Machine Learning meets Knowledge Representation in the Semantic Web

Francesca A. Lisi
lisi@di.uniba.it

Università Degli Studi di Bari
Dipartimento di Informatica
Campus Universitario “E. Quagliariello”
Via E. Orabona, 4 - 70126 Bari - Italy

Tutorial @ AAAI-10
Atlanta
Motivation

- The management of ontologies and rules for the Semantic Web is a very demanding task
- ML algorithms can support this task by partially automating the knowledge acquisition process
Claim

- The logical nature of the KR formalisms underlying ontology and rule languages for the Semantic Web should not be neglected when applying ML algorithms
- ML should provide KR the methodology for inductive reasoning
- The Semantic Web offers a great opportunity for a possible dialogue between ML and KR in AI
Disclaimer

- This tutorial does not provide exhaustive survey of research in either KR or ML for the Semantic Web
- Yet it highlights interesting contributions at the intersection of ML and KR relevant for the Semantic Web context
- Ultimate goal: to show that the Semantic Web is an AI-intensive application area
Overview

http://www.di.uniba.it/~lisi/aaai10/

- The Semantic Web (0:30h)
- Knowledge Representation for the Semantic Web (1:00h)
- Machine Learning for the Semantic Web (1:30h)
- Conclusions (0:30h)
Overview

The Semantic Web

- Vision
- Architecture
- Standards

KR for the Semantic Web
ML for the Semantic Web

Conclusions
The Semantic Web


- Evolving extension of the World Wide Web (WWW) in which WWW content can be expressed not only in natural language, but also in a format that can be read and used by software agents, thus permitting them to find, share and integrate information more easily.

- Vision of the WWW as a universal medium for data, information, and knowledge exchange.
The Semantic Web: architecture

T. Berners-Lee’s invited talk at XML 2000
The Semantic Web: architecture - 2002

The Semantic Web: architecture - 2005

T. Berners-Lee’s invited talk at ISWC 2005
The Semantic Web: standards

What is an ontology?


An Ontology is a

formal specification

of a shared

conceptualization

of a domain of interest

⇒ Executable

⇒ Group of persons

⇒ About concepts

⇒ Between application and „unique truth“
**OWL (Ontology Web Language)**

- **W3C recommendation** (i.e., a standard) for Web ontologies
  - [http://www.w3.org/2004/OWL/](http://www.w3.org/2004/OWL/)
  - 10 Feb 2004

- Developed by the **W3C** WebOnt Working Group

- Mark-up language
  - compatible with RDF/XML exchange format
  - based on earlier languages OIL and DAML+OIL
OWL 2

W3C Recommendation 27 October 2009 – W3C OWL Working Group

Dr. Francesca A. Lisi
OWL 2: profiles

http://www.w3.org/TR/2009/owl2-profiles/

- Tractable ABox reasoning
- Tractable TBox reasoning
- Tractable Query answering
What is a rule?

A rule is an implication of the form

\[ \text{IF antecedent THEN consequent} \]

where the antecedent is a conjunction of conditions and the consequent is a conjunction of facts that hold when the antecedent is satisfied.
SWRL (Semantic Web Rule Language)

- Submitted to W3C for standardization in May 2004
  - http://www.w3.org/Submission/SWRL/

- Mark-up language
  - compatible with RDF/XML exchange format
  - derived by integration of OWL and RuleML
  - but undecidable!

- Never recommended, yet widely used
RIF (Rule Interchange Format)

- W3C RIF Working Group

- W3C Recommendation 22 June 2010
  - RIF Framework for Logic Dialects
  - RIF Core Dialect
  - RIF Basic Logic Dialect
  - RIF Production Rule Dialect
  - RIF Datatypes and Built-Ins 1.0
  - RIF RDF and OWL Compatibility
Overview

• The Semantic Web

• KR for the Semantic Web
  ▶ Description Logics (DLs)
  ▶ Clausal Logics (CLs)
  ▶ Hybrid DL-CL languages

• ML for the Semantic Web

• Conclusions
What are Description Logics?


- DLs are decidable variable-free fragments of First Order Logic (FOL)
  - Describe domain in terms of concepts (classes), roles (properties, relationships) and individuals
- DLs provide a family of logic based formalisms for Knowledge Representation and Reasoning (KR&R)
  - Descendants of semantic networks and KL-ONE

Dr. Francesca A. Lisi
DL Basics

- Atomic concepts
  - unary predicates/formulae with one free variable
  - E.g., Person, Doctor, HappyParent

- Atomic roles
  - binary predicates/formulae with two free variables
  - E.g., hasChild, loves

- Individuals
  - constants
  - E.g., John, Mary, Italy

- Operators (for forming complex concepts and roles from atomic ones) restricted so that:
  - Satisfiability/subsumption is decidable and, if possible, of low complexity
**ALC syntax**

<table>
<thead>
<tr>
<th>atomic concept</th>
<th>A</th>
<th>Human</th>
</tr>
</thead>
<tbody>
<tr>
<td>atomic role</td>
<td>R</td>
<td>likes</td>
</tr>
<tr>
<td>conjunction</td>
<td>C ∩ D</td>
<td>Human ∩ Male</td>
</tr>
<tr>
<td>disjunction</td>
<td>C ∪ D</td>
<td>Nice ∪ Rich</td>
</tr>
<tr>
<td>negation</td>
<td>¬C</td>
<td>¬Meat</td>
</tr>
<tr>
<td>existential restriction</td>
<td>∃R.C</td>
<td>∃hasChild.Human</td>
</tr>
<tr>
<td>value restriction</td>
<td>∀R.C</td>
<td>∀hasChild.Nice</td>
</tr>
</tbody>
</table>

E.g., person all of whose children are either Doctors or have a child who is a Doctor:

\[ \text{Person} \sqcap \forall \text{hasChild.}(\text{Doctor} \sqcap \exists \text{hasChild.Doctor}) \]
The DL Family

- $\textit{ALC}$ (Schmidt-Schauss and Smolka, 1991) is the smallest expressive DL
- $\textit{S}$ often used for $\textit{ALC}$ extended with transitive roles ($R_+$)
- Additional letters indicate other extensions, e.g.:
  - $\mathcal{H}$ for role hierarchy (e.g., hasDaughter $\sqsubseteq$ hasChild)
  - $\mathcal{O}$ for nominals/singleton classes (e.g., \{Italy\})
  - $\mathcal{I}$ for inverse roles (e.g., isChildOf $\equiv$ hasChild$^-$)
  - $\mathcal{N}$ for number restrictions (e.g., $\geq 2$ hasChild, $\leq 3$ hasChild)
  - $\mathcal{Q}$ for qualified number restrictions (e.g., $\geq 2$ hasChild.Doctor)
  - $\mathcal{F}$ for functional number restrictions (e.g., $\leq 1$ hasMother)
- $\textit{S} +$ role hierarchy ($\mathcal{H}$) + inverse ($\mathcal{I}$) + QNR ($\mathcal{Q}$) = $\textit{SHIQ}$
**DL Semantics**

### Interpretation Function $\mathcal{I}$

- **Individuals** $i^\mathcal{I} \in \Delta^\mathcal{I}$
  - John
  - Mary

- **Concepts** $C^\mathcal{I} \subseteq \Delta^\mathcal{I}$
  - Lawyer
  - Doctor
  - Vehicle

- **Roles** $r^\mathcal{I} \subseteq \Delta^\mathcal{I} \times \Delta^\mathcal{I}$
  - hasChild
  - owns

### Interpretation Domain $\Delta^\mathcal{I}$

- $(\text{Lawyer} \cap \text{Doctor})$
DL Semantics: Unique Names Assumption (UNA)


\[ a^I \neq b^I \text{ if } a \neq b \]
**Mapping DLs to FOL**

- Most DLs are decidable fragments of FOL
  - $\text{ALC}$ is a fragment of FOL with two variables (L2)

- For mapping $\text{ALC}$ to FOL introduce:
  - A unary predicate $A$ for a concept name $A$
  - A binary relation $R$ for a role name $R$

- Translate complex concepts $C$, $D$ as follows:
  - $\forall x(A) = A(x)$
  - $\forall x(C \sqcap D) = \forall x(C) \land \forall x(D)$
  - $\forall x(C \sqcup D) = \forall x(C) \lor \forall x(D)$
  - $\exists y.R(x,y) \land t_y(C) = \exists y.R(x,y) \land t_x(C)$
  - $\forall y.R(x,y) \Rightarrow t_y(C) = \forall y.R(x,y) \Rightarrow t_x(C)$
### ALC semantics

<table>
<thead>
<tr>
<th></th>
<th>Atomic Concept</th>
<th>Atomic Role</th>
<th>Conjunction</th>
<th>Disjunction</th>
<th>Negation</th>
<th>Existential Restriction</th>
<th>Value Restriction</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>atomic concept</strong></td>
<td>A</td>
<td>A^I ⊆ Δ^I</td>
<td>C ∩ D</td>
<td>C ∪ D</td>
<td>C̄</td>
<td>∃ R.C</td>
<td>∀ R.C</td>
</tr>
<tr>
<td><strong>atomic role</strong></td>
<td>R</td>
<td>R^I ⊆ Δ^I × Δ^I</td>
<td>C \ D</td>
<td>C \ D</td>
<td>Δ \ C^I</td>
<td>{x</td>
<td>∃ y. ⟨x, y⟩ ∈ R^I ∧ y ∈ C^I}</td>
</tr>
</tbody>
</table>
DL Deduction Rules

Tableau calculus

- Applies rules that correspond to DL constructors
  - E.g., John:(Person ∩ Doctor) → John:Person and John:Doctor

