Base de connaissances RDF Outils pour bases de données syntaxiques structurées

Nicolas Mazziotta
Université de Liège/U. Stuttgart, ILR

20 janvier 2016, Paris

Plan

Objectifs

Objectifs

Partager connaissances techniques

- Langages d'encodage, outils de traitement
- En cours (« bricolage » parfois)

Objectifs

Partager connaissances techniques

- Langages d'encodage, outils de traitement
- En cours (« bricolage » parfois)

Type de données

- Données syntaxiques structurées (structuration complexe)
- Données croisées SRCMF $\times$ BFM

Objectifs

Partager connaissances techniques

- Langages d'encodage, outils de traitement
- En cours (« bricolage » parfois)

Type de données

- Données syntaxiques structurées (structuration complexe)
- Données croisées SRCMF $\times$ BFM

Importance de la théorisation

- Comprendre la structure des données
- Comprendre les enjeux épistémologiques

Problématisation théorique

Enjeu théorique

- Question générale : représentation des connaissances (constructs)

Problématisation théorique

Enjeu théorique

- Question générale : représentation des connaissances (constructs)
- Assertions sur les relations
- Relations entre relations (réification possible)

Problématisation théorique

Enjeu théorique

- Question générale : représentation des connaissances (constructs)
- Assertions sur les relations
- Relations entre relations (réification possible)

Encoder la syntaxe

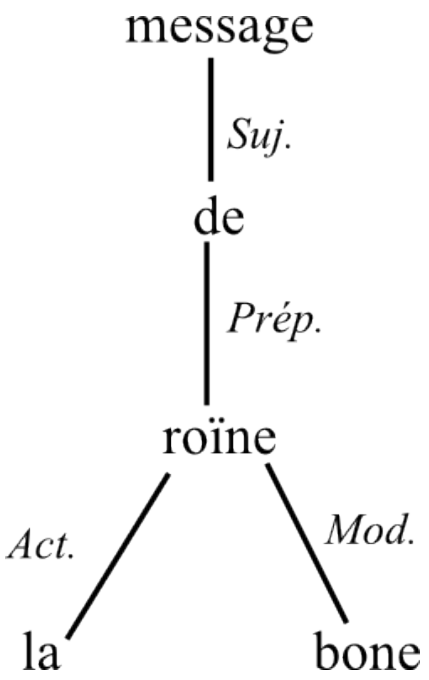
Problématisation théorique

Enjeu théorique

- Question générale : représentation des connaissances (constructs)
- Assertions sur les relations
- Relations entre relations (réification possible)

Encoder la syntaxe

- **arbre** (PSG, MTTSurfaceSyntax,. . .) : nœuds + arêtes orientées entre les nœuds (un nœud ne peut être la cible de plusieurs arêtes)



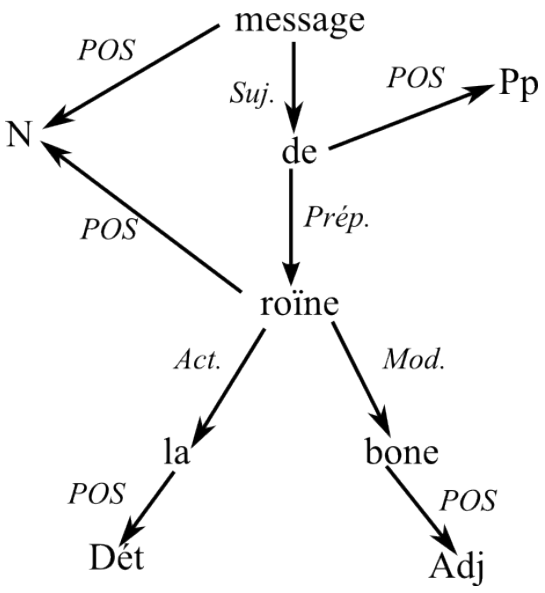
Problématisation théorique

Enjeu théorique

- Question générale : représentation des connaissances (constructs)
- Assertions sur les relations
- Relations entre relations (réification possible)

Encoder la syntaxe

- arbre (PSG, MTTSurfaceSyntax,. . .) : nœuds + arêtes orientées entre les nœuds (un nœud ne peut être la cible de plusieurs arêtes)
- **graphe** orienté acyclique (Word Grammar, LFG,. . .) : nœuds + arêtes orientées entre les nœuds sans cycle



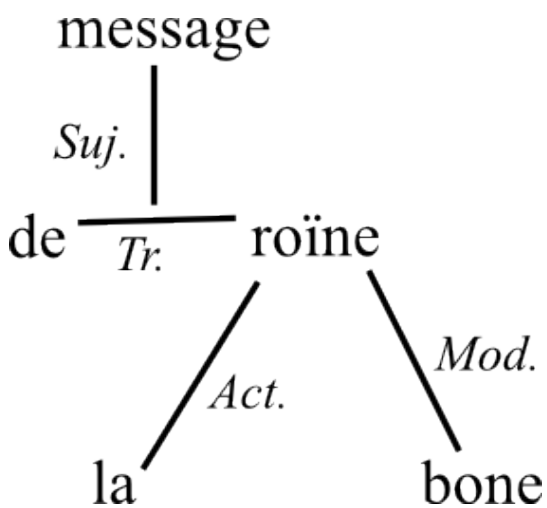
Problématisation théorique

Enjeu théorique

- Question générale : représentation des connaissances (constructs)
- Assertions sur les relations
- Relations entre relations (réification possible)

Encoder la syntaxe

- arbre (PSG, MTTSurfaceSyntax,. . .) : nœuds + arêtes orientées entre les nœuds (un nœud ne peut être la cible de plusieurs arêtes)
- graphe orienté acyclique (Word Grammar, LFG,. . .) : nœuds + arêtes orientées entre les nœuds sans cycle
- **polygraphe** (Nida, Tesnière (translation)) graphe, mais avec éventuellement des arêtes entre le arêtes



