Exemples de relations et d'interfaçages entre ontologies et d'autres approches/outils

Philippe A. Martin

philippe.martin@univ-reunion.fr

Université de La Réunion

Laboratoire d'Informatique et de Mathématiques (LIM)

(autres membres du LIM à AIME : Noël Conruyt, David Grosser)

www.phmartin.info/slides/aime2022/

Plan

1. **Quelques définitions et exemples**

2. **Génération de relations par règles, fonctions et acteurs**

3. **Début d'organisation de processus et méthodes de catégorisation/reconnaissance d'individus (e.g. poissons, coraux, ...)**

1. Quelques définitions informelles ...

Représentation de connaissances (RC) :

représentation/organisation (partielle) d'informations via

– une/des logique(s), e.g. la logique de description “SROIQ(D)”

– des relations sémantiques

(subtype, exclusion, =>, part, instrument, result, time, place, ...)

et parfois autres (e.g. relations lexicales)

→ *formules logiques (et graphes) représentant des informations.*

Base de connaissances (BC) :

1 ontologie : termes formels (identificateurs) + RCs les définissant

et 1 base de faits : RCs sur des objets qui ne sont pas des types

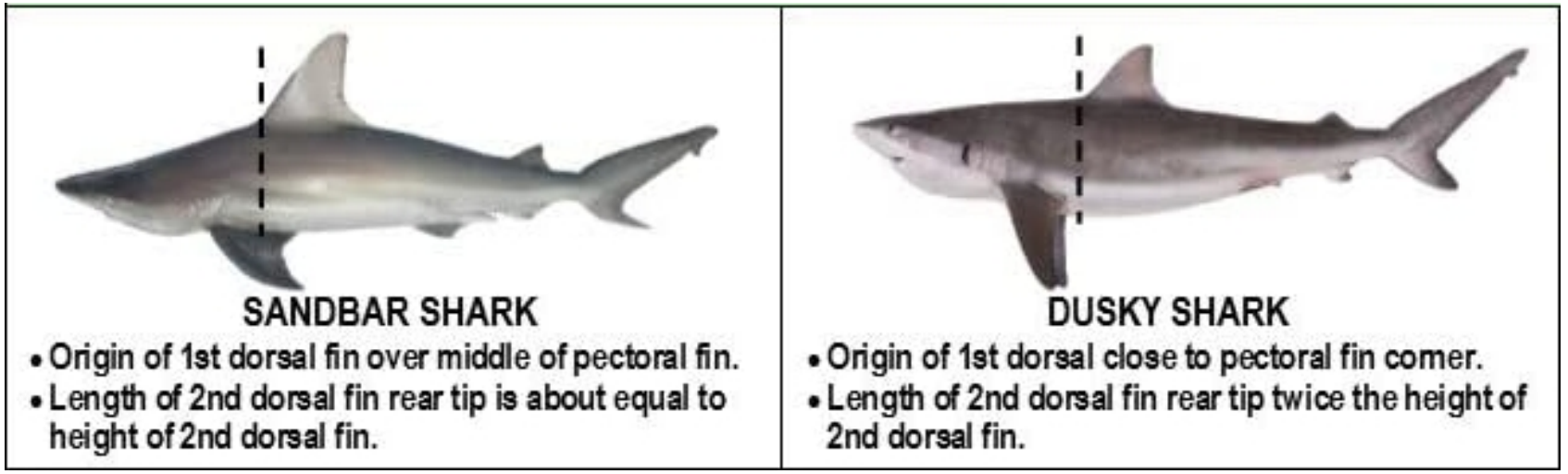
→ plus qu'une base de données (BdD : 1 schéma et des données) car

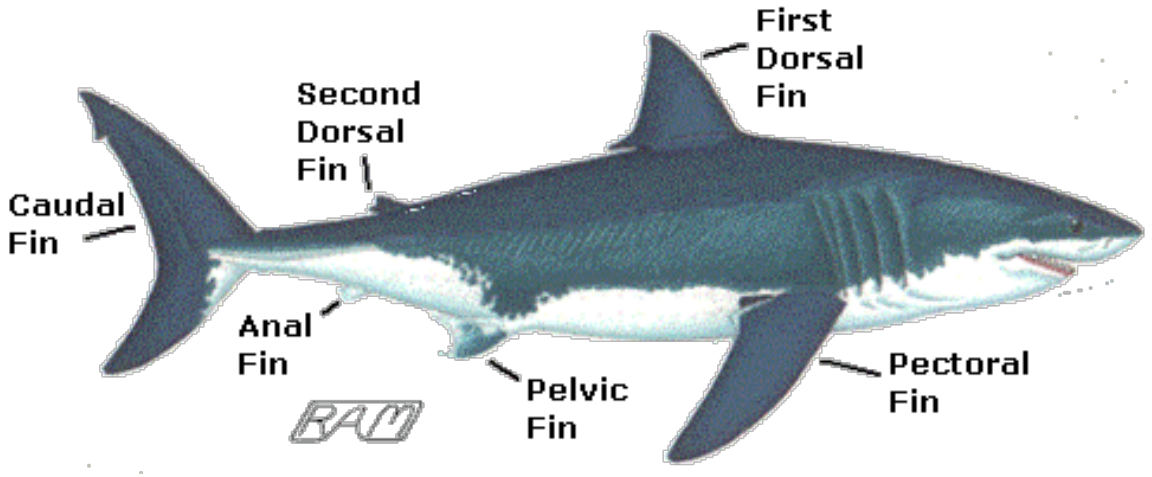
- les utilisateurs peuvent ajouter/définir de nouveaux termes et relier/organiser les objets comme souhaités
- les relations (définitions, faits, ...) permettent des inférences logiques