- Stops when no more rules applicable or clash occurs
  - Clash is an obvious contradiction, e.g., A(\(x\)), \(\neg A(\(x\))\)

- Some rules are nondeterministic (e.g., \(\sqcup\), \(\exists\))
  - In practice, this means search

- Cycle check (blocking) often needed to ensure termination
**ALC Deduction Rules**

An algorithm based on **tableau calculus** for **ALC**

- Tries to build a (tree) model $\mathcal{I}$ for input concept $C$
- Breaks down $C$ syntactically, inferring constraints on elements in $\mathcal{I}$
- Applies inference rules corresponding to **ALC** constructors (e.g. $\rightarrow \exists$)
- Works non-deterministically in PSpace
- Stops when a clash, i.e. a contradiction, occurs ($C$ is inconsistent) or no other rule can be applied ($C$ is consistent)
DL Knowledge Bases

Knowledge Base $\Sigma$

- **Terminological part**
  - Intensional knowledge
  - In the form of axioms

- **Assertional part**
  - Extensional knowledge
  - In the form of assertions

Tbox $\mathcal{T}$

Abox $\mathcal{A}$
**ALC Knowledge Bases:**

**syntax**

**Tbox**

- **equality axioms**
  - $A \equiv C$
  - $\text{Father} \equiv \text{Man} \sqcap \exists \text{hasChild} \cdot \text{Human}$

- **inclusion axioms**
  - $C \sqsubseteq D$
  - $\exists \text{favourite} \cdot \text{Brewery} \sqsubseteq \exists \text{drinks} \cdot \text{Beer}$

**ABox**

- **concept assertions**
  - $a : C$
  - $\text{john} : \text{Father}$

- **role assertions**
  - $\langle a, b \rangle : R$
  - $\langle \text{john}, \text{bill} \rangle : \text{has-child}$
Open World Assumption (OWA)

- The information in an Abox is generally considered to be incomplete (*open world*)
- An Abox represents possibly infinitely many interpretations, namely its models
- Query answering requires nontrivial reasoning
- Classical negation!
**ALC Knowledge Bases:**

**semantics**

An interpretation $\mathcal{I}_o = (\Delta^\mathcal{I}, .^\mathcal{I})$ satisfies

- an equality axiom $A \equiv C$ iff $A^\mathcal{I} \equiv C^\mathcal{I}$
- an inclusion axiom $C \subseteq D$ iff $C^\mathcal{I} \subseteq D^\mathcal{I}$
- a Tbox $\mathcal{T}$ iff $\mathcal{I}$ satisfies all axioms in $\mathcal{T}$
- a concept assertion $a : C$ iff $a^\mathcal{I} \in C^\mathcal{I}$
- a role assertion $<a, b> : R$ iff $<a^\mathcal{I}, b^\mathcal{I}> \in R^\mathcal{I}$
- a ABox $A$ iff $\mathcal{I}$ satisfies all assertions in $A$
DL-based KR&R systems

Knowledge base $\Sigma$

Reasoning services

Tbox $\mathcal{T}$

Abox $\mathcal{A}$
**DL-based KR&R systems: standard reasoning tasks**

### Subsumption
- .. of concepts C and D (C \sqsubseteq D)
  - Is $C^\mathcal{I} \subseteq D^\mathcal{I}$ in all interpretations $\mathcal{I}$?
- .. of concepts C and D w.r.t. a TBox $\mathcal{T}$ ($C \sqsubseteq^\mathcal{T} D$)
  - Is $C^\mathcal{I} \subseteq D^\mathcal{I}$ in all models $\mathcal{I}$ of $\mathcal{T}$?

### Consistency
- .. of a concept C w.r.t. a TBox $\mathcal{T}$
  - Is there a model $\mathcal{I}$ of $\mathcal{T}$ with $C^\mathcal{I} \neq \emptyset$?
- .. of a ABox $\mathcal{A}$
  - Is there a model $\mathcal{I}$ of $\mathcal{A}$?
- .. of a KB ($\mathcal{T}$, $\mathcal{A}$)
  - Is there a model $\mathcal{I}$ of both $\mathcal{T}$ and $\mathcal{A}$?
**DL-based KR&R systems: standard reasoning tasks (2)**

- Subsumption and consistency are closely related
  - $C \subseteq_T D$ iff $C \cap \neg D$ is inconsistent w.r.t. $\mathcal{T}$
  - $C$ is consistent w.r.t. $\mathcal{T}$ iff not $C \subseteq_T A \cap \neg A$

- Algorithms for checking consistency w.r.t TBoxes suffice
  - Based on tableau calculus
  - Decidability is important
  - Complexity between $P$ and $\text{ExpTime}$

**Instance check**
- .. of an individual $a$ and a concept $C$ w.r.t. a KB $\Sigma$
  - Is $a : C$ derivable from $\Sigma$? Or equivalently,
  - Is $\Sigma \cup \{a : \neg C\}$ consistent?
**ALC-based KR&R systems:**

example of instance check

\[ \Sigma = \text{DairyProduct} \sqsubseteq \text{Product}, \text{product11:}\text{DairyProduct}, \text{etc.} \]

- Is \text{product11:}\text{Product} derivable from \( \Sigma \)?
- Or equivalently
- Is \( \Sigma \cup \{ \text{product11:}\neg \text{Product} \} \) consistent?

```
\begin{align*}
\text{product11:}\neg \text{Product} & \quad \text{DairyProduct} \sqsubseteq \text{Product} \\
\text{product11:}\neg \text{DairyProduct} \sqcup \text{Product} & \quad \leftarrow \sqsubseteq \\
\text{product11:}\neg \text{DairyProduct} & \quad \leftarrow \sqcup \\
\text{product11:}\neg \text{DairyProduct} & \quad \rightarrow \\
\text{product11:}\bot & \quad \rightarrow \\
\text{product11:}\bot & \quad \rightarrow \\
\end{align*}
```
**DL-based KR&R systems: non-standard reasoning tasks**

**Most Specific Concept (MSC)**


- Intuitively, the MSC of individuals in an ABox is a concept description that represents all the properties of the individuals including the concept assertions they occur in and their relationship to other individuals.
- The existence of MSC is not guaranteed for all DLs.
  - Approximation of MSC is possible!
- However, if the MSC exists, it is uniquely determined up to equivalence.
DL-based KR&R systems: non-standard reasoning tasks (2)

Least Common Subsumer (LCS)


- The LCS of a given sequence of concept descriptions is
  - Intuitively, a concept description that represents the properties that all the elements of the sequence have in common
  - More formally, the MSC description that subsumes the given concept descriptions

- The existence of the LCS for a given sequence of concept descriptions is not guaranteed but ..

- .. if an LCS exists, then it is uniquely determined up to equivalence
### OWL DL

<table>
<thead>
<tr>
<th>Constructor</th>
<th>DL Syntax</th>
<th>Example</th>
<th>FOL Syntax</th>
</tr>
</thead>
<tbody>
<tr>
<td>intersectionOf</td>
<td>$C_1 \sqcap \ldots \sqcap C_n$</td>
<td>Human $\sqcap$ Male</td>
<td>$C_1(x) \land \ldots \land C_n(x)$</td>
</tr>
<tr>
<td>unionOf</td>
<td>$C_1 \sqcup \ldots \sqcup C_n$</td>
<td>Doctor $\sqcup$ Lawyer</td>
<td>$C_1(x) \lor \ldots \lor C_n(x)$</td>
</tr>
<tr>
<td>complementOf</td>
<td>$\neg C$</td>
<td>$\neg$ Male</td>
<td>$\neg C(x)$</td>
</tr>
<tr>
<td>oneOf</td>
<td>${x_1} \sqcup \ldots \sqcup {x_n}$</td>
<td>${john} \sqcup {mary}$</td>
<td>$x = x_1 \lor \ldots \lor x = x_n$</td>
</tr>
<tr>
<td>allValuesFrom</td>
<td>$\forall P.C$</td>
<td>$\forall$ hasChild.Doctor</td>
<td>$\forall y.P(x, y) \rightarrow C(y)$</td>
</tr>
<tr>
<td>someValuesFrom</td>
<td>$\exists P.C$</td>
<td>$\exists$ hasChild.Lawyer</td>
<td>$\exists y.P(x, y) \land C(y)$</td>
</tr>
<tr>
<td>maxCardinality</td>
<td>$\leq n P$</td>
<td>$\leq 1$ hasChild</td>
<td>$\exists y.P(x, y)$</td>
</tr>
<tr>
<td>minCardinality</td>
<td>$\geq n P$</td>
<td>$\geq 2$ hasChild</td>
<td>$\exists y.P(x, y)$</td>
</tr>
</tbody>
</table>

- $C$ is a concept (class); $P$ is a role (property); $x$ is an individual name
- XML-S datatypes as well as classes in $\forall P.C$ and $\exists P.C$
- Restricted form of DL concrete domains
**OWL DL (2)**

<table>
<thead>
<tr>
<th>OWL Syntax</th>
<th>DL Syntax</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>subClassOf</td>
<td>$C_1 \sqsubseteq C_2$</td>
<td>Human $\sqsubseteq$ Animal $\sqcap$ Biped</td>
</tr>
<tr>
<td>equivalentClass</td>
<td>$C_1 \equiv C_2$</td>
<td>Man $\equiv$ Human $\sqcap$ Male</td>
</tr>
<tr>
<td>subPropertyOf</td>
<td>$P_1 \sqsubseteq P_2$</td>
<td>hasDaughter $\sqsubseteq$ hasChild</td>
</tr>
<tr>
<td>equivalentProperty</td>
<td>$P_1 \equiv P_2$</td>
<td>cost $\equiv$ price</td>
</tr>
<tr>
<td>transitiveProperty</td>
<td>$P^+ \sqsubseteq P$</td>
<td>ancestor$^+ \sqsubseteq$ ancestor</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>OWL Syntax</th>
<th>DL Syntax</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>type</td>
<td>$a : C$</td>
<td>John $: \text{Happy-Father}$</td>
</tr>
<tr>
<td>property</td>
<td>$\langle a, b \rangle : R$</td>
<td>$\langle \text{John, Mary} \rangle : \text{has-child}$</td>
</tr>
</tbody>
</table>

