

# Web ngữ nghĩa

Soạn bởi: Nguyễn Bá Ngọc

## Chương 7

Hà Nội-2021

Chương 7.

Luật

# Nội dung


7.1. Khái niệm luật

7.2. Datalog như ngôn ngữ luật bậc nhất

7.3. Kết hợp luật với OWL DL

7.4. Định dạng trao đổi luật RIF

# Nội dung



7.1. Khái niệm luật

7.2. Datalog như ngôn ngữ luật bậc nhất

7.3. Kết hợp luật với OWL DL

7.4. Định dạng trao đổi luật RIF

# Demo suy diễn với luật trong Apache Jena

@prefix owl: <http://www.w3.org/2002/07/owl#> .

@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .

@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .

@prefix : <http://semweb.edu.vn/rule-demo#> .

:a :p :b.

:b :p :c.

:b :p :d.

@prefix : <http://semweb.edu.vn/rule-demo#> .

[rule1: (?a :p ?b) (?b :p ?c) -> (?a :p ?c)]

# Tùy chỉnh Fuseki

```
ja:reasoner [  
    ja:rulesFrom <file:demo.rules> ;  
].
```

# Mở rộng tập luật

# Example rule file

@prefix pre: <http://jena.hpl.hp.com/prefix#>.

@include <RDFS>.

[rule1: (?f pre:father ?a) (?u pre:brother ?f) -> (?u pre:uncle  
?a)]

# Thử nghiệm suy diễn

Cho dữ kiện:

:john :fatherOf :mary .

:laura :motherOf :mary .

:peter :brotherOf :john .

:david :brotherOf :laura .

Yêu cầu: Tìm chú, bác của Mary (:mary)

a) Sử dụng owl;

b) Sử dụng luật.



# Thử nghiệm suy diễn<sub>(2)</sub>

# 1

```
:uncleOf owl:propertyChainAxiom (:brotherOf :motherOf);  
      owl:propertyChainAxiom (:brotherOf :fatherOf).
```

# 2

```
:MaryFather owl:equivalentClass [a owl:Restriction; owl:onProperty  
:fatherOf; owl:hasValue :mary].  
:MaryMother owl:equivalentClass [a owl:Restriction; owl:onProperty  
:motherOf; owl:hasValue :mary].  
:ParentOfMary owl:equivalentClass [  
  owl:unionOf (:MaryFather :MaryMother)  
].  
:UncleOfMary owl:equivalentClass [  
  a owl:Restriction;  
  owl:onProperty :brotherOf;  
  owl:someValuesFrom :ParentOfMary  
].
```

# Thử nghiệm suy diễn<sub>(3)</sub>

# Lập trình lô-gic

<https://swish.swi-prolog.org/>

# Luật là gì?

- Luật lô-gic (1 phần của lô-gic bậc nhất):
  - $F \rightarrow G$  tương đương với  $\neg F \vee G$
  - Mở rộng lô-gic của cơ sở tri thức  $\rightsquigarrow$  tĩnh/ngoại tuyến
  - Thế giới mở
  - Ngôn ngữ khai báo (declarative/describing)
- Luật sinh (production rules):
  - if X then Y else Z
  - Các lệnh thực thi  $\rightsquigarrow$  động/trực tuyến
  - Thao tác (ý nghĩa = hiệu ứng khi thực hiện)
- Lập trình lô-gic (ví dụ, PROLOG, F-Logic):
  - `man(X) <- person(X) AND NOT woman(X)`
  - Tiệm cận ngữ nghĩa lô-gic với các khía cạnh tính toán
  - Thường là thế giới đóng
  - Bán khai báo

# Luật trong Lô-gic bậc nhất

- Luật như công thức suy diễn

$$\underbrace{H \leftarrow}_{\text{đầu}} \underbrace{A_1 \wedge A_2 \wedge \dots \wedge A_n}_{\text{thân}}$$

- $\rightsquigarrow$  có giá trị chân lý tương đương với

$$H \vee \neg A_1 \vee \neg A_2 \vee \dots \vee \neg A_n$$

- Suy diễn thường được viết theo chiều từ phải qua trái
- Được phép sử dụng các hằng, biến và hàm ký tự
- Các định lượng cho biến thường được bỏ qua - Các biến tự do thường được mặc định là các đại lượng toàn thể (luật hợp lệ với tất cả các giá trị của biến)

## Ví dụ 7.1. Luật

$\text{hasUncle}(x,z) \leftarrow \text{hasParent}(x, y) \wedge \text{hasBrother}(y,z)$

- Chúng ta sử dụng các tên ngắn gọn (hasUncle) thay cho các IRIs như: <http://example.org/Example#hasUncle>
- và các ký hiệu  $x, y, z$  cho các biến

# Biến đổi Lloyd-Topor

- Liên kết mặc định cho các phần tử liên tiếp trong phần đầu luật là hội (và):

$$H_1, H_2, \dots, H_m \leftarrow A_1, A_2, \dots, A_n$$

Tương đương với

$$\left\{ \begin{array}{l} H_1 \leftarrow A_1, A_2, \dots, A_n \\ H_2 \leftarrow A_1, A_2, \dots, A_n \\ \dots \\ H_m \leftarrow A_1, A_2, \dots, A_n \end{array} \right.$$

- Những biến đổi như vậy được gọi là biến đổi Lloyd-Topor

# Các luật tuyển

- Một số hệ luật sử dụng phép tuyển

→ Một số phần tử trong phần đầu được coi là các lựa chọn tương đương:

$$H_1, H_2, \dots, H_m \leftarrow A_1, A_2, \dots, A_n$$

tương đương với

$$H_1 \vee H_2 \vee \dots \vee H_m \leftarrow A_1 \wedge A_2 \wedge \dots \wedge A_n$$

tương đương với

$$H_1 \vee H_2 \vee \dots \vee H_m \vee \neg A_1 \vee \neg A_2 \vee \dots \vee \neg A_n$$

→ (không đi vào chi tiết trong bài giảng này)

# Phân loại mệnh đề

Trong lô-gic bậc nhất:

- Mệnh đề: Gồm tuyển và phủ định của các mệnh đề đơn vị
  - $\neg \text{Person}(x) \vee \text{Woman}(x) \vee \text{Man}(x)$
- Mệnh đề Horn: Mệnh đề với tối đa 1 phần tử khẳng định (non-negated)
  - $\leftarrow \text{Man}(x) \wedge \text{Woman}(x)$
  - $\neg \text{Woman}(x) \vee \neg \text{Man}(x)$
- Mệnh đề xác định: Mệnh đề Horn với đúng 1 phần tử khẳng định
  - $\text{Father}(x) \leftarrow \text{Man}(x) \wedge \text{hasChild}(x, y)$
  - $\text{Father}(x) \vee \neg \text{Man}(x) \vee \neg \text{hasChild}(x, y)$
- Dữ kiện: Mệnh đề chứa đúng 1 phần tử khẳng định
  - $\text{Woman}(\text{gisela})$