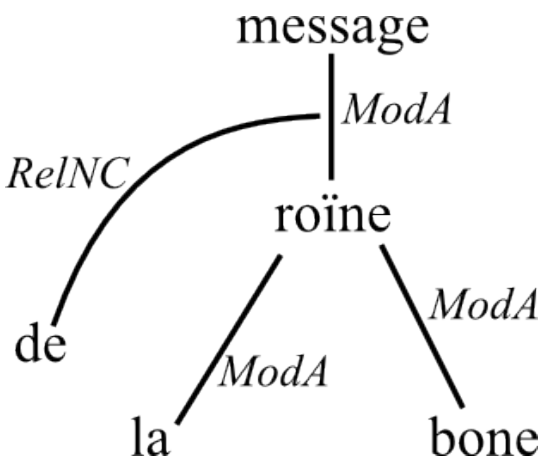
Problématisation théorique

Enjeu théorique

- Question générale : représentation des connaissances (constructs)
- Assertions sur les relations
- Relations entre relations (réification possible)

Encoder la syntaxe

- arbre (PSG, MTTSurfaceSyntax,. . .) : nœuds + arêtes orientées entre les nœuds (un nœud ne peut être la cible de plusieurs arêtes)
- graphe orienté acyclique (Word Grammar, LFG,. . .) : nœuds + arêtes orientées entre les nœuds sans cycle
- **polygraphe** (Nida, Tesnière (translation)) graphe, mais avec éventuellement des arêtes entre le arêtes



Structures de SRCMF et principe de réification

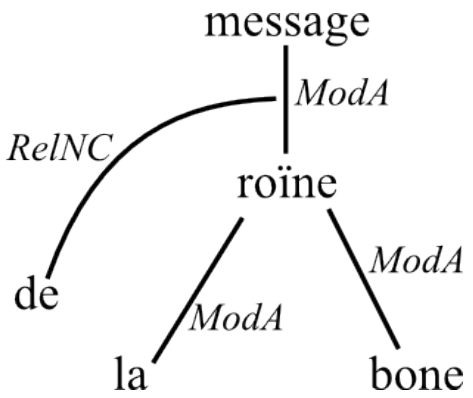
Relateurs

- Dans le modèle théorique SRCMF, **prépositions et conjonctions dépendent** des structures et leur permettent d'assumer une fonction (Kahane/Mazziotta 2015 (Depling))

Structures de SRCMF et principe de réification

Relateurs

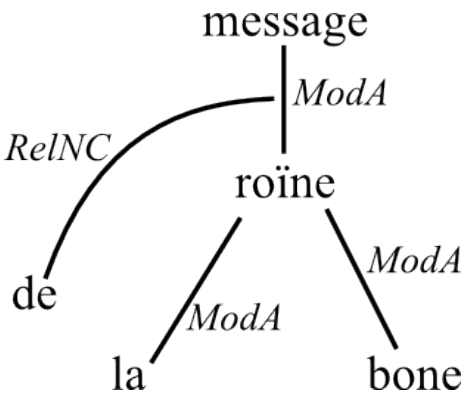
- Dans le modèle théorique SRCMF, **prépositions et conjonctions dépendent** des structures et leur permettent d'assumer une fonction (Kahane/Mazziotta 2015 (Depling))
- Un terme dépend d'une relation ⇒ polygraphe



Structures de SRCMF et principe de réification

Relateurs

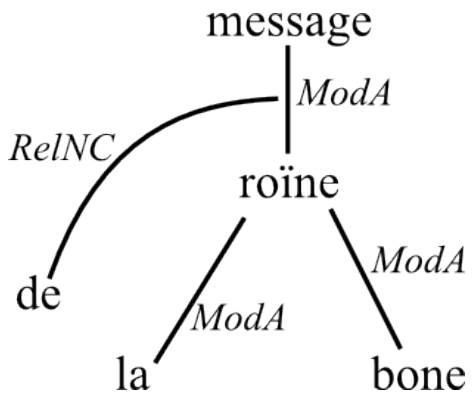
- Dans le modèle théorique SRCMF, **prépositions et conjonctions dépendent** des structures et leur permettent d'assumer une fonction (Kahane/Mazziotta 2015 (Depling))
- Un terme dépend d'une relation ⇒ polygraphe
- MAIS aucun outil ne gère les polygraphes ⇒ besoin de **réifier** les relations

