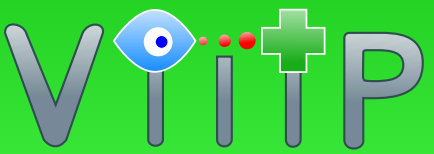
Programmation orientée ontologie en Python

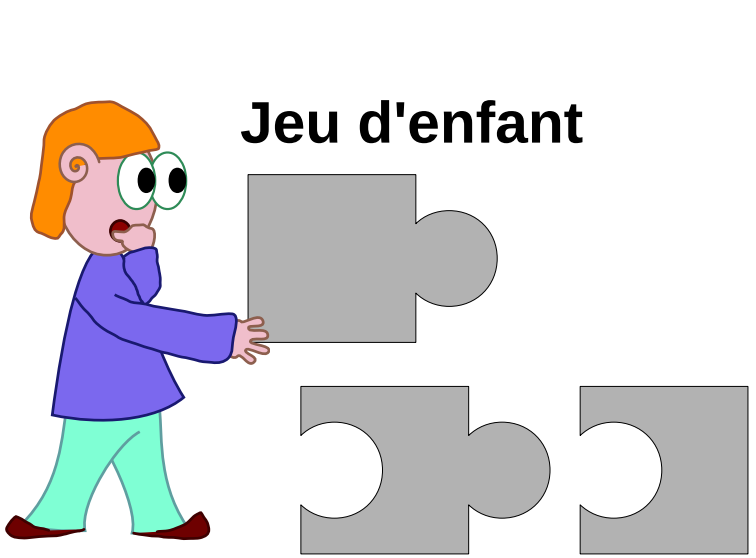


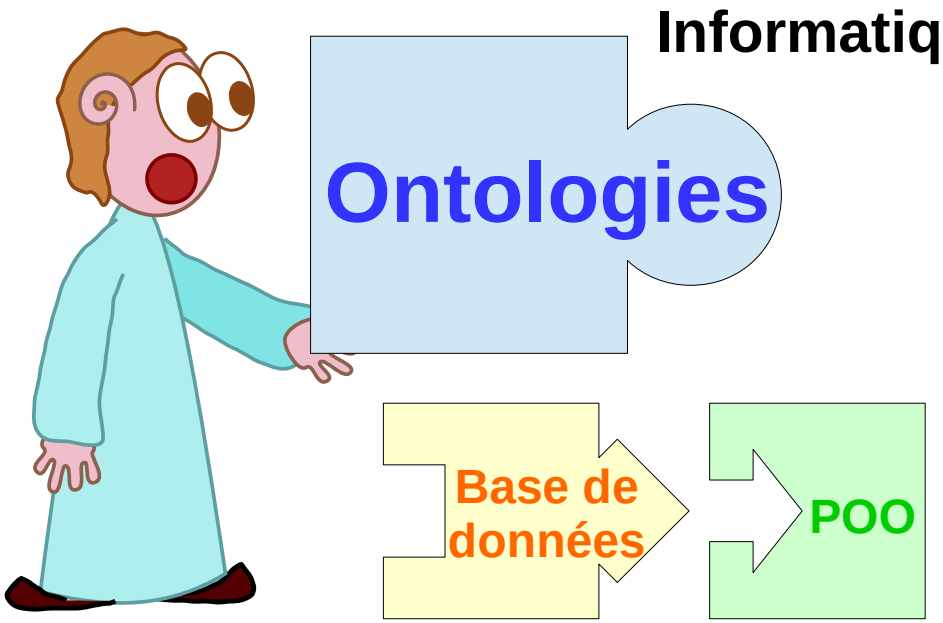
Jean-Baptiste LAMY
Hélène BERTHELOT

LIMICS, Université Paris 13, Sorbonne Paris Cité, Université Paris 6,
INSERM UMR_S 1142, 74 rue Marcel Cachin, 93017 Bobigny, France