# Hàm ký tự

Luật có thể chứa các hàm ký tự:

$\text{hasUncle}(x, y) \leftarrow \text{hasBrother}(\text{mother}(x), y)$

$\text{hasFather}(x, \text{father}(x)) \leftarrow \text{Person}(x)$

- ↪ Các phần tử mới được sinh
- ↪ không đi vào chi tiết trong bài giảng này
- ↪ Tham khảo thêm lập trình lô-gic

# Nội dung

7.1. Khái niệm luật

7.2. Datalog như ngôn ngữ luật bậc nhất

7.3. Kết hợp luật với OWL DL

7.4. Định dạng trao đổi luật RIF

# Tổng quan về Datalog

Các luật Horn không chứa hàm ký tự  $\rightsquigarrow$  Các luật Datalog

- Ngôn ngữ luật lô-gic, ban đầu được sử dụng cho các CSDL suy diễn
- Cơ sở tri thức bao gồm các mệnh đề
- Khả quyết
- Hiệu quả đối với các tập dữ liệu lớn, độ phức tạp kết hợp là hàm mũ
- Có nhiều nghiên cứu trong những năm 198x.

# Ngữ nghĩa của Datalog

Ba cách tương đương để biểu diễn ngữ nghĩa

- Mô hình lý thuyết
- Chứng minh lý thuyết
- Các điểm cố định

# Ngữ nghĩa mô hình lý thuyết của Datalog

Mỗi luật Datalog:

$$\rho: R_1(u_1) \leftarrow R_2(u_2), \dots, R_n(u_n)$$

Tương đương với 1 câu lô-gic bậc nhất

$$\forall x_1, \dots, x_n. (R_1(u_1) \leftarrow R_2(u_2) \wedge \dots \wedge R_n(u_n))$$

- $x_1, \dots, x_n$  là các biến luật và  $\leftarrow$  là suy diễn lô-gic
- Một trường hợp  $I$  thỏa mãn  $\rho$ , được viết là  $I \models \rho$ , khi và chỉ khi đối với tất cả các bộ giá trị

$$R_1(v(u_1)) \leftarrow R_2(v(u_2)), \dots, R_n(v(u_n))$$

$R_1(v(u_1))$  thỏa mãn khi  $R_2(v(u_2)), \dots, R_n(v(u_n))$  được đáp ứng.

# Ngữ nghĩa mô hình-lý thuyết của Datalog<sub>(2)</sub>

- Một trường hợp I là mô hình của 1 chương trình Datalog P, nếu I thỏa mãn tất cả các luật trong P (được xem như các công thức FOL)
- Câu hỏi: Có luôn tồn tại mô hình như vậy?
- Nếu có, làm sao để xây dựng nó?

# Ngữ nghĩa lý thuyết-chứng minh

Dựa trên các chứng minh của các dữ kiện:

Cho :

$$E(a, b), E(b, c), E(c, d)$$

$$T(x, y) \leftarrow E(x, y) \quad (1)$$

$$T(x, y) \leftarrow E(x, z) \wedge T(z, y) \quad (2)$$

(a)  $E(c, d)$  là 1 dữ kiện được cho

(b)  $T(c, d)$  được suy ra từ (1) và (a)

(c)  $E(b, c)$  là một dữ kiện được cho

(d)  $T(b, d)$  được suy ra từ (c), (b) và (2)

(e) ...

# Ngữ nghĩa lý thuyết-chứng minh<sub>(2)</sub>

- Chương trình có thể được coi như nhà máy cung cấp tất cả các dữ kiện có thể kiểm chứng (suy diễn dữ kiện mới từ các dữ kiện đã biết theo hướng từ đáy lên bằng cách áp dụng luật)
- Tương đương: Từ trên xuống, bắt đầu với dữ kiện cần được chứng minh, tìm kiếm các bổ đề cần thiết cho chứng minh ( $\leadsto$  phân giải)



# Ngữ nghĩa lý thuyết-chứng minh<sub>(3)</sub>

*Một dữ kiện được coi là có thể kiểm chứng nếu có cây chứng minh cho nó*

## **Định nghĩa:**

Cây chứng minh cho 1 dữ kiện A trong 1 trường hợp I và 1 chương trình Datalog P là 1 cây có nhãn trong đó:

1. Mỗi nút đều được gán nhãn bằng 1 dữ kiện
2. Mỗi nút lá đều được gán nhãn bằng 1 dữ kiện từ I
3. Gốc được gán nhãn với A
4. Với mỗi nút nội tồn tại 1 thể hiện  $A_1 \leftarrow A_2, \dots, A_n$  của 1 luật trong P, sao cho nút đó được gán nhãn  $A_1$  và các con của nó là  $A_2, \dots, A_n$

# Ngữ nghĩa lý thuyết-chứng minh<sub>(4)</sub>

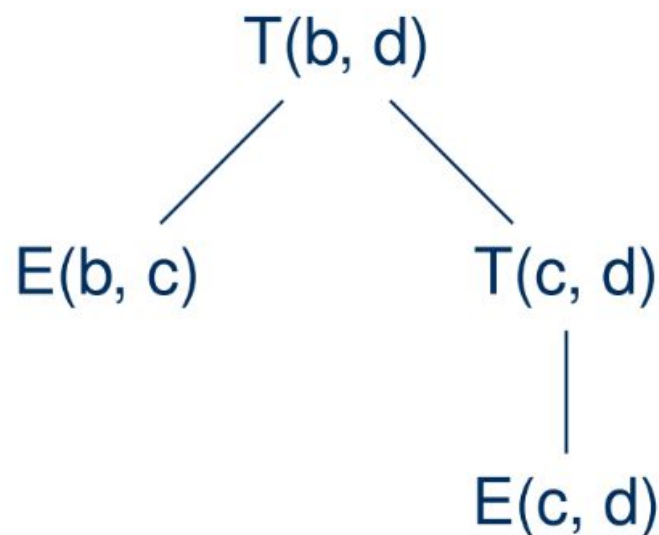
Dựa trên các chứng minh của các dữ kiện:

Cho :  $E(a, b), E(b, c), E(c, d)$

$T(x, y) \leftarrow E(x, y)$  (1)

$T(x, y) \leftarrow E(x, z) \wedge T(z, y)$  (2)

- (a)  $E(c, d)$  là 1 dữ kiện được cho
- (b)  $T(c, d)$  được suy ra từ (1) và (a)
- (c)  $E(b, c)$  là 1 dữ kiện được cho
- (d)  $T(b, d)$  được suy ra từ (c), (b) và (2)
- (e) ...