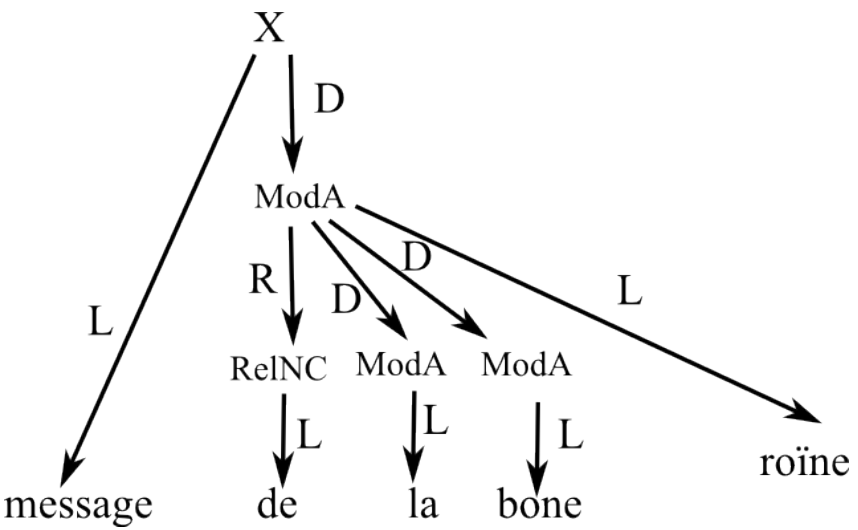


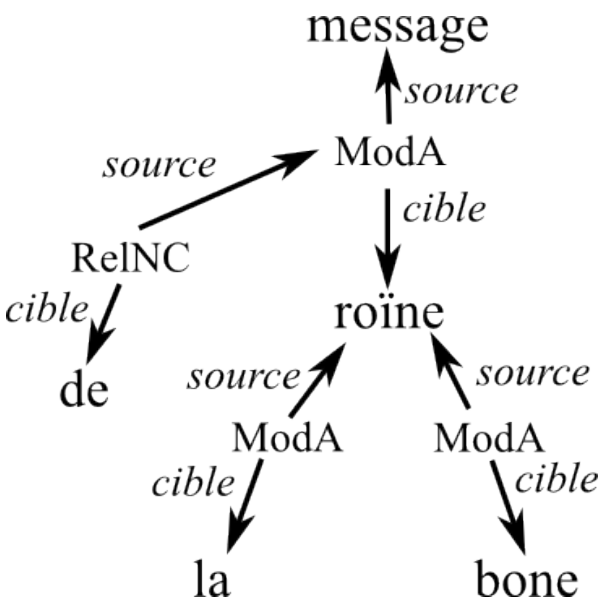
Structures de SRCMF et principe de réification

Relateurs

- Dans le modèle théorique SRCMF, **prépositions et conjonctions dépendent** des structures et leur permettent d'assumer une fonction (Kahane/Mazziotta 2015 (Depling))
- Un terme dépend d'une relation ⇒ polygraphe
- MAIS aucun outil ne gère les polygraphes ⇒ besoin de **réifier** les relations



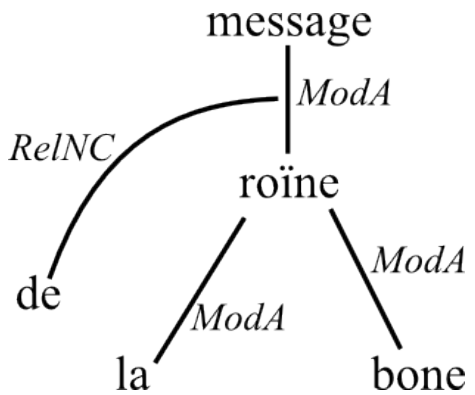


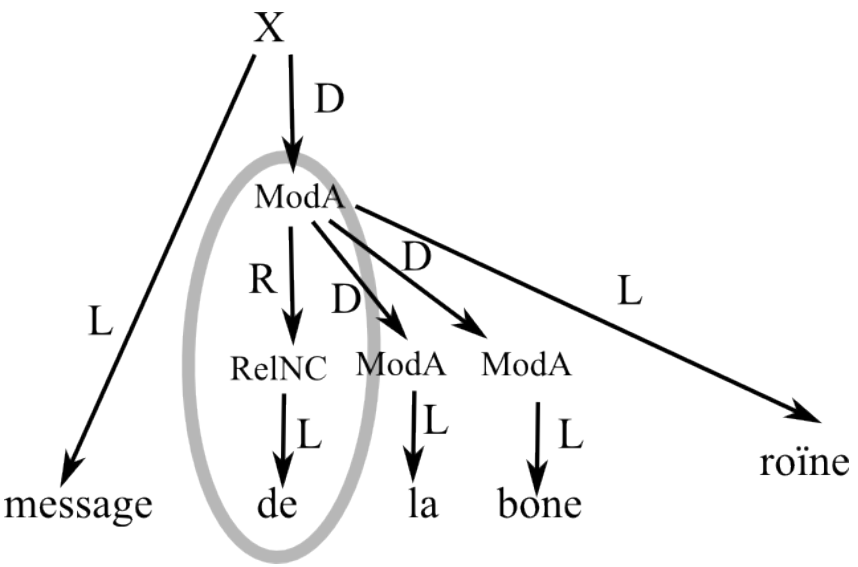


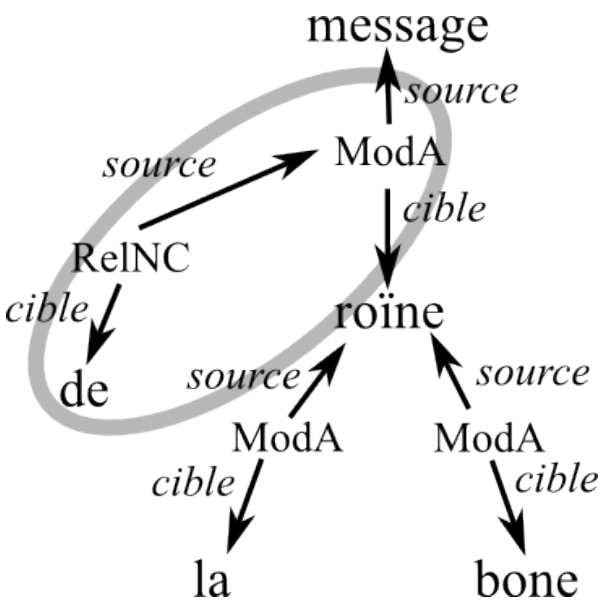
Structures de SRCMF et principe de réification

Relateurs

- Dans le modèle théorique SRCMF, **prépositions et conjonctions dépendent** des structures et leur permettent d'assumer une fonction (Kahane/Mazziotta 2015 (Depling))
- Un terme dépend d'une relation ⇒ polygraphe
- MAIS aucun outil ne gère les polygraphes ⇒ besoin de **réifier** les relations





