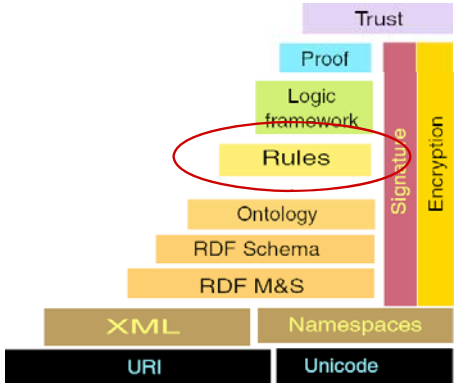
TẦNG TRI THỨC DỰA LUẬT VÀ LOGIC



Biểu diễn tri thức

- BDTT là cơ sở của các hệ thống thông minh
- Vai trò của ontology trong BDTT
  - Để khai báo các tri thức về thế giới
  - Ontology đưa khai báo vào các loại (khái niệm, vai trò, …)
- Luật và suy diễn
  - Luật suy diễn cho phép suy ra các tri thức ẩn (*procedural knowledge)* từ các tri thức rõ (*declarative knowledge*)
  - Luật cho phép diễn tả các ràng buộc giữa các đối tượng

Luật Horn

- Là tập con của First Order Logic
  - Biểu thức Horn là phép hợp của các biểu thức đơn với 1 giá trị khẳng định
    - $(\forall)\ \neg B_1 \vee \neg B_2 \vee \ldots \vee \neg B_n \vee H$
  - Tương đương với
    - $(\forall)\ B_1 \wedge B_2 \wedge \ldots \wedge B_n \rightarrow H$

Các biểu thức đơn

- Các hằng số với các giá trị xác định
  - *a, b, john,…*
- Các biến
  - *x, y,…*
- Các hàm trả về giá trị với các tham số nhất định
  - *f*(*x*), *fatherOf*(*john*),…
- Các hằng, biến, hàm (gọi là các thuật ngữ)
- Các vị từ liên kết các thuật ngữ
  - *p*(*x, a*), *marriage*(*mary, john*),…
- Nếu *p* là vị từ, t là thuật ngữ, khi đó $p(t_1, t_2,\ldots)$ là biểu thức đơn
- Nếu t_1, t_2 là thuật ngữ, khi đó $t_1 = t_2$ là biểu thức đơn
  - *f*(*x*) = a, *marc* = *fatherOf*(*john*)

5

Lập trình logic – logic programming (1)

- Là mở rộng của logic Horn logic
- Luật là kết hợp của các các biến với
  - Biến dương là biểu thức nguyên tử: *p*(*x*), *q*(*x*),...
  - Biến âm là phủ định của biểu thức nguyên tử: not *p*(*x*), not *q*(*x*),...
    - not (negation-as-failure) ≠ ¬
  - Biến cơ sở là biến không có tham số
- Luật Horn (H :- $B_1,\ldots,B_n$) or ($H \leftarrow B_1 \wedge \ldots \wedge B_n$)
  - *H là biến dương*
  - $B_1,\ldots,B_n$ là các biến

6

Lập trình logic (2)

- Một sự kiện là 1 biểu thức nguyên tử (luật không có thân)
  - *person*(*john*)
- ĐÍch hoặc câu truy vấn là luật không có phần đầu, biểu diễn bởi (?- $B_1,\ldots,B_n$)
  - ?- *person*(*x*)
- LP không có phủ định tương đương với tập con của FOL (Horn Logic Programs)
- Datalog là tập con của LP
  - Không có ký hiệu hàm
  - Không có phủ định

7

Các đặc tính của DLP

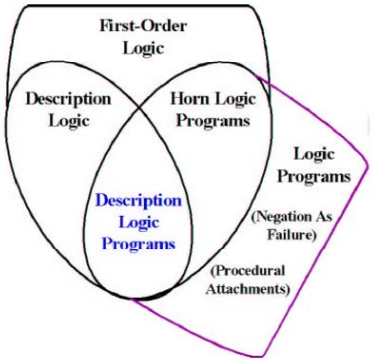
- Tập con RDFS của DL cho phép các phát biểu sau:
  - Lớp C là lớp con của lớp D.
  - Miền của thuộc tính P là lớp C
  - Giới hạn phạm vi của thuộc tính P là lớp D.
  - Thuộc tính P là thuộc tính con của thuộc tính Q
  - A là một giá trị của lớp C.
  - (a,b) là một giá trị của thuộc tính P.
- DLP có thể biểu diễn:
  - Sử dụng kết nối Intersection trong mô tả lớp
  - Khai báo thuộc tính P là truyền ứng (Transitive).
  - Khai báo thuộc tính P là đối xứng (Symmetric).
- DLP có thể biểu diễn hầu hết các đặc tính của DL
- Các vấn đề kỹ thuật trong LP:
  - Xử lý tính bằng nhau (vd, tính duy nhất của tên)