1. … exemple

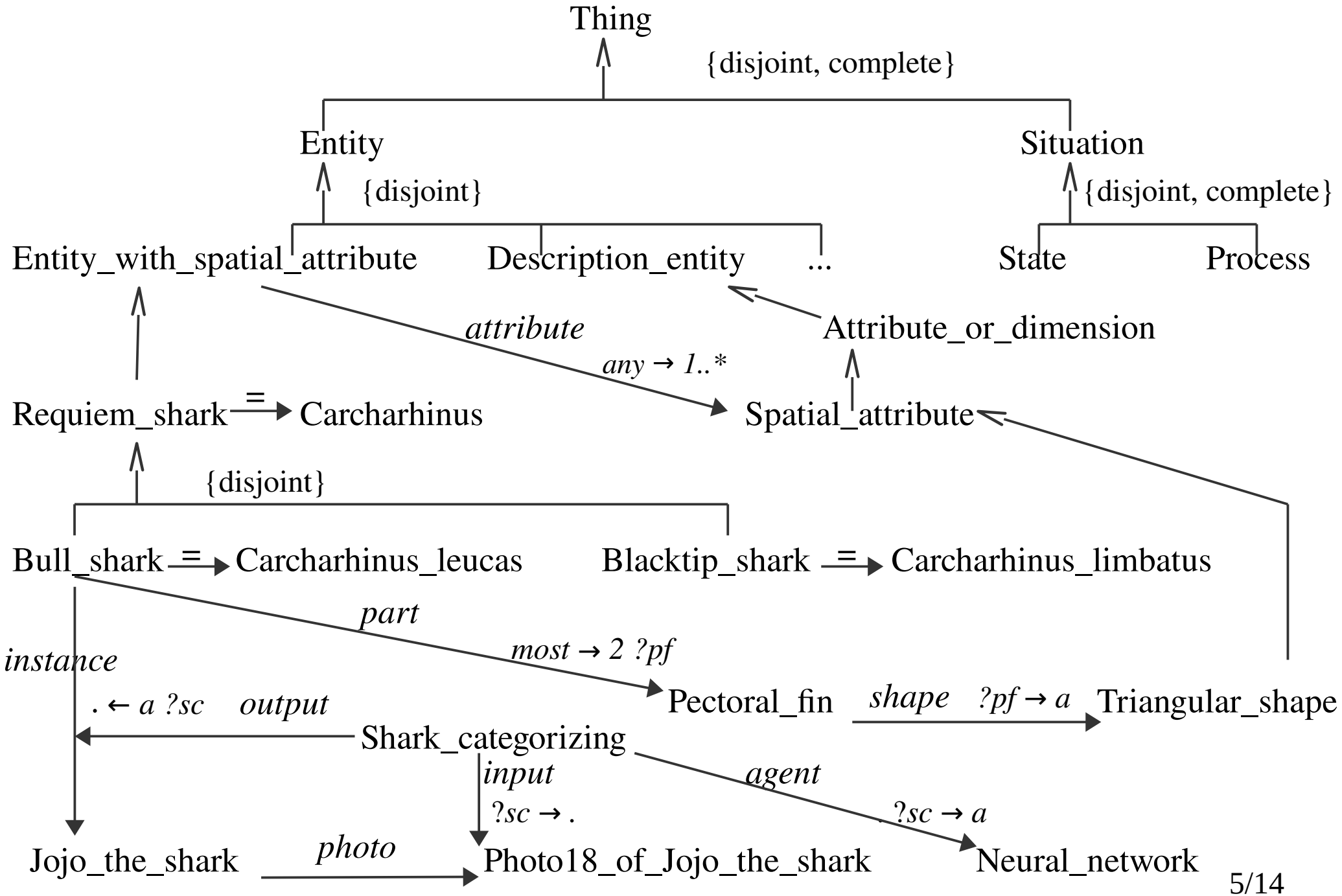
L'exemple des 2 pages suivantes fut inspiré par le texte et la 1ère image ci-dessous qui étaient inclus dans le précédent exposé par Morgan Mangeas.