# Ngữ nghĩa dựa trên các điểm cố định

Thiết lập ngữ nghĩa của 1 chương trình Datalog như lời giải của 1 phương trình điểm cố định

- Định nghĩa tiến trình (lặp cho tới khi chạm điểm cố định)
- Cho 1 trường hợp  $I$  và 1 chương trình Datalog  $P$ , chúng ta gọi 1 dữ kiện  $A$  là hệ quả trực tiếp cho  $P$  và  $I$ , nếu:
  - (1)  $A$  nằm trong  $I$  hoặc
  - (2)  $A \leftarrow A_1, \dots, A_n$  là thể hiện của 1 luật từ  $P$ , sao cho  $A_1, \dots, A_n \in I$

# Ngữ nghĩa dựa trên các điểm cố định<sub>(2)</sub>

- Có thể tính toán tất cả các hệ quả trực tiếp bắt đầu từ 1 trường hợp.
- Tương tự ngữ nghĩa lý thuyết-chứng minh nhưng các chứng minh ngắn luôn được phát hiện trước các chứng minh dài.

# Các khẳng định ngoại và khẳng định nội

- Từ góc nhìn CSDL (khác với lập trình lô-gic) có thể phân biệt dữ kiện và luật
- Trong các luật, chúng ta phân biệt các thuộc tính ngoại và thuộc tính nội
- Các thuộc tính ngoại (còn được gọi là CSDL ngoại - edb) là các thuộc tính không xuất hiện trong đầu luật
  - $T(x, y) \leftarrow E(x, z) \wedge T(z, y)$  - Quan hệ E
- Các thuộc tính nội (còn được gọi là CSDL nội - idb) là các thuộc tính có xuất hiện trong đầu luật
  - $T(x, y) \leftarrow E(x, z) \wedge T(z, y)$  - Quan hệ T
- Ngữ nghĩa của chương trình Datalog có thể được hiểu như ánh xạ của các thể hiện của các thuộc tính edb tới các thể hiện của các thuộc tính idb.

# Datalog trong thực tiễn

Datalog trong thực tiễn:

- Có nhiều triển khai
- Được tùy chỉnh để tương thích với Web ngữ nghĩa: Các kiểu XSD, URIs (ví dụ  $\rightarrow$  IRIs)

Các mở rộng của Datalog:

- Cho phép sử dụng tuyển trong đầu luật
- Các phủ định không đơn điệu (không có ngữ nghĩa FOL)/non-monotonic negation

# Thực thi chương trình Datalog

- Thực thi từ dưới lên hoặc từ trên xuống
- Thực thi trực tiếp vs. biên dịch thành chương trình hiệu năng cao
- Trong bài giảng này chúng ta sẽ tìm hiểu:
  - Thực thi đơn giản từ dưới lên
  - Thực thi bán đơn giản từ dưới lên

# Sinh đảo ngược

Sinh đảo ngược: Reverse-Same-Generation (rsg)

Cho chương trình Datalog:

$\text{rsg}(x, y) \leftarrow \text{flat}(x, y)$

$\text{rsg}(x, y) \leftarrow \text{up}(x, x_1), \text{rsg}(y_1, x_1), \text{down}(y_1, y)$

Cho dữ liệu:

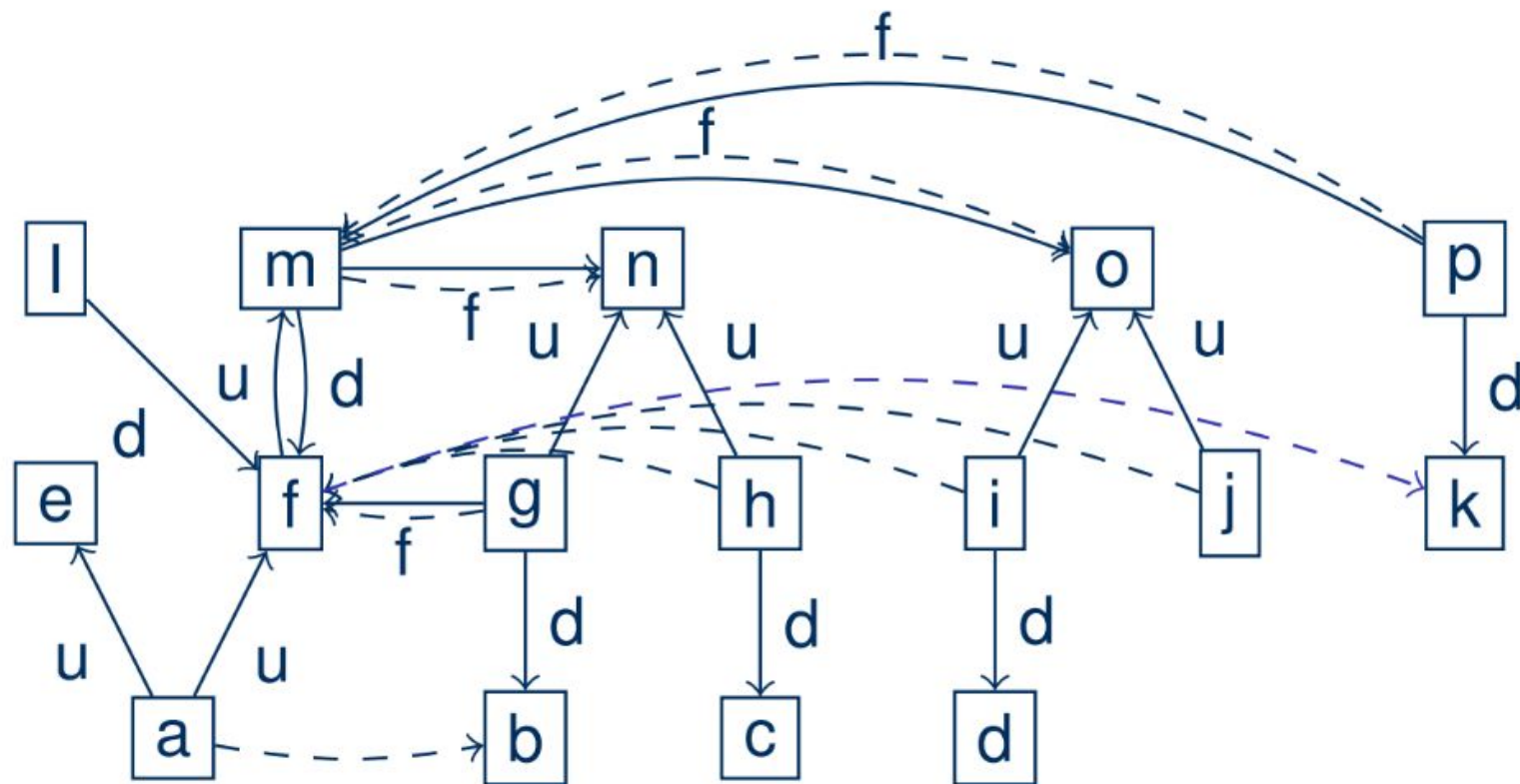
up		
	a	e
	a	f
	f	m
	g	n
	h	n
	i	o
	j	o

flat		
	g	f
	m	n
	m	o
	p	m

down		
	l	f
	m	f
	g	b
	h	c
	i	d
	p	k



# Trực quan hóa sơ đồ sinh đảo ngược



$$rsg(x, y) \leftarrow flat(x, y)$$

$$rsg(x, y) \leftarrow up(x, x_1), rsg(y_1, x_1), down(y_1, y)$$

# Giải thuật đơn giản tính rsg

$$rsg(x, y) \leftarrow flat(x, y)$$

$$rsg(x, y) \leftarrow up(x, x_1), rsg(y_1, x_1), down(y_1, y)$$

---

## Algorithm 1 RSG

---

$rsg := \emptyset$

**repeat**

$$rsg := rsg \cup flat \cup \pi_{16}(\sigma_{2=4}(\sigma_{3=5}(up \times rsg \times down)))$$

