INSA INSTITUT NATIONAL DES SCIENCES APPLIQUÉES LYON

Cours Web Sémantique- 4IF

Partie 5 : Raisonner : inférences et RIF

Sylvie Calabretto et Mehdi Kaytoue

Plan du cours

- Introduction au web sémantique
- Décrire : RDF
- Représenter les connaissances : ontologies, RDF-S et OWL
- Interroger : SPARQL
- Raisonner : inférences et RIF

Pourquoi des règles dans le web sémantique ?

- **Il existe des conditions que OWL ne peut pas exprimer**

  Règles de Horn: (P1 ∧ P2 ∧ …) → C

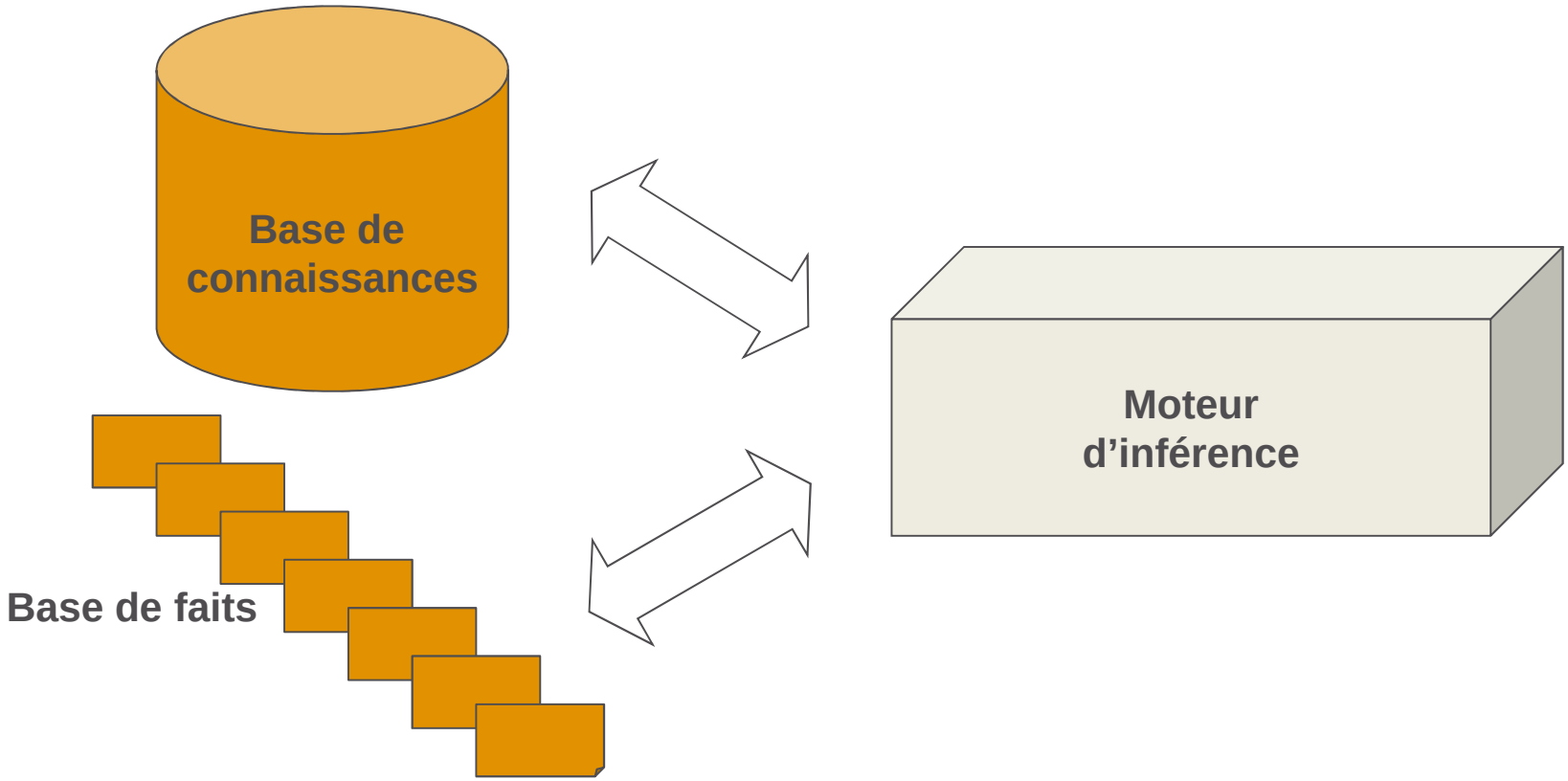
  Exemple :

  SI instance(X, Professor) ET possede_diplome(X, PhD)

  ALORS peut_etre_membre(X, JuryDoctorat)

- **Les langages de règles du web sémantique sont plus faciles à manipuler que le langage OWL**
- **Les langages de règles et les moteurs d'inférence du web sémantique trouvent leurs origines dans les systèmes experts/systèmes à base de connaissances**

Architecture initiale d'un Système à Base de Connaissances (SBC)



SBC à partir de règles

- **Il s'agit des SBC historiques que l'on appelait initialement « systèmes experts »**

  Les connaissances expertes sont représentées par des règles (règles de production) de la forme

  Si (prémisses) Alors (conclusions)
  - Prémisses = conditions de déclenchement de la règle
  - Conclusions = actions de la règle

  Les connaissances sont déclaratives (révisables en principe)

  L'ensemble des règles forme « la base de connaissances ».

  Les faits décrivent ce qui est vrai dans la situation d'exploitation de la base de règles (base de faits).

SBC à partir de règles : les systèmes experts

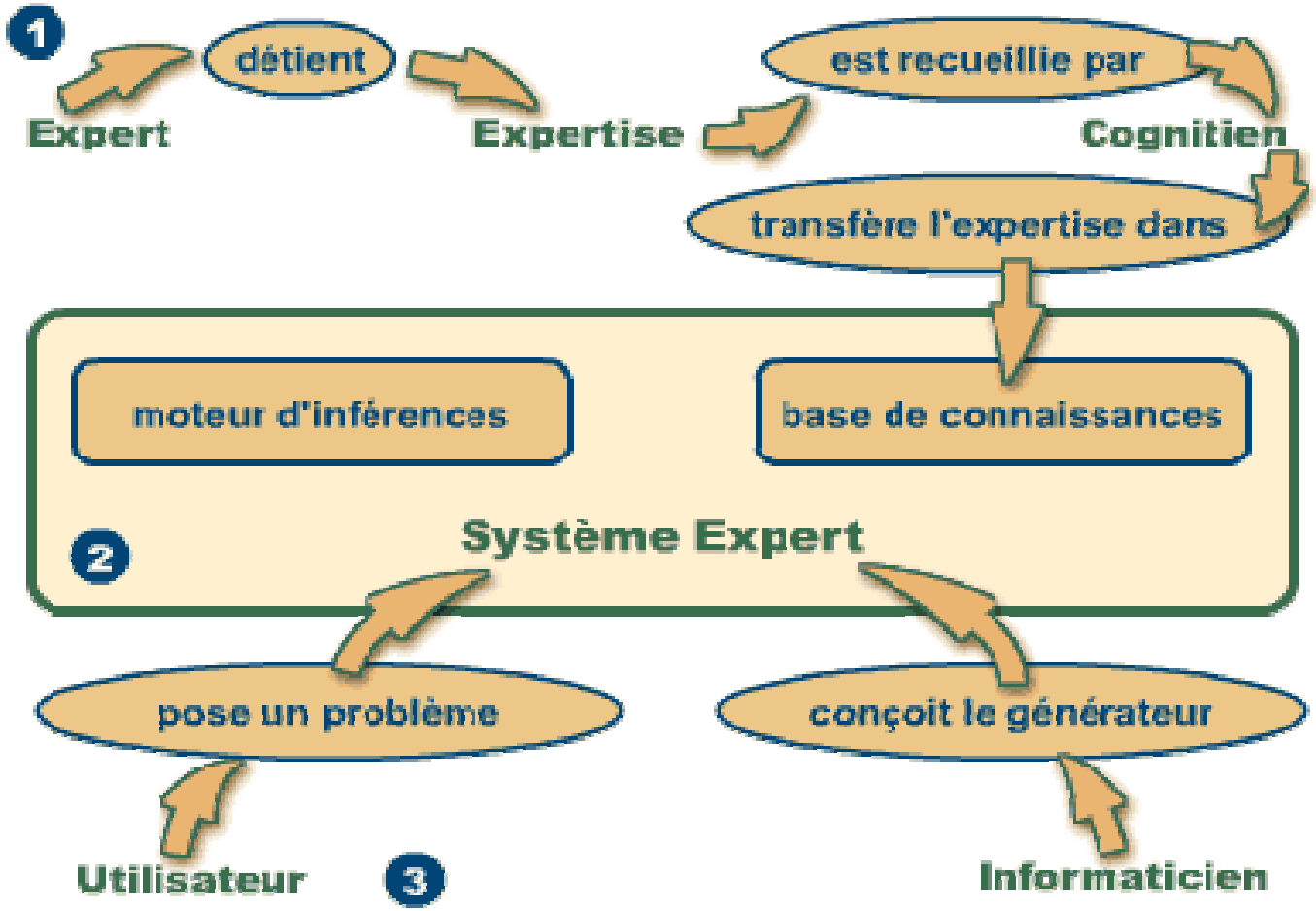
- **Un système expert se compose de 3 parties :**
  - Une base de faits
  - Une base de connaissances ou base de règles
  - Un moteur d'inférence : le moteur d'inférence est capable d'utiliser faits et règles pour produire de nouveaux faits, jusqu'à parvenir à la réponse à la question experte posée.