HOW TO RECOGNIZE THE BULLDOG SHARK ?
The bulldog shark has a massive body, a grey back and a lighter belly.
The head is rounded, the eyes are small, **the pectoral fins are** long and **triangular in shape.** The first dorsal fin is wide. The caudal fin is heterocercal.





1. … exemple dans une notation graphique (~UML)



1. … le même exemple dans une notation textuelle (FL)

```
Thing
  \. p{ (Situation  \.  p{State  Process} )        // "p{...}": subtype partition
        (Entity                                    // "e{...}": exclusive subtypes
           \. e{ (Description_entity \. (Attribute_or_dimension
                                          \. (Spatial_attribute \. Triangular_shape) ) )
                 (Entity_with_spatial_attribute   part:=  1..* Spatial_attribute,
                   \. (Requiem_shark = Carcharhinus,
                       \. e{ (Blacktip_shark = Carcharhinus_limbatus)
                             (Bull_shark = Carcharhinus_leucas,
                                part _[most -> 2]:
                                                ^(Pectoral_fin  shape: a Triangular_shape),
                                instance: (Jojo_the_shark
                                             photo: Photo18_of_Jojo_The_Shark ?p18)
                                           _[output of: (a Shark_categorizing input: ?p18,
                                                          agent: a Neural_network) ]  )

                             } ) )
                 } )
     }.
```

2. Génération de relations – 2.1. ... via des règles

Exemple de règle (très arbitraire) **en FL** (3 versions équivalentes) **:**

[each Shark ?s [?s *part*: (a Pectoral_fin *shape*: a Triangular_shape))]
=> [?s *type*: Bull_shark]].

each ^(Shark *part*: (a Pectoral_fin *shape*: a Triangular_shape)) *type*: Bull_shark.

Bull_shark .(?s) :.<= [?s *type*: Shark, *part*: (a Pectoral_fin
shape: a Triangular_shape)].

→ à.p.d. F1: [Jojo_the_shark *part*: (a Pectoral_fin *shape*: a Triangular_shape)]
un moteur d'inférences qui comprend la règle précédente (R1) peut déduire
[Jojo_the_shark *type*: Bull_shark].

Une relation peut avoir des “contextes”, e.g. une probabilité et des sources :
[[[Jojo_the_shark *type*: Bull_shark] *probability*: 86%] .<= {F1,R1}]

... et “each” dans R1 peut aussi être remplacé par “86%” ou “au moins 86%”.

Une règle peut exploiter des valeurs symboliques (logiques, qualitatives, ...)
et/ou numériques (via des relations/fonctions sur ces valeurs).

2. Génération de relations – 2.2. ... via des fonctions

Exemple (en FL avec ici une syntaxe proche de C++ et de Javascript)
de fonctions générant une relation :

```
bool  is_a_Bullshark  ( Shark  ?s )
{  return   ?? [ ?s  part: (a Pectoral_fin  shape: a Triangular_shape)) ];   }
```

Exemple d'appel de cette fonction sur tous les individus de type Shark
(c'est le moteur d'inférences – ou interpréteur - qui exécute cette commande) :

```
for ( ?s : Shark ) {  if (is_a_Bullshark(?s))  assert( [?s type: Bull_shark] );  }
```

Une fonction peut utiliser des règles (via le moteur d'inférences, comme ci-dessus)
et des règles peuvent appeler des fonctions (mais cela peut parfois restreindre la direction des inférences : chaînage avant/arrière [what-if/how-to questions]).

De plus, les correspondances de Curry-Howard relient "fonctions et propositions",
et donc "généralisations de types/fonctions et implications logiques", et donc
- les inférences des moteurs d'inférences exploitant des règles logiques, avec
- celles des interpréteurs exploitant des signatures de fonctions

↔ autre façon de mélanger les deux : langage Curry, démonstrateur Coq, ...

2. Génération de relations – 2.3. ... via des “acteurs”

Dans une BC, un “acteur” [Sowa, 1984] est la représentation d'un appel d'une fonction qui utilise un programme extérieur (e.g. un SGBD ou un système basé sur un réseau neuronal).
Un “type d'acteur” est donc une fonction utilisant un programme extérieur.

Exemple de type d'acteur (en FL) **:**

```
bool  is_a_Bullshark  ( Shark  ?s )
{ for ( ?fileName  in  ? ?f [?s  photo: a File_name ?f] )
     if ( call ( “http://www.webkb.org/bin/is_a_Bullshark?photo=” + ?fileName ) )
       return true;
  return false;
}
```

Cette fonction est appelable exactement comme dans la page précédente.

