mauvais époux pour Yvette et un bon époux pour Juliette !

## Exemple 14\_5

Cet exemple illustre la relation possible entre deux êtres qui ne soit pas une relation d'inclusion (composition).

Il représente le fait qu'une personne puisse être propriétaire d'un livre :

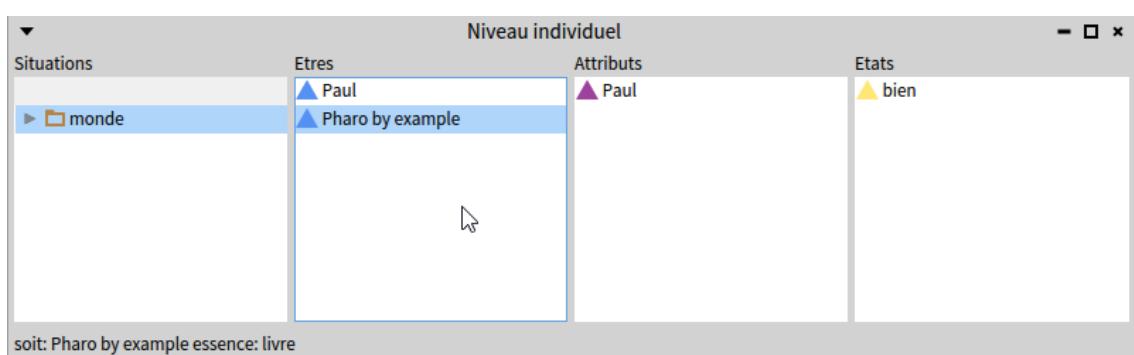
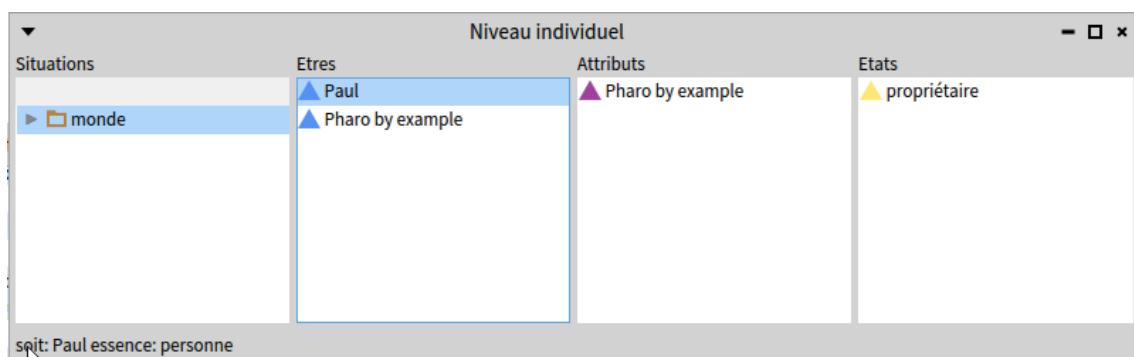
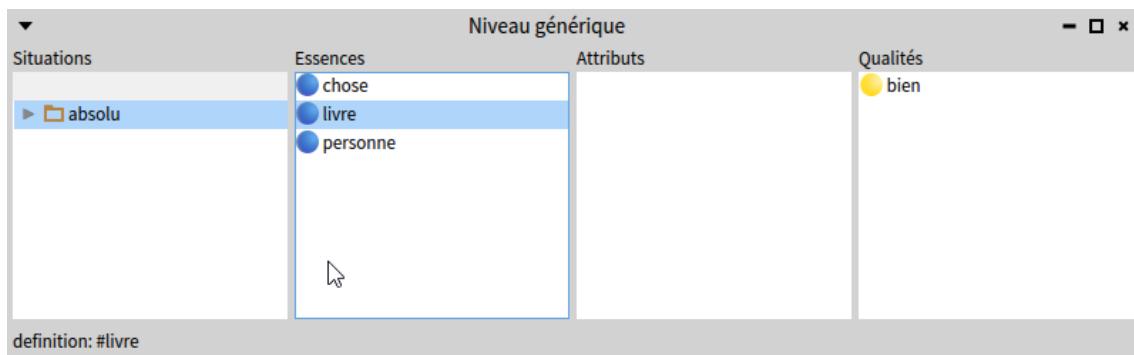
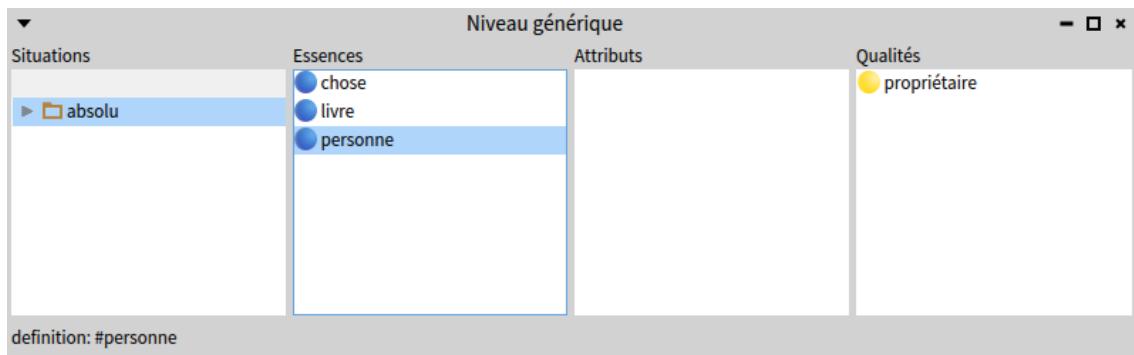


```

iceo definition: #personne .
iceo definition: #livre .
iceo definitionSituation: #possession .
iceo definitionQualite: #propriétaire situation: possession .
iceo definitionQualite: #bien situation: possession .
( possession get: #propriétaire ) associationQualite: ( possession get:#bien ) .
personne peutEtre: ( possession get: #propriétaire ). livre peutEtre: (
    possession get: #bien ) .
iceo soit: #Paul essence: personne .
iceo soit: 'Pharo by example' essence: livre .
iceo soit: 'possession de Paul' situationGenerique: possession .
(monde get: #Paul) affecteEtat: ( possession get: #propriétaire )
    dansSituation: (monde getSituation: 'possession de Paul') .
(monde get: 'Pharo by example') affecteEtat: ( possession get: #bien )
    dansSituation: (monde get: 'possession de Paul') .
((monde get: #Paul) getEtat: #propriétaire dansSituation: (monde get: 'possession de Paul')) associationEtat: ((monde get: 'Pharo by example') getEtat: #bien dansSituation: (monde get: 'possession de Paul')) .

```

Le résultat de cette construction apparaît dans les fenêtres suivantes :



On voit que l'attribut 'Pharo by example' de Paul lui est conféré par son état de propriétaire (icône violette).

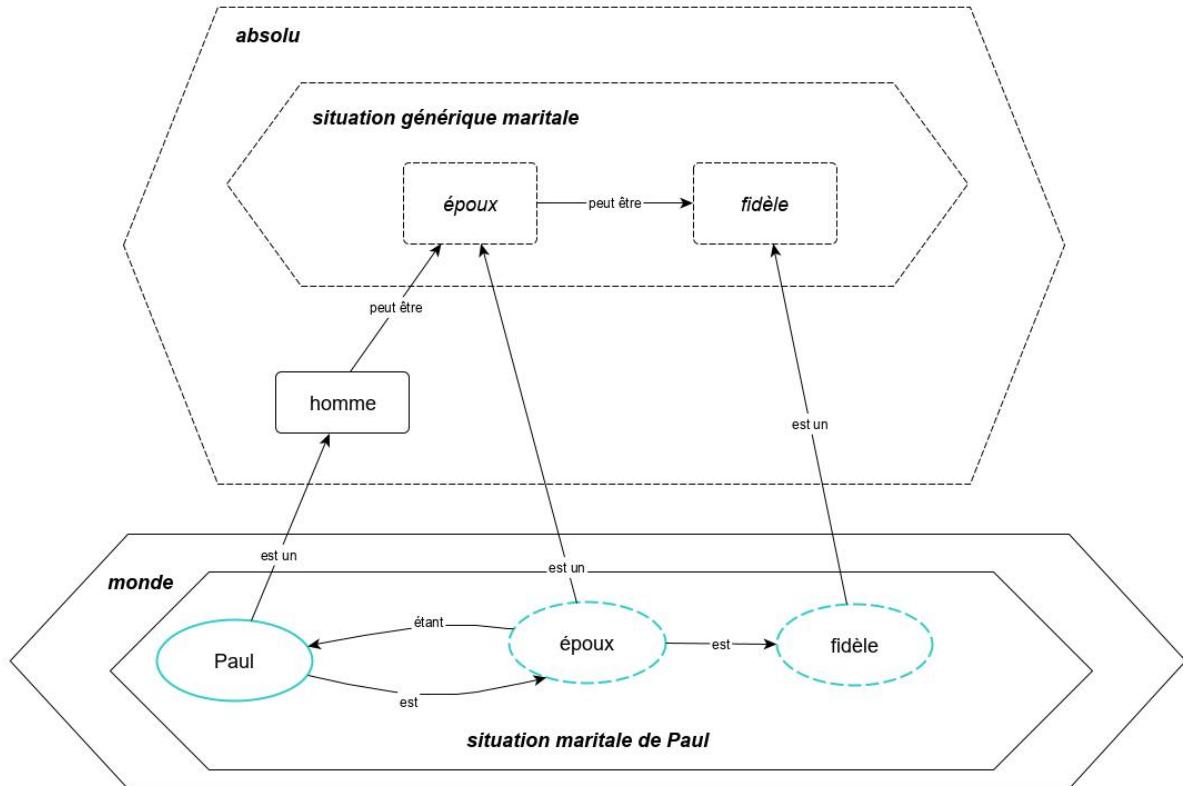
Le fait de dire que le propriétaire d'un livre est une personne (comme nous l'avons fait dans l'exemple 7) est un raccourci de langage.

Bien sûr, un livre ne rentre pas dans la composition d'une personne, et réciproquement !

La même logique s'appliquerait pour dire que Paul a un crayon, une veste, ...