**until** fixpoint reached

---

$$rsg^{i+1} := rsg^i \cup flat \cup \pi_{16}(\sigma_{2=4}(\sigma_{3=5}(up \times rsg \times down)))$$

Level 0:  $\emptyset$

Level 1:  $\{(g, f), (m, n), (m, o), (p, m)\}$

Level 2:  $\{\text{Level 1}\} \cup \{(a, b), (h, f), (i, f), (j, f), (f, k)\}$

Level 3:  $\{\text{Level 2}\} \cup \{(a, c), (a, d)\}$

Level 4:  $\{\text{Level 3}\}$

# Giải thuật thực thi đơn giản: Các đặc điểm

- Dư thừa tính toán (tất cả các phần tử thuộc mức trước được sử dụng trong tính toán)
- Ở mỗi mức tất cả phần tử thuộc lớp trước được tính lại
- Đơn điệu/tăng dần (rsg được mở rộng và mở rộng lại nhiều lần)

# Giải thuật bán đơn giản tính rsg

Tập trung vào các dữ kiện mới được tính từ bước trước

Giải thuật 2 RSG'

---

$$\begin{aligned}\Delta_{rsg}^1(x, y) &:= flat(x, y) \\ \Delta_{rsg}^{i+1}(x, y) &:= up(x, x_1), \Delta_{rsg}^i(y_1, x_1), down(y_1, y)\end{aligned}$$

---

- Không đệ quy
- Tập luật là vô hạn
- với mỗi đầu vào  $I$  và  $\Delta_{rsg}^i$  (các mẫu tính được ở bước  $i$ )
$$rsg^{i+1} - rsg^i \subseteq \Delta_{rsg}^{i+1} \subseteq rsg^{i+1}$$
- $RSG(I)(rsg) = \bigcup_{1 \leq i} (\Delta_{rsg}^i)$
- Ít dư thừa hơn

# Một cải tiến

Quan sát:  $\Delta_{rsg}^{i+1} \neq rsg^{i+1} - rsg^i$

Ví dụ:  $(g, f) \in \Delta_{rsg}^2, (g, f) \notin rsg^2 - rsg^1$

$\leadsto rsg(g, f) \in rsg^1$ , bởi vì  $flat(g, f)$ ,

$\leadsto rsg(g, f) \in \Delta_{rsg}^2$ , bởi vì  $up(g, n), rsg(m, f), down(m, f)$

- Ý tưởng: sử dụng  $rsg^i - rsg^{i-1}$  thay cho  $\Delta_{rsg}^i$  trong luật thứ 2 của RSG'.

## Giải thuật 3. RSG''

---

$$\Delta_{rsg}^1(x, y) := flat(x, y)$$

$$rsg^1 := \Delta_{rsg}^1$$

$$tmp_{rsg}^{i+1}(x, y) := up(x, x_1), \Delta_{rsg}^i(y_1, x_1), down(y_1, y)$$

$$\Delta_{rsg}^{i+1}(x, y) := tmp_{rsg}^{i+1} - rsg^i$$

$$rsg^{i+1} := rsg^i \cup \Delta_{rsg}^{i+1}$$

---

# Luật RDFS với thuộc tính Triple và các thuộc tính cụ thể

# Thuộc tính Triple

$$\frac{a \text{ rdfs:domain } x \ . \ u \ a \ y \ .}{u \text{ rdf:type } x \ .} \text{ rdfs2}$$

$$\text{Triple}(u, \text{rdf : type}, x) \leftarrow \text{Triple}(a, \text{rdfs : domain}, x) \wedge \text{Triple}(u, a, y)$$

*Chỉ sử dụng 1 thuộc tính làm giảm tiềm năng tối ưu hóa*

# Các thuộc tính cụ thể

$$\frac{a \text{ rdfs:domain } x \ . \ u \ a \ y \ .}{u \text{ rdf:type } x \ .} \text{ rdfs2}$$
$$\text{type}(u, x) \leftarrow \text{domain}(a, x) \wedge \text{rel}(u, a, y)$$



# Các bộ ba tiên đề như những dữ kiện

`type(rdf:type, rdf:Property)`

`type(rdf:subject, rdf:Property)`

`type(rdf:predicate, rdf:Property)`

`type(rdf:object, rdf:Property)`

`type(rdf:first, rdf:Property)`

`type(rdf:rest, rdf:Property)`

`type(rdf:value, rdf:Property)`

`type(rdf: 1, rdf:Property)`

`type(rdf: 2, rdf:Property)`

`type(. . . , rdf:Property)`

`type(rdf:nil, rdf:List)`

. . . (Thêm các bộ ba mệnh đề RDFS)

→ Chỉ cần những `rdf:_i` xuất hiện trong các đồ thị  $G_1$  và  $G_2$ , nếu cần quyết định  $G_1 \models G_2$

# Luật suy diễn RDFS

$$\frac{u \ a \ y}{a \ \text{rdf:type} \ \text{rdf:Property}} \text{rdf1}$$
$$\rightsquigarrow \text{type}(a, \text{rdf:Property}) \leftarrow \text{rel}(u, a, y)$$

$$\frac{a \ \text{rdfs:domain} \ x \ . \ u \ a \ y \ .}{u \ \text{rdf:type} \ x \ .} \text{rdfs2}$$
$$\rightsquigarrow \text{type}(u, x) \leftarrow \text{domain}(a, x) \wedge \text{rel}(u, a, y)$$

$$\frac{a \ \text{rdfs:range} \ x \ . \ u \ a \ v \ .}{v \ \text{rdf:type} \ x \ .} \text{rdfs3}$$
$$\rightsquigarrow \text{type}(v, x) \leftarrow \text{range}(a, x) \wedge \text{rel}(u, a, v)$$

$$\frac{u \ a \ x \ .}{u \ \text{rdf:type} \ \text{rdfs:Resource} \ .} \text{rdfs4a}$$
$$\rightsquigarrow \text{type}(u, \text{rdfs:Resource}) \leftarrow \text{rel}(u, a, x)$$

**a, b** là các IRIs  
**u, v** - IRI hoặc nút rỗng  
**x, y** - IRI, nút rỗng  
hoặc hằng  
**I** - hằng giá trị  
**\_n**: nút rỗng