SBC à partir de règles : les systèmes experts

- **Un système expert est un outil capable de reproduire les mécanismes cognitifs d'un expert dans un domaine particulier**
- **Plus précisément, un système expert est un logiciel capable de répondre à des questions, en effectuant un raisonnement à partir de faits et de règles connus.**

  Il peut servir notamment comme outil d'aide à la décision

SBC à partir de règles : les systèmes experts



Système à Base de Connaissances : Historique

- **Les premiers systèmes experts voient le jour aux USA dans les années 1970**
- **MYCIN, qui manipulait de l'expertise dans le domaine médical, est l'un des plus connus**
- **Exemple de règle :**

Si :
- la coloration de l'organisme est GRAM négatif
- sa morphologie est un bâtonnet
- il est aérobie

Alors :
- il est vraisemblable (0,8) que l'organisme est un Enterobactariaceae

Système à Base de Connaissances : Mécanismes de raisonnement

- **La plupart des systèmes experts existants reposent sur des mécanismes de logique formelle et utilisent le raisonnement déductif (déduction)**
- **Déduction :**

  Si A est vrai (fait ou prémisse) et si on sait que A -> B (règle)

  Alors B est vrai (nouveau fait ou conclusion)

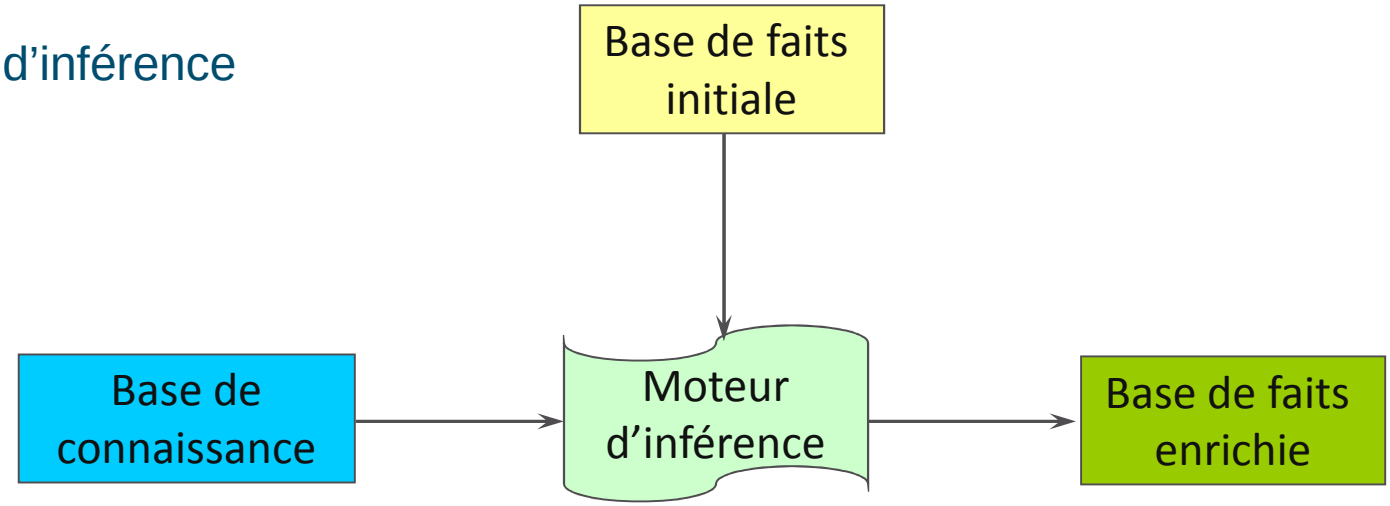
- **Les systèmes experts les plus simples s'appuient sur la logique des propositions (logique d'ordre 0)**

  Dans cette logique, on n'utilise que des propositions, qui sont vraies ou fausses

- **D 'autres systèmes s'appuient sur la logique du premier ordre (logique d'ordre 1)**

Système à Base de Connaissances : Raisonnement à base de règles

- **3 composants essentiels :**
  - Un ensemble de règles
  - Un ensemble de faits
  - Un moteur d'inférence



Système à Base de Connaissances : Base de règles

- **La base de règles rassemble la connaissance et le savoir-faire de l'expert.**
- **Elle n'évolue donc pas au cours d'une session de travail.**
- **Une règle se présente sous la forme : si X alors Y**

  X est la prémisse : c'est une conjonction de conditions, i.e. une suite de comparaison d'attributs et de valeurs à l'aide d'opérateurs.

  Y est la conclusion : la conclusion est une affectation.

  Exemple de règle :

```
si      l'âge du patient < 18          et
si      il a de la fièvre > 39°        et
si      présence d'un germe X
alors   le patient a peut-être une méningite
```

Système à Base de Connaissances : Base de faits

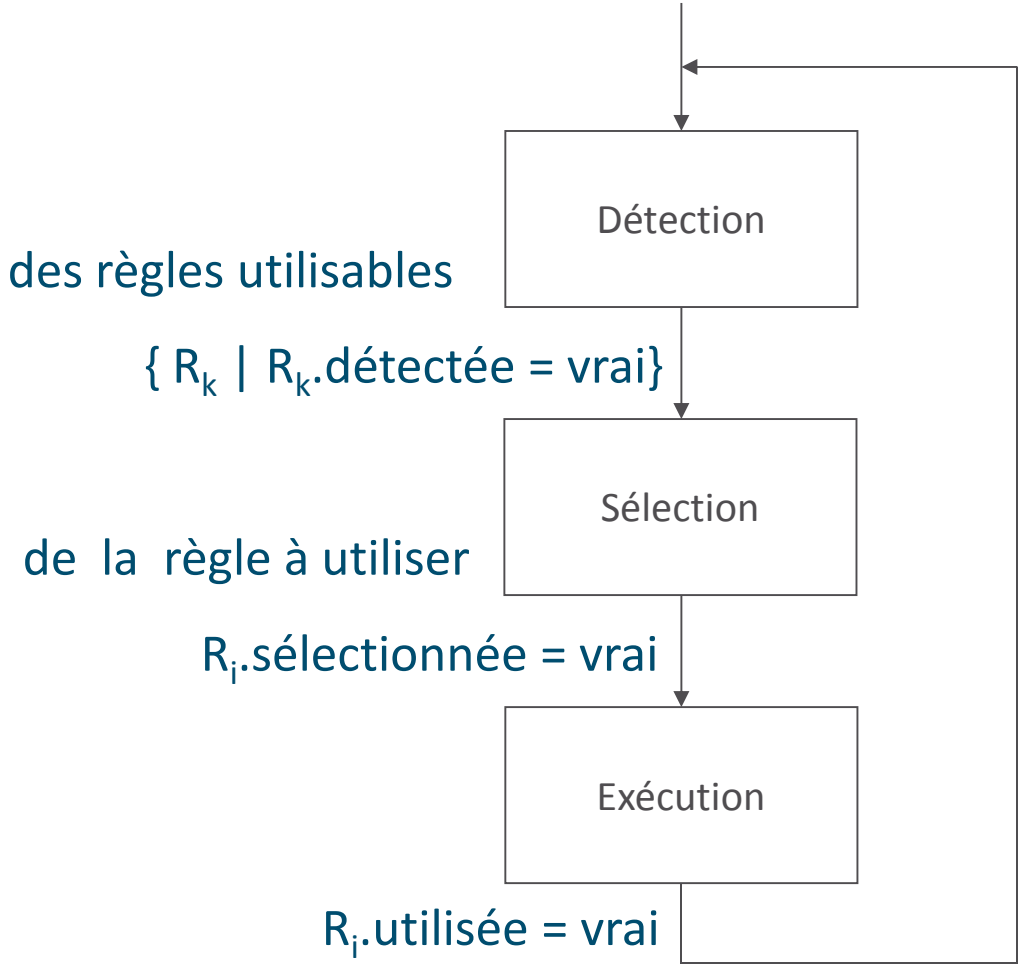
- La base de faits constitue la mémoire de travail du SBC.
- Elle est variable au cours de l'exécution et vidée lorsque l'exécution se termine.
- Au début de la session, elle contient tout ce que l'on sait à propos du cas examiné avant toute intervention du moteur d'inférence.
- A la fin, elle est complétée par les faits déduits par le moteur ou demandés à l'utilisateur.
- Exemple de base de faits :

```
Âge est 6
Fièvre est 40°
Germe X
Sexe féminin
```