## Exemple 15 : sur la notion d'état dans un état

Considérons le graphe suivant :



```

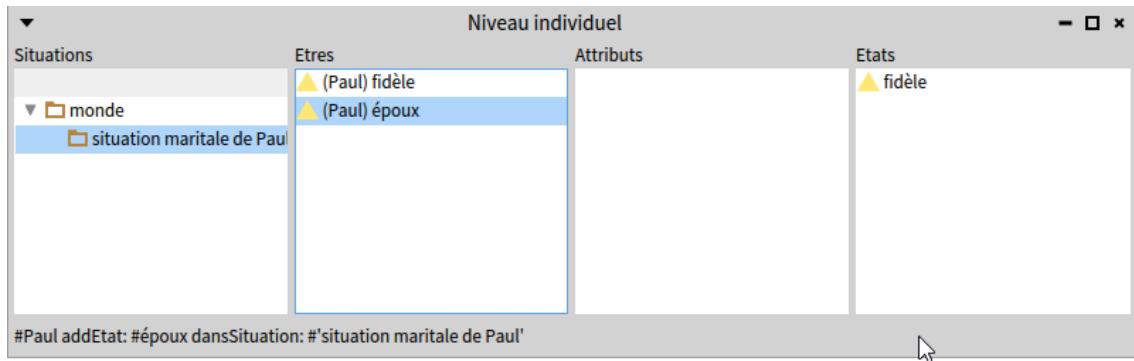
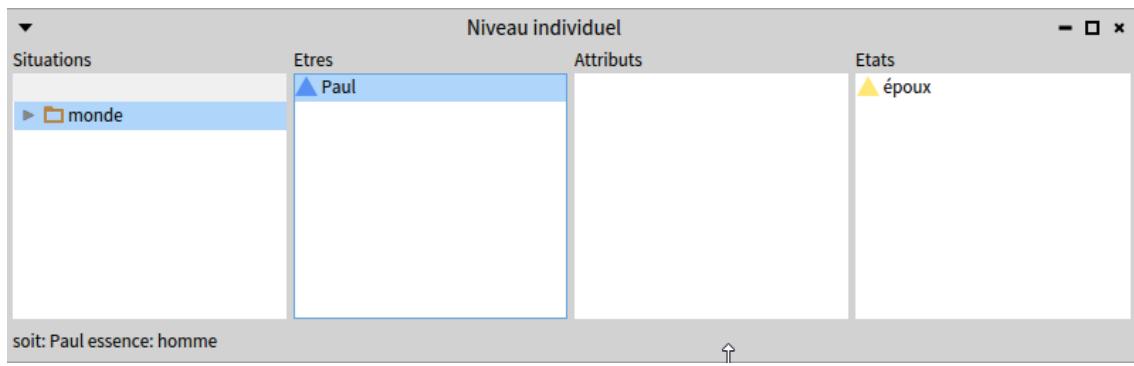
iceo definition: #homme.
iceo definitionSituation: 'situation générique maritale'.
iceo definitionQualite: #époux situation:(absolu getSituation: 'situation g
éénérique maritale').
iceo definitionQualite: #fidèle situation: (absolu getSituation:'situation g
éénérique maritale').
homme peutEtre: ((absolu 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) affecteEtat: ((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')) affecteEtat: ((absolu getSituation: 'situation géné
rique 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(fidèle)

Ceci est visualisé dans les fenêtres suivantes :

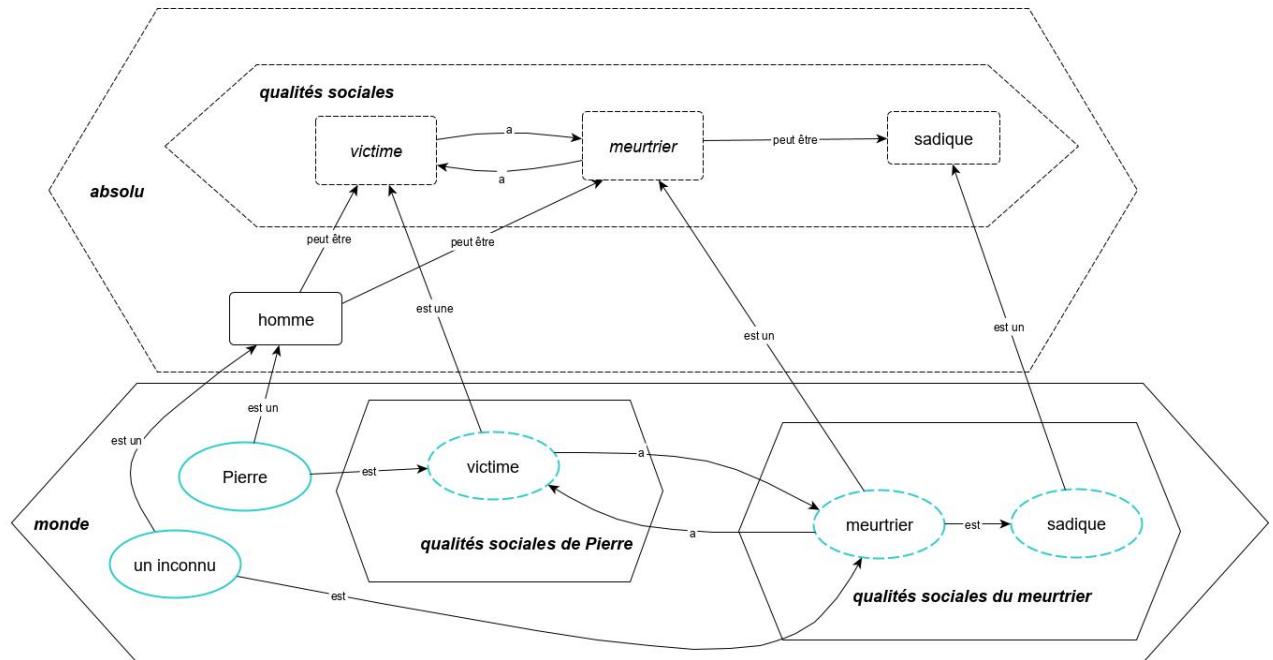


Noter que l'expression " ((monde get: #Paul) getEtat: #fidèle dansSituation: (monde get: 'situation maritale de Paul')) "

ne donnerait 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 :



Il 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) affecteEtat: ((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)
    affecteEtat: ((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).

"((monde get: 'qualités sociales du meurtrier') get: #meurtrier) getEtats"
donne : an OrderedCollection(sadique)

" (monde get: #Pierre) getEtats" donne : an OrderedCollection(victime)

```

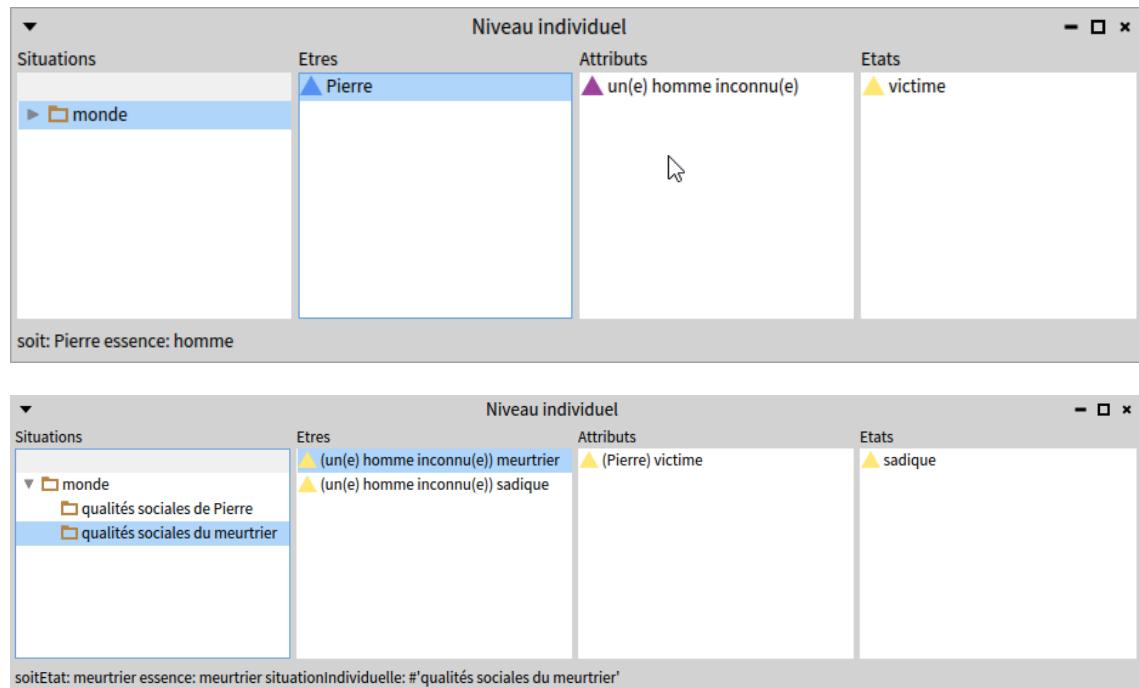
Enfin, l'expression

```

"((monde get: 'qualités sociales du meurtrier') get: #meurtrier) getEtant"
donne : 'un(e) homme inconnu(e)'

```

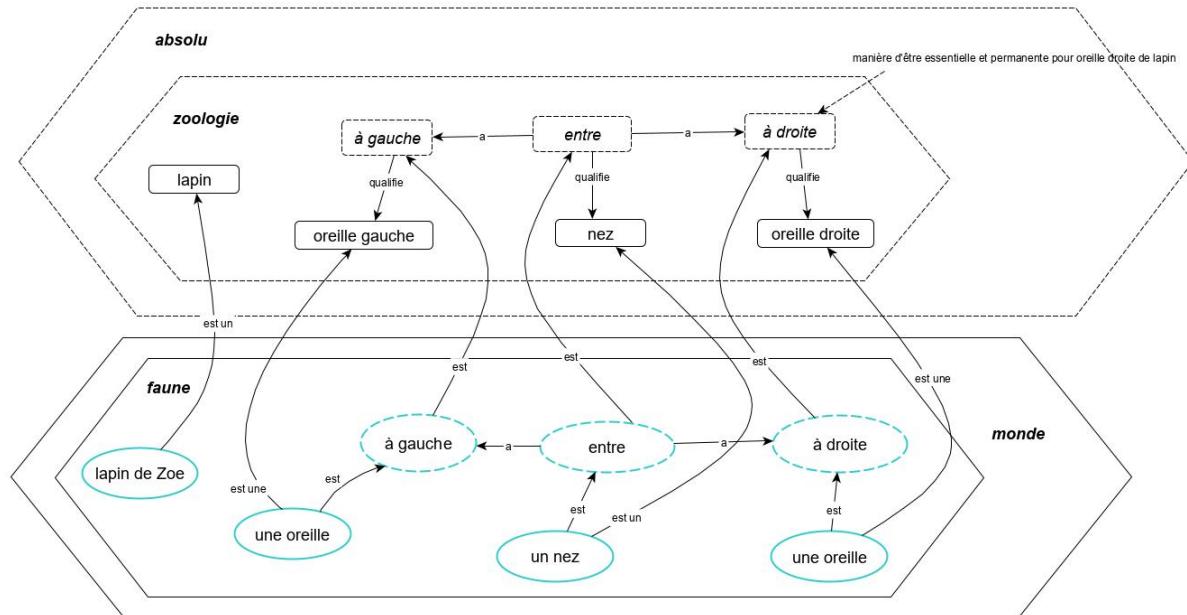
Ceci est visualisé dans les fenêtres suivantes :



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:
        'à droite') getQualite: 'entre') .