IC 2015 – Rennes

Introduction





- Les **ontologies formelles** structurent un domaine de connaissance
  - pour réaliser des inférences, relier les connaissances entre elles
- Il est difficile d'interfacer une ontologie avec un programme informatique :
  - Ex : peupler l'ontologie à partir de diverses sources (BD,...)
  - Ex : générer un site Internet dynamique à partir des inférences issues de l'ontologie
- 2 approches possibles

Exemple en Java

```
public static float getPurchasesSum(RDFIndividual customer) {
    OWLModel owlModel = customer.getOWLModel();
    float sum = 0;
    RDFProperty purchasesProperty = owlModel.getRDFProperty("purchases");
    RDFProperty productProperty = owlModel.getRDFProperty("product");
    RDFProperty priceProperty = owlModel.getRDFProperty("price");
    Iterator purchases = customer.listPropertyValues(purchasesProperty);
    while(purchases.hasNext()) {
        RDFIndividual purchase = (RDFIndividual) purchases.next();
        RDFIndividual product;
        product = (RDFIndividual) purchase.getPropertyValue(productProperty);
        Float price = (Float) product.getPropertyValue(priceProperty);
        sum += price.floatValue();
    }
    return sum;
}
```

API OWL classique (16 lignes)
Deux modèles distincts

```
public static float getPurchasesSum(Customer customer) {
    float sum = 0;
    Iterator purchases = customer.listPurchases();
    while (purchases.hasNext()) {
        Purchase purchase = (Purchase) purchases.next();
        Product product = purchase.getProduct();
        sum += product.getPrice();
    }
    return sum;
}
```

Programmation orienté ontologie
(hypothétique, 10 lignes)
Les classes de l'ontologie sont des classes du langage de programmation

D'après *A Semantic Web Primer for Object-Oriented Software Developers*, W3C Working Group Note

Modèles objet et ontologies

- Un vocabulaire commun : **classes, instances, propriétés,...**
- Des applications différentes : programmation *vs* inférence
- Et des définitions différences :

Modèles objet	Ontologies
Hypothèse du monde fermé : Disjonctions entre classes implicites : deux classes sont disjointes si elles n'ont pas de filles en commun Distinctions entre instances implicites : deux instances sont obligatoirement distinctes Une instance appartient à une seule classe Une propriété est définie pour une classe donnée	**Hypothèse du monde ouvert :** Disjonctions entre classes explicites : deux classes peuvent être disjointes ou non Distinctions entre instances explicites : deux instances peuvent être distinctes ou non Une instance peut appartenir à plusieurs classes Les propriétés sont des entités à part entières, définies indépendamment des classes