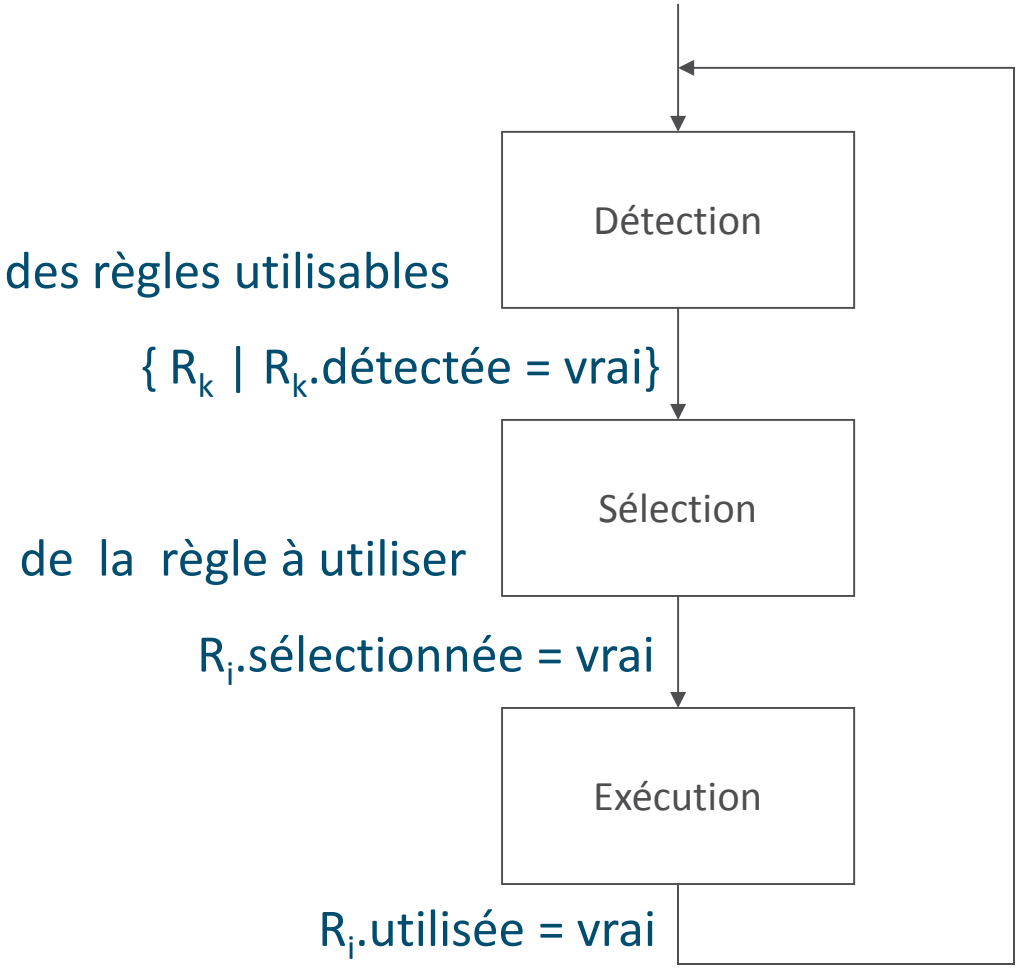
Système à Base de Connaissances : Moteur d'inférence

- **Il existe de nombreux types de moteurs, capables de traiter différentes formes de règles logiques pour déduire de nouveaux faits à partir de la base de connaissances.**
- **On distingue 3 catégories, basées sur la manière dont les problèmes sont résolus :**
  - Les moteurs à « chaînage avant »
  - Les moteurs à « chaînage arrière »
  - Les moteurs à « chaînage mixte »
- **Certains moteurs d'inférence peuvent être partiellement pilotés ou contrôlés par des méta-règles qui modifient leur fonctionnement et leurs modalités de raisonnement**

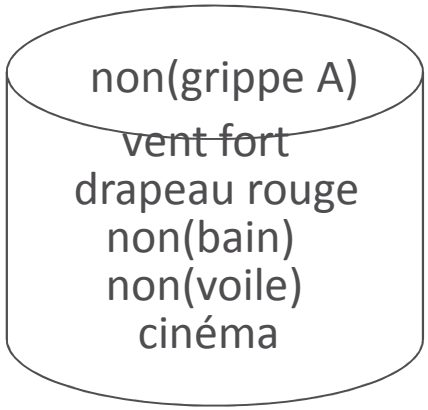
Moteur d'Inférences : raisonneur déductif automatique