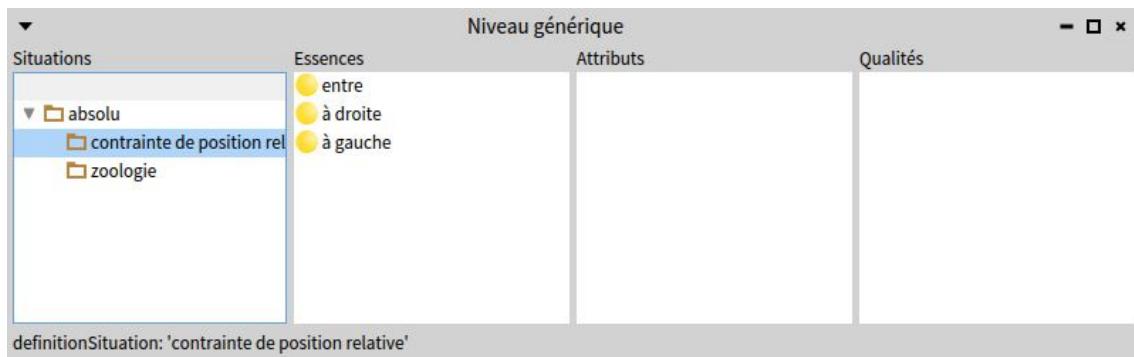
```

```

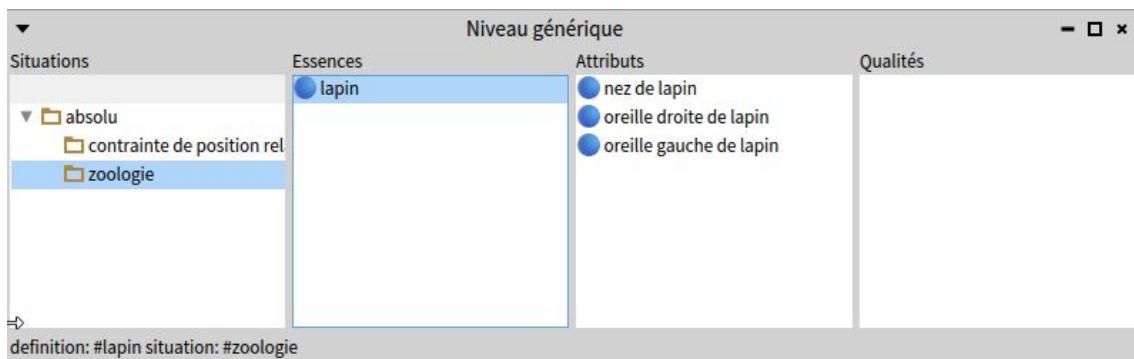
        oreille droite') getQualite: 'à droite').
iceo soit: #faune situationGenerique: zoologie
iceo soit: 'Lapin de Zoé' essence: (zoologie get: #lapin)
situationIndividuelle: (monde get: #faune)
(((monde get: 'Lapin de Zoé') getEtreAttribut: #nez) getEtat: #entre)
associationEtat: (((monde get: 'Lapin de 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')

```

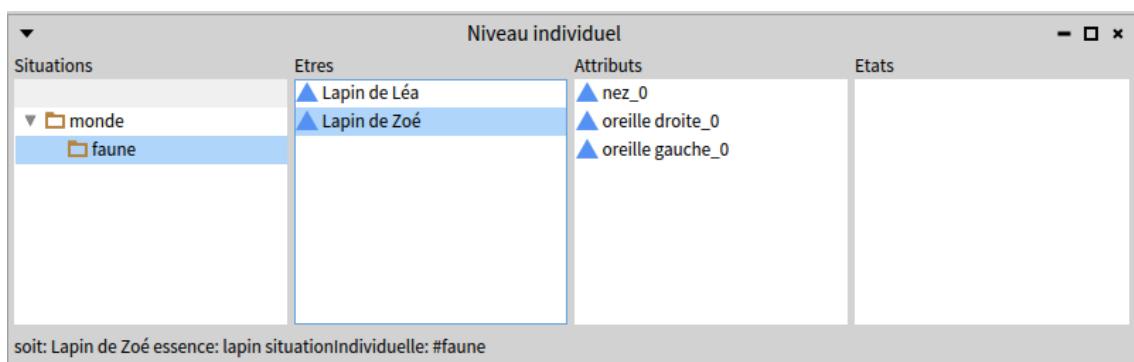
Un SgBrowser visualise les manières d'être définies dans la situation générique "contrainte de position relative" :



L'essence lapin est définie dans la situation zoologie :



Voici une visualisation du lapin de Zoé :



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

L'inspection de

```
"((monde get: #faune) get: 'Lapin de Zoé') getEtreAttribut: #nez) getEtats at: 1"
```

donne :

Inspector on entre		
an entre (entre)		
Raw	Breakpoints	Meta
: Variable		: Value
↳ self		entre
▶ { } structure		an OrderedCollection [2 items] (à gauche à droite)
▶ ⓒ situationDefinition		nez_0
❼ isSituation		false
❼ isEtat		true
❼ etats		nil
▶ ⓒ etant		nez_0
▶ ⌂ nom		#entre

Faisons l'inspection du nez :

Inspector on nez_0		
a nez (nez_0)		
Raw	Breakpoints	Meta
: Variable		: Value
❼ self		nez_0
❼ structure		nil
▶ ⓒ situationDefinition		Lapin de Zoé
❼ isSituation		false
❼ isEtat		false
▶ ⌂ etats		an OrderedCollection [1 item] (entre)
❼ etant		nil
▶ ⌂ nom		#nez_0

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

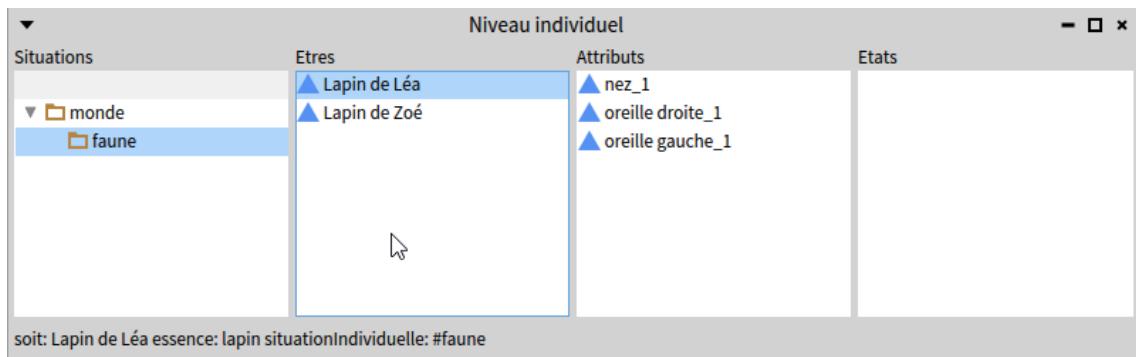
a à gauche (à gauche)		
Raw	Breakpoints	Meta
: Variable		: Value
❼ self		à gauche
▶ { } structure		an OrderedCollection [1 item] (entre)
▶ ⓒ situationDefinition		oreille gauche_0
❼ isSituation		false
❼ isEtat		true
❼ etats		nil
▶ ⓒ etant		oreille gauche_0

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

Créons 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 le nez et les oreilles des deux lapins sont différents, ce qui est logique.

Mais qu'en est-il de leurs états ? Faisons un test d'identité 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 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.

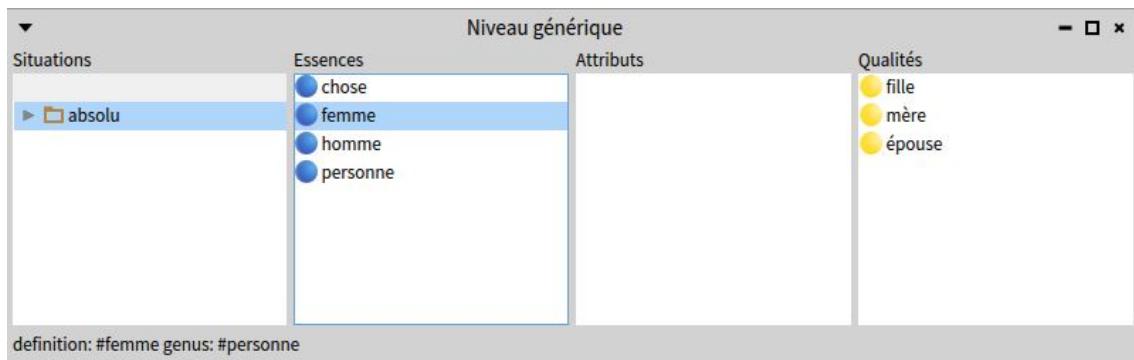
## Exemple 18 : sur la notion de famille

Cet exemple repart de l'exemple 13 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 :

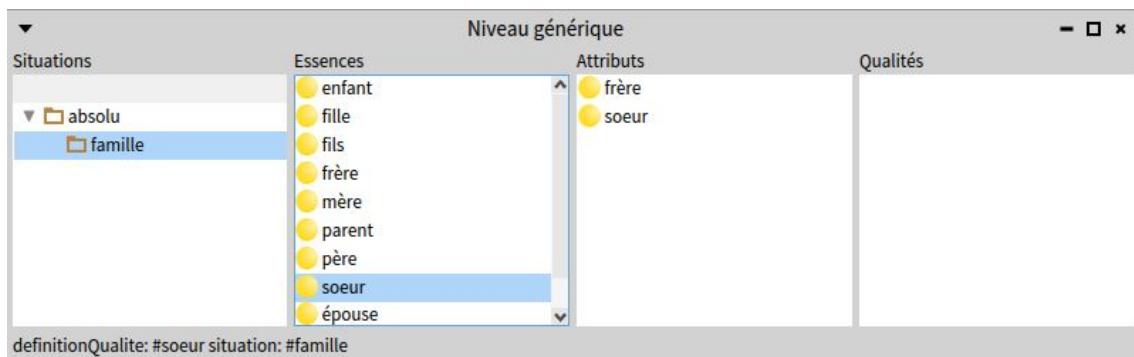
```
iceo definition: #personne.  
iceo definition: #femme genus: personne.  
iceo definition: #homme genus: personne.  
iceo definitionSituation: 'famille'.  
iceo definitionQualite: #parent situation: (absolu getSituation: #famille).  
iceo definitionQualite: #enfant situation: (absolu getSituation: #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).
```

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

Ainsi par exemple, voici les manières d'être possibles pour l'essence femme :



et les attributs possibles de la manière d'être soeur :



Au niveau individuel commençons par créer les deux époux Charles et Hermeline :

