

# 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

# 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  $\leadsto$  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  $\leadsto$  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)`
  - Kết nối 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 định 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

→ Trong đó các 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

# Vì sao chúng ta cần luật?

- Luật là 1 cách biểu diễn tri thức khác
- Bên cạnh các tính năng tương tự OWL, luật có thể 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.



# 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*

- Luật Datalog chỉ chứa toán tử hội, hằng ký tự, và các biến với định lượng toàn thể. Không chứa phép tuyển, phủ định, định lượng tồn tại, hoặc hàm ký tự.
- Datalog ban đầu được nghiên cứu cho các CSDL suy diễn (ngôn ngữ truy vấn) và TTNT (biểu diễn tri thức).
- 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.

# Cú pháp của Datalog

- Mỗi hằng hoặc biến là 1 từ. Mỗi công thức dạng  $R(t_1, \dots, t_n)$  trong đó  $R$  là 1 thuộc tính (hoặc quan hệ) bậc  $n$ , và  $t_1, \dots, t_n$  là các từ, được gọi là 1 nguyên tử.
- Luật Datalog là biểu thức có dạng:

$$H \leftarrow B_1 \wedge \dots \wedge B_m$$

Trong đó  $H$  và  $B_1, \dots, B_m$  là các nguyên tử.  $H$  được gọi là đầu hoặc kết luận;  $B_1 \wedge \dots \wedge B_m$  được gọi là thân luật/tiền đề.

- Luật với thân luật rỗng ( $m = 0$ ) được gọi là dữ kiện.
- Luật nền là luật không có biến (tất cả các từ đều là hằng).
- Tập luật Datalog được gọi là 1 chương trình Datalog.

## Ví dụ 7.2. Chương trình Datalog

father(alice, bob)

mother(alice, carla)

mother(ewan, carla)

father(carla, david)

Parent(x, y)  $\leftarrow$  father(x, y)

Parent(x, y)  $\leftarrow$  mother(x, y)

Ancestor(x, y)  $\leftarrow$  Parent(x, y)

Ancestor(x, z)  $\leftarrow$  Parent(x, y)  $\wedge$  Ancestor(y, z)

SameGeneration(x, x)

SameGeneration(x, y)  $\leftarrow$  Parent(x, v)  $\wedge$  Parent(y, w)  $\wedge$

SameGeneration(v, w)

# Ngữ nghĩa của Datalog

Các phương pháp biểu diễn ngữ nghĩa của Datalog

- Mô hình lý thuyết
- Chứng minh lý thuyết
- 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 công thức 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,  $\leftarrow$  là suy diễn lô-gic
- Có thể coi luật Datalog như công thức suy diễn trong lô-gic bậc nhất.
- Ngữ nghĩa mô hình lý thuyết được xây dựng trên cơ sở các biểu diễn và các ánh xạ
  - (Tham khảo ngữ nghĩa mô hình-lý thuyết của *RDF*, *RDFS* và *OWL*)

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

- Một biểu diễn  $\mathcal{I}$  là mô hình của 1 chương trình Datalog  $P$ , nếu  $\mathcal{I}$  là mô hình của tất cả các luật trong  $P$ .

*Có luôn tồn tại mô hình như vậy? Nếu có thì làm sao để xây dựng nó?*

# Ngữ nghĩa lý thuyết-chứng minh

*Khả năng suy diễn các nguyên tử nền thường được quan tâm*

- Nếu phần thân luật được đáp ứng bởi các nguyên tử nền đã có, thì luật được áp dụng và nguyên tử nền tương ứng với phần đầu của nó được suy ra.
- Tập nguyên tử nền có thể được suy ra từ 1 chương trình Datalog  $P$  là tập nhỏ nhất của các nguyên tử  $A$  sao cho có 1 luật  $H \leftarrow B_1 \wedge \dots \wedge B_n$  và 1 thể hiện nền  $\theta$  trong đó:
  - $A = H\theta$ , và
  - với mỗi  $i \in \{1, \dots, n\}$ ,  $B_i\theta$  có thể được suy ra từ  $P$ .
- Lưu ý:
  - $n = 0$  đối với dữ kiện nền.
  - Nếu các biến trong đầu luật không xuất hiện trong thân luật, thì chúng có thể là bất kỳ hằng nào được biết.