8

Ví dụ

- Các sự kiện về quan hệ
  - mother(mary, john)
  - father(marc, john)
  - male(john)
- Luật về quan hệ
  - parent(X,Y) :- mother(X,Y)
  - parent(X,Y) :- father(X,Y)
  - female(X) :- mother(X,Y)
  - male(X) :- father(X,Y)
- Truy vấn
  - ?- female(mary)
  - ?- parent(x, john)

Expressivity overlaps



Kết hợp luật với ontology

- DLs cho phép biểu diễn tri thức khai báo
- LPs cho phép biểu diễn tri thức thủ tục gắn với biểu diễn tri thức
- DLP (Description Logic Programs) là cách đơn giản nhất để kết hợp DLs với logic Horn
  - Phần OWL có thể định nghĩa dưới dạng Horn
  - Phần logic Horn có thể định nghĩa dưới dạng OWL

Khác biệt giữa DL và DLP

- DLP là tập con của DL.
- Ví dụ của DL không biểu diễn một cách hoàn chỉnh trong DLP:
  - Khai báo lớp con của biểu thức liên kết lớp qua phép hợp:
    - (Human $\cap$ Adult) $\subseteq$ (Man $\cup$ Woman)
  - Khai báo lớp con của biểu thức liên kết lớp qua lượng từ tồn tại:
    - Radio $\subseteq \exists$hasPart.Tuner
- Tại sao không? Vì: LP/Horn, và do đó DLP, không thể biểu diễn được

Khác biệt giữa LP và DLP

- DLP là tập con của Horn LP.
- Ví dụ về Horn LP không biểu diễn được bằng DLP:
  - Luật liên quan nhiều biến:

PotentialLoveInterestBetween(?X,?Y)← Man(?X) ∧ Woman(?Y).

  - Chuỗi (ngoài phép lan truyền đơn giản) để sinh giá trị thuộc tính.

InvolvedIn(?Company, ?Industry)
← Subsidiary(?Company, ?Unit) ∧ AreaOf(?Unit, ?Industry).

- Tại sao? Tính quyết định cuả DLs phụ thuộc chủ yếu vào thuộc tính của mô hình cây
  - DL không dùng để biểu diễn nhiều hơn một biến

DLP có thể làm gì

- Các luật LP trên các DL ontologies.
  - Dịch các luật LP sang DL ontologies và ngược lại
  - Sử dụng các luật LP cho các phần của DL
  - Tạo các ontologies trong LP
  - Tạo các luật trong DL.
  - Dịch các kết luận LP sang DL
  - Dịch các kết luận DL sang LP

Ưu điểm của DLP

- Mô hình hóa: Sử dụng DL hoặc luật
- Cài đặt: sử dụng cơ chế suy luận của DL hoặc hệ thống suy diễn dựa luật
  - Dịch các luật LP sang DL ontologies và ngược lại
  - Tạo các ontologies trong LP (hoặc luật trong DL)
  - Linh động, có thể sử dụng nhiều công cụ khác nhau (vd, khai thác các công cụ LP/DB để chạy các ontology quy mô lớn)
- Khả năng biểu diễn: OWL ontologies thường chỉ dùng rất ít các phép biểu diễn ngoài DLP

Chuyển từ DL sang Horn logic (1)

- (C rdfs:subClassOf D)
  - $C \sqsubseteq D \Leftrightarrow D(x) \leftarrow C(x)$
- (Q rdfs:subPropertyOf P)
  - $Q \sqsubseteq P \Leftrightarrow P(x, y) \leftarrow Q(x, y)$
- (P rdfs:range C)
  - $f \sqsubseteq \forall P.C \Leftrightarrow C(y) \leftarrow P(x, y)$
- (P rdfs:domain C)
  - $f \sqsubseteq \forall P^{-}.C \Leftrightarrow C(y) \leftarrow P(y, x)$
- (a rdf:type C)
  - $a:C \Leftrightarrow C(a)$
- (a P b)
  - $(a, b): P \Leftrightarrow P(a, b)$
- (C owl:equivalentClass D)
  - $C \equiv D \Leftrightarrow D(x) \leftarrow C(x); C(x) \leftarrow D(x)$
- (Q owl:equivalentProperty P)
  - $Q \equiv P \Leftrightarrow P(x, y) \leftarrow Q(x, y); Q(x, y) \leftarrow P(x, y)$