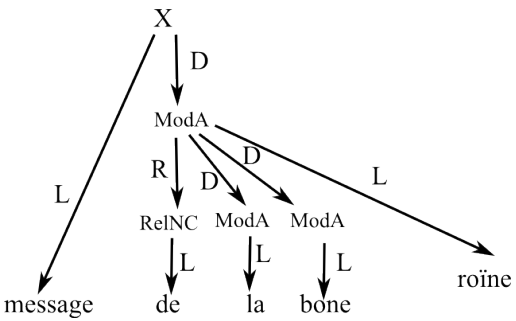


TigerXML

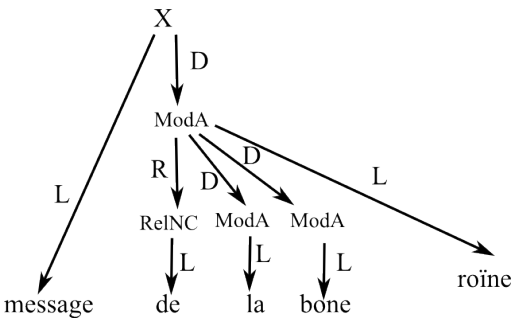
TigerXML encode principalement des arbres

- Distinction entre les nœuds terminaux/non terminaux.
- Nœud et arêtes sont complètement distincts
- Les nœuds ont des propriétés distinctes des arêtes auxquelles ils sont liés

Langage de requête



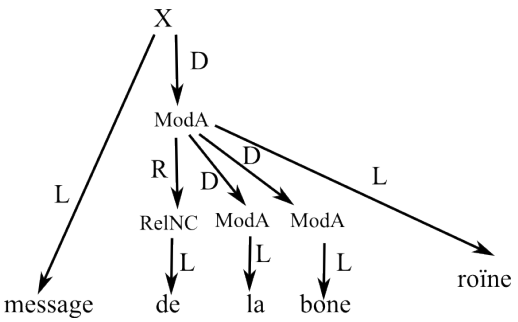
Langage de requête



Langage de requête

- Nœuds : mots, fonctions
  [ ]

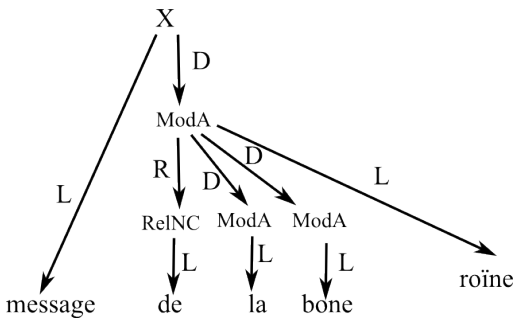
Langage de requête



Langage de requête

- Nœuds : mots, fonctions
  `[]`
- Propriétés : caractéristiques des nœuds
  `[x='y']`

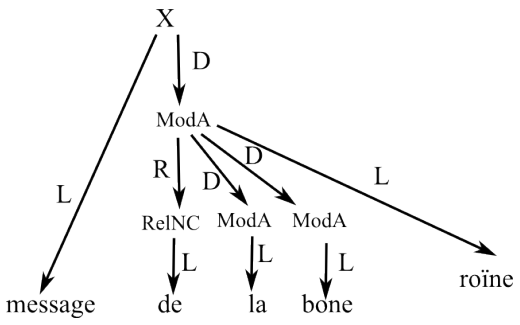
Langage de requête



Langage de requête

- Nœuds : mots, fonctions
  `[]`
- Propriétés : caractéristiques des nœuds
  `[x='y']`
- Arêtes (primaires/secondaires) : entre les nœuds
  `>x` et `>∼x`

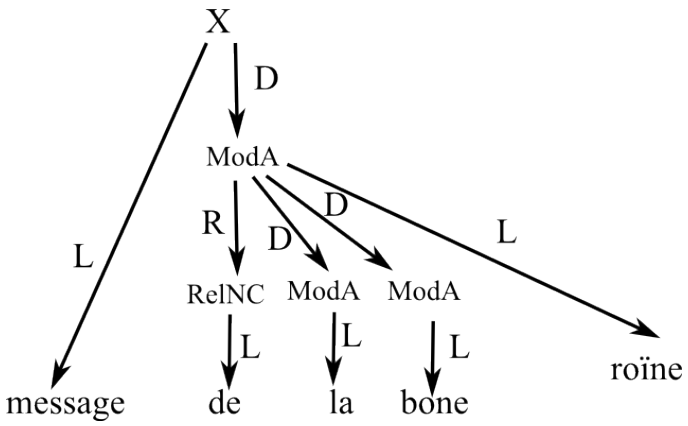
Langage de requête



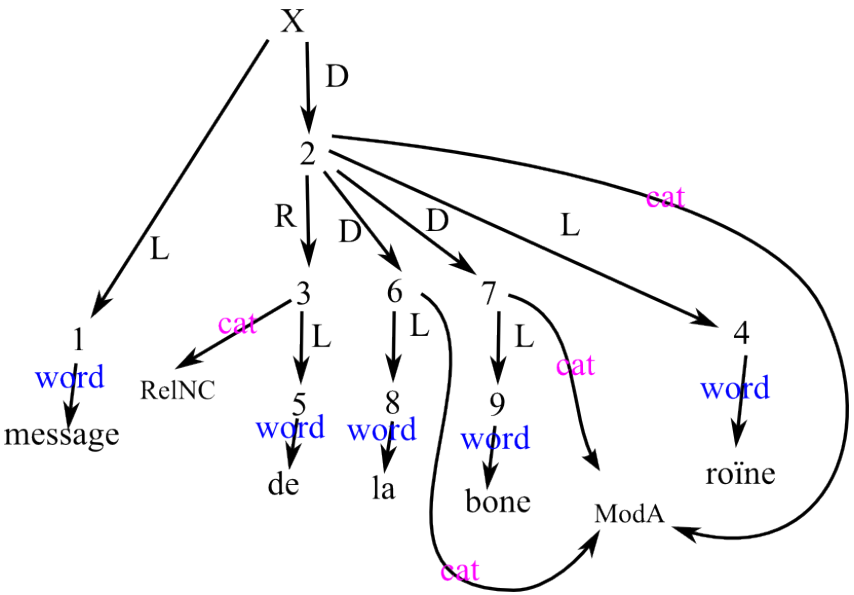
Langage de requête

- Nœuds : mots, fonctions
  `[]`
- Propriétés : caractéristiques des nœuds
  `[x='y']`
- Arêtes (primaires/secondaires) : entre les nœuds
  `>x` et `>~x`
- Variables : enregistrent les nœuds (traits et contexte)
  `#x: []`