## Ví dụ 7.3. Chứng minh

Cho :

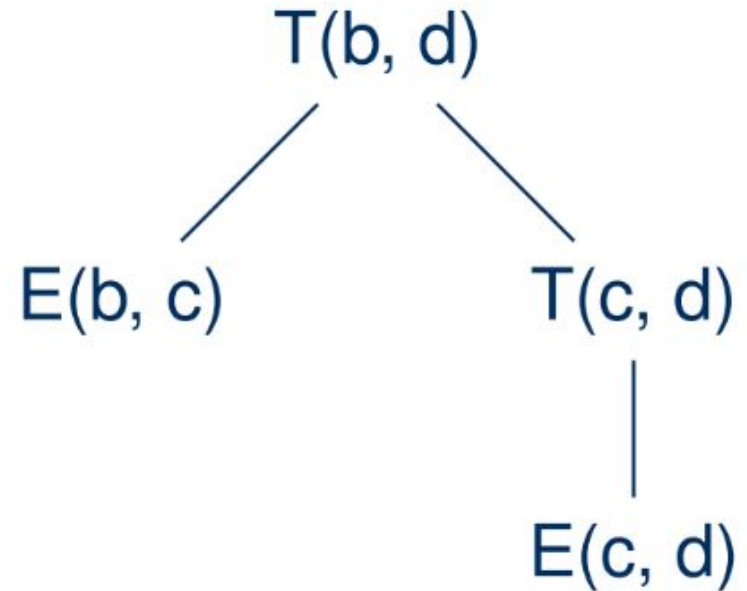
- $E(a, b), E(b, c), E(c, d)$
- $T(x, y) \leftarrow E(x, y) \quad (1)$
- $T(x, y) \leftarrow E(x, z) \wedge T(z, y) \quad (2)$

Có thể suy ra:

- (a)  $T(c, d)$  được suy ra từ (1) và  $E(c, d)$
- (b)  $T(b, d)$  được suy ra từ  $E(b, c)$ , (a) và (2)
- (c) ...

## Ví dụ 7.3. Chứng minh<sub>(2)</sub>

*Có thể biểu diễn các minh chứng theo cấu trúc cây, được gọi là cây chứng minh*



$E(b, c)$  và  $E(c, d)$  là các dữ kiện nền được cho.

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

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

(c) ...

# Cây chứng minh

## Định nghĩa:

Cây chứng minh cho 1 dữ kiện  $A$  từ 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 nền
3. Gốc được gán nhãn  $A$
4. Mỗi nút nội tương ứng với 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  $A_2, \dots, A_n$

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

# Ngữ nghĩa dựa trên điểm cố định

## Định nghĩa:

*Nền của 1 chương trình Datalog  $P$  -  $\text{ground}(P)$  là tập luật nền thu được từ các luật trong  $P$  bằng cách thay tất cả các biến bằng các hằng trong không gian hằng.*

- Toán tử hệ quả trực tiếp  $T_P$  ánh xạ tập dữ kiện nền  $I$  tới tập dữ kiện nền  $T_P(I)$ 
  - $T_P(I) = \{H \mid H \leftarrow B_1 \wedge \dots \wedge B_n \in \text{ground}(P) \text{ and } B_1, \dots, B_n \in I\}$
  - Điểm cố định tối thiểu của  $T_P$ : Tập  $L$  nhỏ nhất sao cho  $T_P(L) = L$
  - Tính từ đáy:  $T_P^0 = \emptyset$  và  $T_P^{i+1} = T_P(T_P^i)$
  - Điểm cố định tối thiểu của  $T_P$  là  $T_P^\infty = \bigcup_{i \geq 0} T_P^i$