- **OWL ontology** equivalent to DL KB
- **SHIQ** is the basis for **OWL**
  - OWL DL $\approx$ **SHIQ** extended with nominals (i.e., **SHOIQ**)
  - OWL Lite $\approx$ **SHIQ** with only functional restrictions (i.e., **SHIF**)

Dr. Francesca A. Lisi
**OWL DL: an example**

- European customers are customers living in European countries

- \( \text{EuropeanCustomer} \equiv \text{Customer} \cap \forall \text{livesIn.EuropeanCountry} \)

- \(<\text{owl:Class rdf:ID="EuropeanCustomer" }>
  <\text{owl:equivalentClass}>
  <\text{owl:intersectionOf rdf:parseType=" collection"}>
    <\text{owl:Class rdf:about="#Customer" }>
    <\text{owl:restriction}>
      <\text{owl:onProperty rdfResource="#livesIn" }>
      <\text{owl:allValuesFrom rdf:resource="#EuropeanCountry" }>
      <\text{owl:allValuesFrom}>
      <\text{owl:restriction}>
      <\text{owl:intersectionOf}>
      <\text{owl:equivalentClass}>
    </\text{owl:Class}>
Description Logics: 
Bibliography (only the essential)


Overview

- The Semantic Web
- **KR for the Semantic Web**
  - Description Logics (DLs)
  - Clausal Logics (CLs)
  - Hybrid DL-CL languages
- ML for the Semantic Web
- Conclusions
What is Horn Clausal Logic?

- Horn clausal logic (HCL) is the FOL fragment that contains universally quantified disjunctions of literals with at most one positive literal.
- It is at the basis of Logic Programming and Deductive Databases.
**HCL syntax**

- **Clausal language** $\mathcal{L} = \text{the set of constant, variable, functor and predicate symbols}

- **Term**: Constant / Variable / Function applied to a term
- **Atom**: Predicate applied to n terms
- **Literal**: (negated) atom
- **Horn Clause** allows the two following equivalent notations
  - $\forall X \forall Y (p(X, Y) \lor \neg q(X, a) \lor \neg r(Y, f(a)))$
  - $p(X, Y) \leftarrow q(X, a), r(Y, f(a))$

- **Definite clause (rule)**: only one literal in the head
- **Unit clause (fact)**: rule without head
**HCL Semantics**

Herbrand model theory

- **Herbrand universe** $U_H = \text{the set of all ground terms that can be formed out from the constants and function symbols in } \mathcal{L}$

- **Herbrand base** $B_H = \text{the set of all ground atoms that can be formed out from terms in } U_H \text{ and predicates in } \mathcal{L}$

- **Herbrand interpretation** $I_H = \text{subset of } B_H \text{ containing all atoms that are true in } I_H$
HCL Deduction Rules

SLD-resolution

2 opposite literals (up to a substitution) : \( l_i \theta_1 = \neg k_j \theta_2 \)

\[
(\lor l_1 \lor \ldots \lor l_i \lor \ldots \lor l_n \lor k_1 \lor \ldots \lor k_j \lor \ldots \lor k_m) \theta_1 \theta_2
\]

E.g., \( p(X) :- q(X) \) and \( q(X) :- r(X,Y) \) yield \( p(X) :- r(X,Y) \)
\( p(X) :- q(X) \) and \( q(a) \) yield \( p(a) \).

\( \checkmark \) complete by refutation!

Dr. Francesca A. Lisi
**Datalog**


- It is a function-free fragment of HCL (more precisely of definite clauses)
- It is used as logical language for relational databases
- Query answering by SLD-refutation
Deductive databases

Dr. Francesca A. Lisi
Closed World Assumption (CWA)

- The information in a database is generally considered to be complete (*closed world*)
- A database instance represents exactly one interpretation, namely the one where classes and relations in the schema are interpreted by the objects and the tuples in the instance
- Negation As Failure: what is unknown is false
**Datalog:**
example of query answering

\[ \Pi = \text{item(OrderID, ProductID)} \leftarrow \text{orderDetail(OrderID, ProductID,\_,\_,\_,\_)} \]

orderDetail(order10248, product11, ‘£14’,12,0.00)

*Etc.*

- Is item(order10248, product11) derivable from \( \Pi \)?
- Is \( \Pi \cup \{\neg \text{item(order10248, product11)}\} \) consistent?

\[ \leftarrow \text{item(order10248, product11)} \quad \text{item(OrderID, ProductID)} \leftarrow \text{orderDetail(OrderID, ProductID,\_,\_,\_,\_)} \]

\[ \{ \text{OrderID/order10248, ProductID/ product11} \} \]

\[ \leftarrow \text{orderDetail(order10248, product11,\_,\_,\_,\_)} \quad \text{orderDetail(order10248, product11, ‘£14’,12,0.00)} \]

\[ \{\} \]

Dr. Francesca A. Lisi
Clausal Logics: Bibliography (only the essential)


Overview

-The Semantic Web
-\textit{KR for the Semantic Web}
  -Description Logics (DLs)
  -Clausal Logics (CLs)
  -Hybrid DL-CL languages
-ML for the Semantic Web
-Conclusions
**DLs vs CLs**

- **Different expressive power** (Borgida, 1996)
  - No relations of arbitrary arity or arbitrary joins between relations in DLs
  - No exist. quant. in HCL

- **Different semantics** (Rosati, 2005)
  - OWA for DLs
  - CWA for HCL

- **Can they be combined?** Yes, but integration can be easily undecidable if unrestricted
Integrating DLs and CLs

Hybrid DL-HCL KR systems

- **CARIN** (Levy & Rousset, 1998)
  - Any DL+HCL
  - Unsafe
  - Decidable for some simple DL (e.g., \(\text{ALCNR}\))

- **\(\mathcal{AL}\)-log** (Donini et al., 1998)
  - \(\mathcal{ALC}\)+Datalog
  - Safe
  - Decidable

- **\(\mathcal{DL}\)+log\(^{\land}\)** (Rosati, 2006)
  - Any DL+ Datalog\(^{\land}\)
  - Weakly-safe
  - Decidable for some v.e. DL (e.g., \(\text{SHIQ}\))
The KR framework of $\mathcal{AL}$-log: syntax

\[ \mathcal{B} = \langle \Sigma, \Pi \rangle \]

$\mathcal{ALC}$ knowledge base

constrained Datalog program

constrained Datalog clauses

\[ \alpha_0 \leftarrow \alpha_1, \ldots, \alpha_m \& \gamma_1, \ldots, \gamma_n \]

where $\alpha_i$ are Datalog literals and $\gamma_j$ are constraints ($\mathcal{ALC}$ concepts from $\Sigma$ used as “typing constraints” for variables)

- item(OrderID, ProductID) $\leftarrow$ orderDetail(OrderID, ProductID,_,_,_,_)
  & OrderID:Order, ProductID:Product

- Safeness conditions:
  - Only positive Datalog literals in the body
  - Only one Datalog literal in the head
  - Constraints must refer to variables occurring in the Datalog part
  - Variables in the Datalog part can be constrained
The KR framework of $\mathcal{AL}$-log: semantics

$J = (I_\Sigma, I_H)$

- $J$ satisfies $\mathcal{B}$ iff
  - it satisfies $\Sigma$, and
  - for each clause $\alpha_0 \leftarrow \alpha_1, \ldots, \alpha_m \& \gamma_1, \ldots, \gamma_n$, for each of its ground instances $\alpha'_0 \leftarrow \alpha'_1, \ldots, \alpha'_m \& \gamma'_1, \ldots, \gamma'_n$, either there exists one $\gamma'_i$, $1 \leq i \leq n$, that is not satisfied by $J$ or $\alpha'_0 \leftarrow \alpha'_1, \ldots, \alpha'_m$ is satisfied by $J$

- OWA of $\mathcal{ALC}$ and CWA of Datalog do not interfere (safeness)
- UNA holds for $\mathcal{ALC}$ and ground Datalog

Dr. Francesca A. Lisi
The KR framework of $\mathcal{AL}$-log: reasoning

Query answering

- Atomic queries (only Datalog)
- Constrained SLD-resolution = SLD-resolution (Datalog part) + tableau calculus ($\mathcal{ALC}$ part)
  - Decidable
  - Sound and complete by refutation

- Queries are answered by constrained SLD-refutation
  - For each ground instance $Q'$ of the query $Q$,
  - Collect the set of all constrained SLD-derivations $d_1, d_2, \ldots, d_m$ of bounded length (with $d_i = Q_{i_0} \ldots Q_{i_n}$) for $Q'$ in $\Sigma$
  - Then check whether $\Sigma \vdash \text{disj}(Q_{1_{n_1}}, \ldots, Q_{m_{n_m}})$
The KR framework of $AL$-log: example of query answering

Assuming that this is the only SLD-derivation for the query, the existential entailment problem boils down to prove that

$\Sigma \cup \{ \text{order10248:}\neg\text{Order, product11:}\neg\text{Product} \}$

is unsatisfiable!
The KR framework of CARIN: syntax and semantics

- $\Sigma$ is based on any DL (but good results for $ALCNR$)
- $\Pi$ contain Horn rules, i.e. definite clauses, where DL literals:
  - can be built from either concept or role predicates
  - are allowed in rule heads