Moteur d'Inférences : **Marche Avant**

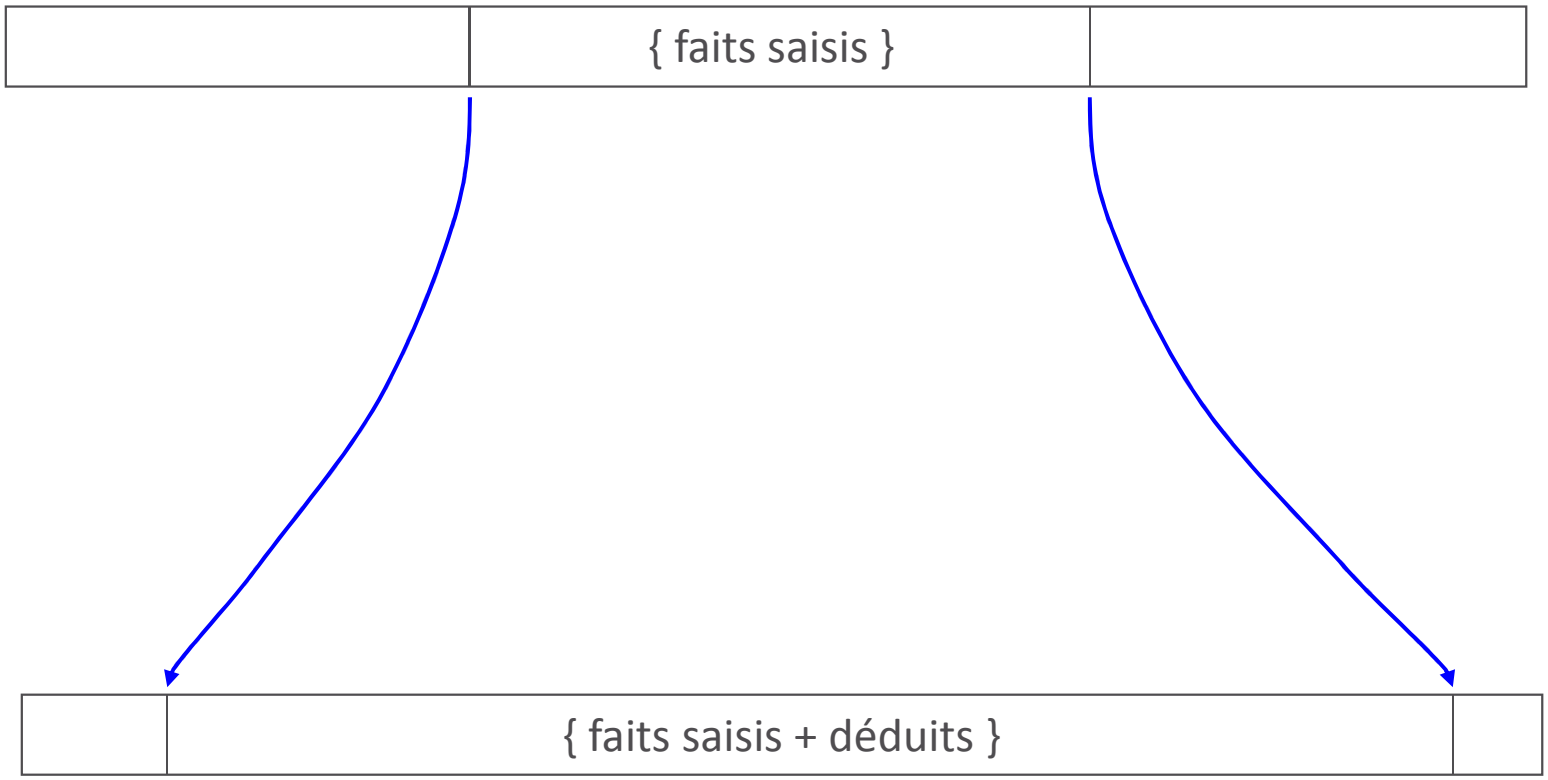


R1 : vent fort => drapeau rouge
R2 : drapeau rouge => non(bain) ∧ non(voile)
R3 : grippe A => non(bain)
R4 : pollution => non(bain)
R5 : grippe A => non(cinéma)
R6 : non(bain) ∧ non(grippe A) ∧ non(voile) => cinéma



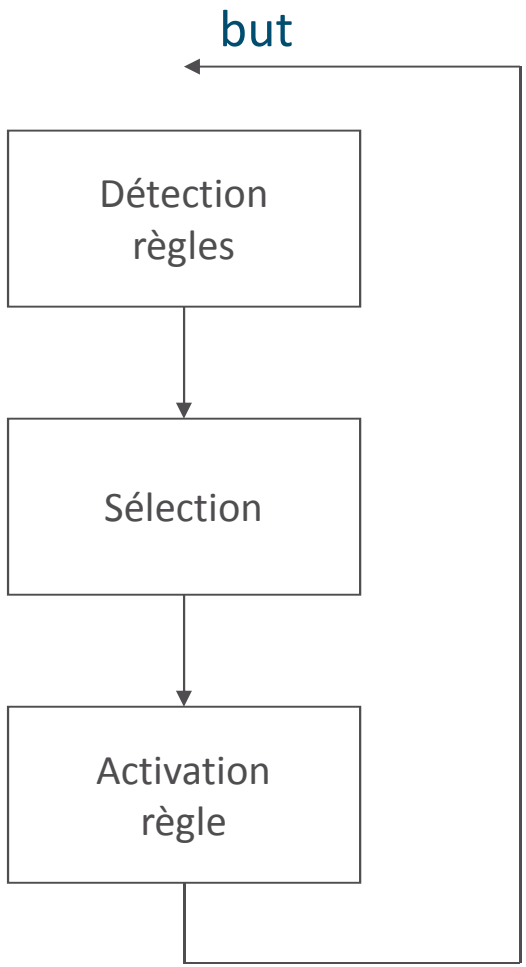
Question détection R3 ? Réponse : Oui
détection R6
Question détection grippe A ? Réponse : Non
sélection R6
sélection R1
exécution R6 => cinéma
détection R2
exécution R1 => drapeau rouge
sélection R2
exécution R2 => non(bain) ∧ non(voile)

Marche Avant : propagation jusqu'à 'saturation' des faits connus

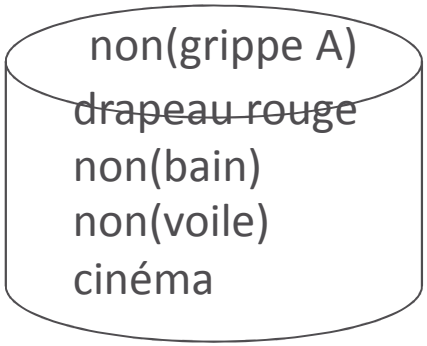


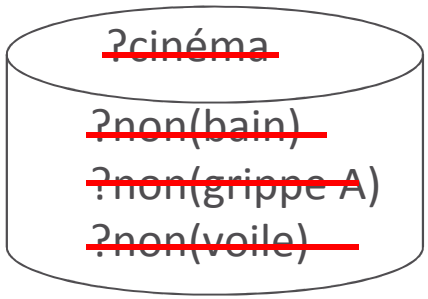
Moteur d'Inférences : Marche Arrière

De l'ensemble des règles qui concluent sur le but



R1 : vent fort => drapeau rouge
R2 : drapeau rouge => non(bain) $\wedge$ non(voile)
R3 : grippe A => non(bain)
R4 : pollution => non(bain)
R5 : grippe A => non(cinéma)
R6 : non(bain) $\wedge$ non(grippe A) $\wedge$ non(voile) => cinéma





sélection R2, R3, R4

Question : Est ce que pollution ?

Réponse : ?