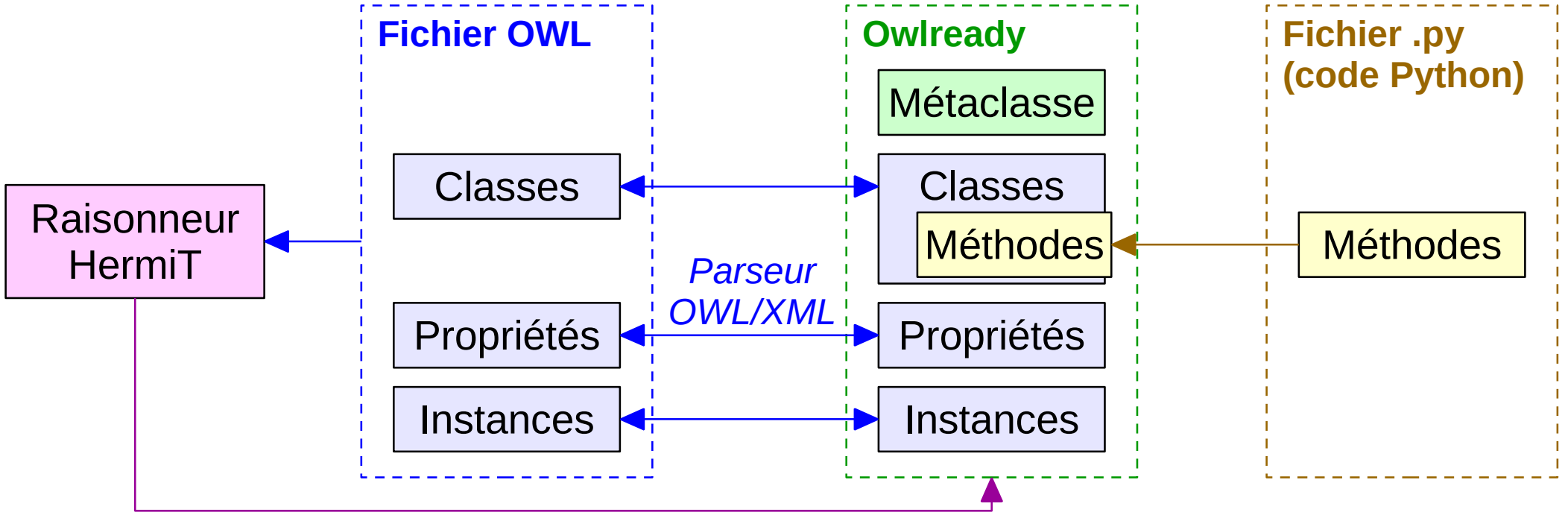
Modèles objet et ontologies

- Python : un langage objet dynamique.
- Les langages dynamiques sont plus proches des ontologies que les langages statiques :

Java (langage statique)	**Python** (langage dynamique)	**OWL** (Ontology Web Language)
Monde fermé	Monde fermé	Monde ouvert
Héritage simple	Héritage multiple	Héritage multiple
Une instance ne peut pas changer de classe Une classe ne peut pas changer de superclasse	Une instance peut changer de classe Une classe peut changer de superclasse	Une instance peut être reclassée (raisonneur) Une classe peut changer de superclasse (idem)
Une propriété est définie pour une classe donnée	Une propriété est définie pour une classe, mais parfois considérée indépendamment (*Duck-typing*)	Les propriétés sont indépendantes des classes

Owlready : architecture générale

- **Owlready** (anciennement Ontopy) : un module Python pour la programmation orientée ontologie
- **Objectif** : intégration la plus transparente possible entre Python et OWL : manipuler des entités et individus OWL comme s'il s'agissait d'objets Python
- Ontologies au format OWL / XML, méthodes en Python (fichier .py)
- HermiT 1.3.8 (version modifiée avec d'avantages d'options en ligne de commande)
- Owlready définit des métaclasses qui réunissent tous ces éléments



Adaptation du modèle objet Python à OWL

- Méthodes redéfinies au niveau des métaclasses :

Méthode	Effet	Raison de la redéfinition
`C.__new__`	Crée un nouvel objet	Combiner la nouvelle classe à la classe OWL de même nom, si elle existe
`C.__instancecheck__`	Teste si un objet est une instance de la classe	Prendre en compte les classes équivalentes OWL
`C.__subclasscheck__`	Teste si une classe est une sous-classe de la classe	Prendre en compte les classes équivalentes OWL
`C.mro`	Calcule l'ordre de résolution des méthodes (*method resolution order*, MRO) notamment en cas d'héritage multiple	Ne pas déclencher d'erreur en cas de MRO temporairement incorrect lors du chargement de l'ontologie (les classes parentes étant ajoutées une à une)
`i.__setattr__`	Modifie un attribut de l'objet	Mettre à jour les propriétés inverses
`i.__getattr__`	Obtient un attribut de l'objet (appelé uniquement pour les attributs inexistants)	Retourner une liste vide si la propriété n'a pas été renseignée, ou None pour une propriété fonctionnelle

`C` : classe, `i` : instance

Exemple d'utilisation

- Les ontologies sont chargées à partir d'un « *onto_path* » qui se comporte comme le *class path* (Java) ou le *python path* (Python)
  - À défaut, elles sont téléchargées

```
>>> from owlready import *
>>> onto_path.append("/chemin/local/vers/les/ontos")
>>> onto = get_ontology("http://www.lesfleursdunormal.fr/
                  static/_downloads/crepes_et_galettes.owl")
>>> onto.load()
```