- The semantics naturally follows as in $AL$-log
The KR framework of CARIN: reasoning

Query answering

- Atomic queries (built from either concept, role or ordinary predicates)
- Constrained SLD-resolution = SLD-resolution (HCL part) + tableau calculus (DL part)
  - Complete by refutation for non-recursive CARIN-\(\text{ALCNR}\)
  - Decidable for the non-recursive case
  - Undecidable for the recursive case, unless weaken the DL part or impose rules to be role-safe
The KR framework of $\text{DL} + \text{log}^{-\forall}$: syntax

$\text{DL} + \text{log} \ \text{KB} = \text{DL KB extended with Datalog}^{-\forall} \ \text{rules}$

\[
p_1(x_1) \vee ... \vee p_n(x_n) \leftarrow r_1(y_1), ..., r_m(y_m), s_1(z_1), ..., s_k(z_k), \neg u_1(w_1), ..., \neg u_h(w_h)
\]

satisfying the following properties

- **Datalog safeness:** every variable occurring in a rule must appear in at least one of the atoms $r_1(y_1), ..., r_m(y_m), s_1(z_1), ..., s_k(z_k)$
- **DL weak safeness:** every head variable of a rule must appear in at least one of the atoms $r_1(y_1), ..., r_m(y_m)$
The KR framework of $DL+\log \neg \vee$ semantics

- FOL-semantics
  - OWA for both DL and Datalog predicates

- NM-semantics: extends stable model semantics of Datalog$\neg \vee$
  - OWA for DL-predicates
  - CWA for Datalog-predicates

- In both semantics, entailment can be reduced to satisfiability

- In Datalog$\neg \vee$, FOL-semantics equivalent to NM-semantics
The KR framework of $DL+log$:

reasoning

- CQ answering can be reduced to satisfiability
- NM-satisfiability of $DL+log$ KBs combines
  - **Consistency in Datalog**: A Datalog program is consistent if it has a stable model
  - **Boolean CQ/UCQ containment problem in DLs**: Given a DL-TBox $T$, a Boolean CQ $Q_1$ and a Boolean UCQ $Q_2$ over the alphabet of concept and role names, $Q_1$ is contained in $Q_2$ wrt $T$, denoted by $T |= Q_1 \subseteq Q_2$, iff, for every model $I$ of $T$, if $Q_1$ is satisfied in $I$ then $Q_2$ is satisfied in $I$.

- The decidability of reasoning in $DL+log$ depends on the decidability of the Boolean CQ/UCQ containment problem in DL
Hybrid DL-HCL KR&R Systems: Bibliography

Hybrid DL-HCL KR&R Systems: Bibliography (2)

Hybrid DL-HCL KR&R Systems: Bibliography (3)


Overview

- The Semantic Web
- KR for the Semantic Web
- **ML for the Semantic Web**
  - ML in CLs
  - ML with DLs
  - ML with hybrid DL-CL languages
- Conclusions
Machine Learning


- It aims at building computer programs able to learn
  - A computer program learns if it improves its performance at some task through experience (Mitchell, 1997)
  - More formally: A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks in T as measured by P, improves with experience E (Mitchell, 1997).
  - More generally: Any change in a system that allows it to perform better the second time on repetition of the same task or on task drawn from the same population (Simon, 1983).

- It relies on several inferences but notably on induction
What is induction?

Induction or inductive reasoning, sometimes called inductive logic, is the process of reasoning in which the premises of an argument are believed to support the conclusion but do not ensure it. It is used to ascribe properties or relations to types based on tokens (i.e., on one or a small number of observations or experiences); or to formulate laws based on limited observations of recurring phenomenal patterns.
What is induction? (2)

This tomato is red

This tomato is also red

Distinguish:

- weak induction: all observed tomatoes are red
- strong induction: all tomatoes are red

All tomatoes are red
What is induction? (3)

- Weak induction: conclusion is entailed by (follows deductively from) observations
  - cannot be wrong

- Strong induction: conclusion does not follow deductively from observations
  - could be wrong!
  - logic does not provide justification
  - probability theory may
What is induction? (4)

**Induction**
(generalise from observed facts)

\[
\text{Human(Socrates)} \quad \rightarrow \quad \text{Mortal(Socrates)}
\]

\[
\text{Mortal(x)} \quad \leftrightarrow \quad \text{Human(x)}
\]

**Deduction**

\[
\text{Human(Socrates)} \quad \rightarrow \quad \text{Mortal(Socrates)}
\]

\[
\text{Mortal(x)} \quad \leftrightarrow \quad \text{Human(x)}
\]
Generalization

- (Inductive) Generalization is a type of induction.
- It proceeds from a premise about a sample to a conclusion about the population.
  - The proportion Q of the sample has attribute A.
  - Therefore
    - The proportion Q of the population has attribute A.
- How great the support which the premises provide for the conclusion is dependent on (a) the number of individuals in the sample group compared to the number in the population; and (b) the randomness of the sample. The hasty generalization and biased sample are fallacies related to generalization.
Inductive Learning

- **Inductive Learning Hypothesis** Any hypothesis found to approximate the target function well over a sufficiently large set of training examples will also approximate the target function well over other unobserved examples.

- **Concept Learning** is a case of Inductive Learning where inductive generalization plays a key role
  - Acquiring the definition of a general category given a sample of positive and negative examples of the category
Concept learning

Given:
- an instance space
- some unknown concept = subset of instance space

Task: learn concept definition from examples (= labelled instances)
- Could be defined extensionally or intensionally
- Usually interested in intensional definition
  - otherwise no generalisation possible
Concept learning

- Hypothesis $h = \text{concept definition}$
  - can be represented intensionally: $h$
  - or extensionally (as set of examples): $\text{ext}(h)$
- Hypothesis $h$ covers example $e$ iff $e \in \text{ext}(h)$
- Given a set of (positive and negative) examples $E = \langle E^+, E^- \rangle$, $h$ is consistent with $E$ if $E^+ \subseteq \text{ext}(h)$ and $\text{ext}(h) \cap E^- = \emptyset$
Version spaces

Given a set of instances $E$ and a hypothesis space $H$, the version space is the set of all $h \in H$ consistent with $E$

- contains all hypotheses in $H$ that might be the correct target concept

Some inductive algorithms exist that, given $H$ and $E$, compute the version space $VS(H,E)$
Version spaces:
Properties

- If target concept \( c \in H \), and \( E \) contains no noise, then \( c \in \text{VS}(H,E) \)
  - If \( \text{VS}(H,E) \) is singleton: one solution
  - Usually multiple solutions

- If \( H = 2^I \) with \( I \) instance space:
  - i.e., all possible concepts in \( H \)
  - then: no generalisation possible
  - \( H \) is called *inductive bias*
**Version spaces:**

*Example*


- Usually illustrated with conjunctive concept definitions

<table>
<thead>
<tr>
<th>Sky</th>
<th>AirTemp</th>
<th>Humidity</th>
<th>Wind</th>
<th>Water</th>
<th>Forecast</th>
<th>EnjoySport</th>
</tr>
</thead>
<tbody>
<tr>
<td>sunny</td>
<td>warm</td>
<td>normal</td>
<td>strong</td>
<td>warm</td>
<td>same</td>
<td>yes</td>
</tr>
<tr>
<td>…</td>
<td>…</td>
<td>…</td>
<td>…</td>
<td>…</td>
<td>…</td>
<td>…</td>
</tr>
</tbody>
</table>

- Concept represented as if-then-rule:
  - \(<\text{Sunny},\text{Warm},?,?,?,?>\)
  - IF Sky=sunny AND AirTemp=warm THEN EnjoySports=yes
Version spaces:
Example

\(<?,?,?,?,?,?,?>\)

\(<\text{Sunny},?,?,?,?,?,?>\)

\(<?,\text{Warm},?,?,?,?>\) ... \(<?,?,?,?,?,?,?\text{,Same}>\)

... ...

... ...

... ...

... ...

... ...

... ...

... ...

... ...

... ...

... ...

... ...

\(<\text{Sunny, Warm, Normal, Strong, Warm, Same}>\)

... ...