Inversement (et similairement), pour qu'un SGBC (système de gestion de BC) puisse être exploité par un autre programme (e.g. un système de reconnaissance par réseau neuronal), il faut que ce SGBC ait une "interface de programmation" [API] ou une interface réseau/Web (e.g., utilisant le protocole GET/POST comme illustré ci-dessus).

3. Début d'organisation de processus et méthodes de catégorisation/reconnaissance d'individus

Automatic_categorization *//non-manual attribution of a type to an individual object*
\. p{ Automatic_categorization_not_based_on_learning
(Automatic_categorization_based_on_learning
\. **Automatic_categorization_based_on_supervised_learning**
Automatic_categorization_based_on_unsupervised_learning)
}.

Les 2 pages suivantes illustrent la spécialisation de ces 2 derniers types.
Il serait intéressant d'approfondir cela en représentant les types de méthodes et d'outils qui reconnaissent le mieux des individus en fonction de leurs types (types de poissons, coraux, ...), de leurs environnements (emplacement, ...) et des entrées utilisées (photos, vidéos, indicateurs, ...). Cela permettrait ensuite d'effectuer des choix de méthodes et d'outils en fonction de ces critères.

3. Début d'organisation de processus et méthodes de catégorisation/reconnaissance d'individus

Début de classification de **méthodes d'apprentissage supervisées**
par quelqu'un ayant seulement lu https://en.wikipedia.org/wiki/Supervised_learning

Supervised_learning_method
\. Support-vector_machine_based_supervised_learning_method
Linear_regression_based_supervised_learning_method
Logistic_regression_based_supervised_learning_method
Naive_Bayes_based_supervised_learning_method
Linear_discriminant_analysis_based_supervised_learning_method
Decision_tree_induction
K-nearest_neighbor_algorithm_based_supervised_learning_method
Neural_network_based_supervised_learning_method
Similarity_learning_based_supervised_learning_method .

3. Début d'organisation de processus et méthodes de catégorisation/reconnaissance d'individus

Début de classification de **méthodes d'apprentissage non supervisées**
par quelqu'un ayant seulement lu https://en.wikipedia.org/wiki/Unsupervised_learning

Unsupervised_learning_method
\. **Unsupervised_learning_method_for_neural_network //cf. page suivante**
(Unsupervised_learning_probabilistic_method
\. (Cluster_analysis
\. Hierarchical_clustering K-means Mixture-models_based_cluster_analysis)
(Anomaly-detection_based_unsupervised_learning_probabilistic_method
\. Local_outlier_factor Isolation_forest)
(Learning-latent-variable-model_based_unsupervised_learning_method
\. Expectation-maximization_based_unsupervised_learning_method
Method-of-moments_based_unsupervised_learning_method
(Blind-signal-separation_based_unsupervised_learning_method
\. Principal_component_based_unsupervised_learning_method
Independent_component_based_unsupervised_learning_method
Non-negative-matrix-factorization_based_unsupervised_learning_method
Singular_value_decomposition_based_unsupervised_learning_method
))).

3. Début d'organisation de processus et méthodes de catégorisation/reconnaissance d'individus

Suite du début de classification de **méthodes d'apprentissage non supervisées** par quelqu'un ayant seulement lu https://en.wikipedia.org/wiki/Unsupervised_learning

Unsupervised_learning_method_for_neural_network

\. Backpropagation_based_unsupervised_learning_method
Hopfield-learning-rule_based_unsupervised_learning_method
Boltzmann-learning-rule_based_unsupervised_learning_method
Contrastive-divergence_based_unsupervised_learning_method
Wake-sleep_based_unsupervised_learning_method
Variational-inference_based_unsupervised_learning_method
Maximum-likelihood_based_unsupervised_learning_method
Maximum-a-posteriori_based_unsupervised_learning_method
Gibbs-sampling_based_unsupervised_learning_method
Backpropagating-reconstruction-error_based_unsupervised_learning_method .

3. Début d'organisation de processus et méthodes de catégorisation/reconnaissance d'individus

Suite du début de classification de **types de réseaux neuronaux**
par quelqu'un ayant seulement lu https://en.wikipedia.org/wiki/Unsupervised_learning

Neural_network_used_in_unsupervised_learning
\. (Hopfield-inspired_neural_network
\. Hopfield_Network
(Stochastic_Hopfield_Net
\. Boltzmann_machine
Restricted_Boltzmann_machine //RBM
Stacked_Boltzmann_neural_network
(Helmholtz_inspired_neural_network
\. Helmholtz_machine
(Feed-forward_neural_network
\. Autoencoder_neural_network
Variational-autoencoder_neural_network))) //VAE
Deep-belief_neural_network)
(Probabilistic-model-inspired_neural_network
\. Deep-belief_neural_network Sigmoid-belief_net).