# Luật suy diễn RDFS (2)

$$\frac{u \ a \ v \ .}{v \ \text{rdf:type} \ \text{rdfs:Resource} \ .} \text{ rdfs4b}$$
$$\rightsquigarrow \text{type}(v, \text{rdfs:Resource}) \leftarrow \text{rel}(u, a, v)$$
$$\frac{u \ \text{rdfs:subPropertyOf} \ v \ . \quad v \ \text{rdfs:subPropertyOf} \ x \ .}{u \ \text{rdfs:subPropertyOf} \ x \ .} \text{ rdfs5}$$
$$\rightsquigarrow \text{subPropertyOf}(u, x) \leftarrow \text{subPropertyOf}(u, v) \wedge \text{subPropertyOf}(v, x)$$
$$\frac{u \ \text{rdf:type} \ \text{rdf:Property} \ .}{u \ \text{rdfs:subPropertyOf} \ u \ .} \text{ rdfs6}$$
$$\rightsquigarrow \text{subPropertyOf}(u, u) \leftarrow \text{type}(u, \text{rdf:Property})$$
$$\frac{a \ \text{rdfs:subPropertyOf} \ b \ . \quad u \ a \ y \ .}{u \ b \ y \ .} \text{ rdfs7}$$
$$\rightsquigarrow \text{rel}(u, b, y) \leftarrow \text{subPropertyOf}(a, b) \wedge \text{rel}(u, a, y)$$

**a, b** là các IRIs  
**u, v** - IRI hoặc nút rỗng  
**x, y** - IRI, nút rỗng  
hoặc hằng  
**l** - hằng giá trị  
**\_n**: nút rỗng

# Luật suy diễn RDFS<sub>(3)</sub>

$$\frac{u \text{ rdf:type rdfs:Class .}}{u \text{ rdf:subClassOf rdfs:Resource .}} \text{ rdfs8}$$

$\rightsquigarrow \text{subClassOf}(u, \text{rdfs:Resource}) \leftarrow \text{type}(u, \text{rdfs:Class})$

$$\frac{u \text{ rdfs:subClassOf } x . \quad v \text{ rdf:type } u .}{v \text{ rdf:type } x .} \text{ rdfs9}$$

$\rightsquigarrow \text{type}(v, x) \leftarrow \text{subClassOf}(u, x) \wedge \text{type}(v, u)$

$$\frac{u \text{ rdf:type rdfs:Class .}}{u \text{ rdfs:subClassOf } u .} \text{ rdfs10}$$

$\rightsquigarrow \text{subClassOf}(u, u) \leftarrow \text{type}(u, \text{rdfs:Class})$

$$\frac{u \text{ rdfs:subClassOf } v . \quad v \text{ rdfs:subClassOf } x .}{u \text{ rdfs:subClassOf } x .} \text{ rdfs11}$$

$\rightsquigarrow \text{subClassOf}(u, x) \leftarrow \text{subClassOf}(u, v) \wedge \text{subClassOf}(v, x)$

**a, b** là các IRIs  
**u, v** - IRI hoặc nút rỗng  
**x, y** - IRI, nút rỗng  
hoặc hằng  
**l** - hằng giá trị  
**\_n**: nút rỗng

# Luật suy diễn RDFS<sub>(4)</sub>

$$\frac{u \text{ rdf:type } \text{rdfs:ContainerMembershipProperty} \quad .}{u \text{ rdfs:subPropertyOf } \text{rdfs:member} \quad .} \text{rdfs12}$$

$\rightsquigarrow \text{subPropertyOf}(u, \text{rdfs:member}) \leftarrow \text{type}(u, \text{rdfs:ContainerMembershipProperty})$

**a, b** là các IRIs

**u, v** - IRI hoặc nút rỗng

**x, y** - IRI, nút rỗng  
hoặc hằng

**l** - hằng giá trị

**\_n**: nút rỗng

# Nội dung

7.1. Khái niệm luật

7.2. Datalog như ngôn ngữ luật bậc nhất

7.3. Kết hợp luật với OWL DL

7.4. Định dạng trao đổi luật RIF

# OWL và Luật

- Các hệ thống dựa trên luật có vai trò quan trọng và là giải pháp hiệu quả để biểu diễn tri thức và suy diễn.
- Tuy nhiên khi kết hợp luật với OWL có thể tạo ra nhiều lỗi
- Chúng ta sẽ tìm hiểu về các vấn đề và một số giải pháp
  - SWRL: một đề xuất đã được sử dụng rộng rãi
  - RIF: một quy chuẩn chưa được sử dụng rộng rãi

# Web ngữ nghĩa và Lô-gic

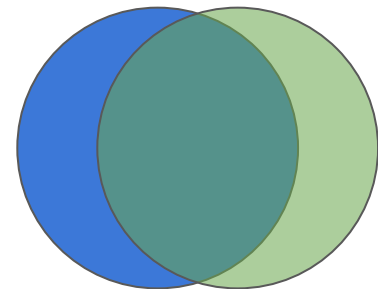
- Lô-gic giữ vai trò nền tảng của Web ngữ nghĩa
- Những hệ lô-gic nào?
  - OWL Full = Lô-gic bậc nhất (FOL)
  - OWL DL = Lô-gic mô tả
  - N3 rules  $\sim$  các luật lập trình lô-gic (LP - Logic programming)
  - SWRL  $\sim$  DL + LP
  - Các lựa chọn khác, ví dụ, lô-gic mặc định, lô-gic mờ, lô-gic xác suất, v.v...
- Các thành phần này gắn kết với nhau như thế nào và các hệ quả là gì?



# Cấu trúc mô tả và luật

- Các ontologies OWL dựa trên DL ( $\Rightarrow$  dựa trên FOL)
  - Web là một môi trường mở
  - Tái sử dụng/trao đổi
  - Ontology là một mô hình dễ hiểu
- Có nhiều hệ luật dựa trên lập trình lô-gic
  - Để đạt được tính khả quyết, các ngôn ngữ ontology không cung cấp khả năng diễn đạt như chúng ta muốn. Nhưng luật có khả năng diễn đạt cao.
  - Đã có các phần mềm thực hiện tốt việc suy diễn.
  - Các luật dễ đọc và trực quan hơn.

# Lô-gic mô tả vs. luật Horn



- Không cái nào là tập con của cái nào
- Không thể diễn đạt trong DL: Người học và sống ở cùng 1 thành phố là các sinh viên địa phương
  - Nhưng dễ dàng diễn đạt với luật:  
 $\text{localStud}(X) \leftarrow \text{studiesAt}(X, U), \text{loc}(U, L), \text{lives}(X, L)$
- Không thể diễn đạt bằng luật Horn: Người chỉ có thể là nam hoặc nữ
  - Dễ dàng diễn đạt bằng OWL DL  
`:Person owl:disjointUnionOf (:Man :Woman) .`

# Ngôn ngữ luật cho Web ngữ nghĩa - SWRL

- SWRL là sự kết hợp của DL và lô-gic horn cùng với nhiều hàm định sẵn (ví dụ, các hàm toán học)
- Được gửi tới W3C năm 2004, nhưng không trở thành quy chuẩn (dẫn đến RIF).
  - Vấn đề: Đặc tả đầy đủ của SWRL không khả quyết.
- SWRL được đặc tả tốt và các tập con được hỗ trợ rộng rãi (ví dụ, trong Pellet, HermiT).
- Dựa trên OWL: Các luật sử dụng các từ là các khái niệm OWL (lớp, thuộc tính, phần tử, hằng giá trị, v.v...)