```

iceo soit: #Gonthier situationGenerique: famille .
iceo soit: #Hermeline essence: femme .
iceo soit: #Charles essence: homme .
(monde get: #Hermeline) affecteEtat: (famille get: #épouse) dansSituation:
    (monde getSituation: #Gonthier) .
(monde get: #Charles) affecteEtat: (famille get: #époux) dansSituation: (
    monde getSituation: #Gonthier) .
((monde get: #Charles) getEtat: #époux) associationEtat: ((monde get: #
    Hermeline) getEtat: #épouse dansSituation: (monde getSituation: #
    Gonthier)) .

```

ICEO étant implanté en Smalltalk, il est possible de mixer du code ICEO avec du code Smalltalk.

Nous en avons profité pour écrire une méthode de l'essence femme qui correspond à la mise au monde d'un enfant :

```

femme compile: ' metAuMonde: prénom famille: uneFamille essence: uneEssence
| époux |
  ico soit: prénom essence: uneEssence.
  (self getEtat: #épouse) notNil ifTrue: [époux := ((self getEtresAttributsQuiSont: (famille get: #époux)) at: 1].
  (self getEtat: #mère) ifNil: [self affecteEtat: (famille get: #mère) dansSituation: uneFamille].
  (époux getEtat: #père) ifNil: [époux affecteEtat: (famille get: #père) dansSituation: uneFamille].
  uneEssence getNom == #homme
    ifTrue: [ (monde get: prénom) affecteEtat: (famille get: #fils) dansSituation: uneFamille.
    (self getEtat: #mère) associationEtat: ((monde get: prénom) getEtat: #fils dansSituation: uneFamille).
    (époux getEtat: #père) associationEtat: ((monde get: prénom) getEtat: #fils dansSituation: uneFamille).

  "Si la femme a déjà des enfants, le dernier né est leur frère"
  "l"étant de ce frère est l"état fils du nouveau né"
  "l"état frère de l"état fils du nouveau né sera ici le même pour tous les autres enfants"
  ((self getEtresAttributsQuiSont: (famille get: #enfant)) size > 0) ifTrue: [
    ((monde get: prénom) getEtat: #frère) ifNil: [ ((monde get: prénom) getEtat: #fils) affecteEtat: (famille get: #frère) dansSituation: uneFamille].
    ((self getEtresAttributsQuiSont: (famille get: #enfant)) copyWithout: (monde get: prénom)) do: [:each |
      (each getEtat: #fille) ifNotNil: [
        ((each getEtat: #fille) getEtat: #soeur) ifNil: [(each getEtat: #fille) affecteEtat: (famille get: #soeur) dansSituation: uneFamille].
        ((each getEtat: #fille) getEtat: #soeur) associationEtat: (((monde get: prénom) getEtat: #fils) getEtat: #frère)].
      (each getEtat: #fils) ifNotNil: [
        ((each getEtat: #fils) getEtat: #frère) ifNil: [(each getEtat: #fils) affecteEtat: (famille get: #frère) dansSituation: uneFamille].
        ((each getEtat: #fils) getEtat: #frère) associationEtat: (((monde get: prénom) getEtat: #fils) getEtat: #frère)]]].
    ifFalse: [ (monde get: prénom) affecteEtat: (famille get: #fille) dansSituation: uneFamille.
    (self getEtat: #mère) associationEtat: ((monde get: prénom) getEtat: #fille dansSituation: uneFamille).
    (époux getEtat: #père) associationEtat: ((monde get: prénom) getEtat: #fille dansSituation: uneFamille).

  "Si la femme a déjà des enfants, le dernier né est leur soeur"
  "l"étant de cette soeur est l"état fille du nouveau né"
  "l"état soeur de l"état fille du nouveau né sera ici le même pour tous les autres enfants"
  ((self getEtresAttributsQuiSont: (famille get: #enfant)) size > 0) ifTrue: [
    ((monde get: prénom) getEtat: #soeur) ifNil: [ ((monde get: prénom) getEtat: #fille) affecteEtat: (famille get: #soeur) dansSituation: uneFamille].
    ((self getEtresAttributsQuiSont: (famille get: #enfant)) copyWithout: (monde get: prénom)) do: [:each |
      (each getEtat: #fille) ifNotNil: [
        ((each getEtat: #fille) getEtat: #soeur) ifNil: [(each getEtat: #fille) affecteEtat: (famille get: #soeur) dansSituation: uneFamille].
        ((each getEtat: #fille) getEtat: #soeur) associationEtat: (((monde get: prénom) getEtat: #fille) getEtat: #soeur)].
      (each getEtat: #fils) ifNotNil: [
        ((each getEtat: #fils) getEtat: #frère) ifNil: [(each getEtat: #fils) affecteEtat: (famille get: #frère) dansSituation: uneFamille].
        ((each getEtat: #fils) getEtat: #frère) associationEtat: (((monde get: prénom) getEtat: #fille) getEtat: #soeur)]]]].
  ].

```

Hermeline met au monde ses quatre enfants :

```

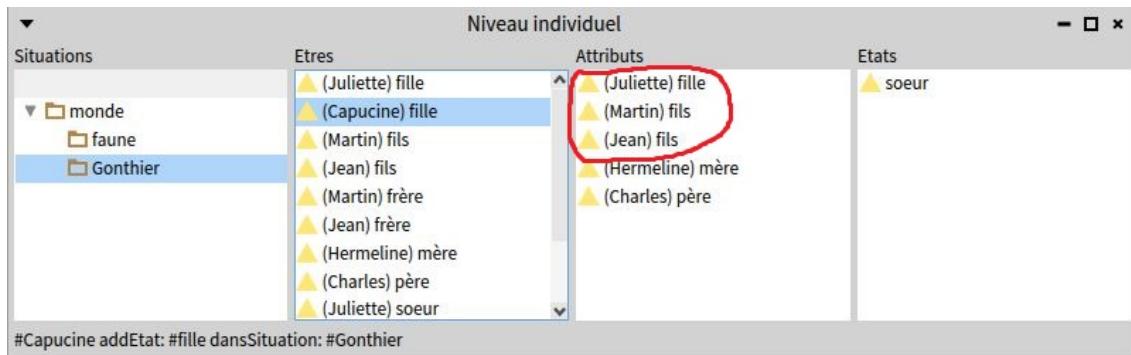
(monde get: #Hermeline) metAuMonde: #Capucine famille: (monde get: #Gonthier) essence: femme.
(monde get: #Hermeline) metAuMonde: #Jean famille: (monde get: #Gonthier) essence: homme.
(monde get: #Hermeline) metAuMonde: #Martin famille: (monde get: #Gonthier) essence: homme.
(monde get: #Hermeline) metAuMonde: #Juliette famille: (monde get: #Gonthier) essence: femme.

```

Au niveau individuel, Capucine est une fille :



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



Nous en avons surchargé la méthode "printOn: aStream" de la classe Object de Smalltalk au niveau de l'essence chose, pour afficher le nom des membres d'une famille avec leur nom de famille :

```

chose compile: ' printOn: aStream
self isEtat
ifTrue: [self getEtats
        detect: [:e | e getSituationDefinition class getNom == #famille]
        ifFound: [:x | aStream nextPutAll: self getEtat 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))
```

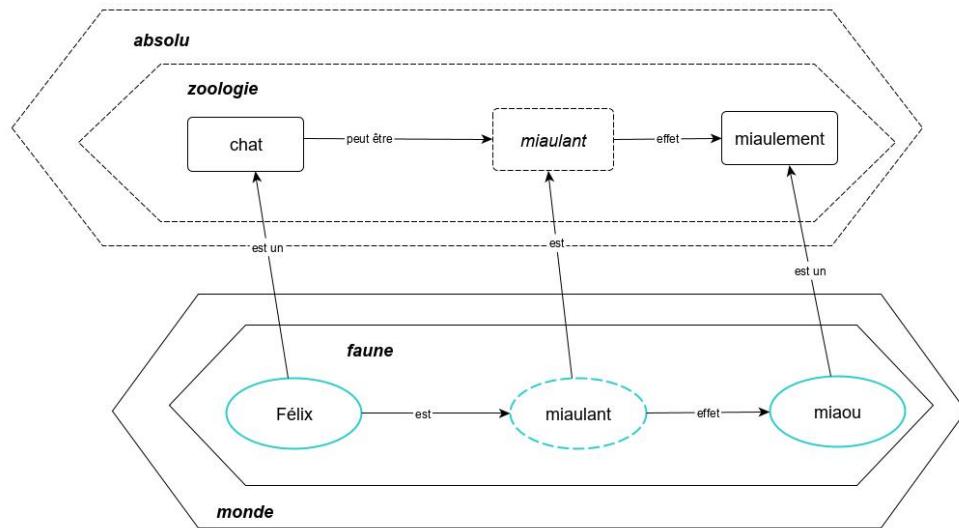
donne : an OrderedCollection(Jean Gonthier)

Noter que si Capucine possède, en tant que soeur, une soeur et deux frères, il ne serait pas possible de préciser ici qu'elle est une gentille soeur pour Juliette et une soeur autoritaire par rapport à Martin.

Pour cela, il faudrait procéder comme nous l'avons fait dans l'exemple 14\_4 pour différencier les états d'époux de Paul polygame par rapport à ses deux épouses Juliette et Yvette.

## Exemple 19 : sur la notion d'action

Considérons le graphe suivant :

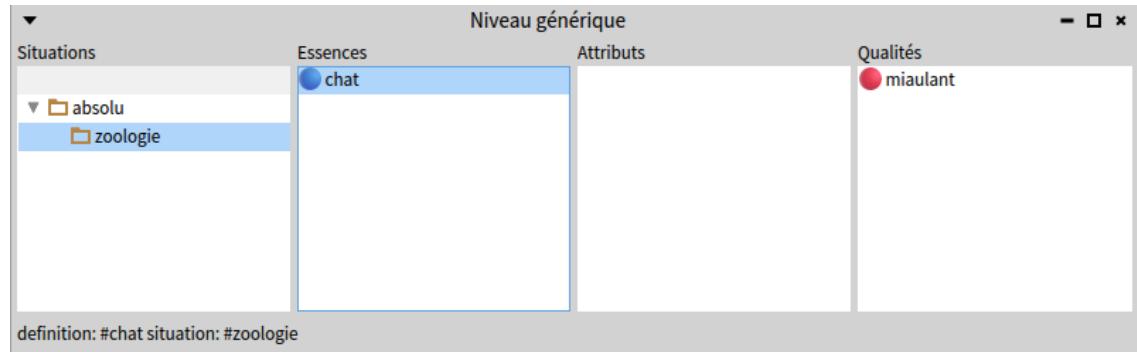