*Nguyên tử nền  $A$  được suy ra từ  $P$  khi và chỉ khi  $A \in T_P^\infty$*

# Giải thuật tìm điểm cố định tối thiểu

*Xét giải thuật thô sơ nhất*

1.  $T_P^0 = \emptyset$
2.  $i = 0$
3. do
4.    $T_P^{i+1} = \emptyset$
5.   for  $H \leftarrow B_1 \wedge \dots \wedge B_\ell \in P$ :
6.     for each  $(\{B_1, \dots, B_\ell\})\theta$ :
7.        $T_P^{i+1} = T_P^{i+1} \cup \{H\theta\}$
8.    $i = i + 1$
9. while  $T_P^{i-1} \neq T_P^i$

Ghi chú:

*$\theta$  là ánh xạ thay thế các biến bằng các hằng trong không gian hằng.*

# Ví dụ 7.4. Tìm điểm cố định

Xét chương trình Datalog:

$e(1, 2) \ e(2, 3) \ e(3, 4) \ e(4, 5)$

$T(x, y) \leftarrow e(x, y) \quad (R1)$

$T(x, z) \leftarrow T(x, y) \wedge T(y, z) \quad (R2)$

Chúng ta có:

$$T_P^0 = \emptyset$$

$$T_P^1 = \{T(1, 2), T(2, 3), T(3, 4), T(4, 5)\}$$

$$T_P^2 = T_P^1 \cup \{T(1, 3), T(2, 4), T(3, 5)\}$$

$$T_P^3 = T_P^2 \cup \{T(1, 4), T(2, 5), T(1, 5)\}$$

$$T_P^4 = T_P^3 = T_P^\infty$$

*Có vấn đề gì với khối lượng tính toán? Số lần khớp luật dư thừa?*

*Chúng ta chưa tìm hiểu các giải thuật khó nhưng ưu việt hơn trong bài giảng này*

# Các mở rộng của Datalog

*Trong thực tế các tùy chỉnh có thể bổ xung thêm các tính năng khác nhau*

- Tùy chỉnh để tương thích với Web ngữ nghĩa
  - Các kiểu XSD
  - Sử dụng URIs
  - V.V..
- Cho phép sử dụng phép tuyển trong đầu luật
- Các phủ định không đơn điệu (không có ngữ nghĩa FOL)/non-monotonic negation
- V.V..

# Mô phỏng ngữ nghĩa RDFS bằng luật



# Các luật RDF và RDFS

**rdfs2:**  $\text{rdf:type}(x, C) \leftarrow \text{rdfs:domain}(p, C) \wedge p(x, y)$

**rdfs3:**  $\text{rdf:type}(y, D) \leftarrow \text{rdfs:range}(p, D) \wedge p(x, y)$

**rdfs4a:**  $\text{rdf:type}(x, \text{rdfs:Resource}) \leftarrow p(x, y)$

**rdfs4b:**  $\text{rdf:type}(y, \text{rdfs:Resource}) \leftarrow p(x, y)$

**rdfs5:**  $\text{rdfs:subPropertyOf}(p, r) \leftarrow \text{rdfs:subPropertyOf}(p, q) \wedge$   
 $\text{rdfs:subPropertyOf}(q, r)$

**rdfs6:**  $\text{rdfs:subPropertyOf}(p, p) \leftarrow \text{rdf:type}(p, \text{rdf:Property})$

**rdfs7:**  $q(x, y) \leftarrow \text{rdfs:subPropertyOf}(p, q) \wedge p(x, y)$

**rdfs8:**  $\text{rdfs:subClassOf}(C, \text{rdfs:Resource}) \leftarrow \text{rdf:type}(C, \text{rdfs:Class})$

**rdfs9:**  $\text{rdf:type}(x, D) \leftarrow \text{rdfs:subClassOf}(C, D) \wedge \text{rdf:type}(x, C)$

**rdfs10:**  $\text{rdfs:subClassOf}(C, C) \leftarrow \text{rdf:type}(C, \text{rdfs:Class})$

**rdfs11:**  $\text{rdfs:subClassOf}(C, E) \leftarrow \text{rdfs:subClassOf}(C, D) \wedge$   
 $\text{rdfs:subClassOf}(D, E)$