\(<\emptyset, \emptyset, \emptyset, \emptyset, \emptyset, \emptyset, \emptyset>\)
Version spaces: 
The importance of generalization

- Generality is a notion central to version space algorithms
  - $h$ is more general than $h'$ ( $h \geq h'$ ) iff $\text{ext}(h') \subseteq \text{ext}(h)$

- Properties of VS($H,E$) w.r.t. generality:
  - if $s \in \text{VS}(H,E)$, $g \in \text{VS}(H,E)$ and $g \geq h \geq s$, then $h \in \text{VS}(H,E)$
  - $\Rightarrow$ VS can be represented by its borders

- Even when not VS itself, but only one element of it is computed, generality can be used for search
  - properties allow to prune search space
    - if $h$ covers negatives, then any $g \geq h$ also covers negatives
    - if $h$ does not cover some positives, then any $s \leq h$ does not cover those positives either
**Generalization as search**


- *Generalization as search* through a partially ordered space of hypotheses
- The goal of this search is to find the hypothesis that best fits the training examples
- The hypothesis language biases the hypothesis space
  - The more *expressive* the hypothesis language is, the more *structured* the hypothesis space must be
Generalization as search (2)

Instances $X$

- $x_1 = \langle \text{Sunny. Warm. High. Strong. Cool. Same} \rangle$
- $x_2 = \langle \text{Sunny. Warm. High. Light. Warm. Same} \rangle$

Hypotheses $H$

- $h_1 = \langle \text{Sunny. ?. ?. Strong. ?. ?} \rangle$
- $h_2 = \langle \text{Sunny. ?. ?. ?. ?} \rangle$
- $h_3 = \langle \text{Sunny. ?. ?. ?. Cool. ?} \rangle$
Generalization as search (3)

Instances $X$

Hypotheses $H$

$x_1 = \langle \text{Sunny Warm Normal Strong Warm Same} \rangle, +$
$x_2 = \langle \text{Sunny Warm High Strong Warm Same} \rangle, +$
$x_3 = \langle \text{Rainy Cold High Strong Warm Change} \rangle, -$ (or $-$)
$x_4 = \langle \text{Sunny Warm High Strong Cool Change} \rangle, +$

$h_0 = \langle \emptyset, \emptyset, \emptyset, \emptyset, \emptyset \rangle$
$h_1 = \langle \text{Sunny Warm Normal Strong Warm Same} \rangle$
$h_2 = \langle \text{Sunny Warm ? Strong Warm Same} \rangle$
$h_3 = \langle \text{Sunny Warm ? Strong Warm Same} \rangle$
$h_4 = \langle \text{Sunny Warm ? Strong ? ?} \rangle$
Machine Learning: bibliography (only the essential)

Overview

- The Semantic Web
- KR for the Semantic Web
- *ML for the Semantic Web*
  - ML in CLs
  - ML with DLs
  - ML with hybrid DL-CL languages
- Conclusions
Inductive Logic Programming


- Induction of rules from examples and background knowledge within the representation framework of HCL (Muggleton, 1990)
- Scope of induction: discrimination
- Class of tasks: prediction
ILP Example: “Bongard problems”

- Simplified version of Bongard problems used as benchmarks in ILP
  - Bongard: a Russian scientist studying pattern recognition
  - Bongard problem: Given some pictures, find patterns in them

- E.g. we want to find a set of hypotheses (clausal theory) that is complete and consistent with the following set of (positive and negative) examples
  - Complete=covers all positive examples
  - Consistent=covers no negative example
Negative examples

neg(ex1):- contains(ex1,o1),contains(o1,o2),triangle(o1),
points(o1,down),square(o2).

Positive examples

pos(ex2):- contains(ex2,o3),contains(o3,o4),triangle(o3),
points(o3,down),square(o4), contains(ex2,o5),
contains(o5,o6), circle(o5),triangle(o6), points(o6,up).

pos(X):- contains(X,O1),contains(O1,O2),
triangle(O1), points(O1,down),square(O2)?
Negative examples

Positive examples

\( \text{pos}(X) :\neg \, \text{contains}(X,O1), \text{contains}(O1,O2), \text{circle}(O1), \text{triangle}(O2), \text{points}(O2, \text{up})? \)
Induction in ILP

Induction as inverted deduction

INDUCTION

DEDUCTION

Facts
events
observations

theories
rules
models

Dr. Francesca A. Lisi
Inverse resolution


 Resolution implements \( |- \) for clausal theories

 Inverting it allows to generalize a clausal theory

 Pros:

 - In principle very powerful

 Cons:

 - Gives rise to huge search space
 - Returns not unique results

 E.g., \( \text{father}(j,p):-\text{male}(j) \) and \( \text{parent}(j,p) \) yields \( \text{father}(j,p):-\text{male}(j),\text{parent}(j,p) \)

 or \( \text{father}(X,Y):-\text{male}(X),\text{parent}(X,Y) \) or ...

 Need for a ordered hypothesis space
**Induction in ILP (2)**

**Induction as generalization**

- Exploits results obtained in Concept Learning
- Bunch of techniques for:
  - Structuring
    - Generality orders
  - Searching
    - Refinement operators
  - Bounding
    - Declarative bias

the space of hypotheses when the hypothesis language is defined over HCL
Generality orders in ILP: \( \theta \)-subsumption


\( \theta \)-subsumption implements |- for single clauses

\( C_1 \theta \)-subsumes \( C_2 \) (denoted \( C_1 \leq \theta C_2 \)) if and only if there exists a variable substitution \( \theta \) such that \( C_1 \theta \subseteq C_2 \)

- to check this, first write clauses as disjunctions
  \[ a, b, c \leftarrow d, e, f \iff a \lor b \lor c \lor \neg d \lor \neg e \lor \neg f \]
  - then try to replace variables with constants or other variables

- Most often used in ILP
Generality orders in ILP: θ-subsumption (2)

Example:

\( \text{\textgreater c}_1 = \text{father}(X,Y) : - \text{parent}(X,Y) \)
\( \text{\textgreater c}_2 = \text{father}(X,Y) : - \text{parent}(X,Y), \text{male}(X) \)

\( \text{\textgreater for } \theta = \{\} : \text{c}_1 \theta \subseteq \text{c}_2 \Rightarrow \text{c}_1 \theta\text{-subsumes c}_2 \)

\( \text{\textgreater c}_3 = \text{father}(\text{luc},Y) : - \text{parent}(\text{luc},Y) \)

\( \text{\textgreater for } \theta = \{X/\text{luc}\} : \text{c}_1 \theta = \text{c}_3 \Rightarrow \text{c}_1 \theta\text{-subsumes c}_3 \)

\( \text{\textgreater c}_2 \text{ and c}_3 \text{ do not } \theta\text{-subsume one another} \)
Generality orders in ILP: \(\theta\)-subsumption (3)

Another (slightly more complicated) example:

\[ \uparrow c1 = p(X,Y) :- q(X,Y) \]
\[ \uparrow c2 = p(X,Y) :- q(X,Y), q(Y,X) \]
\[ \uparrow c3 = p(Z,Z) :- q(Z,Z) \]
\[ \uparrow c4 = p(a,a) :- q(a,a) \]

Which clauses \(\theta\)-subsumed by which?
Generality orders in ILP: θ-subsumption

Logical properties

- Sound: if $c_1$ θ-subsumes $c_2$ then $c_1 \models c_2$
- Incomplete: possibly $c_1 \models c_2$ without $c_1$ θ-subsuming $c_2$ (but only for recursive clauses)
  - $\downarrow c_1 : p(f(X)) :- p(X)$
  - $\downarrow c_2 : p(f(f(X))) :- p(X)$

- Checking θ-subsumption is decidable but NP-complete
Generality orders in ILP: 
\( \Theta \)-subsumption

**Algebraic properties**

- It is a semi-order relation
  - I.e. transitive and reflexive, not anti-symmetric
- It generates equivalence classes
  - equivalence class: \( c_1 \sim c_2 \) iff \( c_1 \leq_\Theta c_2 \) and \( c_2 \leq_\Theta c_1 \)
  - \( c_1 \) and \( c_2 \) are then called *syntactic variants*
  - \( c_1 \) is *reduced clause* of \( c_2 \) iff \( c_1 \) contains minimal subset of literals of \( c_2 \) that is still equivalent with \( c_2 \)
  - each equivalence class represented by its reduced clause
Generality orders in ILP: 
$\theta$-subsumption

**Algebraic properties (cont.)**

- It generates a partial order on those equivalence classes
  - If $c_1$ and $c_2$ in different equivalence classes, either $c_1 \leq_\theta c_2$ or $c_2 \leq_\theta c_1$ or neither $\Rightarrow$ anti-symmetry $\Rightarrow$ partial order

- Thus, reduced clauses form a lattice
  - Least/greatest upper/lower bound of two clauses always exists and is unique
  - Infinite chains $c_1 \leq_\theta c_2 \leq_\theta c_3 \leq_\theta \ldots \leq_\theta c$ exist

- Looking for good hypothesis $\Rightarrow$ traversing this lattice
**Generality orders in ILP: relative subsumption**


- Given two clauses $C_1$ and $C_2$ and a clausal theory $B$, $C_1$ *subsumes* $C_2$ *relative to* $B$ (denoted by $C_1 \geq_B C_2$) if and only if there exists a variable substitution $\theta$ for $C_1$ such that $B |= \forall (C_1 \theta \Rightarrow C_2)$

- Used in ILP if $B$ is a set of facts

- Semantic notion of generality
Generality orders in ILP: generalized subsumption


- $\mathcal{B}$ background knowledge
- $C_1, C_2$ two definite clauses
- $\sigma$ a Skolem substitution for $C_2$ w.r.t. $\{C_1\} \cup \mathcal{B}$

$C_1 \geq_\mathcal{B} C_2$ iff there exists a substitution $\theta$ for $C_1$ such that

- $\text{head}(C_1)\theta = \text{head}(C_2)$
- $\mathcal{B} \cup \text{body}(C_2)\sigma \vdash \text{body}(C_1)\theta\sigma$
- $\text{body}(C_1)\theta\sigma$ is ground.
Generality orders in ILP: generalized subsumption (2)

- Background knowledge $B$
  - pet(X):-cat(X)
  - pet(X):-dog(X)
  - small(X):-cat(X)

- Clauses:
  - $C_1 = \text{cuddlypet}(X) :- \text{small}(X), \text{pet}(X)$
  - $C_2 = \text{cuddlypet}(X) :- \text{cat}(X)$

- Semantic generality!!
  - $C_1 \geq_B C_2$
  - $\theta$- subsumption fails
Refinement operators

Heuristics-based searches (greedy, beam, exhaustive…)

VS
Refinement operators: properties

- How to traverse hypothesis space so that
  - no hypotheses are generated more than once?
  - no hypotheses are skipped?