```

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)



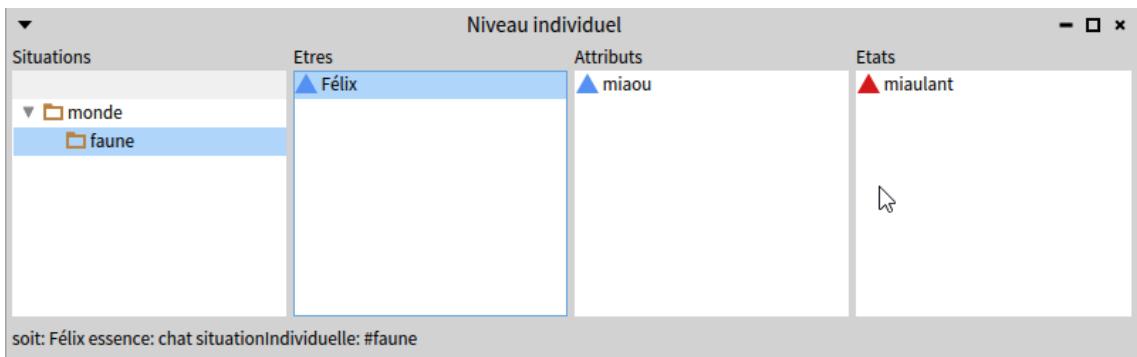
mais son effectivité est intermittente (un chat ne miaule pas sans arrêt).

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) affecteEtatEssentiel: (( 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 Smalltalk 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) affecteEtatEssentiel: ((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 :

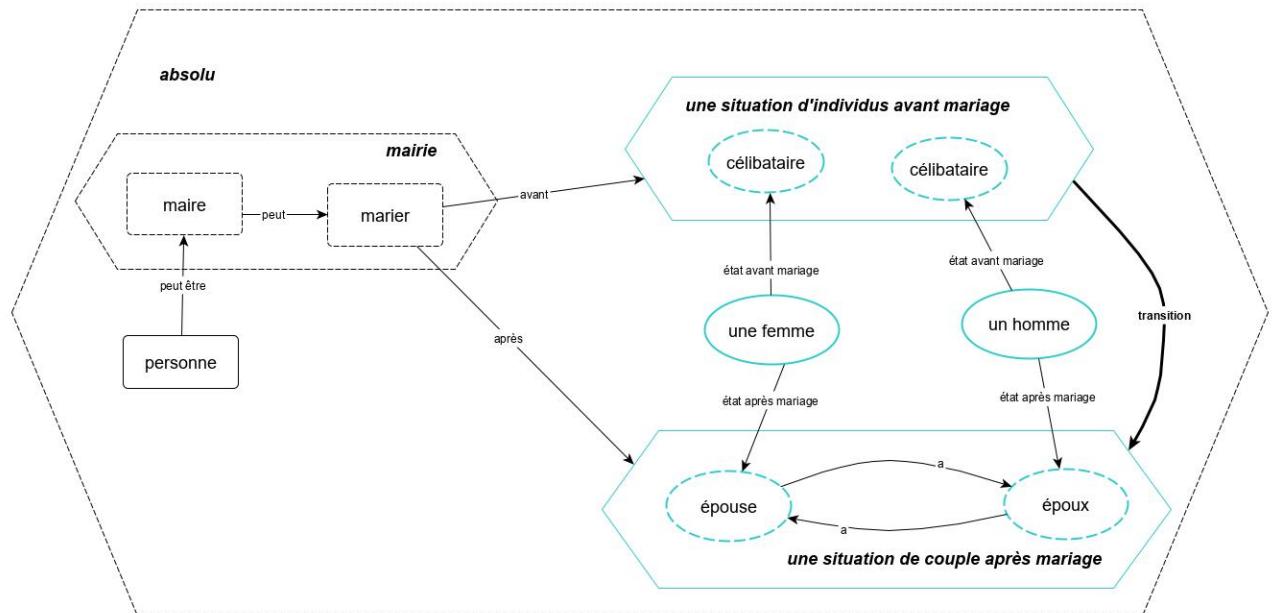
```
((monde get: #faune) get: #Maya) getEtat: #miaulant) miaule
```

donne : Meow

## Exemple 20 : 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) .
iceo definitionQualiteEssentielle: #mariant pour: ((absolu getSituation: #mairie) get: #maire) effectivite: #intermittente .
((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) .

```

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

```

(((absolu getSituation: #mairie) get: #maire) getQualite: #mariant) compile: 'marie: 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
                affecteEtat: (s2 class get: #époux)
                dansSituation: s2]
        ifFalse: [uneFemme := etat getEtant.
            uneFemme
                affecteEtat: (s2 class get: #épouse)
                dansSituation: s2]].
(unHomme getEtat: #époux dansSituation: s2)
associationEtat: (uneFemme getEtat: #épouse dansSituation: s2)'.

```

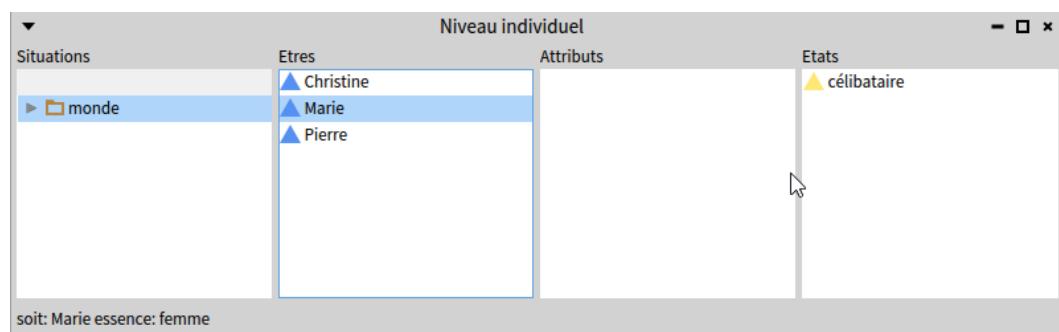
Christine, Marie et Pierre sont créés de la manière suivante :

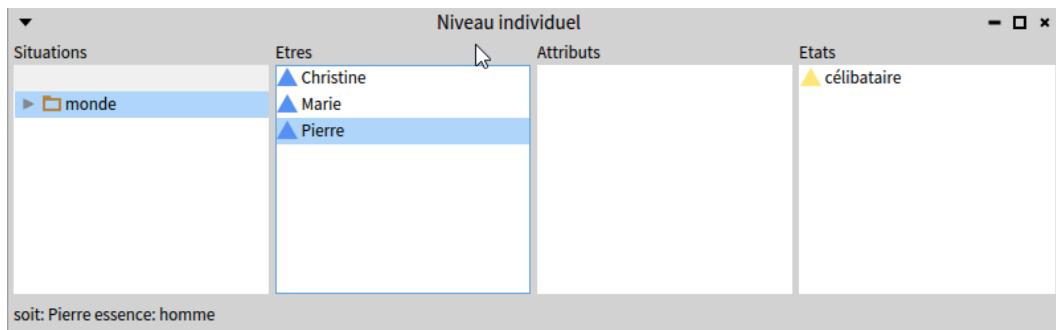
```

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) affecteEtat: ((absolu getSituation: #mairie) get: #maire)
dansSituation: (monde getSituation: 'une mairie').
((monde get: #Christine) getEtat: #maire) affecteEtatEssentiel: (((absolu
getSituation: #mairie) get: #maire) getQualite: #mariant).
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) affecteEtat: ((absolu getSituation: 'situation d'
individu') get: #célibataire) dansSituation: (monde getSituation: 'situation de Pierre et Marie avant mariage').
(monde get: #Pierre) affecteEtat: ((absolu getSituation: 'situation d'
individu') get: #célibataire) dansSituation: (monde getSituation: 'situation de Pierre et Marie avant mariage').
personne peutEtre: ((absolu getSituation: #mairie) get: #maire).