# Các luật RDF và RDFS<sub>(2)</sub>

**rdfs12:** `rdfs:subPropertyOf(p, rdfs:member) ←`  
`rdf:type(p, rdfs:ContainerMembershipProperty)`

**rdfs13:** `rdfs:subClassOf(x, rdfs:Literal) ← rdf:type(x, rdfs:Datatype)`

# 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

# 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?

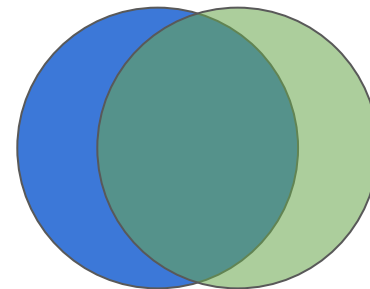
# OWL và luật

- 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.
- Cơ sở chung - lô-gic bậc nhất là tiền đề để kết hợp OWL và luật.
- Tuy nhiên trong trường hợp tổng quát kết hợp luật với OWL có thể tạo ra 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

# OWL và luật<sub>(2)</sub>

- Các ontologies OWL dựa trên DL ( $\Rightarrow$  dựa trên FOL)
  - Web là một môi trường mở
  - Tái sử dụng/trao đổi
  - Ontology chứa các mô tả với các diễn đạt tự nhiên.
- 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 có thể 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 1 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

# Các giới hạn của SWRL

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

# 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:  $A \equiv \exists r_A. \text{Self}$

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

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

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

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

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

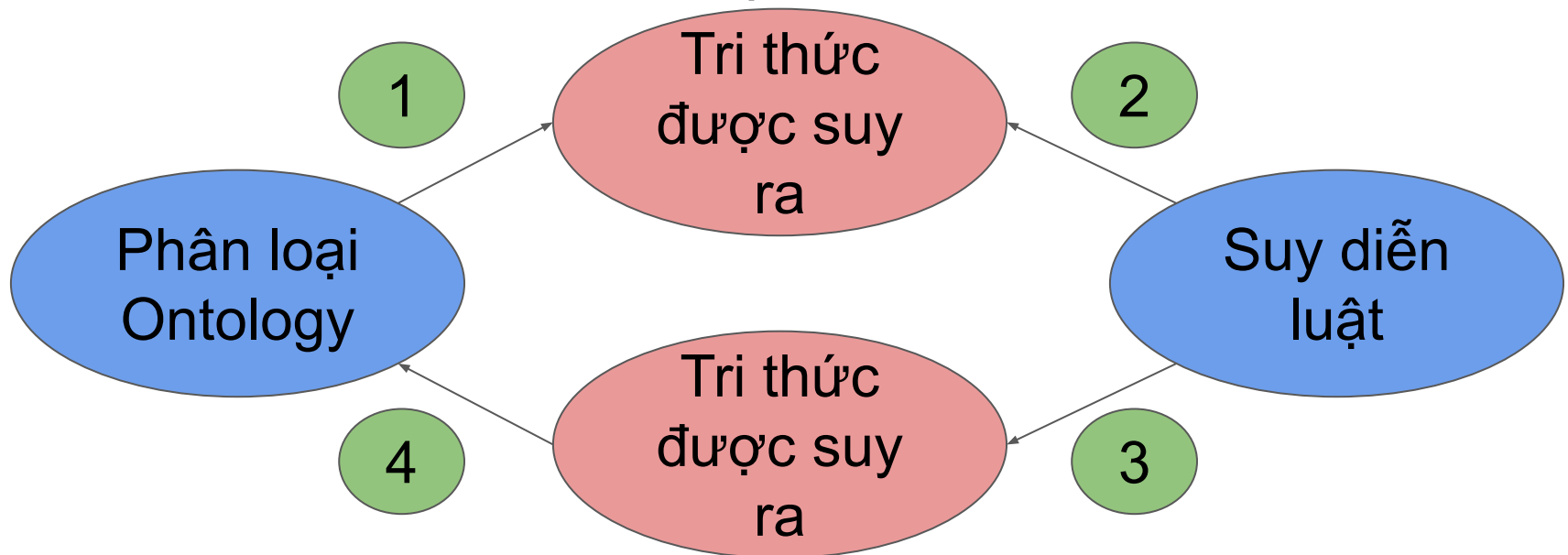
$A(x) \wedge B(y) \wedge r(x, y) \rightarrow s(x, y)$  Trở thành  $r_A \circ r \circ r_B \sqsubseteq s$