- Properties of refinement operators
  - globally complete: each point in lattice is reachable from top
  - locally complete: each point directly below \( c \) is in \( \rho(c) \) (useful for greedy systems)
  - optimal: no point in lattice is reached twice (useful for exhaustive systems)
  - minimal, proper, ...
Refinement operators: lgg


- **Bottom-up search in clausal spaces**
  - Starts from 2 clauses and compute least general generalisation (lgg)
  
  - i.e., given 2 clauses, return most specific single clause that is more general than both of them

- We shall consider only the case of clausal spaces ordered according to $\theta$-subsumption
  - lgg under $\theta$-subsumption
Refinement operators: \textit{lgg}

\section*{Definition of \textit{lgg} of terms:}
\begin{itemize}
\item $\text{lgg}(f(s_1,\ldots,s_n), f(t_1,\ldots,t_n)) = f(\text{lgg}(s_1,t_1),\ldots,\text{lgg}(s_n,t_n))$
\item $\text{lgg}(f(s_1,\ldots,s_n), g(t_1,\ldots,t_n)) = V$
\end{itemize}
\textit{e.g.:} $\text{lgg}(a,b) = X; \text{lgg}(f(X),g(Y)) = Z; \text{lgg}(f(a,b,a),f(c,c,c))=f(X,Y,X); \ldots$

\section*{Definition of \textit{lgg} of literals:}
\begin{itemize}
\item $\text{lgg}(p(s_1,\ldots,s_n), p(t_1,\ldots,t_n)) = p(\text{lgg}(s_1,t_1),\ldots,\text{lgg}(s_n,t_n))$
\item $\text{lgg}(\neg p(\ldots), \neg p(\ldots)) = \neg \text{lgg}(p(\ldots),p(\ldots))$
\item $\text{lgg}(p(s_1,\ldots,s_n), q(t_1,\ldots,t_n))$ is undefined
\item $\text{lgg}(p(\ldots), \neg p(\ldots))$ and $\text{lgg}(\neg p(\ldots),p(\ldots))$ are undefined
\end{itemize}

\section*{Definition of \textit{lgg} of clauses:}
\begin{itemize}
\item $\text{lgg}(c_1,c_2) = \{\text{lgg}(l_1, l_2) \mid l_1 \in c_1, l_2 \in c_2 \text{ and } \text{lgg}(l_1,l_2) \text{ defined}\}$
Refinement operators: \textit{lgg}

Example:

\( \blacktriangleright f(t,a) : - p(t,a), m(t), f(a) \)
\( \blacktriangleright f(j,p) : - p(j,p), m(j), m(p) \)
\( \blacktriangleright lgg = f(X,Y) : - p(X,Y), m(X), m(Z) \)
**Refinement operators: relative lgg**


- relative to "background theory" B
  - assume B is a set of facts
- \( r\text{lgg}(e_1, e_2) = \text{lgg}(e_1 :- B, e_2 :- B) \)
- method to compute:
  - change facts into clauses with body B
  - compute lgg of clauses
  - remove B, reduce

Used in in the ILP system Golem (Muggleton & Feng)
Refinement operators: example

Given the following 2 simple Bongard configurations, find least general clause that would predict both to be positive

1. pos(1).
   contains(1,o1).
   contains(1,o2).
   triangle(o1).
   points(o1,down).
   circle(o2).

2. pos(2).
   contains(2,o3).
   contains(2,o3).
   triangle(o3).
   points(o3,down).
Refinement operators: example

Method 1: represent example by clause; compute lgg of examples

\[
\text{pos}(1) : - \text{contains}(1,o1), \text{contains}(1,o2), \text{triangle}(o1), \\
\quad \text{points}(o1,\text{down}), \text{circle}(o2). \\
\text{pos}(2) : - \text{contains}(2,o3), \text{triangle}(o3), \text{points}(o3,\text{down}).
\]

\[
\text{lgg}( \\
(\text{pos}(1) : - \text{contains}(1,o1), \text{contains}(1,o2), \text{triangle}(o1), \\
\quad \text{points}(o1,\text{down}), \text{circle}(o2)) , \\
(\text{pos}(2) : - \text{contains}(2,o3), \text{triangle}(o3), \text{points}(o3, \text{down})) ) \\
= \text{pos}(X) : - \text{contains}(X,Y), \text{triangle}(Y), \text{points}(Y,\text{down})
\]
Refinement operators: example

Method 2: represent class of example by fact, other properties in background; compute rlgg

Examples:
- pos(1).
- pos(2).

Background:
- contains(1,o1).
- contains(2,o3).
- contains(1,o2).
- triangle(o1).
- triangle(o3).
- points(o1,down).
- points(o3,down).
- circle(o2).

rlgg(pos(1), pos(2)) = ? (exercise)
Refinement operators: Shapiro’s specialization operator


*Top down search in clausal spaces ordered according to theta-subsumption:*

- $\rho(c)$ yields set of refinements of $c$
- Theory: $\rho(c) = \{c' \mid c' \text{ is a maximally general specialisation of } c\}$
- Practice: $\rho(c) \subseteq \{c \cup \{l\} \mid l \text{ is a literal}\} \cup \{c\theta \mid \theta \text{ is a substitution}\}$