# SWRL

- Các lớp OWL là các khẳng định 1 ngôi, các thuộc tính là các khẳng định 2 ngôi.  
 $\text{brother}(\text{?p}, \text{?s}) \leftarrow \text{sibling}(\text{?p}, \text{?s}) \wedge \text{Man}(\text{?s})$
- Các tính toán toán có thể dẫn đến các giá trị Boolean hoặc xác định giá trị cho các biến
  - $\text{swrlb:greaterThan}(\text{?age2}, \text{?age1}) \# \text{age2} > \text{age1}$
  - $\text{swrlb:subtract}(\text{?n1}, \text{?n2}, \text{?diff}) \# \text{diff} = \text{n1} - \text{n2}$
- Các khẳng định SWRL cho các mệnh đề OWL và các tập dữ liệu
  - $\text{differentFrom}(\text{?x}, \text{?y}), \text{sameAs}(\text{?x}, \text{?y}), \text{xsd:int}(\text{?x}), [3, 4, 5](\text{?x}), \dots$

# SWRL<sub>(2)</sub>

- SWRL định nghĩa một tập các khẳng định có sẵn cho phép so sánh, tính toán toán học, thao tác chuỗi, v.v..
- Ví dụ:
  - `Person(?p), hasAge(?p, ?age), swrlb:greaterThan(?age, 18) -> Adult(?p)`
  - `Person(?p), bornOnDate(?p, ?date), xsd:date(?date), swrlb:date(?date, ?year, ?month, ?day, ?timezone) -> bornInYear(?p, ?year)`
- Một số mô-tơ suy diễn (ví dụ, Pellet) cho phép bạn tự mở rộng tập những thành phần dựng sẵn

Xem: <https://www.w3.org/Submission/2004/SUBM-SWRL-20040521/>

# Các hạn chế của SWRL đầy đủ

- Nguồn gốc chính của tính phức tạp:  
Các biểu thức OWL bất kỳ (ví dụ, các giới hạn) có thể xuất hiện trong đầu hoặc thân của một luật.
- Bổ xung khả năng diễn tả đáng kể cho OWL, nhưng dẫn đến tính không khả quyết  
Không có mô-tơ suy diễn nào cho ra các kết luận tuyệt đối giống với ngữ nghĩa của SWRL

# Các tập con của SWRL

- Khó khăn: Xác định các tập ngôn ngữ con của SWRL với sự cân đối hợp lý giữa khả năng diễn đạt và khả năng tính toán
- Ứng viên OWL DL + Luật DL an toàn
  - Tất cả các biến phải xuất hiện trong phần tử phi lô-gic mô tả trong thân luật.

# Các luật DL an toàn

- Các mô-tơ suy diễn tiêu chuẩn chỉ hỗ trợ các luật DL an toàn: Các biến luật chỉ được gán với các phần tử đã biết (tức là, owl:NamedIndividual trong Owl 2).
- Ví dụ:  $:Vehicle(?v) \wedge :Motor(?m) \wedge :hasMotor(?v,?m) \rightarrow :MotorVehicle(?v)$
- Trong đó:  $:Car = :Vehicle \text{ and some } :hasMotor \text{ Motor}$  và  $:x \text{ a } :Car$
- Mô-tơ suy diễn sẽ không gán ?m với một xe máy nếu nó không phải là một phần tử đã biết.
- Vì vậy luật không thể kết luận  $MotorVehicle(:x)$



# Các giới hạn của SWRL

Các luật SWRL không hỗ trợ nhiều tính năng hữu ích của 1 số hệ thống dựa trên luật

- Suy diễn mặc định
- Tính ưu tiên của luật
- Phủ định = sai (ví dụ, ngữ nghĩa thế giới đóng)
- Các cấu trúc dữ liệu
- ....

Các hạn chế dẫn tới RIF - Rule Interchange Format

# Các luật trong OWL

Có thể biểu diễn được những luật nào trong OWL?

DL	Luật
$A \sqsubseteq B$	$A(x) \rightarrow B(x)$
$r \sqsubseteq s$	$r(x, y) \rightarrow s(x, y)$
$A \sqcap \exists R. \exists S. B \sqsubseteq C$	$A(x) \wedge R(x, y) \wedge S(y, x) \wedge B(z) \rightarrow C(x)$
$A \sqsubseteq \forall R. B$	$A(x) \wedge R(x, y) \rightarrow B(y)$
$A \sqsubseteq \neg B \sqcup C$	$A(x) \wedge B(x) \rightarrow C(x)$
$\top \sqsubseteq \leq 1 R. T$	$R(x, y) \wedge R(x, z) \rightarrow y = z$
$A \sqcap \exists R. \{b\} \sqsubseteq C$	$A(x) \wedge R(x, b) \rightarrow C(x)$
$\{a\} \equiv \{b\}$	$\rightarrow a = b$
$A \sqcap B \sqsubseteq \perp$	$A(x) \wedge B(x) \rightarrow \text{false}$
$A \sqsubseteq B \wedge C$	$A(x) \rightarrow B(x) \text{ và } A(x) \rightarrow C(x)$
$A \sqcup B \rightarrow C$	$A(x) \rightarrow C(x) \text{ và } B(x) \rightarrow C(x)$

# Chuyển đổi lớp và thuộc tính

$\text{Elephant}(x) \wedge \text{Mouse}(y) \rightarrow \text{biggerThan}(x, y)$

Thuộc tính tương ứng với khái niệm A:  $A \equiv \exists r_A. \text{Self}$

$\text{Elephant} \equiv \exists r_{\text{Elephant}}. \text{Self}$

$\text{Mouse} \equiv \exists r_{\text{Mouse}}. \text{Self}$

$r_{\text{Elephant}} \circ u \circ r_{\text{Mouse}} \sqsubseteq \text{biggerThan}$

$A(x) \wedge r(x, y) \rightarrow s(x, y)$  Trở thành  $r_A \circ r \sqsubseteq s$

$A(y) \wedge r(x, y) \rightarrow s(x, y)$  Trở thành  $r \circ r_A \sqsubseteq s$

$A(x) \wedge B(y) \wedge r(x, y) \rightarrow s(x, y)$  Trở thành  $r_A \circ r \circ r_B \sqsubseteq s$

$\text{Woman}(x) \wedge \text{marriedTo}(x, y) \wedge \text{Man}(y) \rightarrow \text{hasHusband}(x, y)$

$r_{\text{Woman}} \circ \text{marriedTo} \circ r_{\text{Man}} \sqsubseteq \text{hasHusband}$