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

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

# Các giới hạn

- Các hỗ trợ suy diễn theo luật có thể không tương thích với các lớp 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
  - V.V..

# 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



# 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 1 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, vì sao không phát triển 1 ngôn ngữ chung để trao đổi luật?
  - và phát triển các cơ chế ánh xạ tới các ngôn ngữ luật chuyên dụng
- Các định dạng trao đổi luật tiêu biểu hiện nay:
  - 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
- Có mạng lưới mở gồm những cá nhân và các tổ chức từ các môi trường doanh nghiệp và hàn lâm.

[\[https://www.w3.org/2005/rules/wg/wiki/RuleML\]](https://www.w3.org/2005/rules/wg/wiki/RuleML)

# 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, 1 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
  - PRD - Production Rules Dialect: Thêm vào các hành động với hiệu ứng phụ trong kết luận luật
- Có ánh xạ tới RDF

[<https://www.w3.org/TR/2013/NOTE-rif-primer-20130205/>]

# Ví dụ 7.5. 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)

)

)

)

# Các luật RDFS bằng RIF

**rdfs2:**  $\text{rdf:type}(x, C) \leftarrow \text{rdfs:domain}(p, C) \wedge p(x, y)$

Forall  $?x ?y ?p ?C$  ( $?x[\text{rdf:type} \rightarrow ?C] :- \text{And} (?x[\text{rdfs:domain} \rightarrow ?C] ?x[?p \rightarrow ?y])$ )

**rdfs3:**  $\text{rdf:type}(y, D) \leftarrow \text{rdfs:range}(p, D) \wedge p(x, y)$

Forall  $?x ?y ?p ?D$  ( $?y[\text{rdf:type} \rightarrow ?D] :- \text{And} (?p[\text{rdfs:range} \rightarrow ?D] ?x[?p \rightarrow ?y])$ )

**rdfs4a:**  $\text{rdf:type}(x, \text{rdfs:Resource}) \leftarrow p(x, y)$

Forall  $?x$  ( $?x[\text{rdf:type} \rightarrow \text{rdfs:Resource}] :- \text{Exists } ?p ?y (?x[?p \rightarrow ?y])$ )

**rdfs4b:**  $\text{rdf:type}(y, \text{rdfs:Resource}) \leftarrow p(x, y)$

Forall  $?y$  ( $?y[\text{rdf:type} \rightarrow \text{rdfs:Resource}] :- \text{Exists } ?p ?x (?x[?p \rightarrow ?y])$ )

**rdfs5:**  $\text{rdfs:subPropertyOf}(p, r) \leftarrow \text{rdfs:subPropertyOf}(p, q) \wedge \text{rdfs:subPropertyOf}(q, r)$

Forall  $?p ?q ?r$  ( $?p[\text{rdfs:subPropertyOf} \rightarrow ?r] :- \text{And} (?p[\text{rdfs:subPropertyOf} \rightarrow ?q] ?q[\text{rdfs:subPropertyOf} \rightarrow ?r])$ )