Chuyển từ DL sang Horn logic (2)

- (Q owl:inverseOf P)
  - Q ≡ P^- ⇔ Q(y, x) ← P(x, y)
- (P rdf:type owl:TransitiveProperty)
  - P^+ ≡ P ⇔ P(x, z) ← P(x, y) ∧ P(y, z)
- owl:intersectionOf
  - C1 ″ C2 ≡ D ⇔ D(x) ← C1(x) ∧ C2(x)
  - C ≡ D1 ″ D2 ⇔ D1(x) ← C(x); D2(x) ← C(x)
- owl:unionOf
  - C1 ⊂ C2 ≡ D ⇔ D(x) ← C1(x); D(x) ← C2(x)
  - C ≡ D1 ⊂ C2 ⇔ impossible translation
- owl:allValuesFrom
  - C ≡ ∀P.D ⇔ (D(y) ← P(x, y)) ← C(x)
  - ∀P.C ≡ D ⇔ impossible translation
- owl:someValuesFrom
  - C ≡ ∃P.D ⇔ impossible translation
  - ∃P.C ≡ D ⇔ D(x) ← P(x, y) ∧ C(y)
- owl:complementOf (negation), owl:minCardinality, owl:maxCardinality không thể dịch được

Semantic Web Rule Language

Semantic Web Rule Language (SWRL):

- Kết hợp ontologies và luật:
  - Ontologies: OWL-DL
  - Rules: RuleML

SWRL = OWL-DL + RuleML

- OWL-DL: không có biến
  - tương ứng với SHOIN(D)
- RuleML: sử dụng biến.

RuleML

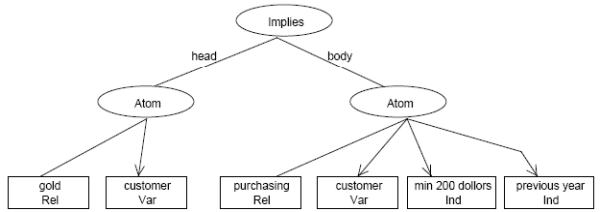
- RuleML, ngôn ngữ datalog của mệnh đề Horn:
  1. Datalog là tập con của Prolog:
     - Function-free: cách biểu diễn P(f(2),5) không hợp lệ
  2. Mệnh đề Horn (hợp của các ký hiệu và có tối đa 1 ký hiệu dương), vd
     - ¬p ∨ ¬ q ∨ . . . ∨ ¬ t ∨ u có thể viết thành,
     - p ∧ q ∧ ldots ∧ t → u
     - Chỉ có phép giao của các phần tử

Ví dụ của RuleML

1 quan hệ n-ary (n= 0, 1, 2, …) trong RuleML.

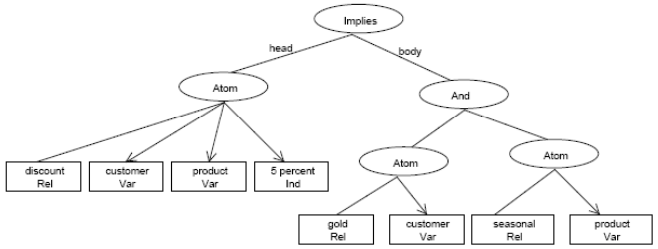
VD: A customer is gold if her purchasing has been minimum 200 dollars in the previous year.

- head (unary relationship): A customer is gold.
- body (3-ary relationship): Her purchasing . . .



Luật với nhiều phần tử

- Luật với nhiều phép and trong thân:
  - The discount for a customer on a product is 5% if the customer is gold and the product is seasonal.



Luật RuleML trong SWRL

- Chỉ có các quan hệ unary/binary
  - Các ví dụ ở slide trước không phải là luật SWRL
  - Các quan hệ n-ary relations được chuyển thành quan hệ nhị phân.
- Luật với nhiều phép and trong thân khá phổ biến. Vd
  - The discount for a customer on a product is 5% if the customer is gold and the product is seasonal.