# Nội dung

7.1. Khái niệm luật

7.2. Datalog như ngôn ngữ luật bậc nhất

7.3. Kết hợp luật với OWL DL

7.4. Định dạng trao đổi luật RIF



# Cách tiếp cận trao đổi luật

- Ngăn xếp RDF của W3C là 1 giải pháp tích hợp cho phép mã hóa và trao đổi tri thức
  - Hỗ trợ OWL (DL) dẫn đến một số hạn chế
  - Ví dụ, ngăn cản sự tiếp nhận của quy chuẩn luật OWL
- Có các cách tiếp cận khác để chuẩn hóa ngôn ngữ luật phục vụ trao đổi tri thức
  - RuleML: Ngôn ngữ đánh dấu luật, một cách tiếp cận XML để biểu diễn luật
  - RIF: Quy chuẩn W3C để trao đổi luật
  - Luôn không tương thích với OWL

# Nhiều ngôn ngữ luật khác nhau

- Có nhiều hệ ngôn ngữ luật: Lô-gic, Lập trình lô-gic, luật sinh, thủ tục, v.v...
  - Các phiên bản trong một hệ có thể khác biệt về cú pháp, ngữ nghĩa hoặc các khía cạnh khác
- Jess là ngôn ngữ luật sinh
  - `(defrule r42 (parent ?a ?b) (male ?a) => (assert(father ?a ?b)))`
- Prolog - ngôn ngữ lập trình lô-gic
  - `father(A, B) :- parent(A, B), Male(A) .`
- Định dạng lô-gic phổ thông / Common Logic
  - `(=> (and (parent ?a ?b)(male ?a)) (father ?a ?b))`

# Định dạng trao đổi luật

- Thay vì tạo  $N^2$  công cụ chuyển đổi cho  $N$  ngôn ngữ, chúng ta có thể
  - Phát triển một ngôn ngữ chung để trao đổi luật
  - Cho mỗi ngôn ngữ nhập/xuất các ánh xạ tới đó
- Hai định dạng hiện đại để biểu diễn luật
  - RuleML: Ngôn ngữ đánh dấu luật, một cách tiếp cận XML để biểu diễn luật
  - RIF: Định dạng trao đổi luật, một quy chuẩn W3C để trao đổi luật.

# RuleML

- Mục tiêu của RuleML: Biểu diễn đồng thời cả các luật tiến (từ dưới lên) và lui (từ trên xuống) bằng XML
- Các nỗ lực bắt đầu từ 2001
- Một mạng lưới mở những các nhân và tổ chức từ các môi trường doanh nghiệp và hàn lâm.

Xem: <http://ruleml.org>



# RIF

- Định dạng trao đổi luật của W3C
- Ba phiên bản: Core, BLD, và PRD
  - Core: Tập con chung của hầu hết các mô-tơ luật, một ngôn ngữ datalog an toàn với nhiều dựng sẵn.
  - BLD - Basic Logic Dialect: Thêm vào các hàm lô-gic, tính tương đương và các tham số có tên, tương đương lô-gic horn dương
  - PRD - Production Rules Dialect: Thêm vào các hành động với hiệu ứng phụ trong kết luận luật
- Có ánh xạ tới RDF

## Ví dụ 7.2. Luật được biểu diễn bằng RIF

Document(  
Prefix(rdfs <http://www.w3.org/2000/01/rdf-schema#>)  
Prefix(imdbrel <http://example.com/imdbrelations#>)  
Prefix(dbpedia <http://dbpedia.org/ontology/>)

Group(  
Forall ?Actor ?Film ?Role (  
If And(imdbrel:playsRole(?Actor ?Role)  
imdbrel:roleInFilm(?Role ?Film))  
Then dbpedia:starring(?Film ?Actor) ) ) )

Xem: <http://w3.org/2005/rules/wiki/Primer>

# Tài liệu RIF có thể chứa dữ kiện

Document(

Prefix(bio <http://example.com/biology#>)

Prefix(phil <http://example.com/philosophers#>)

Group(

  If bio:human(?x)

  Then bio:mortal(?x) )

Group(

  bio:human(phil:Socrates) ))

Dẫn đến kết luận: bio:mortal(phil:Socrates)

## Ví dụ 7.3. RIF (PRD)

Document(

Prefix(rdfs <http://www.w3.org/2000/01/rdf-schema#>)

Prefix(imdbrelf <http://example.com/fauximdbrelations#>)

Prefix(dbpediaf <http://example.com/fauxibdbrelations>)

Prefix(ibdbrelf <http://example.com/fauxibdbrelations#>)

Group(

Forall ?Actor (

If Or(Exists ?Film (imdbrelf:winAward(?Actor ?Film))

Exists ?Play (ibdbrelf:winAward(?Actor ?Play)) )

Then assert(dbpediaf:awardWinner(?Actor)) )

imdbrelf:winAward(RobertoBenigni LifelsBeautiful) ))

# Vì sao chúng ta cần YAKL

- YAKL - Yet another knowledge language, một ngôn ngữ biểu diễn tri thức khác
- Luật tốt cho biểu diễn tri thức
- Luật có các tính năng mạnh không được và không thể được hỗ trợ bởi OWL
  - Các luật không đơn điệu
  - Suy diễn mặc định
  - Các hàm tùy ý, bao gồm các hàm có hiệu ứng phụ
  - V.V..

# Các luật không đơn điệu

- Các luật không đơn điệu sử dụng toán tử "không thể chứng minh"
- Có thể được sử dụng để triển khai suy diễn mặc định
  - Cho rằng  $P(X)$  đúng với một số  $X$  ngoại trừ bạn có thể chứng minh nó không đúng
  - Cho rằng chim có thể bay ngoại trừ bạn biết rằng nó không thể

# Đơn điều

canFly(X) :- bird(X)

bird(X) := eagle(X)

bird(X) :- penguin(X)

eagle(sam)

penguin(tux)