**rdfs6:**  $\text{rdfs:subPropertyOf}(p, p) \leftarrow \text{rdf:type}(p, \text{rdf:Property})$

Forall  $?p$  ( $?p[\text{rdfs:subPropertyOf} \rightarrow ?p] :- ?p[\text{rdf:type} \rightarrow \text{rdf:Property}]$ )

**rdfs7:**  $q(x, y) \leftarrow \text{rdfs:subPropertyOf}(p, q) \wedge p(x, y)$

Forall  $?x ?y ?p ?q$  ( $?x[?q \rightarrow ?y] :- \text{And} (?p[\text{rdfs:subPropertyOf} \rightarrow ?q] ?x[?p \rightarrow ?y])$ )

*Ghi chú:* Cấu trúc  $s[p \rightarrow o]$  được gọi là khung, tương đương với bộ-3 (s p o)



# Các luật RDF và RDFS bằng RIF<sub>(2)</sub>

**rdfs8:** *rdfs:subClassOf*(C, *rdfs:Resource*) ← *rdf:type*(C, *rdfs:Class*)

Forall ?C (?C[*rdfs:subClassOf* -> *rdfs:Resource*] :- ?C[*rdf:type* -> *rdfs:Class*])

**rdfs9:** *rdf:type*(x, D) ← *rdfs:subClassOf*(C, D) ∧ *rdf:type*(x, C)

Forall ?x ?C ?D (?x[*rdf:type*->?D] :- And(?C[*rdfs:subClassOf*->?D]  
?x[*rdf:type*->?C]))

**rdfs10:** *rdfs:subClassOf*(C, C) ← *rdf:type*(C, *rdfs:Class*)

Forall ?C (?C[*rdfs:subClassOf* -> ?C] :- ?C[*rdf:type* -> *rdfs:Class*])

**rdfs11:** *rdfs:subClassOf*(C, E) ← *rdfs:subClassOf*(C, D) ∧ *rdfs:subClassOf*(D, E)

Forall ?C ?D ?E (?C[*rdfs:subClassOf* -> ?E] :- And (?C[*rdfs:subClassOf* -> ?D]  
?D[*rdfs:subClassOf*->?E]))

**rdfs12:** *rdfs:subPropertyOf*(p, *rdfs:member*) ←

*rdf:type*(p, *rdfs:ContainerMembershipProperty*)

Forall ?p (?p[*rdfs:subPropertyOf* -> *rdfs:member*] :-

?p[*rdf:type* -> *rdfs:ContainerMembershipProperty*])

**rdfs13:** *rdfs:subClassOf*(x, *rdfs:Literal*) ← *rdf:type*(x, *rdfs:Datatype*)

Forall ?x (?x[*rdfs:subClassOf* -> *rdfs:Literal*] :- ?x[*rdf:type* -> *rdfs:Datatype*])