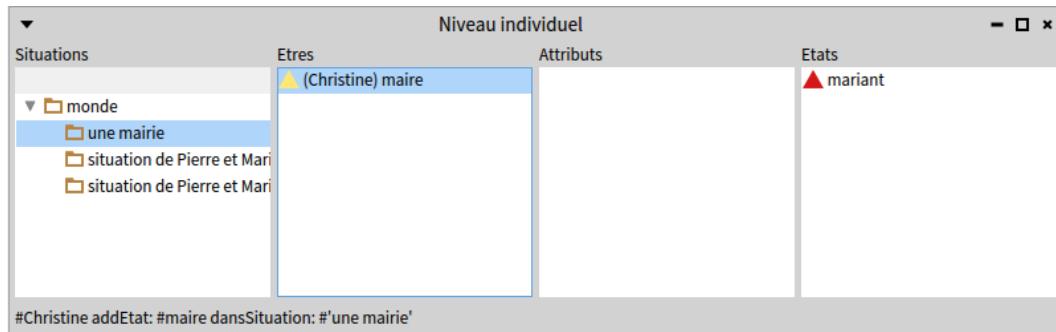
```

Avant leur mariage, Pierre et Marie sont célibataires :





et Christine en tant que maire est dans l'état essentiel et intermittent mariant :

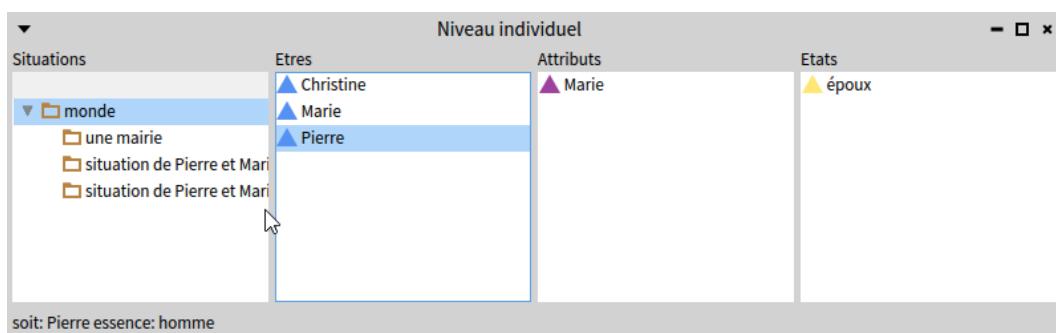
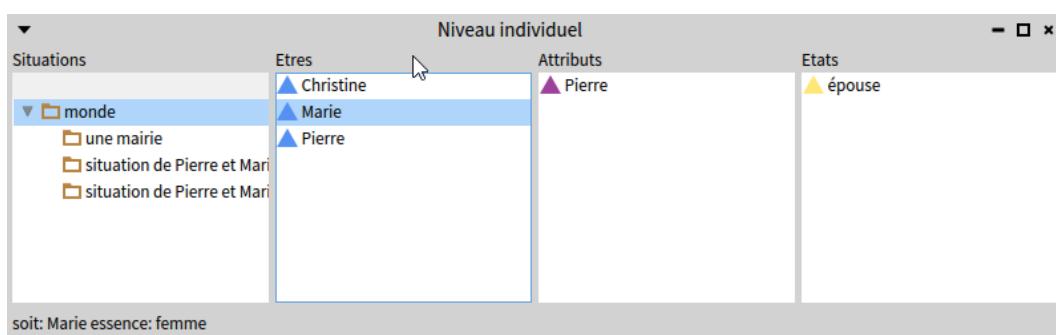


L'action de marier de Christine en tant que maire dans l'état mariant va les faire passer dans une situation de couple, avec les états d'épouse et d'époux :

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

Sélectionner cette instruction et faire un "Do it".

Après mariage on constate que Marie et Pierre ont bien été mariés par Christine en tant que maire.



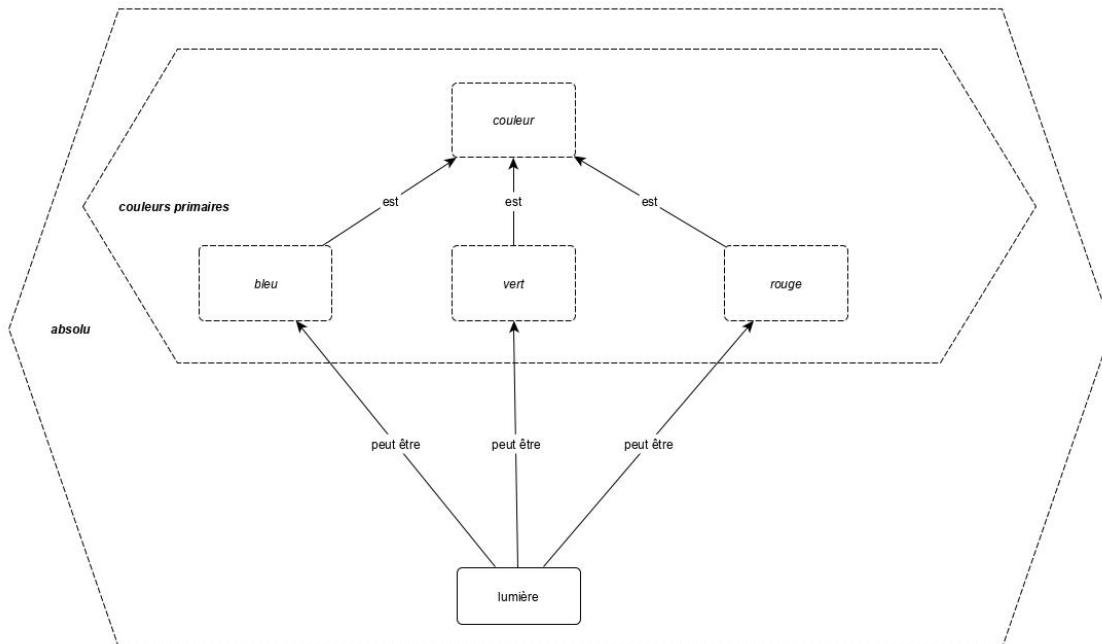
## Exemple 21 : sur la notion de couleur

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.

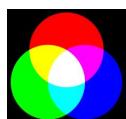
Les couleurs primaires bleu, vert où rouge sont celles des trois (spectres de) lumières perçues et distinguées par l'œil humain que nous qualifierons de "monochromatiques"

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.

Nous qualifierons ces lumières de "polychromatiques"



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.

La palette des couleurs primaires et secondaires peut être définie de la manière 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 :

Niveau générique			
Situations	Essences	Attributs	Qualités
▼ absolu	blanc		
colorée	bleu		
palette de couleurs	cyan		
	jaune		
	magenta		
	noir		
	rouge		
	vert		

definitionSituation: 'palette de couleurs'

Voici comment peuvent se définir les différentes lumières :

```
iceo definition: #lumière.
iceo definition: 'lumière monochromatique' genus: lumière.
iceo definition: 'lumière polychromatique' genus: lumière.
iceo definition: 'lumière bleu' genus: (absolu get: 'lumière monochromatique').
iceo definitionQualitéEssentielle: 'bleu' pour: (absolu get: 'lumière bleu
') effectivité: #permanente.
iceo definition: 'lumière vert' genus: (absolu get: 'lumière
monochromatique').
iceo definitionQualitéEssentielle: 'vert' pour: (absolu get: 'lumière vert
') effectivité: #permanente.
iceo definition: 'lumière rouge' genus: (absolu get: 'lumière
monochromatique').
iceo definitionQualitéEssentielle: 'rouge' pour: (absolu get: 'lumière
rouge') effectivité: #permanente.
iceo definition: 'lumière cyan' genus: (absolu get: 'lumière
```

```

polychromatique') .
iceo definitionQualiteEssentielle: 'cyan' pour: (absolu get: 'lumière cyan'
') effectivite: #permanente.
(absolu get: 'lumière cyan') referenceEssence: (absolu get: 'lumière bleu')
cardinalite: 1.
(absolu get: 'lumière cyan') referenceEssence: (absolu get: 'lumière vert')
cardinalite: 1.
iceo definition: 'lumière magenta' genus: (absolu get: 'lumière
polychromatique').
iceo definitionQualiteEssentielle: 'magenta' pour: (absolu get: 'lumière
magenta') effectivite: #permanente.
(absolu get: 'lumière magenta') referenceEssence: (absolu get: 'lumière
bleu') cardinalite: 1.
(absolu get: 'lumière magenta') referenceEssence: (absolu get: 'lumière
rouge') cardinalite: 1.
iceo definition: 'lumière jaune' genus: (absolu get: 'lumière
polychromatique').
iceo definitionQualiteEssentielle: 'jaune' pour: (absolu get: 'lumière
jaune') effectivite: #permanente.
(absolu get: 'lumière jaune') referenceEssence: (absolu get: 'lumière vert
') cardinalite: 1.
(absolu get: 'lumière jaune') referenceEssence: (absolu get: 'lumière rouge
') cardinalite: 1.
iceo definition: 'lumière blanc' genus: (absolu get: 'lumière
polychromatique'). iceo definitionQualiteEssentielle: 'blanc' pour: (
absolu get: 'lumière blanc') effectivite: #permanente.
(absolu get: 'lumière blanc') referenceEssence: (absolu get: 'lumière vert
') cardinalite: 1.
(absolu get: 'lumière blanc') referenceEssence: (absolu get: 'lumière rouge
') cardinalite: 1.
(absolu get: 'lumière blanc') referenceEssence: (absolu get: 'lumière bleu
') cardinalite: 1.
iceo definition: 'lumière noir' genus: (absolu get: 'lumière
polychromatique').
iceo definitionQualiteEssentielle: 'noir' pour: (absolu get: 'lumière noir
') effectivite: #permanente.

```

L'ensemble de lumières ainsi défini est :

Niveau générique			
Situations	Essences	Attributs	Qualités
▼ absolu <ul style="list-style-type: none"> <li>colorée</li> <li>palette de couleurs</li> </ul>	<ul style="list-style-type: none"> <li>chose</li> <li>lumière</li> <li>lumière blanc</li> <li>lumière bleu</li> <li>lumière cyan</li> <li>lumière jaune</li> <li>lumière magenta</li> <li>lumière monochromatique</li> <li>lumière noir</li> <li>lumière polychromatique</li> <li>lumière rouge</li> <li>lumière vert</li> <li>objet</li> <li>torche</li> </ul>	<ul style="list-style-type: none"> <li>lumière rouge</li> <li>lumière vert</li> </ul>	<ul style="list-style-type: none"> <li>jaune</li> </ul>

definition: 'lumière jaune' genus: #'lumière polychromatique'

On voit que par exemple la lumière jaune possède la qualité essentielle d'être jaune, et qu'elle est composée des lumières rouge et vert.

## Couleur des objets

La couleur d'un objet est celle de la lumière qu'il reflète 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 apparaîtra 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 paraît 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') affecteEtat: ((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.torche referenceEssence: lumière.
iceo soit: 'une torche' essence: torche.
(monde get: 'une torche') attributionEtre: 'une lumière' essence: (absolu
    get: 'lumière magenta').
```

Le code Smalltalk suivant 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
| | c s t |
s := (absolu getElements: lumière) select: [:each |
  (each getGenus == (absolu get: "lumière monochromatique")) or: [each getGenus == (absolu get: "lumière polychromatique")]].
t := s detect: [:each |
  | := each.
  | getGenus == (absolu get: "lumière monochromatique")

ifTrue: [c := OrderedCollection with: | ]
ifFalse: [ c := | getEssencesAttributs sort: [:a :b | a getNom < b getNom] ] .
c = ((uneTorche lumièresEmises intersection: self lumièresReflétées) sort: [:a :b | a getNom < b getNom]) ] .
  ^t getQualites at: 1 '.

```

Cette méthode utilise deux autres méthodes qui déterminent respectivement les lumières reflétées par un objet et les lumières émises par une torche :

```

objet compile: 'lumièresReflétées
| |
| := (absolu get: "lumière ", (self getEtatEssence: ((absolu getSituation: #colorée) get: #couleur)) getNom).
| getGenus == (absolu get: "lumière monochromatique") ifTrue: [^ OrderedCollection with: | ]
ifFalse: [ ^ | getEssencesAttributs sort: [:a :b | a getNom < b getNom] ] '.

torche compile: 'lumièresEmises
| |
| := (self getEtreAttribut: #lumière) getEssence.
| getGenus == (absolu get: "lumière monochromatique") ifTrue: [^ OrderedCollection with: | ]
ifFalse: [ ^ | getEssencesAttributs sort: [:a :b | a getNom < b getNom] ] '.

```

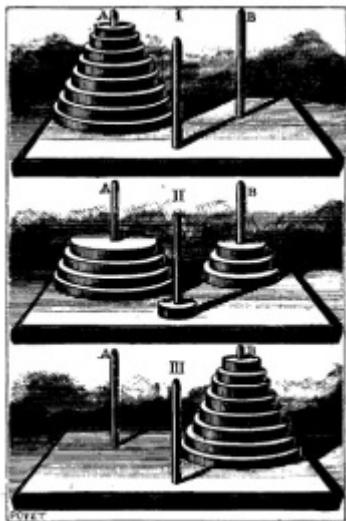
Ainsi par exemple, pour un objet de couleur jaune éclairé par de la lumière magenta, l'expression :