Cú pháp SWRL

Các phần tử SWRL được định nghĩa như sau:

Atom $\leftarrow$ C(i)| D(v) | R(I, j) | U(I, v) | builtIn(p, v_1, …, v_n) | i = j | i $\neq$ j

C = Class D = Data type

R = Object Property U = Data type Property

i,j = Object variable names or Object individual names

V_1, …, v_n = Data type variable names or Data type value names

p = Built-in names

Cú pháp SWRL

Cú pháp luật SWRL:

- a $\leftarrow b_1$, …, b_n trong đó,
  - a : head (1 atom) b_s: body (nhiều atom)

1 CSTT SWRL (k) được định nghĩa như sau:

- k = (Σ,P) trong đó,
  - Σ = CSTT của SHOIN(D)
  - P = tập luật

Ngữ nghĩa SWRL

- Cho I = (Δ^I, Δ^D, $.^I$, $.^D$) trong đó
- I = phép dịch
- Δ^I = miền dịch đối tượng
- Δ^D = miền dịch dữ liệu
- $.^I$ = hàm dịch đối tượng
- $.^D$ = hàm dịch kiểu dữ liệu
- $\Delta^I \cap \Delta^D = \varnothing$, sao cho
  - $V_{IX} \rightarrow P(\Delta^I)$ $V_{DX} \rightarrow P(\Delta^D)$ trong đó,
  - V_{IX} = biến đối tượng V_{DX} = biến kiểu dữ liệu
  - P = phép toán lực lượng

Ngữ nghĩa SWRL

- Bảng ràng buộc B(I) đối với các phần tử SWRL

SWRL Atoms	Condition on Interpretation
$C(i)$	$i^I \in C^I$
$R(i,j)$	$(i^I, j^I) \in R^I$
$U(i,v)$	$(i^I, v^D) \in U^I$
$D(v)$	$v^D \in D^D$
$builtIn(p, v_1, \ldots, v_n)$	$(v_1^D, \ldots, v_n^D \in p^D)$
$i = j$	$i^I = j^I$
$i \neq j$	$i^I \neq j^I$

Ngữ nghĩa SWRL

- Các phần tử SWRL trong phần trước (antecedent) thỏa nếu:
  - Nó rỗng (đúng hiển nhiên)
  - Hoặc mọi phần tử của nó thỏa
- 1 phần tử SWRL trong phần sau (consequent) thỏa nếu:
  - Nó rỗng
  - Hoặc nó thỏa
- 1 luật thỏa phép dịch I nếu
  - Mọi ràng buộc B(I) thỏa phần trước
  - B(I) thỏa phần sau

Ví dụ về SWRL

- Trong 1 định dạng bảng, các thuật ngữ SWRL như sau:

$C(i)$	$R(i,j)$	$D(v)$	U(i,v)	$builtIn(p, v_1, \ldots, v_n)$	$i = j$	$i \neq j$

- Biến được xác định qua dấu ? trong luật
- Ví dụ:
- FastComputer(?c) ← Computer(?c) ∧ hasCPU(?c, ?cpu) ∧ hasSpeed(?cpu, ?sp) ∧ HighSpeed(?sp)
- FastComputer(?c): thuật ngữ C(i) trong bảng
- hasCPU(?c, ?cpu): thuật ngữ R(i, j) trong bảng

Diễn tả luật không sử dụng SWRL

Có thể diễn tả một số luật chỉ sử dụng DL:

VD: với luật

FastComputer(?c) ← Computer(?c) ∧ hasCPU(?c, ?cpu) ∧ hasSpeed(?cpu, ?sp) ∧ HighSpeed(?sp)

Luật chỉ sử dụng DL:

Computer ∩ ∃hasCPU. ∃hasSpeed.HighSpeed ⊆ FastComputer

Dịch luật từ SWRL sang DL,

- Phụ thuộc vào số biến dựa vào các biến chung giữa phần trước và phần sau

Dịch luật từ SWRL sang DL

Số biến chung giữa phần trước và phần sau:

- Có thể dịch được nếu:
  - 2 phần chung nhau 0 biến, nhưng có ít nhất 1 cá thể chung
  - 2 phần chung nhau 1 biến
- Không dịch được nếu:
  - 2 phần chung nhau >= 2 biến

Quá trình dịch từ SWRL sang DL

- Phần trước và phần sau trở thành các truy vấn giao
- Truy vấn kết quả được dịch thành diễn tả lớp
  - Sử dụng kỹ thuật rolling-up
- Phần trước trở thành lớp con của phần sau

Dịch luật từ SWRL sang DL

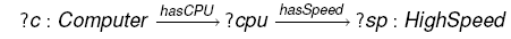
FastComputer(?c) ← Computer(?c) ∧ hasCPU(?c, ?cpu) ∧ hasSpeed(?cpu, ?sp) ∧ HighSpeed(?sp)

1. Phần trước và phần sau trở thành các truy vấn giao

   1a. ?c: FastComputer

   1b. ?c: Computer∧ (?c, ?cpu): hasCPU ∧ (?cpu, ?sp):hasSpeed ∧ ?sp:HighSpeed

   - Các phép giao tạo ra đồ thị có hướng:
     - Mỗi nút là 1 biến hoặc 1 tên
     - Mỗi cạnh là 1 quan hệ
     - 1 đồ thị truy vấn