*Used in many ILP systems*
**Refinement operators:**
**Shapiro’s specialization operator**

```
daughter(X, Y)

daughter(X, X)

......

daughter(X, Y) :- female(X)
daughter(X, Y) :- parent(Y, X)

daughter(X, Y) :- female(X), female(Y)
daughter(X, Y) :- female(X), parent(Y, X)
```

```
daughter(X, Y) :- parent(X, Z)
```

```
...```

Dr. Francesca A. Lisi
Declarative bias


- **Language bias**
  - Specifies and restricts the set of clauses or theories that are permitted (language of hypotheses)

- **Search bias**
  - Concerns the way the system searches through the hypothesis space

- **Validation bias**
  - Determines when the learned theory is acceptable, so when the learning process may stop.
ILP logical settings


Orthogonality of the following two dimensions

Scope of induction
- discriminant vs. characteristic induction

Representation of the observations
- learning from implications vs. learning from interpretations

leads to 4 different logical settings for ILP
ILP logical settings: Predictive vs Descriptive ILP

Prediction

Description
ILP logical settings: Learning from entailment

1 example = a fact $e$ (or clause $e:\neg B$)

Goal:

- Given examples $<E^+,E^->$,
- Find theory $H$ such that
  - $\forall e^+ \in E^+: B \land H \vdash e^+$
  - $\forall e^- \in E^-: B \land H \not\vdash e^-$
ILP logical settings: Learning from entailment (2)

Examples:

\[
\text{pos(1).} \\
\text{pos(2).} \\
\text{:- pos(3).}
\]

Background:

contains(1,o1). \\
contains(1,o2). \\
contains(2,o3). \\
triangle(o1). \\
triangle(o3). \\
poins(o1,down). \\
poins(o3,down). \\
circle(o2). \\
contains(3,o4). \\
circle(o4). \\

\[
\text{pos(X) :- contains(X,Y), triangle(Y), points(Y,down).}
\]
ILP logical settings: Learning from interpretations

- Example = interpretation (set of facts) $e$
  - contains a full description of the example
  - all information that intuitively belongs to the example, is represented in the example, not in background knowledge

- Background = domain knowledge
  - general information concerning the domain, not concerning specific examples
ILP logical settings: Learning from interpretations (2)

Examples:

- pos: \{\text{contains}(o1), \text{contains}(o2), \text{triangle}(o1),\}
  \text{points}(o1,\text{down}), \text{circle}(o2)\}
- pos: \{\text{contains}(o3), \text{triangle}(o3), \text{points}(o3,\text{down})\}
- neg: \{\text{contains}(o4), \text{circle}(o4)\}

Background:

- polygon(X) :- \text{triangle}(X).
- polygon(X) :- \text{square}(X).

\exists Y: \text{contains}(Y), \text{triangle}(Y), \text{points}(Y, \text{down}).

Closed World Assumption made inside interpretations

constraint on pos
ILP logical settings:
Learning from interpretations (3)

Note: when learning from interpretations
1. can dispose of "example identifier"
   - but can also use standard format
2. CWA made for example description
   - i.e., example description is assumed to be complete
3. class of example related to information inside example + background information, NOT to information in other examples

Because of 3rd property, more limited than learning from entailment
- cannot learn relations between different examples, nor recursive clauses

... but also more efficient because of 2nd and 3rd property
- positive PAC-learnability results (De Raedt and Džeroski, 1994, AIJ) vs. negative results for learning from entailment
Inductive Logic Programming: bibliography (only the essential)


Overview

- The Semantic Web
- KR for the Semantic Web
- ML for the Semantic Web
  - ML in CLs
  - ML with DLs
  - ML with hybrid DL-CL languages
- Conclusions
Learning in DLs

- Logic Programming
- ILP
- Machine Learning
- FOL
- DLs

Dr. Francesca A. Lisi
Learnability of DLs


- Learnability of sublanguages of CLASSIC w.r.t. the PAC learning model
- LCS used as a means for inductive learning from examples assumed to be concept descriptions
Learning in CLASSIC


- Learning task: supervised
  - Classified examples: ABox individuals
  - Goal: induce new concepts to be added to the TBox
- Search direction: bottom-up
- Algorithm: **LCSLearn/LCSLearnDISJ**
  1. Apply the MSC operator to compute the minimal Tbox generalizations of the examples
  2. Apply the LCS operator to generalize the MSC descriptions of examples
- Limits: overly specific concept definitions
Learning in BACK


- Learning task: unsupervised
  - Unclassified examples: ABox individuals
  - Goal: induce new concepts to be added to the TBox
- Search direction: bottom-up
- Algorithm: KLUSTER
  1. Cluster the ABox individuals into $n$ mutually disjoint concepts so that $n$ supervised learning problems are obtained
  2. Find a correct definition of each of these concepts as follows:
     1. Compute and evaluate the *most specific generalization* (MSG) of a concept by applying the MSC operator;
     2. Obtain the *most general discrimination* (MGD) of the concept by further generalizing the MSG.
**Refinement operators for DLs**


- Complete and proper refinement operator for \textit{ALER}
- No minimal refinement operators exist for \textit{ALER}
  - Minimality of all refinement steps can be achieved except for those introducing
- Complete refinement operators for \textit{ALER} can not be locally finite
- An upward refinement operator can be obtained by inverting the arrows in the refinement rules of the downward one
Refinement operators for DLs (2)


- Let $\mathcal{L}$ be a DL which allows to express $\top$, $\bot$, $\cap$, $\cup$, $\exists$ and $\forall$
  - E.g. $\mathcal{ALC}$
- Maximal sets of properties of $\mathcal{L}$ refinement operators
  1. {Weakly complete, complete, finite}
  2. {Weakly complete, complete, proper}
  3. {Weakly complete, non-redundant, finite}
  4. {Weakly complete, non-redundant, proper}
  5. {Non-redundant, finite, proper}

- Application: learning in $\mathcal{ALC}$ (Lehmann & Hitzler, 2007a)
Learning in $\text{ALC}$


- **Learning task**: supervised
  - Classified examples: ABox individuals
  - Goal: find a correct Tbox concept definition
- **Search direction**: bottom-up/top-down
- **Algorithm**: YinYang
  1. Apply the MSC operator to compute the minimal Tbox generalizations of the examples
  2. Apply *downward and upward refinement operators* for $\text{ALC}$ to converge towards a correct concept definition
- **Implementation**: [http://www.di.uniba.it/~iannone/yinyang/](http://www.di.uniba.it/~iannone/yinyang/)
Learning in $\mathbb{ALC}(2)$


- Learning task: Unsupervised
  - Unclassified examples: ABox individuals
  - Goal: induce new concepts to be added to the TBox
- Search direction: bottom-up/top-down
- Algorithm: CSKA
  1. Cluster the ABox individuals into mutually disjoint concepts (see KLUSTER)
  2. For each of these concepts find a correct concept definition by applying downward and upward refinement operators for $\mathbb{ALC}$ (see Yin/Yang)
Learning in $\text{ALC}(3)$


- **Learning task**: supervised
  - Classified examples: ABox individuals
  - Goal: find a correct Tbox concept definition
- **Search direction**: top-down
- **Algorithm**: DL-Learner
  - Implements a genetic programming procedure based on refinement operators for ALC whose fitness is computed on the grounds of the covered instances
- **Implementation**: [http://aksw.org/Projects/DLLearner](http://aksw.org/Projects/DLLearner)
Learning in OWL DL

N. Fanizzi, C. d'Amato, F. Esposito (2008a): DL-FOIL: Concept Learning in Description Logics. ILP 2008: 107-121

- Learning task: supervised
  - Classified examples: ABox individuals
  - Goal: find a correct Tbox concept definition
- Search direction: top-down
- Algorithm: DL-FOIL
  - Adapts FOIL to learning in OWL DL
  - Implements a downward refinement operator for DLs
  - Extends the gain function to deal with incomplete examples
- Implementation: upon request
**kNN in DLs**


- **Algorithm: kNN-DL**
  - Instance-based learning system
  - Based on structural/semantic *(dis)similarity* measures

N. Fanizzi, C. d'Amato, F. Esposito. *Instance Based Retrieval by Analogy*. SAC 2007 SDRC Track, 11-15 March 2007, Seoul, Korea

- **Algorithm: DiVS-kNN**
  - Instance-based learning system
  - Based on *disjunctive version space*
**Kernels in DLs**


- **Task**: classification
- **From distances to kernels**
  - Kernel is a similarity measure (can be obtained from distances)
  - Kernel machine = algorithm parameterized by kernels
Learning in DLs: bibliography

Learning in DLs: bibliography (2)


Learning in DLs: bibliography (3)

Learning in DLs: bibliography (4)

Learning in DLs: bibliography (5)

Overview

- The Semantic Web
- KR for the Semantic Web
- *ML for the Semantic Web*
  - ML in CLs
  - ML with DLs
  - ML with hybrid DL-CL languages
- Conclusions
ILP and DL-HCL hybridization
ILP and DL-HCL hybridization: Ontologies as BK

Defining rules is a demanding task

Inducing rules on top of ontologies

≈

Inducing rules by having ontologies as prior knowledge

Machine Learning can partially automate this task
Learning in CARIN-\textsc{ALN}


- **Scope of induction:** prediction
- **Logical setting:** learning from interpretations
- **Language of hypotheses:** definite clauses in CARIN-\textsc{ALN}
- **Generality order:** adaptation of Buntine’s generalized subsumption to CARIN-\textsc{ALN}
- **Coverage relations:** query answering in CARIN-\textsc{ALN}
Learning in CARIN-$\mathcal{ALN}$ (2)


- Method for transforming CARIN-$\mathcal{ALN}$ into Datalog extended with numerical constraints
- Transfer of learnability results known for ILP to learning in CARIN-$\mathcal{ALN}$
**Learning in $\mathcal{AL}$-log**


CoRR abs/0711.1814

- **Scope of induction:** prediction/description
- **Logical setting:** learning from interpretations/learning from implications
- **Language of hypotheses:** constrained Datalog clauses
- **Generality order:** adaptation of Buntine’s generalized subsumption to $\mathcal{AL}$-log
- **Coverage relations:** query answering in $\mathcal{AL}$-log
Learning in AL-log: task of frequent pattern discovery

- **Scope of induction:** description

- **KR framework:** AL-log

- **Generality order:** adaptation of Buntine's *generalized subsumption* to AL-log

- **Algorithm:** upgrade of WARMR (Dehaspe & Toivonen, 1999)

- **Application:** induction of multi-grained descriptions of the individuals of a reference ontology concept wrt other ontology concepts and a relational database