Exemple : rechercher les *ki* relatifs sujets

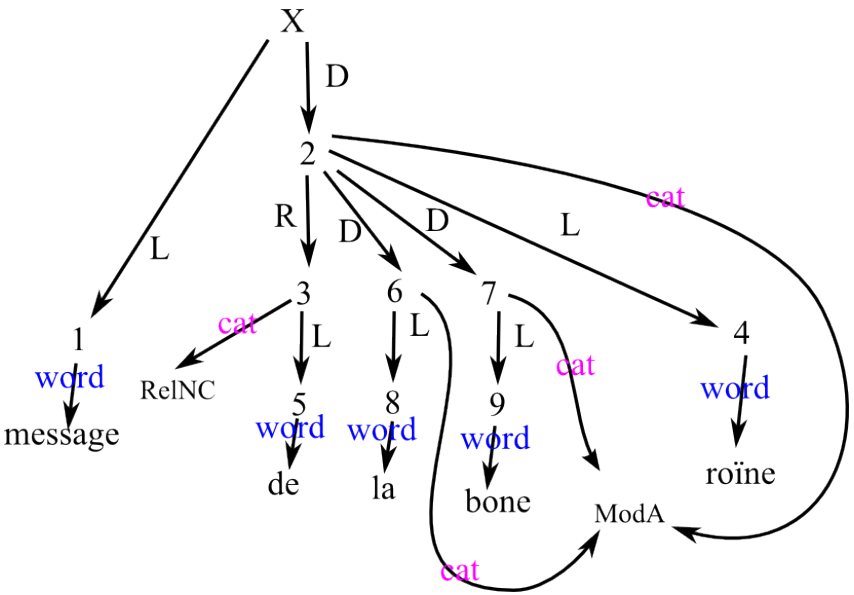


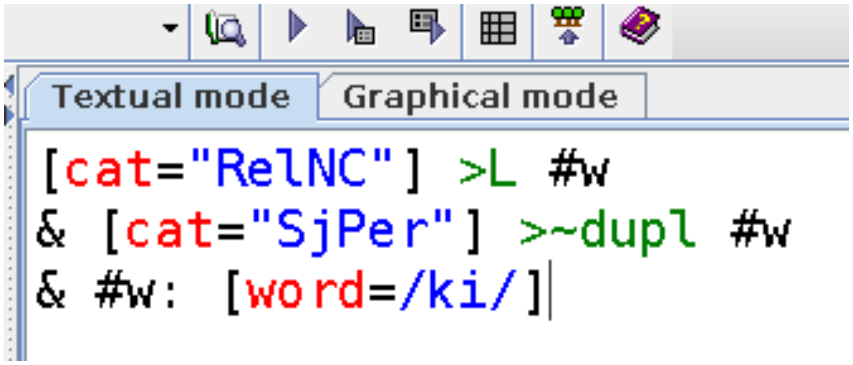
Exemple : rechercher les *ki* relatifs sujets



Traitement différent pour les propriétés

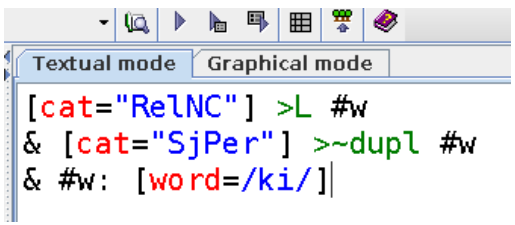
Exemple : rechercher les *ki* relatifs sujets





Traitement différent pour les propriétés

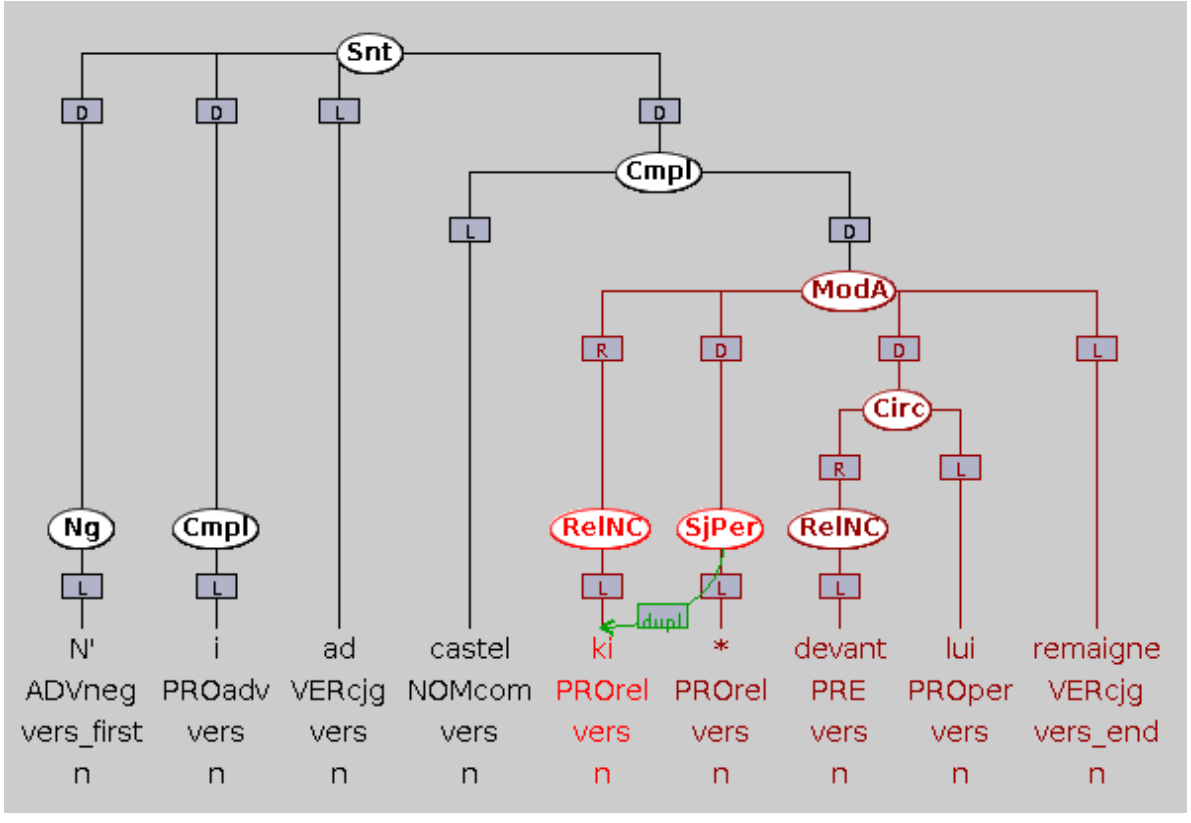
Exemple : rechercher les *ki* relatifs sujets