sélection R3

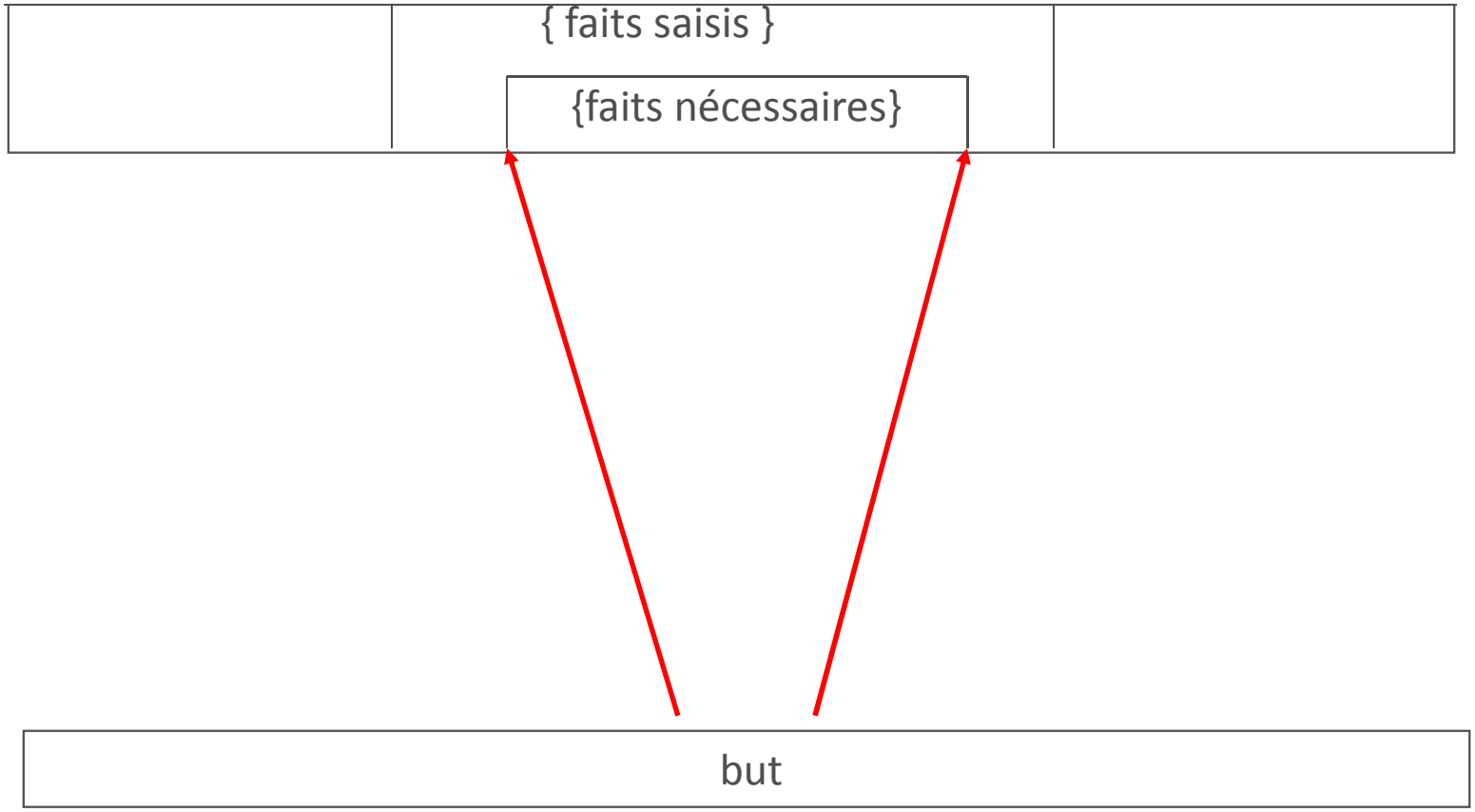
Pas de question à poser

sélection R2

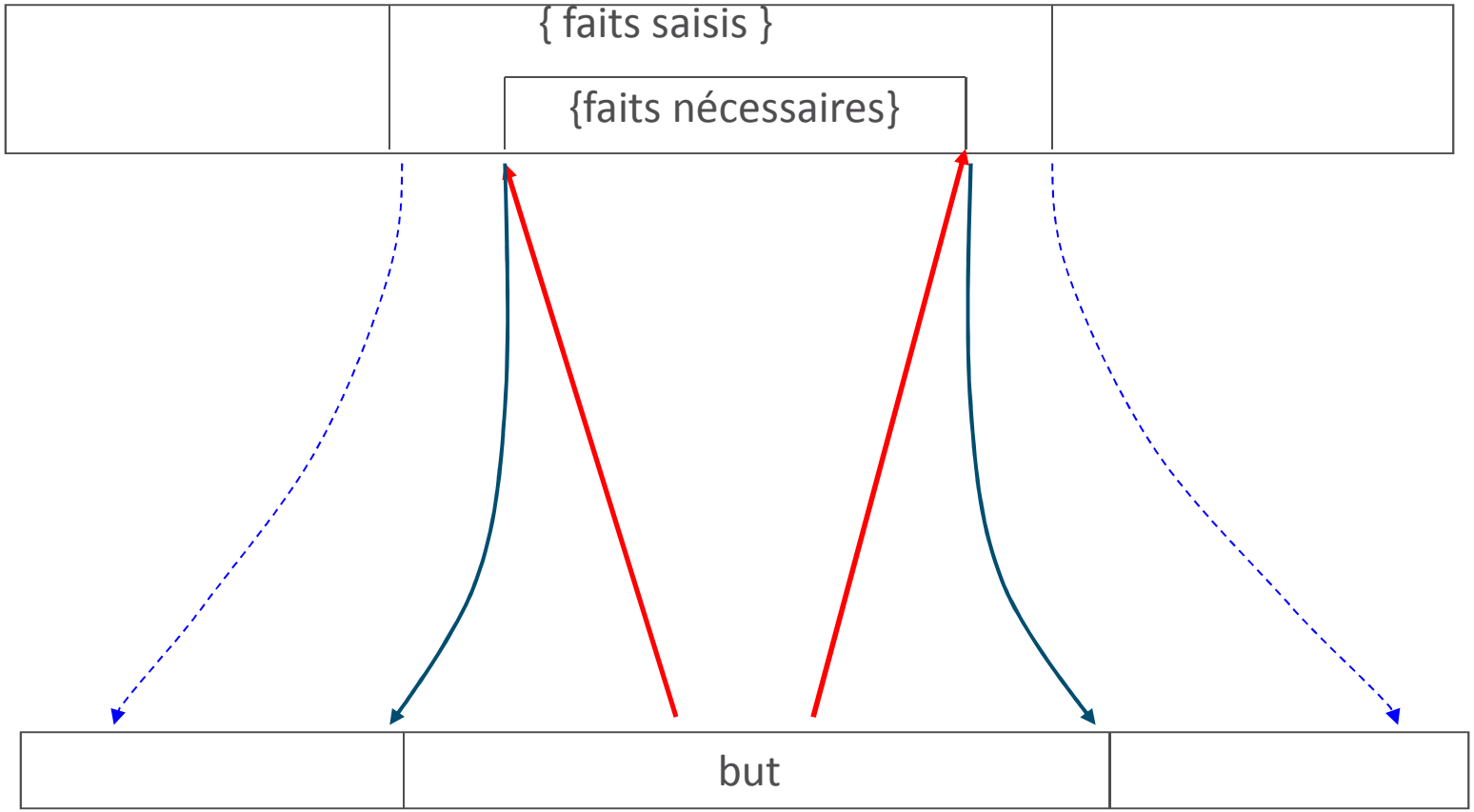
Question : Est-ce que drapeau rouge ?

Réponse : Oui

Marche Arrière : recherche des faits nécessaires à la preuve du but



Mode mixte : recherche puis propagation en avant des faits nécessaires à la preuve du but



Exemple

- Soit la base de connaissances :
- R1 : si B et D et E alors F
- R2 : si D et G alors A
- R3 : si C et F alors A
- R4 : si B alors X
- R5 : si D alors E
- R6 : si A et X alors H
- R7 : si C alors D
- R8 : si X et C alors A
- R9 : si X et B alors D
- Base de faits : B, C
- But : H

Evolution des systèmes à base de connaissances

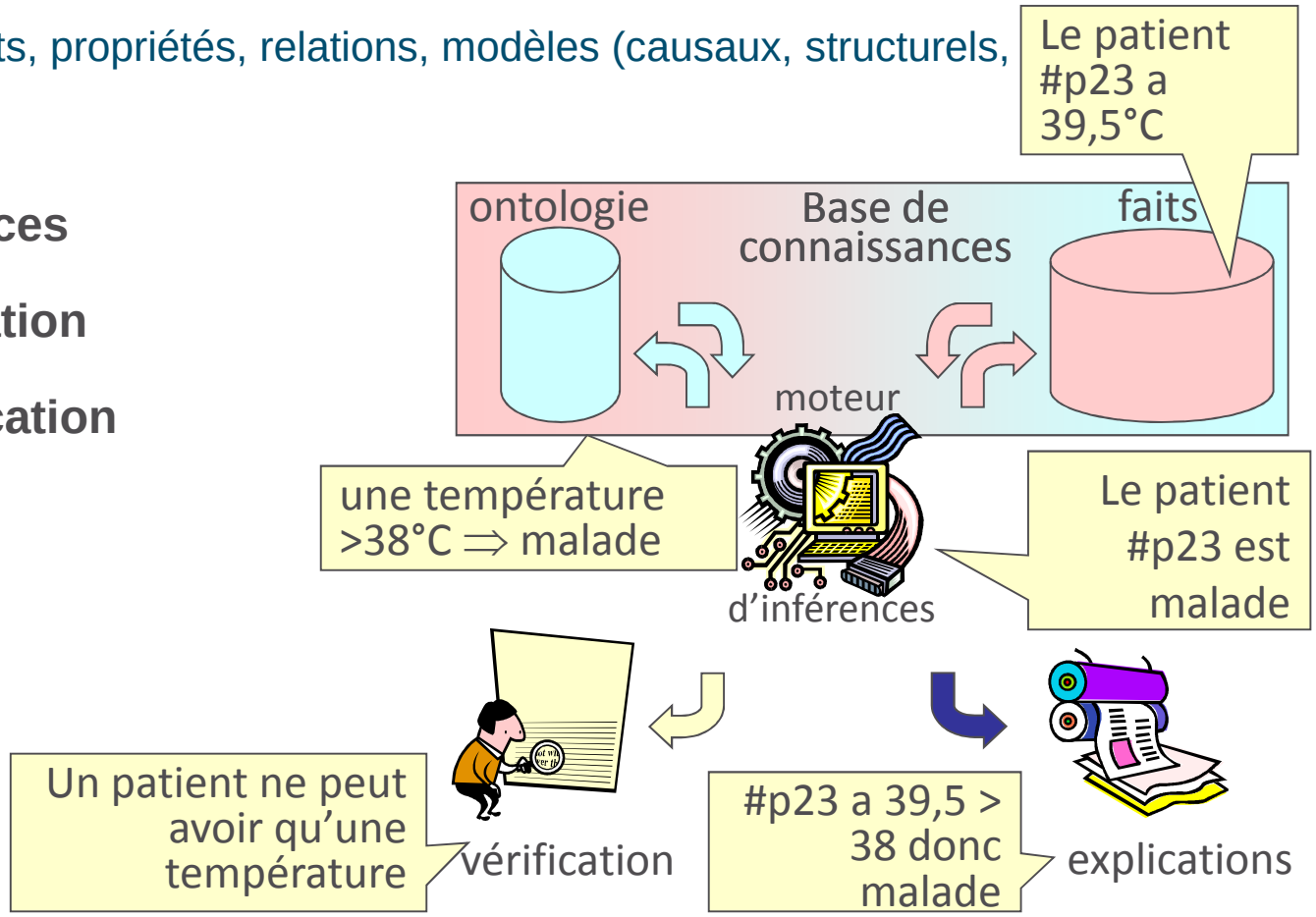
- **Base de connaissances**

  Ontologie : concepts, propriétés, relations, modèles (causaux, structurels,

  Base de faits

- **Moteur d'inférences**
- **Module d'explication**
- **Module de vérification**