Dịch luật từ SWRL sang DL

2. Sử dụng kỹ thuật rolling-up

$?c : Computer \xrightarrow{hasCPU} ?cpu \xrightarrow{hasSpeed} ?sp : HighSpeed$

- Mỗi cạnh ra được biểu diễn dưới dạng 1 lượng từ tồn tại
- Cạnh được biểu diễn như các ràng buộc
- Mỗi cạnh ra (?x, ?y) : R được dịch thành ∃R.Y
- Y là ràng buộc về tên lớp ?y
- ∃hasCPU.∃hasSpeed.HighSpeed

Dịch luật từ SWRL sang DL

Lớp của biến đích ?c : là Computer

- Giao với lớp Computer ∩

Kết quả rolling-up :

Computer ∩ ∃hasCPU.∃hasSpeed.HighSpeed

Lớp của biến đích ?c

?c : FastComputer dịch thành FastComputer

3. Làm phần trước trở thành lớp con của phần sau

Computer ∩ ∃hasCPU.∃hasSpeed.HighSpeed ⊆ FastComputer

Luật SWRL không chuyển sang được DL

- SWRL có thể biểu diễn một số luật mà DL không biểu diễn được

VD:

hasUncle(?nephew,?uncle) ←hasParent(?nephew, ?parent) ∧ hasBrother(?parent,?uncle)

Luật trên không thể dịch sang DL vì

- phần sau có 2 biến khác nhau
- Việc sinh phép thế cho từng biến không đủ để giải quyết

Luật SWRL không chuyển sang được DL

Mặc dù không thể suy diễn hasUncle(Bob,Bill) từ

- hasParent(Bob,Mary) và hasBrother(Mary,Bill),
- Quan hệ hasUncle có thể được dùng trực tiếp hoặc gián tiếp

VD: People whose uncles are all lawyers,

Sử dụng trực tiếp hasUncle: ∀hasUncle.Lawyer

Sử dụng gián tiếp hasUncle: ∀ hasParent. ∀hasBrother.Lawyer

Ví dụ Family

- Family ontology
  - Class Person
  - Các thuộc tính của Person: hasParent, hasBrother, hasUncle
- Luật SWRL
  - hasParent(x1,x2) ∧ hasBrother(x2,x3) → hasUncle(x1,x3)
  - Astract syntax

    Implies(

    Antecedent(hasParent(I-variable(x1) I-variable(x2)) hasBrother(I-variable(x2) I-variable(x3)))

    Consequent(hasUncle(I-variable(x1) I-variable(x3)))

)

SWRL's XML syntax

```
<ruleml:imp>
  <ruleml:_rlab ruleml:href="example"/>
  <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>
```


SWRL's RDF syntax

```
<swrl:Variable rdf:ID="x1"/>
<swrl:Variable rdf:ID="x2"/>
<swrl:Variable rdf:ID="x3"/>
<ruleml:Imp>
  <ruleml:body rdf:parseType="Collection">
    <swrl:IndividualPropertyAtom>
      <swrl:propertyPredicate rdf:resource="hasParent"/>
      <swrl:argument1 rdf:resource="#x1"/>
      <swrl:argument2 rdf:resource="#x2"/>
    </swrl:IndividualPropertyAtom>
 <swrl:IndividualPropertyAtom>
      <swrl:propertyPredicate rdf:resource="hasBrother"/>
      <swrl:argument1 rdf:resource="#x2"/>
      <swrl:argument2 rdf:resource="#x3"/>
    </swrl:IndividualPropertyAtom>
  </ruleml:body>
  <ruleml:head rdf:parseType="Collection">
    <swrl:IndividualPropertyAtom>
      <swrl:propertyPredicate rdf:resource="hasUncle"/>
      <swrl:argument1 rdf:resource="#x1"/>
      <swrl:argument2 rdf:resource="#x3"/>
    </swrl:IndividualPropertyAtom>
  </ruleml:head>
</ruleml:Imp>
```


Further reading

- "Combining Rules and Ontologies. A survey", s Deliverable I3, REWERSE project, 2005. http://rewerse.net/deliverables/m12/i3-d3.pdf
- SWRL: A Semantic Web Rule Language Combining OWL and RuleML: http://www.w3.org/Submission/SWRL/
- B. N. Grosof, I. Horrocks, R. Volz, and S. Decker. Description logic programs: Combining logic programs with description logic. In Proc. Intl. Conf. on the World Wide Web (WWW2003), Budapest, Hungary, 2003.