Exemple : rechercher les *ki* relatifs sujets

Textual mode | Graphical mode

```
[cat="RelNC"] >L #w
& [cat="SjPer"] >~dupl #w
& #w: [word=/ki/]
```



RDF

RDF encode des graphes

- Sous la forme de triplets (suite ordonnée de trois unités) : *sujet*, *prédicat*, *objet*.

RDF

RDF encode des graphes

- Sous la forme de triplets (suite ordonnée de trois unités) : *sujet*, *prédicat*, *objet*.
  Exemples :
  - (Jean) (mange) (une pomme)
  - xxx yyy zzz

RDF

RDF encode des graphes

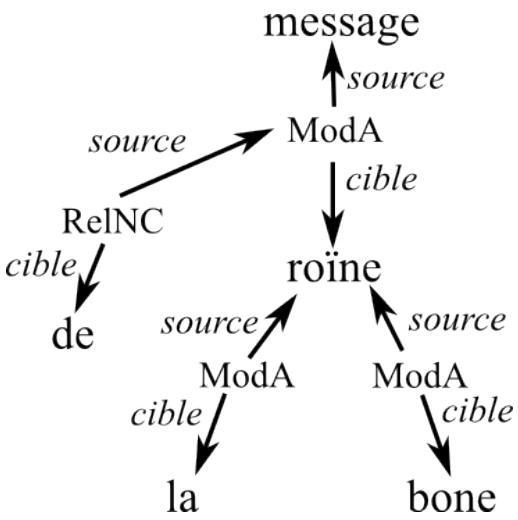
- Sous la forme de triplets (suite ordonnée de trois unités) : *sujet*, *prédicat*, *objet*.
  Exemples :
  - (Jean) (mange) (une pomme)
  - xxx yyy zzz
- Nœud et arêtes sont des *ressources* représentées par une URI

RDF

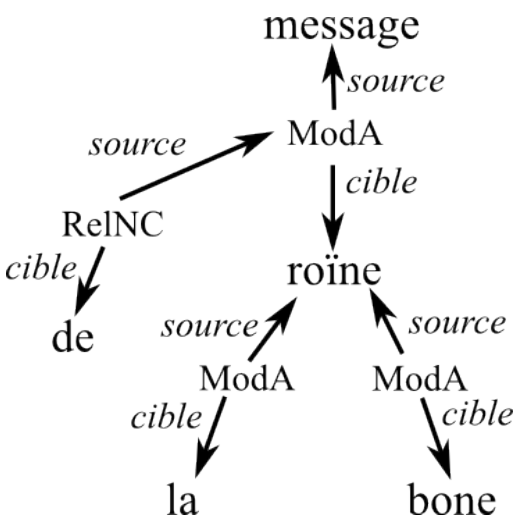
RDF encode des graphes

- Sous la forme de triplets (suite ordonnée de trois unités) : *sujet*, *prédicat*, *objet*.
  Exemples :
  - (Jean) (mange) (une pomme)
  - xxx yyy zzz
- Nœud et arêtes sont des *ressources* représentées par une URI
- Localisent les ressources dans des espaces de nommage (URI)

SPARQL


message
source
source
ModA
cible
RelNC
cible
roïne
de
source
source
ModA
ModA
cible
cible
la
bone