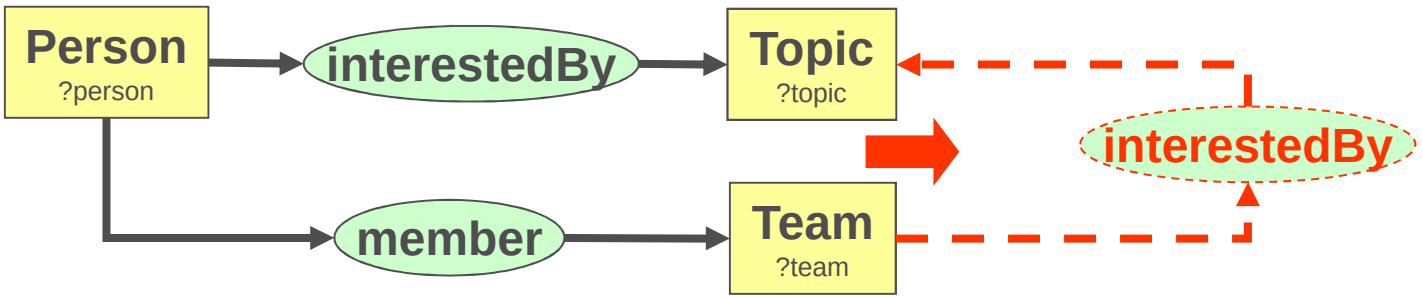
  de la cohérence



Pourquoi des règles pour le web sémantique : Inférer de nouvelles relations

Si un membre d'une équipe a un centre d'intérêt alors l'équipe a aussi ce centre d'intérêt

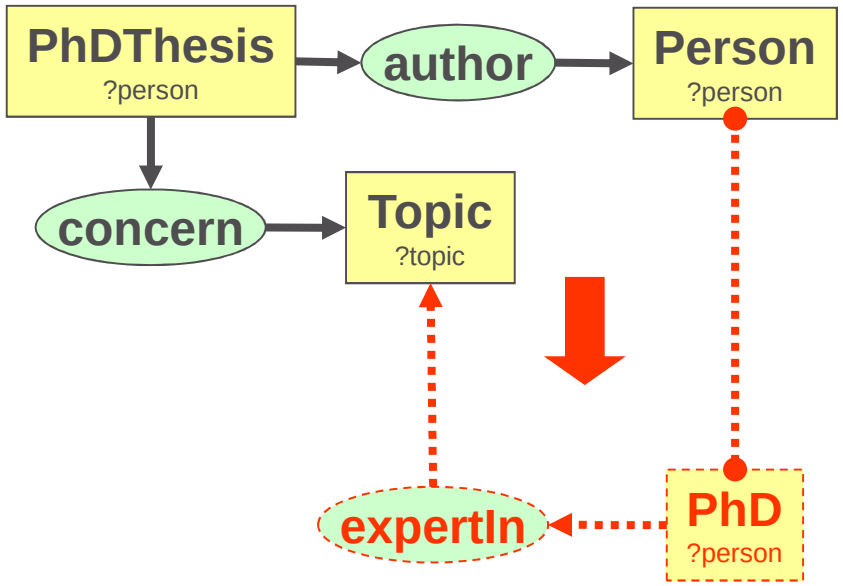
```
?person interestedBy ?topic
?person member ?team
→
?team interestedBy ?topic
```



Pourquoi des règles pour le web sémantique : classer des ressources

Si une personne a écrit une thèse sur un sujet alors c'est un docteur et un expert du sujet.

```
?person author ?doc
?doc rdf:type PhDThesis
?doc concern ?topic
→
?person expertIn ?topic
?person rdf:type PhD
```



Exemple de règle

- **Si deux personnes ont le même nom et le même email , ou le même nom et la même page web alors ces personnes sont identiques**

```
If  {  ?x  rdf:type  foaf:Person.
       ?y  rdf:type  foaf:Person.
       ?x  foaf:name  ?n.
       ?x  foaf:homepage  ?h.
       ?y  foaf:name  ?n.
       ?y  foaf:homepage  ?h.  }
then  {  ?x  =  ?y  }

If  {  ?x  rdf:type  foaf:Person.
       ?y  rdf:type  foaf:Person.
       ?x  foaf:name  ?n.
       ?x  foaf:mailbox  ?h.
       ?y  foaf:name  ?n.
       ?y  foaf:mailbox  ?m.  }
then  {  ?x  =  ?y  }
```

Langages de règles pour le web sémantique

- **RuleML :** Rule Markup Language (2001)
- **SWRL :** A Semantic Web Rule Language Combining OWL and RuleML (2004)
- **…**
- **RIF :** Rule Interchange Format (2010)

RuleML

- **RuleML : Rule Markup Language (2001)**
- **But principal : standard de règles métiers pour le partage et l'échange de règles**
- **Basé sur RDF et XML**
- **Une règle RuleML comporte deux parties**

  head : conséquent et body : condition
- **RuleML supporte le chaînage avant et le chaînage arrière**

RuleML : exemple

- **La réduction pour un client ayant acheté un produit est 5% si le client est premium et le produit ordinaire**

Règle de réduction d'un client suivant son statut

```
<Implies>
    <head>
      <Atom>
        <Rel>discount</Rel>
        <Var>customer</Var>
        <Var>product</Var>
        <Ind>5.0</Ind>
      </Atom>
    </head>
    <body>
     <And>
      <Atom>
        <Rel>premium</Rel>
        <Var>customer</Var>
      </Atom>
      <Atom>
        <Rel>regular</Rel>
        <Var>product</Var>
      </Atom>
     </And>
    </body>
</Implies>
```

SWRL

- **SWRL: Semantic Web Rule Language (2004)**
- **Combinaison de OWL et RuleML**
- **Extension de OWL : SWRL permet aux utilisateurs de rédiger des règles pour raisonner sur les individus OWL et inférer de nouvelles connaissances sur ces individus**
- **antécédent (body) → conséquent (head)**

Exemple 1 de règle SWRL

hasParent(?x1,?x2) ∧ hasBrother(?x2,?x3) ⇒ hasUncle(?x1,?x3)

```
<ruleml:imp>
 <ruleml:_rlab ruleml:href="#example1"/>
 <ruleml:_body>
   <swrlx:individualPropertyAtom  swrlx:property="hasParent">
     <ruleml:var>x1</ruleml:var>
     <ruleml:var>x2</ruleml:var>
   </swrlx:individualPropertyAtom>
   <swrlx:individualPropertyAtom  swrlx:property="hasBrother">
     <ruleml:var>x2</ruleml:var>
     <ruleml:var>x3</ruleml:var>
   </swrlx:individualPropertyAtom>
 </ruleml:_body>
 <ruleml:_head>
   <swrlx:individualPropertyAtom  swrlx:property="hasUncle">
     <ruleml:var>x1</ruleml:var>
     <ruleml:var>x3</ruleml:var>
   </swrlx:individualPropertyAtom>
 </ruleml:_head>
</ruleml:imp>
```