```
(monde get: 'un objet') couleurApparenteEclairePar: (monde get: 'unetorche')
dans: monde
```

retourne : rouge

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

## Exemple 22 : les tours de Hanoï revisitées



La tour d'Hanoï.

Au XIXe siècle, le mathématicien français Edouard Lucas a inventé le jeu des tours de Hanoï, une simple récréation mathématique qui s'est révélée au fil des années une mine de réflexions<sup>1</sup>.

Le jeu des tours de Hanoï est constitué de trois piquets placés verticalement, et de  $n$  disques de taille décroissante. Les disques sont placés initialement par taille décroissante sur le premier piquet. Le but du jeu consiste à déplacer les disques jusqu'à parvenir à la situation finale dans laquelle tous les disques se retrouvent autour d'un autre piquet par ordre de taille décroissante. Les disques se déplacent en suivant deux règles :

- on ne déplace qu'un seul disque à la fois
- un disque ne peut jamais être placé sur un disque plus petit.

Ce problème est un cas d'école d'algorithme dit récursif. Pour  $n$  disques, le nombre minimum de déplacements est  $2^n - 1$

La solution que nous proposons ici est inspirée de celle proposée par Ted Kaehler et Dave Patterson dans le livre "A TASTE OF SMALLTALK".

Dans leur solution, les disques sont des objets qui se déplacent en fonction des messages déterminés par un chef d'orchestre. Aux deux règles du jeu est ajoutée une troisième (qui semble logique pour avoir un minimum de déplacements) interdisant qu'un disque ne revienne en arrière. Leur algorithme exige que des disques fictifs (mock disks) de taille plus grande que celle des autres soient placés sur les piquets vides.

Notre solution diffère sur différents points :

- les piquets sont des situations où évoluent les disques.
- les disques sont des acteurs (implantés sous forme de processus) qui jouent une course de

---

<sup>1</sup>Édouard Lucas, Récréations mathématiques, t. 3, 1892 (réimpr. Librairie Albert Blanchard, 1979)

relais.

- nous avons défini les manières d'être relatives "sous" et "sur" pour les disques (ce qui évite d'avoir recours à des "mock disks" fictifs).

L'algorithme utilisé est également original<sup>1</sup>

Pour l'exemple, le nombre de disques a été fixé à 9.

Ce nombre peut être bien sûr être changé. Notez que si vous fixez ce nombre à 64, le nombre minimum de déplacements sera de  $2^{64}-1$ , soit 18 446 744 073 709 551 615. Il faudra être patient !

Pour que le déplacement des disques soit visible, un délai paramétrable (qui peut être nul) de 10 ms a été imposé entre deux déplacements.

```
iceo definition: #jeu.  
iceo definitionSituation: #piquet.  
  
iceo definitionAttribut: #disque de: jeu cardinalite: 9.  
Smalltalk at: #delay put: 10. "ms"
```

Les piquets, ici au nombre de trois, sont les situations entre lesquelles évoluent les disques.

Au départ, les disques sont empilés sur le premier piquet en tenant compte de leur taille.

Chaque disque possède un comportement qui est une manière d'être essentielle et permanente.

```
iceo definitionQualiteEssentielle: #comportement pour: (jeu  
    getEssenceAttribut: #disque) effectivite: #permanente.
```

Au comportement de chaque disque est associé un process Smalltalk et un sémaphore<sup>2</sup> :