SPARQL

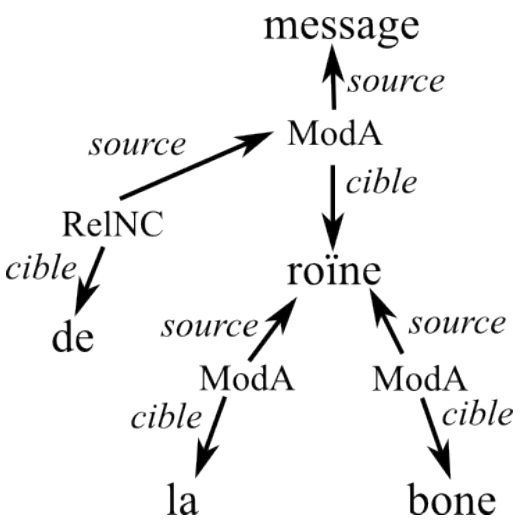


Langage de requête SPARQL

Requête = graphe avec des « jokers » → ressort tout ce qui convient

- ressources : URI
  `http://xyz/w` ou `xyz:w`

SPARQL

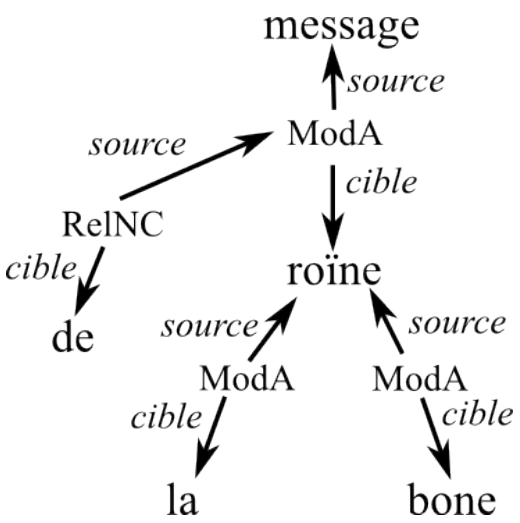


Langage de requête SPARQL

Requête = graphe avec des « jokers » → ressort tout ce qui convient

- ressources : URI
  `http://xyz/w` ou `xyz:w`
- Variables enregistrent les ressources et leur contexte
  `?x`

SPARQL

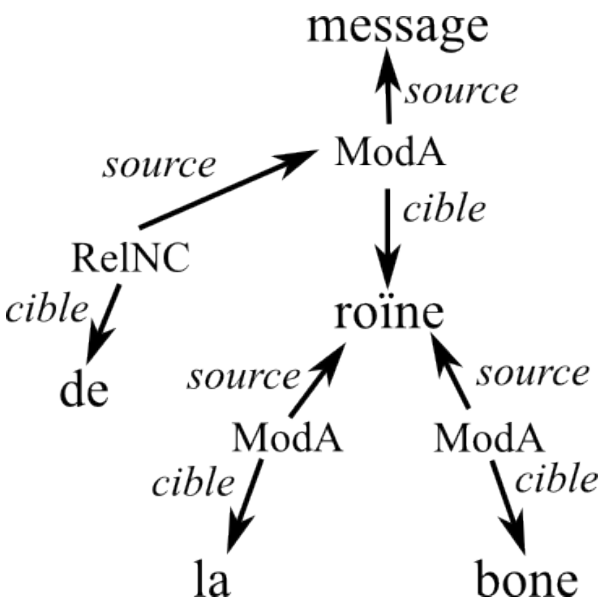


Langage de requête SPARQL

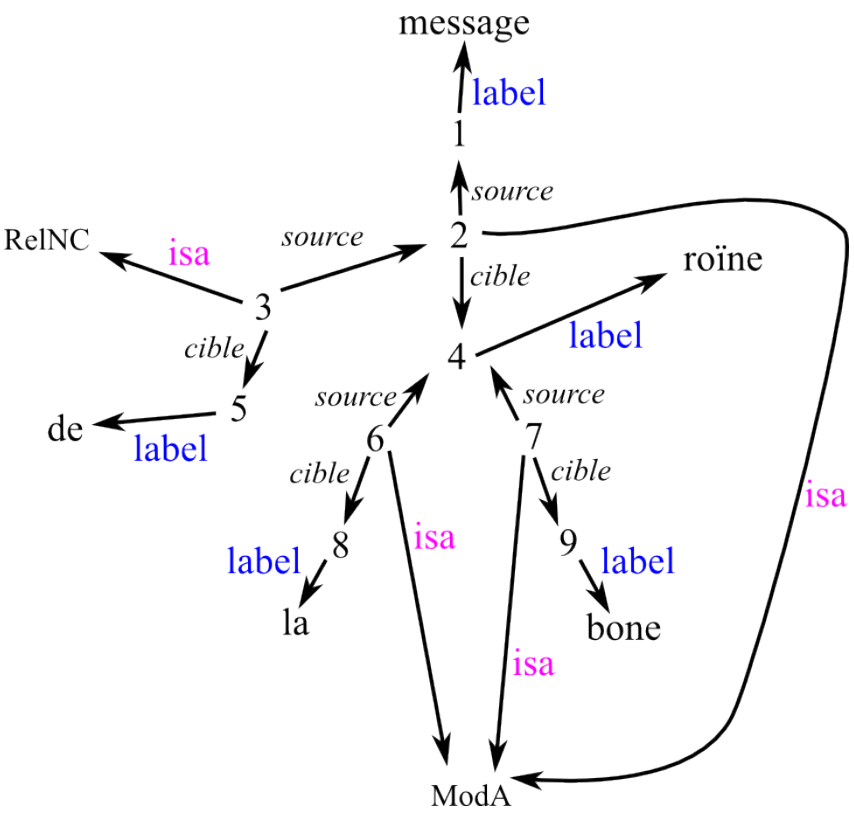
Requête = graphe avec des « jokers » → ressort tout ce qui convient

- ressources : URI
  `http://xyz/w` ou `xyz:w`
- Variables enregistrent les ressources et leur contexte
  `?x`
- Syntaxe simplifiant les « phrases » :
  `RelNC source ModA. RelNC cible de.`
  =
  `RelNC source ModA; cible de.`