Exemple 2 de règle SWRL

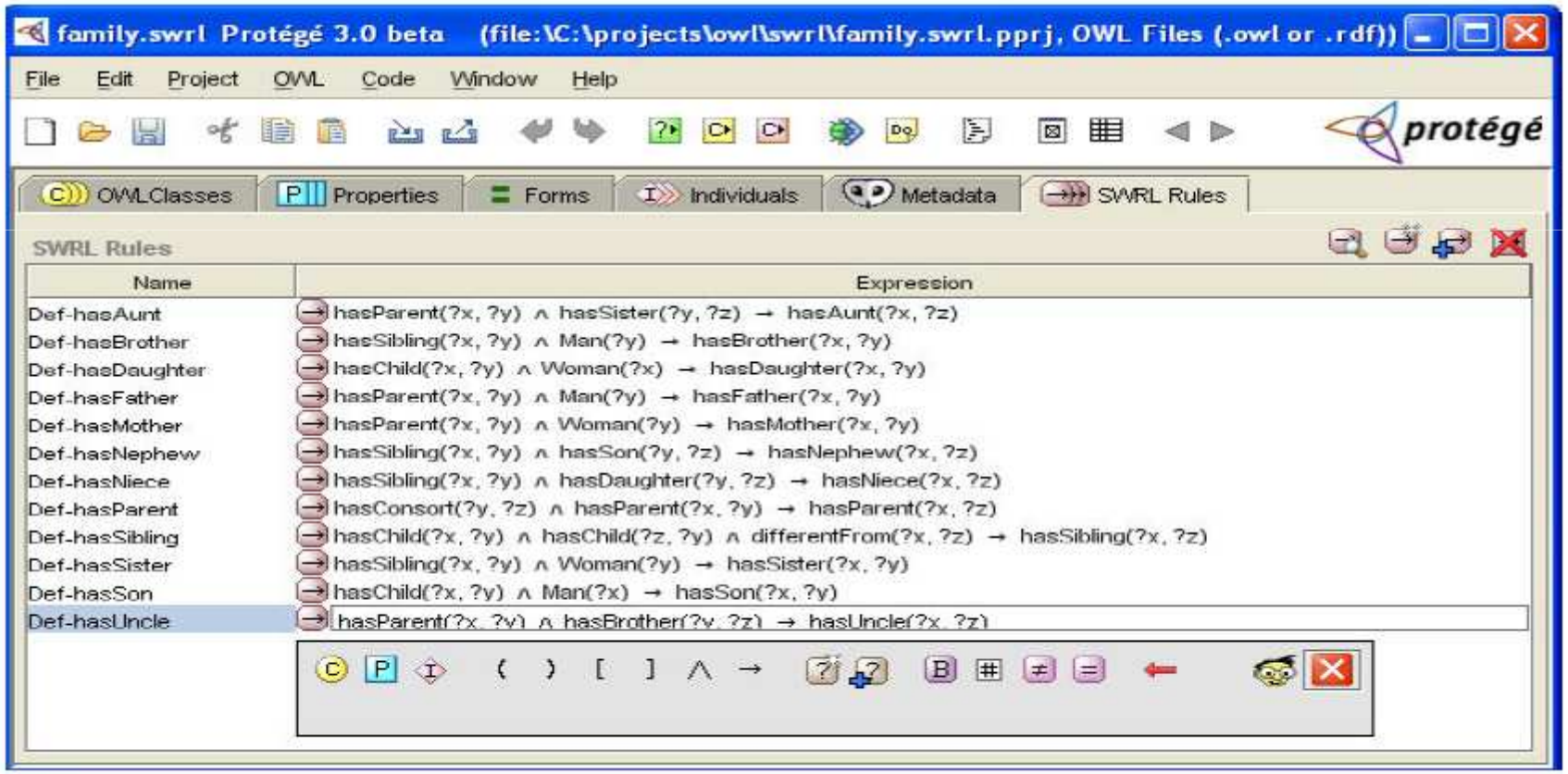
- **X3 est l'oncle de x1 si x2 est un parent de x1, x2 et x3 ont les mêmes parents, et x3 de sexe mâle**

Règle hasOncle

```
<ruleml :imp>
      <ruleml :_rlab ruleml :href="#example1" />
      <ruleml :_body>
        <swrlx :individualPropertyAtom swrlx :property="hasParent">
           <ruleml :var>x1</ruleml :var>
           <ruleml :var>x2</ruleml :var>
        </swrlx :individualPropertyAtom>
        <swrlx :individualPropertyAtom swrlx :property="hasSibling">
           <ruleml :var>x2</ruleml :var>
           <ruleml :var>x3</ruleml :var>
      </swrlx :individualPropertyAtom>
        <swrlx :individualPropertyAtom swrlx :property="hasSex">
           <ruleml :var>x3</ruleml :var>
           <owlx :Individual owlx :name="#male" />
        </swrlx :individualPropertyAtom>
      </ruleml :_body>
      <ruleml :_head>
        <swrlx :individualPropertyAtom swrlx :property="hasUncle">
           <ruleml :var>x1</ruleml :var>
           <ruleml :var>x3</ruleml :var>
        </swrlx :individualPropertyAtom>
      </ruleml :_head>
</ruleml :imp>
```

SWRLTab (Protégé)

- **SWRLTab est une extension de Protégé qui permet l'édition et l'exécution de règles SWRL**



RIF

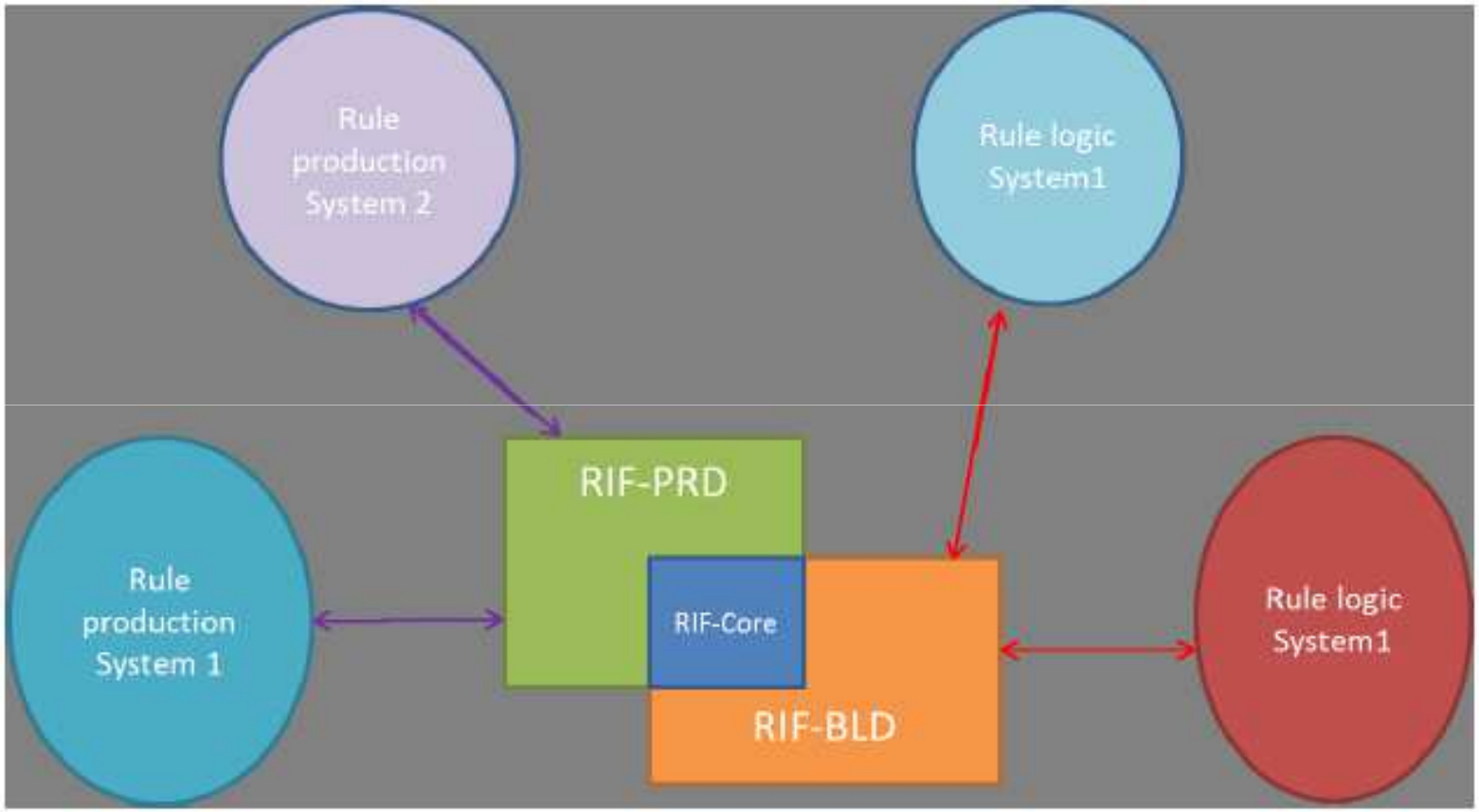
- **RIF : Rule Interchange Format (juin 2010)**
- **Langage de représentation et d'échange de règles sur le web (sémantique) entre différents systèmes de règles (logique ordre 1, PROLOG, Règles de production)**
- **Plusieurs dialectes :**

  RIF-BLD : Basic Logic Dialect

  RIF-PRD : Production Rules Dialect

  RIF-Core : noyau de primitives commun aux dialectes RIF

RIF


Rule production System 2
Rule logic System1
RIF-PRD
Rule production System 1
RIF-Core
RIF-BLD
Rule logic System1