- La classe « Galette » :

```
>>> onto.Galette
```

- Création d'une instance de Galette :

```
>>> ma_galette = onto.Galette()
```

- Accès aux propriétés avec la « *notation pointée* » usuelle :

```
>>> ma_galette.a_pour_garniture = [ onto.Tomate(),
...                                 onto.Viande() ]
```

Exemple d'utilisation

- Création d'une classe « mixte » OWL – Python

```
>>> class GaletteNonVégétarienne(onto.Galette):
...   equivalent_to = [
...     onto.Galette
...   & ( onto.a_pour_garniture(SOME, onto.Viande)
...     | onto.a_pour_garniture(SOME, onto.Poisson)
...     ) ]
...   def manger(self): print("Beurk, je suis végétarien !")
```

Hérite d'une classe OWL

Une définition OWL

Une méthode Python

- Classification avec le raisonneur HermiT :

```
>>> onto.sync_reasoner()


>>> ma_galette.__class__   => onto.GaletteNonVégétarienne

>>> ma_galette.manger()    => Beurk, je suis végétarien !
```

Fonctionnalité d'Owlready

- **Éléments de OWL 2.0 supportés :**
  - Classe
  - Propriété (fonctionnelle, etc)
  - Instance
  - Disjonction, distinction
  - Annotations : dans un dictionnaire à part (=> non héritées)
  - Clef (*has for key*) : construction automatique d'un dictionnaire avec les clefs
  - Types : booléen, entier, flottant, date, chaîne de caractères

- Possibilité de lier une ontologie à un module Python (*via* une annotation)

- **Owlready** est un **logiciel libre** (licence GNU LGPL)

  http://www.lesfleursdunormal.fr/static/informatique/owlready/index_fr.html

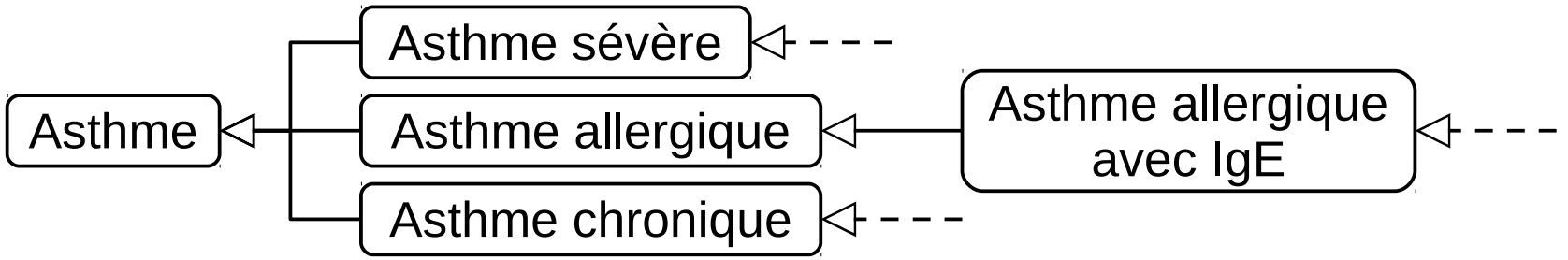
Discussion : comparaison à la littérature

- Peu d'approches sont allées aussi loin dans l'unification entre modèle objet et ontologie
- Approches existantes de programmation orientée ontologie :

	Goldman 2003	**Kalyanpur 2004**	**Koide 2005**	**Babik 2006**	**Stevenson 2011**	**Lamy 2015**
Type	statique	statique	dynamique	dynamique	semi-dynamique	dynamique
Langage	C#	Java	Common Lisp	Python	Java	Python
Classification des classes	non	non	?	?	non	**oui**
Classification des instances	non	non	oui	?	oui	oui
Syntaxe pour définitions OWL	non	non	oui	non	?	oui
Classe *mixte* avec méthodes	non	non	?	non	non	oui

Discussion : « à quoi ça sert ? »

- **Dans le cadre du projet VIIIP** (Visualisation Intégrée de l'Information sur l'Innovation Thérapeutique) :
  1) Peupler l'ontologie à partir des bases de données médicamenteuses
  2) Effectuer des raisonnements pour comparer les médicaments
  3) Générer un site web à partir de l'ontologie et des inférences produites
  - Une contre-indication porte sur une classe de maladie et non sur une (instance) maladie, ex :