```
"définition du rôle de chaque joueur"  
(monde get: #Hanoi) getEtres do: [:each |  
    | started |  
    started := false.  
    (each getEtat: #comportement) setSemaphore: Semaphore new.  
    (each getEtat: #comportement) setProcess: ([:h :d |  
        | s x |  
        s isNil ifTrue: [ s := false ].  
        [ true ] whileTrue: [  
            s ifFalse: [  
                s := true.  
                (d getEtat: #comportement) getSemaphore wait ].  
                x := h meilleurRelais.  
                x notNil ifTrue: [  
                    x key prendLeRelais: x value.  
                    (x key getEtat: #comportement) getSemaphore signal].  
                (d getEtat: #comportement) getSemaphore wait ] ] newProcessWith: (Array with: (monde get: #Hanoi) with: each) ].
```

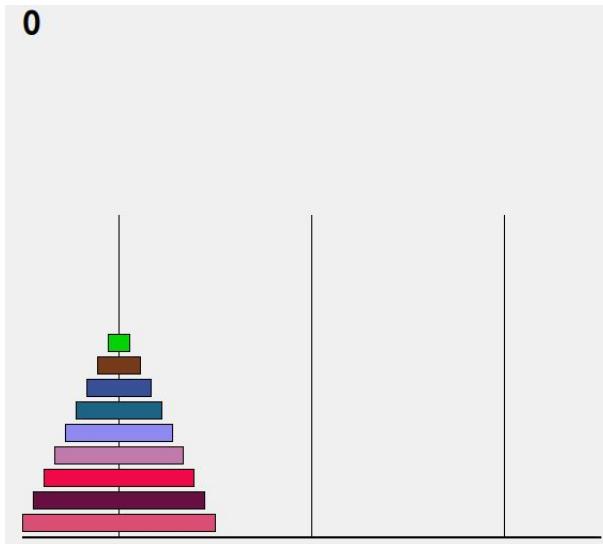
<sup>1</sup>Comme dans l'ouvrage "A taste of Smalltalk" de Ted Kaehler et Dave Patterson, nous ne donnerons pas la preuve que cet algorithme représente une solution mathématique rigoureuse, mais il fonctionne !

<sup>2</sup>Les notions de process et de sémaphore en Smalltalk sont décrits dans le livre "Concurrent Programming in Pharo" de Stéphane Ducasse et Guillermo Polito".

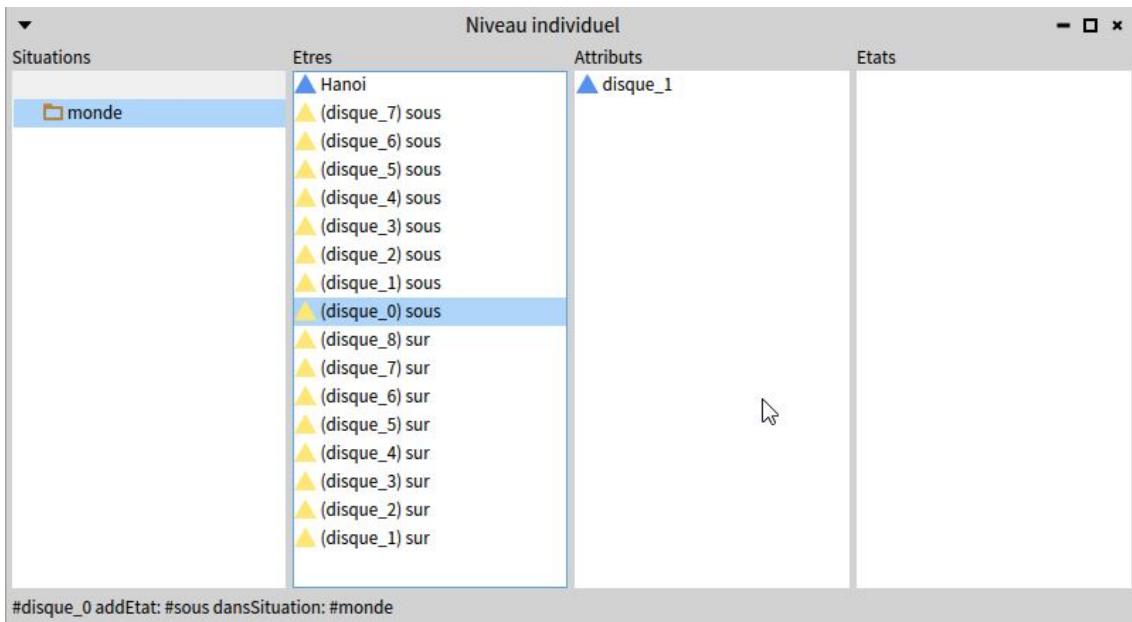
Au départ, les joueurs sont placés dans un état d'attente par "allezAuxStartingBlocks".

Pour initialiser le jeu, sélectionnez et exécutez l'ensemble des instructions de l'exemple (où une grande partie du code est dédiée à la représentation graphique des disques se déplaçant de piquet en piquet).

L'image suivante s'affiche (la couleur des disques est déterminée de manière aléatoire) :



Il peut être intéressant à ce stade d'ouvrir un SiBrowser :

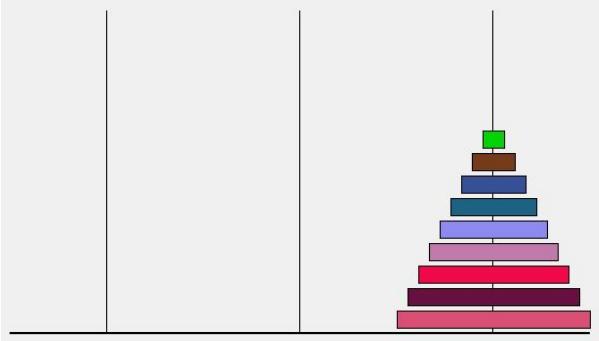


On peut vérifier qu'au départ les disques 0 à 7 sont dans l'état "sous" et que les disques 1 à 8 sont dans l'état "sur".

Un click avec le bouton gauche de la souris sur l'un des disques envoie un signal au plus petit pour lui dire de commencer la course.

Vous voyez les disques se déplacer dans une course de relais qui se termine lorsque les disques ont retrouvé leur configuration initiale, sur un autre piquet.

**511**



Le nombre affiché en haut à gauche de l'image correspond au nombre de déplacements effectués par les disques. Le nombre 511 correspond à la valeur théorique minimale pour ce jeu avec neuf disques ( $2^9-1$ ).

Pour clore le jeu, il suffit d'un click avec le bouton droit de la souris sur l'un des disques (ou le nombre de déplacements affiché).

## Exemple 22\_1 : et avec 4 piquets ?



Le problème des tours de Hanoï avec quatre piquets a été énoncé pour la première fois par Henry Dudeney en 1907; il l'appelle alors le puzzle de Reve (sans faire aucune référence au jeu des tours de Hanoi d'Edouard Lucas, imaginé en 1892 ...).

En 1941, Bonnie M. Stewart et James S. Frame ont proposé une solution mathématique du problème (pour quatre piquets ou plus, bien que le problème avec plus de 4 piquets reste encore en fait une conjecture ouverte).

L'algorithme que nous donnons est original, dans la lignée de celui que nous avons proposé avec 3 piquets.

Avec 3 piquets, un seul disque pouvait se déplacer à chaque étape, et ce sur un seul piquet.

Dans le cas de 4 piquets, les degrés de liberté des disques sont plus grands : 1 ou 2 disques peuvent prendre le relais à chaque étape, et ce sur 1 ou 2 piquets cibles possibles.

Nous qualifierons notre algorithme d'altruiste : à chaque étape, le disque qui prend le relais est généralement le plus grand possible parmi ceux qui peuvent se déplacer, mais le piquet cible qu'il choisit pour son déplacement vise à libérer la place pour le plus grand disque possible à l'étape suivante.

Pour cela, chaque disque est doté de capacités d'anticipation sur la situation des autres disques après son déplacement<sup>1</sup>

Le nombre de mouvements nécessaires pour parvenir à déplacer  $n$  disques du piquet où ils se situent initialement à leur position finale est beaucoup plus petit que dans le cas de 3 piquets.

Ainsi, 33 mouvements (au lieu de 511) suffisent dans le cas de 8 disques, 41 pour 9 disques, 49 pour 10 disques, ...

Dans le jeu, les différentes étapes ne s'enchaînent pas automatiquement après un certain délai, mais en cliquant avec le bouton gauche de la souris sur le nombre de déplacements affiché.

---

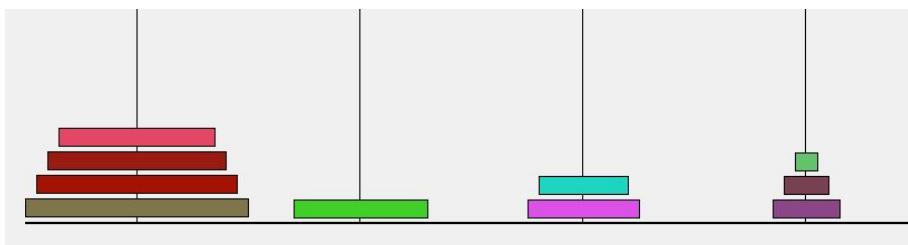
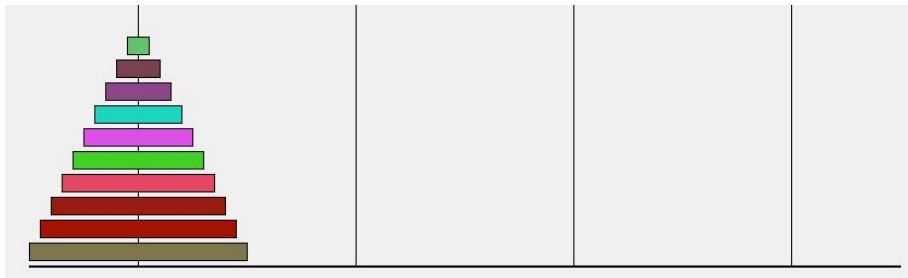
<sup>1</sup>Les disques constituent de fait un système multi-agents.

Pour clore le jeu, il suffit d'un click avec le bouton droit de la souris sur l'un des disques (ou le nombre de déplacements).

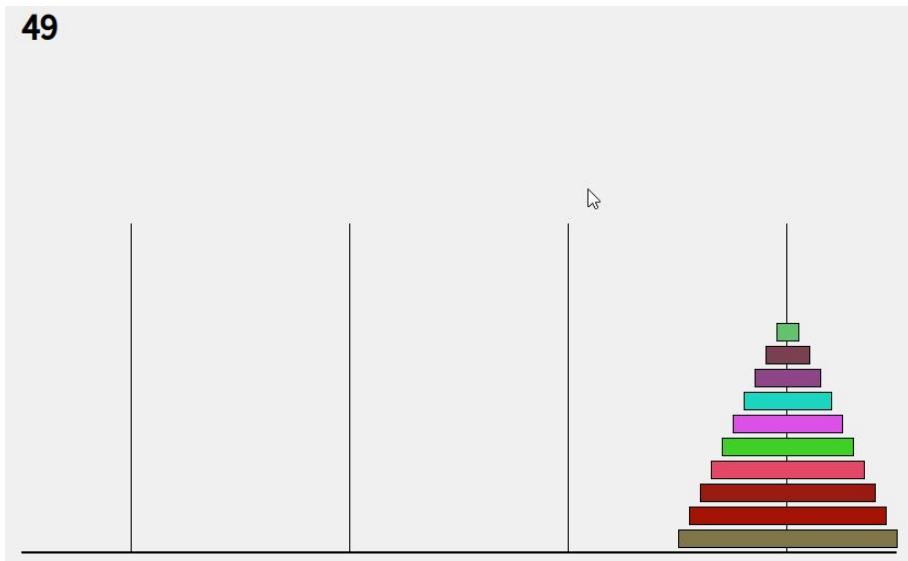
A certaines étapes il convient de privilégier le déplacement du plus petit disque.

Dans notre algorithme, ces étapes sont actuellement fixées en "dur" : par exemple, 3, 11 et 13 dans le cas de 9 disques.

Nous laisserons à notre lecteur le soin de déterminer, dans le cas général, à quelles situations singulières ces étapes correspondent<sup>1</sup>.



49



<sup>1</sup>Le fait de pouvoir exécuter le jeu pas-à-pas constitue un atout majeur pour étudier ces situations.

# 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.

Aucune méthode de l'environnement de base de Pharo n'a été modifiée, à part la méthode "evaluate: aString onError: onErrorBlock" de SpCodePresenter, pour éviter que le compilateur trop pressé n'évalue une expression avant d'avoir terminé d'évaluer la précédente ! (ce qui permet d'évaluer toutes les expressions de chaque exemple en bloc).

Cette méthode attrape les exceptions de classe Oups.

La méthode "validateClassName" de ShiftClassBuilder a également été modifiée pour autoriser que le nom des classes ne commencent pas par une majuscule. Pour permettre l'homonymie des essences, nous avons mis à profit la possibilité de créer des classes de type "newAnonymousSubclass".

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.

Les attributs des essences sont des instances de la classe Association, ce qui permet d'associer à chaque essence attribut une cardinalité pour sa composition.

Pour la composition des manières d'être et des situations génériques, celles-ci sont traitées comme des essences (avec leur differentia), à la différence près que leurs attributs n'ont pas de cardinalité.

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 mét-essence).

Par défaut, la mét-essence d'une essence est l'essence chose et l'essence chose est son propre genus et sa propre mét-essence.



# Contents

<b>Avant-propos</b>	<b>3</b>
<b>Introduction</b>	<b>5</b>
<b>Première partie : sur la notion d'être</b>	<b>9</b>
<b>Etres</b>	<b>11</b>
Existence d'un être . . . . .	11
Situation . . . . .	12
Essence d'un être . . . . .	14
Constitution de l'essence d'un être . . . . .	15
Principe d'héritage des attributs des essences . . . . .	19
Création d'une essence . . . . .	20
Situation générique . . . . .	22
Identification des essences . . . . .	24
Essences abstraites . . . . .	25
Essence et normalité . . . . .	26
Constitution d'un être . . . . .	27
Situation individuelle . . . . .	29
Individus . . . . .	30
Création des êtres . . . . .	30
Identification des êtres . . . . .	32
Essence d'une essence . . . . .	33
Essence chose . . . . .	35
<b>Seconde partie : sur les relations entre les êtres et leurs manières d'être</b>	<b>37</b>
<b>Manière d'être</b>	<b>39</b>
Manières d'être essentielles ou accidentielles . . . . .	40
Relations entre manières d'être . . . . .	43
Subsomption des manières d'être . . . . .	45
Manière d'être d'une manière d'être . . . . .	46
<b>Estat</b>	<b>49</b>
Relations entre états . . . . .	52
Relation entre états différente de la composition . . . . .	53

Etat dans un état . . . . .	53
Effectivité d'une manière d'être . . . . .	55
<b>Contraintes structurelles internes</b>	<b>57</b>
<b>Contraintes externes sur les manières d'être</b>	<b>58</b>
<b>Êtres inconnus</b>	<b>59</b>
<b>Actions</b>	<b>61</b>
Déclenchement d'une action . . . . .	64
Événements . . . . .	64
Changement de situation . . . . .	65
Interactions entre acteurs . . . . .	68
Scénarios . . . . .	69
<b>Pour conclure sur cette présentation des concepts</b>	<b>71</b>
<b>Bibliographie</b>	<b>73</b>
<b>Présentation du langage textuel d'ICEO</b>	<b>77</b>
<b>Implantation en Smalltalk</b>	<b>79</b>
<b>Installation d'ICEO dans Pharo et exemples</b>	<b>81</b>
Installation d'ICEO . . . . .	81
Services accessibles via l'onglet "ICEO" de la fenêtre principale . . . . .	86
Exemple 2 : sur la composition d'une essence . . . . .	88
Exemple 3 : sur la composition d'une essence à partir d'essences existantes . . . . .	90
Exemple 4 : sur le principe d'héritage des attributs des essences . . . . .	92
Exemple 5 : sur le principe d'identification des essences . . . . .	94
Exemple 5_2 . . . . .	95
Exemple 6 : sur le principe d'identification des êtres . . . . .	97
Exemple 7 : sur la notion de métatitre (essence d'une essence) . . . . .	99
Exemple 8 : sur les notions de situation générique et de situation individuelle . . . . .	103
Exemple 9 : sur la notion de manière d'être essentielle et permanente . . . . .	105
Exemple 10 : sur la notion de manière d'être accidentelle . . . . .	107
Exemple 11 : sur les attributs d'une manière d'être . . . . .	109
Exemple 12 : sur les relations entre manières d'être . . . . .	111
Exemple 13 : sur la subsomption des manières d'être . . . . .	113
Exemple 14 : sur la relation entre états . . . . .	116
Exemple 14_2 . . . . .	118
Exemple 14_3 . . . . .	119
Exemple 14_4 . . . . .	120
Exemple 14_5 . . . . .	122
Exemple 15 : sur la notion d'état dans un état . . . . .	124
Exemple 16 : sur la notion d'être inconnu . . . . .	126
Exemple 17 : sur la notion de contrainte structurelle interne . . . . .	128
Exemple 18 : sur la notion de famille . . . . .	132
Exemple 19 : sur la notion d'action . . . . .	137

Exemple 20 : sur les changements de situation . . . . .	139
Exemple 21 : sur la notion de couleur . . . . .	142
Exemple 22 : les tours de Hanoï revisitées . . . . .	147
Exemple 22_1 : et avec 4 piquets ? . . . . .	151
<b>Sur l'implantation d'ICEO en Pharo</b>	<b>153</b>