Learning in \textit{AL-log}: task of frequent pattern discovery (2)

- **Data set:** facts from the on-line CIA World Fact Book
- **Taxonomy:** about countries, religions and languages
- **Thresholds:** \( \text{minsup}^1 = 20\%; \text{minsup}^2 = 13\%; \text{minsup}^3 = 10\% \)
- **Goal:** find frequent patterns describing \((C_{\text{ref}})\) Middle East countries w.r.t. \((C_{\text{trel}})\) 's religions and languages at \((\text{maxG})\) 3 granularity levels

- **Patterns:**
  - Example for the level \(l=2\) of description granularity:
    \[ q(A) \leftarrow \text{speaks}(A,B) \& A: \text{MiddleEastCountry}, B: \text{AfroAsiaticLanguage} \quad \text{-- Support: 20\%} \]
    
    "20\% Middle East countries speak an Afro-Asiatic language"

  - Example for the level \(l=3\) of description granularity:
    \[ q(A) \leftarrow \text{speaks}(A,B), \text{believes}(A,C) \& A: \text{MiddleEastCountry}, B: \text{ArabicLanguage}, C: \text{MuslimReligion} \quad \text{-- Support: 13.3\%} \]
    
    "13.3\% Middle East countries speak Arabic and believe Islam"
Learning in AL-log: task of concept formation

Concept Formation = Clustering + Characterization

- **Assumption**: frequent patterns as clues of data clusters
- **Two-phased method**:
  1. detect *emerging* concepts (known ext, unknown int)
  2. turn emerging concepts into *fully-specified* ones (known ext, known int)

It can rely on an algorithm for frequent pattern discovery!

It can rely on a criterion choice that combines biases!
Learning in $\mathcal{AL}$-log:

**task of concept formation (2)**

Different frequent patterns can have the same answer set!

\[ P = q(A) \leftarrow \text{speaks}(A,B), \text{believes}(A,C) \] & 
A: MiddleEastCountry, B: AfroAsiaticLanguage, C: MonotheisticReligion

\text{answerset}(P, B) = \{\text{IR}', 'SA'\}

\text{ext}(C) = \{\text{IR}', 'SA'\}

\[ Q = q(A) \leftarrow \text{speaks}(A,B), \text{believes}(A,C) \] & 
A: MiddleEastCountry, B: ArabicLanguage, C: MuslimReligion

\text{answerset}(Q, B) = \text{answerset}(P, B)

\text{int}(C) = ?
Learning in $AL$-log: task of concept formation (3)

Examples

Descriptions must have all the variables ontologically constrained by concepts from the 2nd granularity level on

**m.g.d.**

$$\text{int}(C) = q(A) \leftarrow \text{speaks}(A,B), \text{believes}(A,C) \; \& \; A: \text{MiddleEastCountry}, \; B: \text{AfroAsiaticLanguage}, \; C: \text{MonotheisticReligion}$$

$$\text{ext}(C) = \{ \text{IR', 'SA'} \}$$

**m.s.d.**

$$\text{int}(C) = q(A) \leftarrow \text{speaks}(A,B), \text{believes}(A,C) \; \& \; A: \text{MiddleEastCountry}, \; B: \text{ArabicLanguage}, \; C: \text{MuslimReligion}$$

$$\text{ext}(C) = \{ \text{IR', 'SA'} \}$$
Learning in $\mathcal{DL}+\log$\textsuperscript{−}\textsuperscript{∨}

F.A. Lisi (2010). Inductive Logic Programming in Databases: from Datalog to $\mathcal{DL}+\log$\textsuperscript{−}\textsuperscript{∨}. Theory and Practice of Logic Programming 10(3): 331–359.

CoRR abs/ 1003.2586

- **Scope of induction**: discrimination/characterization
- **ILP setting**: learning from entailment
- **Coverage test**: CQ answering in DL+log
Learning in $\mathcal{DL}+\text{log}$: task of rule learning

- **Scope of induction:** prediction
- **KR framework:** $\mathcal{DL}+\text{log}$
- **Generality order:** adaptation of Buntine’s *generalized subsumption* to $\mathcal{DL}+\text{log}$
- **Algorithm:** upgrade of FOIL (Quinlan, 1990)
- **Application:** induction of view definitions for a relational database whose schema is partially defined by means of an ontology
Learning in $\mathcal{DL} + \text{log}^\land \lor$:

task of rule learning (2)

| [A1] RICH$\land$UNMARRIED $\sqsubseteq$ $\exists$ WANTS-TO-MARRY$^\land$.T | UNMARRIED(Mary) UNMARRIED(Joe) |
| [A2] WANTS-TO-MARRY $\sqsubseteq$ LOVES | famous(Mary) famous(Paul) famous(Joe) |

[R1] RICH(X) $\leftarrow$ famous(X), not scientist(X) $\mathcal{K}$

$\mathcal{L}^{\text{happy}}$

$\models$\{famous/1,RICH/1, WANTS-TO-MARRY/2, LOVES/2\}

$\models$\{famous(X), WANTS-TO-MARRY(Y,X)\}

$\mathcal{L}^{\text{LONER}}$

$\models$\{famous/1,scientist/1,UNMARRIED/1\}

$\models$\{LONER(X) $\leftarrow$ scientist(X,Y),UNMARRIED(X)\}
Learning in \( \mathcal{DL+log} \) \( \nabla \vee \): task of rule learning (3)

[A1] \( \text{RICH} \cap \text{UNMARRIED} \subseteq \exists \ WANTS-TO-MARRY \cdot T \)

[A2] \( \text{WANTS-TO-MARRY} \subseteq \text{LOVES} \)

[R1] \( \text{RICH}(X) \leftarrow \text{famous}(X), \text{not scientist}(X) \)

\( \models \)

\( H_1^{\text{happy}} = \text{happy}(A) \leftarrow \text{RICH}(A) \)

\( H_2^{\text{happy}} = \text{happy}(X) \leftarrow \text{famous}(X) \)

\( H_1^{\text{happy}} \models K H_2^{\text{happy}} \)

\( H_2^{\text{happy}} \models K H_1^{\text{happy}} \)
Learning in $\mathcal{DL} + \text{log}$: task of rule learning (3)

A1] RICH \sqcap \text{UNMARRIED} \sqsubseteq \exists \text{WANTS-TO-MARRY}.T

[A2] \text{WANTS-TO-MARRY} \sqsubseteq \text{LOVES}

[R1] \text{RICH}(X) \leftarrow \text{famous}(X), \text{not scientist}(X)

\begin{align*}
\land H_1^{\text{happy}} & = \text{happy}(A) \leftarrow \text{famous}(A), \text{LOVES}(B,A) \\
\land H_2^{\text{happy}} & = \text{happy}(X) \leftarrow \text{famous}(X), \text{WANTS-TO-MARRY}(Y,X) \\
\land H_1^{\text{happy}} & \geq^\mathcal{K} H_2^{\text{happy}} \\
\land H_2^{\text{happy}} & \nleq^\mathcal{K} H_1^{\text{happy}}
\end{align*}
Learning in $\mathcal{DL} + \log$:

**task of rule learning (4)**

$\mathcal{L}_{\text{happy}}$ over $\{\text{famous}/1, \text{RICH}/1, \text{WANTS-TO-MARRY}/2, \text{LOVES}/2\}$

happy(X) ←

- $\langle \text{AddDataLit} \rangle$
- $\langle \text{AddOntoLit} \rangle$
- $\langle \text{AddOntoLit} \rangle$

happy(X) ← famous(X)

happy(X) ← famous(X), LOVES(Y,X)

happy(X) ← famous(X), WANTS-TO-MARRY(Y,X)

$\vdots$

[A2] WANTS-TO-MARRY $\subseteq$ LOVES $K$

Dr. Francesca A. Lisi
Learning in $\mathcal{DL} + \log^{\nabla}$: task of theory discovery

- **Scope of induction**: description

- **KR framework**: full $\mathcal{DL} + \log^{\nabla}$

- **Generality order**: adaptation of Plotkin’s *relative subsumption* to $\mathcal{DL} + \log^{\nabla}$

- **Algorithm**: upgrade of CLAUDIEN (De Raedt & Dehaspe, 1997)

- **Application**: induction of an integrity theory for a relational database whose schema is partially defined by means of an ontology
Learning in $\mathcal{DL} + \text{log}$: task of theory discovery

$\mathcal{DL}$ KB $\Sigma$

- PERSON $\sqsubseteq \exists$ FATHER$^{-1}$ . MALE
- MALE $\sqsubseteq$ PERSON
- FEMALE $\sqsubseteq$ PERSON
- FEMALE $\sqsubseteq \neg$ MALE
- MALE(Bob)
- PERSON(Mary)
- PERSON(Paul)

Datalog program:

- FEMALE(X) ← girl(X)
- MALE(X) ← boy(X)
- boy(Paul)
- girl(Mary)
- enrolled(Paul,c1)
- enrolled(Mary,c1)
- enrolled(Mary,c2)
- enrolled(Bob,c3)

NMSAT-DL+log

NMDISC-DL+log

H

- PERSON(X) ← enrolled(X,c1)
- boy(X) v girl(X) ← enrolled(X,c1)
- ← enrolled(X,c2), MALE(X)
- ← enrolled(X,c2), not girl(X)
- MALE(X) ← enrolled(X,c3)
- ....
Learning in $\mathcal{DL} + \text{log}$: task of theory discovery

$\leftarrow \text{enrolled}(X, c_1) \quad \text{ADD}\text{OntoLit}$

$\text{MALE}(X) \leftarrow \text{enrolled}(X, c_1)$

$\downarrow \text{ADD}\text{DataLit}$

$\text{boy}(X) \leftarrow \text{enrolled}(X, c_1)$

$\text{ADD}\text{DataLit}$

$\text{boy}(X) \lor \text{girl}(X) \leftarrow \text{enrolled}(X, c_1)$

$\leftarrow \text{enrolled}(X, c_2) \quad \text{ADD}\text{DataLit}$

$\leftarrow \text{enrolled}(X, c_2), \text{not girl}(X)$

$\leftarrow \text{enrolled}(X, c_3) \quad \text{ADD}\text{OntoLit}$

$\text{MALE}(X) \leftarrow \text{enrolled}(X, c_3)$

$\downarrow \text{GenOntoLit}$

$\text{PERSON}(X) \leftarrow \text{enrolled}(X, c_3)$
Learning in DL-HCL: Bibliography

Learning in DL-HCL: Bibliography (2)

Learning in DL-HCL: Bibliography (3)

Overview

- The Semantic Web
- KR for the Semantic Web
- ML for the Semantic Web
- Conclusions
What the SW can do for ML/DM

1. Lots and lots of tools to describe and exchange data for later use by ML/DM methods in a canonical way!

2. Using ontological structures to improve the ML/DM tasks

3. Provide background knowledge to guide ML/DM systems

See PriCKLws@ECML/PKDD-07
What ML/DM can do for the SW

1. Learning Ontologies (even if not fully automatic)
2. Learning to map between ontologies
3. Deep Annotation: Reconciling databases and ontologies
4. Annotation by Information Extraction
5. Duplicate recognition
ML meets KR in the SW: research directions in theory

- ILP frameworks for learning/mining in more expressive DLs or different DL-HCL integration schemes (e.g. loosely-coupled)
  ▲ closer to OWL and RIF
- ILP frameworks for learning/mining under uncertainty and vagueness
  ▲ closer to real-world ontologies
- ILP frameworks for learning/mining from multiple contexts
  ▲ Closer to the real scenario of the Semantic Web
ML meets KR in the SW: research directions in practice

- Efficient and scalable implementations
- Interfacing of ILP systems with specialized reasoners for the Semantic Web
  - (Fuzzy) OWL/SWRL reasoners
- Experimental work on big OWL ontologies integrated with a rule base
ML meets KR in the SW: applications for learning in DLs

- Ontology Refinement
- Ontology Matching
- Ontology Merging
- FOAF
- Semantic retrieval
- Etc.
ML meets KR in the SW: applications for learning in DL-HCL

- Ontology Refinement
  - Some concepts are better defined with rules
- Ontology Mapping
- Semantic Web Services
- Business rules
- Policy rules
- Etc.

Potentially all RIF use cases!
Further resources

- Tutorials on the Semantic Web
  - http://km.aifb.uni-karlsruhe.de/ws/prowl2006/
  - http://rease.semanticweb.org/

- Tutorials on ML for the Semantic Web
  - http://www.aifb.uni-karlsruhe.de/WBS/pci/OL_Tutorial_ECML_PKDD_05/
  - http://www.uni-koblenz.de/~staab/Research/Events/ICML05tutorial/icml05tutorial.pdf
Related AAAI-10 tutorials

- SA2: Exploiting Statistical and Relational Information on the Web and in Social Media: Applications, Techniques, and New Frontiers (L. Getoor, L. Mihalkova)
- SA3: Large-Scale Ontology Reasoning and Querying (J. Pan, Guilin Qi, J. Du)
- SP3: Rules on the Semantic Web: Advances in Knowledge Representation and Standards (B. Grosof, M. Dean, M. Kifer)
- MA2: How to Integrate Ontologies and Rules? (T. Eiter, S. Heymans, L. Polo, A. Nazarenko)