*Ghi chú:* cú pháp A :- B tương đương If B Then A

# 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 7.6. 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)))

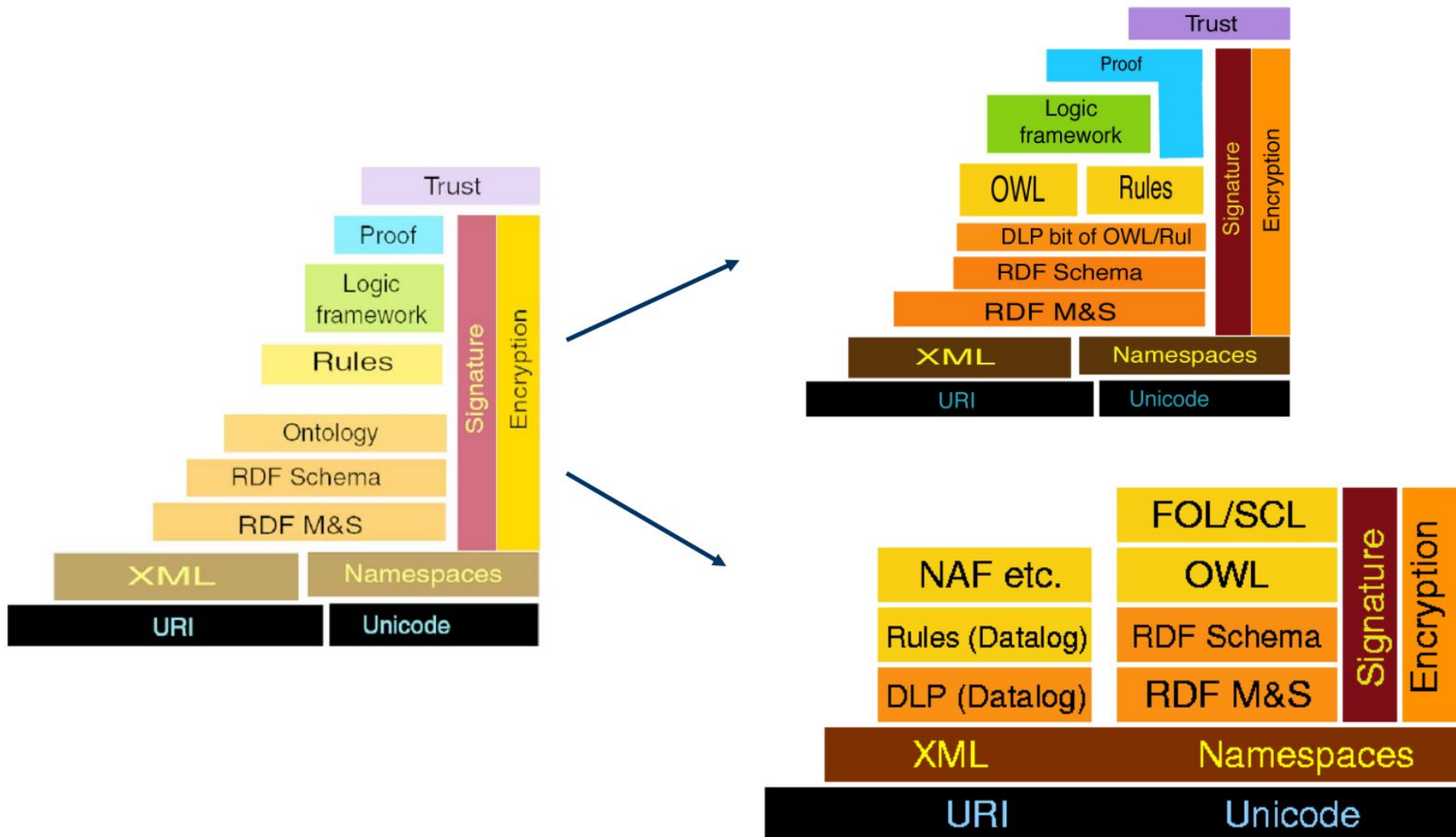
Then assert(dbpediaf:awardWinner(?Actor))

)

imdbrelf:winAward(RobertoBenigni LifelsBeautiful)

))

# Có nhiều phiên bản của Web ngữ nghĩa?



# Phụ lục: Luật trong Apache Jena

Rule     := bare-rule .

      or [ bare-rule ]

      or [ ruleName : bare-rule ]

bare-rule := term, ... term -> hterm, ... hterm   // FW rule

or bhterm <- term, ... term   // BW rule

hterm     := term

      or [ bare-rule ]

term      := (node, node, node)                   // triple pattern

      or (node, node, functor)                   // extended triple pattern

      or builtin(node, ... node)                 // invoke procedural primitive

bhterm     := (node, node, node)                   // triple pattern

functor    := functorName(node, ... node) // structured literal

# Phụ lục: Luật trong Apache Jena<sub>(2)</sub>

node        :=   uri-ref                // e.g. http://foo.com/eg  
or   prefix:localname                // e.g. rdf:type  
or   <uri-ref>                        // e.g. <myscheme:myuri>  
or   ?varname                        // variable  
or   'a literal'                        // a plain string literal  
or   'lex'^^typeURI                // a typed literal, xsd:\* type names supported  
or   number                            // e.g. 42 or 25.5