# Không đơn điệu

`canFly(X) :- bird (X), \+ not(canFly(X))`

`bird(X) :- eagle(X)`

`bird(X) :- penguin(X)`

`not(canFly(X)) :- penguin(X)`

`not(canFly(X)) :- dead(X)`

`eagle(sam)`

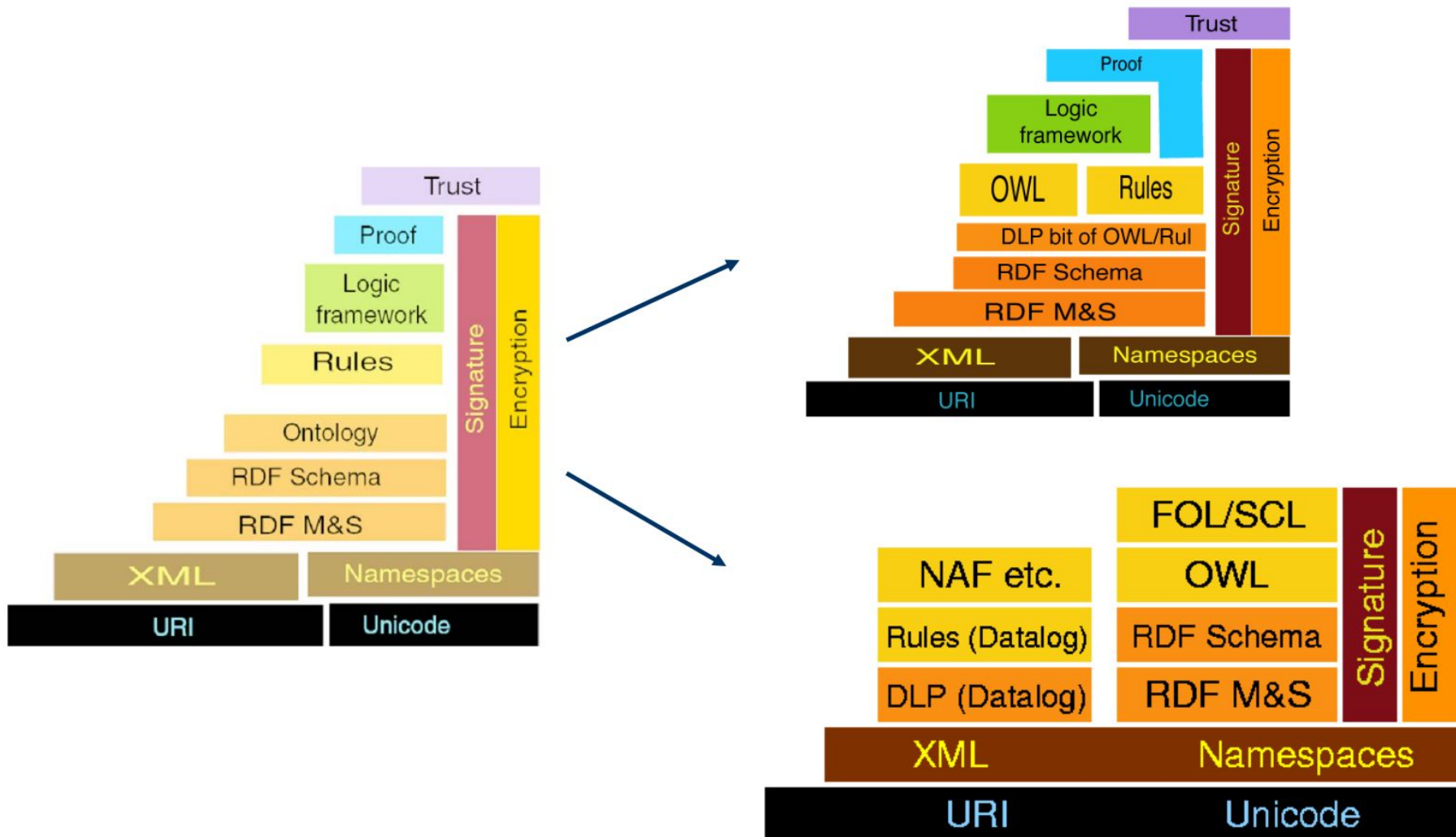
`penguin(tux)`



# Thứ tự ưu tiên luật

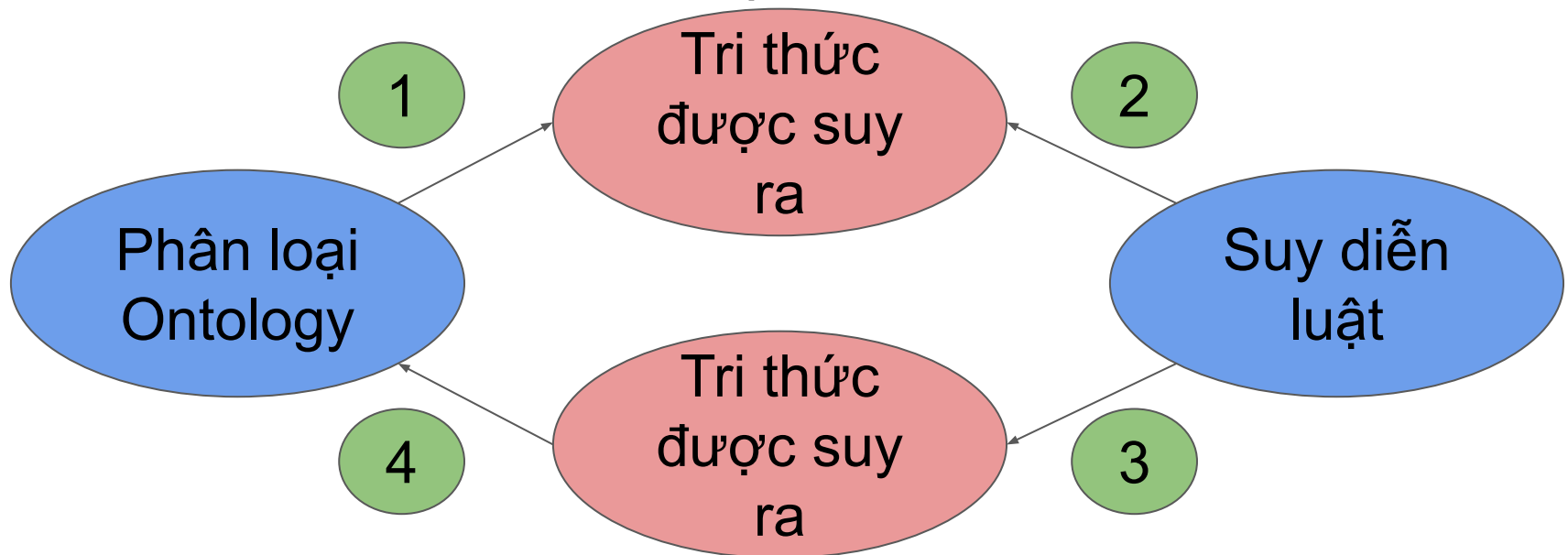
- Cách tiếp cận này có thể được mở rộng để triển khai các hệ thống trong đó luật có thứ tự ưu tiên
- Đặc điểm này khá quen thuộc với người, được sử dụng trong các tổ chức
  - Ví dụ, các quy định chung cho tổ chức cao hơn các quy định riêng của từng phòng ban.

# Có nhiều phiên bản của Web ngữ nghĩa?



# Các giới hạn

- Các hỗ trợ suy diễn theo luật không tương thích với các phân loại OWL
  - Các khẳng định mới được phát hiện bởi luật có thể vi phạm các giới hạn đang có trong ontology
  - Các tri thức mới được suy ra từ phân loại có thể cung cấp tri thức hữu ích cho các luật.



# Các giới hạn<sub>(2)</sub>

- Các giải pháp hiện có: Giải quyết các xung đột có thể xảy ra theo cách thủ công
- Giải pháp lý tưởng: Một mô-đun đồng thời cho cả phân lớp ontology và suy diễn luật
- Sẽ thế nào nếu chúng ta muốn kết hợp các tính năng không đơn điệu với lô-gic cổ điển?
- Giải pháp bộ phận:
  - Bên ngoài thông qua các mô-đun luật phù hợp.
  - Lập trình tập câu trả lời