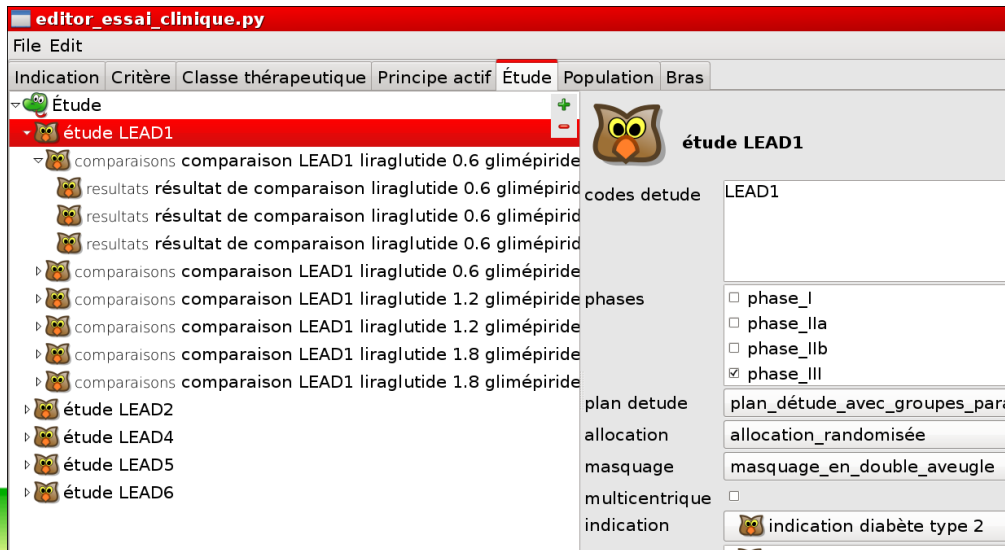
- **Utilisation plus facile des ontologies**
  - Programmeurs débutants pour lesquels OWL API est trop complexe
  - Ex : Étudiant stagiaire de M1 informatique biomédicale
- **Génération automatique** et dynamique des identifiants des instances en fonction de leurs propriétés

Discussion et conclusion

- Limitations :
  - Chargement des ontologies en mémoire vive
  - Les assertions sont obligatoirement rattachées à l'ontologie de la classe sur laquelle elle porte
- Perspectives :
  - Représentation du monde ouvert en langage de programmation, ex :

```
Pizza.has_topping = [onto.TomatoTopping()]  vs
Pizza.has_topping = [onto.TomatoTopping(), Any]
```

  - Génération automatique de boîtes de dialogues pour éditer les instances (*via* les modules Python existants tel que *EditObj*)



Références

Babik M. & Hluchy L. (2006). Deep Integration of Python with Web Ontology Language. In Proceedings of the 2nd workshop on scripting for the semantic web, Budva, Montenegro.

Goldman NM (2003). Ontology-oriented programming : static typing for the inconsistent programmer. In Lecture notes in computer science : the SemanticWeb, ISWC, volume 2870, p. 850–865.

Kalyanpur A., Pastor D., Battle S. & Padget J. (2004). Automatic mapping of OWL ontologies into Java. In Proceedings of the Sixteenth International Conference on Software Engineering & Knowledge Engineering (SEKE'2004), p. 98–103.

Knublauch H., Oberle D., Tetlow P. & Wallace E. (2006). A Semantic Web Primer for Object-Oriented Software Developers. W3C Working Group Note.

Koide S., Aasman J. & Haflich S. (2005). OWL vs. Object Oriented Programming. In the 4th International Semantic Web Conference (ISWC 2005), Workshop on Semantic Web Enabled Software Engineering (SWESE).

Stevenson G. & Dobson S. (2011). Sapphire : Generating Java Runtime Artefacts from OWL Ontologies. In Lecture Notes in Business Information Processing, Advanced Information Systems Engineering Workshops , volume 83, p. 425–436.