RDF et SPARQL

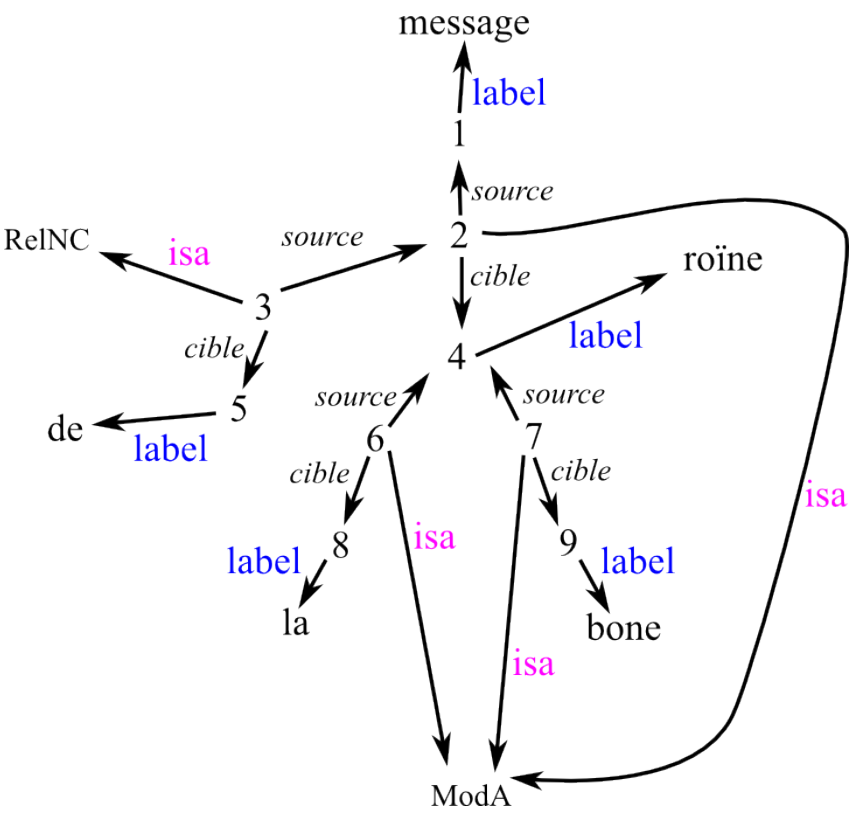


RDF et SPARQL



Traitement identique pour toutes les arêtes

RDF et SPARQL



Traitement identique pour toutes les arêtes

Selected dataset: ds

```
1  SELECT DISTINCT ?focus ?ctxt_rel
2
3  WHERE {
4
5    ?focus    rdfs:label ?lab .
6    ?ctxt_rel a srcmf:Relateur-NonCoordonnant ;
7              sdep:target ?focus .
8
9    FILTER REGEX(STR(?lab),'^ki$')
10
11 } ORDER BY LCASE(?lab)
12 LIMIT 10
```

Exemple : rechercher les *ki* relatifs sujets

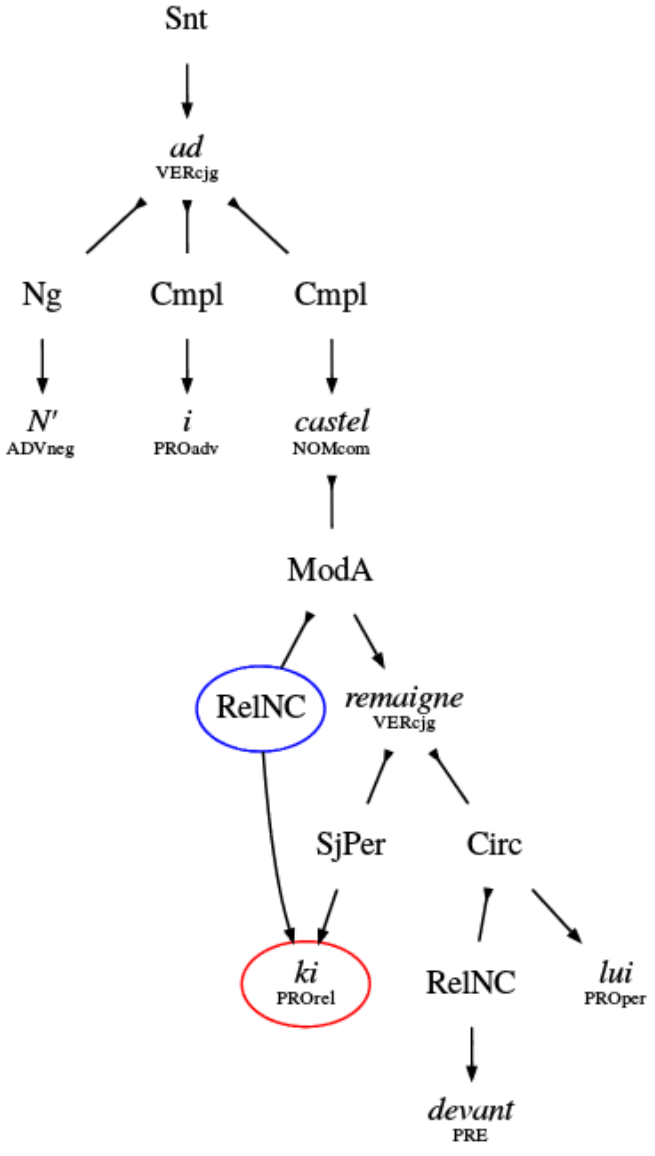
Selected dataset: ds

```
1  SELECT DISTINCT ?focus ?ctxt_rel
2
3  WHERE {
4
5    ?focus    rdfs:label ?lab .
6    ?ctxt_rel a srcmf:Relateur-NonCoordonnant ;
7              sdep:target ?focus .
8
9    FILTER REGEX(STR(?lab),'^ki$')
10
11 } ORDER BY LCASE(?lab)
12 LIMIT 10
```

Exemple : rechercher les *ki* relatifs sujets

Selected dataset: ds

```
1  SELECT DISTINCT ?focus ?ctxt_rel
2
3  WHERE {
4
5    ?focus    rdfs:label ?lab .
6    ?ctxt_rel a srcmf:Relateur-NonCoordonnant ;
7              sdep:target ?focus .
8
9    FILTER REGEX(STR(?lab),'^ki$')
10
11 } ORDER BY LCASE(?lab)
12 LIMIT 10
```



Démonstration

Recherche et enrichissement