[<https://jena.apache.org/documentation/inference/>]

# Các thành phần dựng sẵn

*builtin*

sum(?a, ?b, ?c)

addOne(?a, ?c)

difference(?a, ?b, ?c)

min(?a, ?b, ?c)

max(?a, ?b, ?c)

product(?a, ?b, ?c)

quotient(?a, ?b, ?c)

v.v..

## Ví dụ 7.7. Luật trong Apache Jena

```
[allID: (?C rdf:type owl:Restriction), (?C owl:onProperty ?P),  
      (?C owl:allValuesFrom ?D) ->  
      (?C owl:equivalentClass all(?P, ?D)) ]  
[all2: (?C rdfs:subClassOf all(?P, ?D)) -> print('Rule for ', ?C)  
      [all1b: (?Y rdf:type ?D) <- (?X ?P ?Y), (?X rdf:type ?C) ]  
      ]  
[max1: (?A rdf:type max(?P, 1)), (?A ?P ?B), (?A ?P ?C)  
      -> (?B owl:sameAs ?C) ]
```



# Ví dụ 7.8. Kết hợp Ontology và 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)]

# Demo suy diễn 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.

:c :p :d.

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

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

# Demo suy diễn luật trong Apache Jena<sub>(2)</sub>

@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.

:c :p :d.

:a :p :c. :a :p :d. :b :p :d.

:a :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> ;  
].
```

# Mô phỏng ngữ nghĩa RDFS trong Jena

[**rdfs2:** ( $?x ?p ?y$ ), ( $?p$  rdfs:domain  $?c$ )  $\rightarrow$  ( $?x$  rdf:type  $?c$ )]

[**rdfs3:** ( $?x ?p ?y$ ), ( $?p$  rdfs:range  $?c$ )  $\rightarrow$  ( $?y$  rdf:type  $?c$ )]

[**rdfs5a:** ( $?a$  rdfs:subPropertyOf  $?b$ ), ( $?b$  rdfs:subPropertyOf  $?c$ )  $\rightarrow$   
( $?a$  rdfs:subPropertyOf  $?c$ )]

[**rdfs5b:** ( $?a$  rdf:type rdf:Property)  $\rightarrow$  ( $?a$  rdfs:subPropertyOf  $?a$ )]

[**rdfs6:** ( $?a ?p ?b$ ), ( $?p$  rdfs:subPropertyOf  $?q$ )  $\rightarrow$  ( $?a ?q ?b$ )]

[**rdfs7:** ( $?a$  rdf:type rdfs:Class)  $\rightarrow$  ( $?a$  rdfs:subClassOf  $?a$ )]

[**rdfs8:** ( $?a$  rdfs:subClassOf  $?b$ ), ( $?b$  rdfs:subClassOf  $?c$ )  $\rightarrow$   
( $?a$  rdfs:subClassOf  $?c$ )]

[**rdfs9:** ( $?x$  rdfs:subClassOf  $?y$ ), ( $?a$  rdf:type  $?x$ )  $\rightarrow$  ( $?a$  rdf:type  $?y$ )]

[**rdfs10:** ( $?x$  rdf:type rdfs:ContainerMembershipProperty)  $\rightarrow$   
( $?x$  rdfs:subPropertyOf rdfs:member)]

[<https://github.com/apache/jena/blob/main/jena-core/src/main/resources/etc/rdfs.rules>]

# Mở rộng tập luật trong Jena

# 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.
- c) Lập trình lô-gic

# 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>

# 3 Lập trình lô-gic

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