RIF-BLD

- **RIF-BLD permet de représenter des programmes logiques i.e. des suites de règles d'inférence sur des faits positifs (sans négation). Il correspond à la logique de Horn.**
  - **RIF-BLD utilise deux syntaxes :**
    - Syntaxe abstraite : Logique (proche de PROLOG) :

      For All var* (conclusion :- condition)
    - Syntaxe concrète : XML
- **Le raisonnement repose sur la déduction de nouveau faits par l'instanciation de règles universelles, l'application de la règle de déduction et l'évaluation de formules conjonctives ou disjonctives**

RIF-BLD : instanciation

- **Règle universelle : « tout homme est mortel »**

```
Document(
        Prefix(bio <http://example.com/biology#>)
        Group(
                Forall ?x (bio:mortal(?x) :- bio:human(?x))
        )
)
```

- **Instanciation de la règle pour un célèbre philosophe**

```
Document(
        Prefix(bio <http://example.com/biology#>)
        Prefix(phil  <http://example.com/philosophers#>)
        Group(
                (bio:mortal(phil:Socrates) :- bio:human(phil:Socrates))
        )
)
```

RIF-BLD : déduction

- **Considérons un document comportant la règle précédente et le fait que « Socrates est un homme »**

```
Document(
    Prefix(bio <http://example.com/biology#>)
    Group(
        Forall ?x (bio:mortal(?x) :- bio:human(?x))
        bio:human(phil:Socrates)
    )
)
```

- **On en déduit (à l'aide de la règle de la déduction) que :**

```
Bio:mortal(phil:Socrates)
```

RIF-BLD : règles d’évaluation des formules conjonctives et disjonctives

- **Dans le cas où des faits ou les prémisses de règles ne sont plus des formules atomiques, des règles d’évaluation permettent d’évaluer leur valeur de vérité :**

  Une formule conjonctive est vraie si toutes les formules en conjonction sont vraies

  Une formule disjonctive est vraie si l’une au moins des formules en disjonction est vraie

RIF-BLD : exemple 1

?r est de type ?d si ?r est de type ?c et ?c st une sous-classe de ?d

Exemple (Sémantique de subClassOf)

```
Document( Base(<http ://example.com#>)
Group
(
    Forall ?r ?c ?d (
        ?r# ?d :- AND( ?r#c
                       ?c## ?d)
) )
```

RIF-BLD : exemple 2

If an item is perishable and it is delivered to John more than 10 days after the scheduled delevery date then the item will be rejected by him

If an item is unsolited by Fred then the item willbe rejected by him

Exemple

```
Document( Base(<http ://example.com/people#>) Prefix(cpt <http ://example.com/concepts#>) Prefix(dc
<http ://purl.org/dc/terms/>) Prefix(rif <http ://www.w3.org/2007/rif#>) Prefix(func
<http ://www.w3.org/2007/rif-builtin-function#>) Prefix(pred
<http ://www.w3.org/2007/rif-builtin-predicate#>) Prefix(xs <http ://www.w3.org/2001/XMLSchema#>)
(* "http ://sample.org" ^^rif :iri _pd[dc :publisher -> "http ://www.w3.org/" ^^rif :iri
dc :date -> "2008-04-04" ^^xs :date] *)
Group
(       Forall ?item ?deliverydate ?scheduledate ?diffduration ?diffdays (
        cpt :reject(<John> ?item) :-
            And(cpt :perishable( ?item)
                  cpt :delivered( ?item ?deliverydate <John>)
                  cpt :scheduled( ?item ?scheduledate)
                  ?diffduration = External(func :subtract-dateTimes( ?deliverydate ?scheduledate))
                  ?diffdays = External(func :days-from-duration( ?diffduration))
                  External(pred :numeric-greater-than( ?diffdays 10)))
        )

      Forall ?item (
        cpt :reject(<Fred> ?item) :- cpt :unsolicited( ?item)
        )
)
)
```

RIF-PRD

- **RIF-PRD permet de représenter des règles de production i.e. des règles dont l'application déclenche des actions d'ajout, de modification ou de suppression de faits dans la base**
- **Il existe deux syntaxes :**
  - Syntaxe abstraite : For All var* (if condition then do action)

    Actions possibles : assert, modify, retract
  - Syntaxe concrète XML

RIF-PRD : exemple

Si le client est un client dont le statut n'est pas New, Bronze, Silver, Gold, alors exécuter la fonction print et lui ajouter le statut New

Exemple tiré de RIF-PRD

```
Document( Prefix(ex <http ://example.com/2008/prd#>)
  Group (
   Forall ?customer such that ?customer # ex :Customer
    (If Not(Exists ?status
      (And( ?customer[ex :status-> ?status]
        External(pred :list-contains(List("New" "Bronze" "Silver" "Gold") ?status)) )))
    Then Do( Execute(act :print(External(func :concat("New customer :
" ?customer))))
      Assert( ?customer[ex :status->"New"])))))
```

RIF-Core

- RIF-Core est un sous-langage commun à RIF-BLD et RIF-PRD
- RIF-Core est une restriction syntaxique de RIF-BLD
- RIF-Core est une spécialisation de RIF-PRD

Compatibilité de RIF avec RDF(S), OWL et SPARQL

- Des règles RIF sont échangées avec des graphes RDF et des ontologies RDFS ou OWL (Systèmes à base de règles capables de prendre en compte des ontologies et d'opérer sur des bases de faits RDF)

- Des règles RIF-BLD peuvent être prises en compte dans un moteur de recherche SPARQL

RIF/SPARQL

- **Certaines primitives SPARQL peuvent être vues comme des règles :**

  Le traitement d'une requête SPARQL de la forme CONSTRUCT peut être considéré comme l'application en chaînage avant d'une règle dont la prémisse est la clause WHERE et la conclusion la clause CONSTRUCT de la requête

- **Certains éléments de RIF ont leur équivalent en SPARQL**

RIF/SPARQL

libellé	RIF-BLD	SPARQL
constant	*literal*	*literal*
variable	*'?'varname*	*'?'varname*
conjunction	*AND*	
disjunction	*or*	*UNION*
	:-	*where*
member	a#b	*a rdf :type b*
subclass	a##b	*a rdfs :subClassOf b*
frame	s[p → o]	*s p o*
	'Forall' Var+	*'CONSTRUCT'*
	Exists Var+	*ASK*
	Exists	*Exists*
Group	*Group*	*<rdf :RDF>*
opérateurs	=,<, >,* ...	=,<, >,* ...

RIF/SPARQL

Une personne x est un oncle de z, si x est un frère de y et y parent de z.

RIF-BLD	SPARQL
Forall ?x ?y ?z (*?x[uncleOf → ?z] :-* *AND(?x[brotherOf → ?y]* *?y[parentOf → ?z]))*	*CONSTRUCT { ?x uncleOf ?z}* *WHERE {* *?x brotherOf ?y* *?y parentOf ?z}*

Import de RDF et OWL dans un document RIF

Exemple

Document RDF : http ://example.org/mygraph
Prefix(ex <http ://example.org/example#>
ex :john ex :hasName "John"
Document RIF :
Document(Prefix(ex <http ://example.org/example#>)
Prefix(rdf <http ://www.w3.org/1999/02/22-rdf-syntax-ns#>)
import(<http ://example.org/mygraph><http ://www.w3.org/ns/entailment/RDF>)
Group (
Forall ?x ?y (?x[rdf :type -> ex :NamedEntity] :-
?x[ex :hasName -> ?y])
))

Conclusion

- **Possibilité de manipulation de RDF, RDFS et OWL par RIF**
- **Certains énoncés de RIF ont un équivalent en SPARQL**

- **RIF est actuellement très peu utilisé sur le web sémantique pour plusieurs raisons :**

  Sa relative jeunesse

  Sa grande complexité

  Peu de plate-formes acuellement disponibles pour son utilisation

Bibliographie

- F. Gandon, C. Faron-Zucker, O. Corby. Le Web sémantique. Comment lier les données et les schémas sur le Web ? Dunod eds. 2012.
- D. Kayser, La représentation des connaissances, Hermès (Paris), 1997.
- J.L. Laurière, Intelligence Artificielle (résolution de problèmes par l'homme et la machine), Eyrolles, Paris, 1987.
- M. Stefik, Introduction to Knowledge Systems, Morgan Kaufmann Publishers, San Francisco (CA), 1995.
- A. Thayse et al., Approche logique de l'Intelligence Artificielle (5 tomes), Dunod Informatique (Paris), 1988.