Probabilistic Description Logics: Reasoning and Learning

Riccardo Zese
Università di Ferrara
Email: riccardo.zese@unife.it
Presented Systems

The systems we developed, which I will mention or demonstrate during the talk, are all available at the Machine Learning@unife Web page: 
http://ml.unife.it/

The content of these slides is taken from the book Probabilistic Semantic Web: Reasoning and Learning published by IOSPress. 
Semantic Web

User interface and applications

Trust

Proof

Unifying logic

Taxonomies: RDFS

Rules: RIF/SWRL

Querying:

SPARQL

Ontologies: OWL

Data interchange:

RDF

Syntax: XML+NS

Identifiers: URI/IRI

Character set: UNICODE

Certification

Cryptography

R. Zese

Probabilistic Description Logics
The **Semantic Web** aims at making information available in a form that is understandable and **automatically manageable** by machines.

**Ontologies**, or **knowledge bases**, are engineering artifacts used to this purpose.

**Description Logics** are (decidable) subsets of First Order Logic languages used for modeling (and reasoning upon) ontologies and the Semantic Web.
Big Data
Outline

1. Combining Logic and Probability
2. Inference Algorithms
3. Learning Algorithms
Description Logics [Baader et al., 1994]

- (Decidable) subsets of First Order Logic, basis of OWL
- Open World Assumption
- Individuals, Class concepts, roles, and concept-forming operators

\[ KB = \langle ABox, \; TBox, \; RBox \rangle \]

\begin{align*}
\text{tom} : & \text{Cat} \\
(\text{kevin, tom}) : & \text{hasAnimal} \\
\text{Cat} & \sqsubseteq \text{Pet} \\
\exists \text{hasAnimal}. \text{Pet} & \sqsubseteq \text{NatureLover} \\
\text{hasPet} & \sqsubseteq \text{hasAnimal}
\end{align*}
Description Logics

Some concept-forming operators:

- class conjunction: \( C \sqcap D \)
- class disjunction: \( C \sqcup D \)
- existential restriction: \( \exists R.C \)

Some types of axioms:

- class inclusion/subsumption: \( C \sqsubseteq D \)
- role chains: \( R_1 \circ \ldots \circ R_n \subseteq R \)
- inverse roles: \( R \equiv S^- \)

Decidability (and tractability) is a central issue of DLs
Description Logics

- DLs differ depending on which operators are admitted; more operators means:
  - higher expressivity
  - higher computational costs
  - the logic might to be *undecidable*
  - ...

- [http://www.cs.man.ac.uk/~ezolin/dl/](http://www.cs.man.ac.uk/~ezolin/dl/), a nice Web application that shows decidability and complexity issues depending on which operator you decide to include/exclude in your logic, with extensive references
\( K_{SW} \): A simple example in \( \text{ALC} \)

A long time ago in a galaxy far, far away...

- The class of Stormtrooper is that of soldiers of Galactic Empire having at least one blaster equipped, and also only fighting Ewoks

\[
\text{Stormtrooper} \equiv \exists \text{hasEquip}.\text{Blaster} \land \forall \text{hasFought}.\text{Ewok}
\]

- The class of blaster is the union of blaster rifles and blaster pistols

\[
\text{Blaster} \equiv \text{Blaster}_\text{Rifle} \sqcup \text{Blaster}_\text{Pistol}
\]

- FN-2199 is equipped also with a blaster, but not all the foes he has fought were Ewoks

\[
\text{FN-2199} : \exists \text{hasEquip}.\text{Blaster} \land \neg (\forall \text{hasFought}.\text{Ewok})
\]

- We can check whether \( K_{SW} \models \text{FN-2199} : \text{Stormtrooper} \), and it is not but...
Meet FN-2199
OWL: Web Ontology Language

- Developed by the W3C for representing ontologies

Three different sublanguages:

- **OWL-Lite**, based on $SHIF$, good for thesauri or hierarchies
- **OWL DL**, based on $SHOIN'$, more expressive, computational complete, decidable, availability of practical reasoning algorithms
- **OWL Full**, highly expressive, also high-order logic, undecidable
- **OWL 1.1** and **OWL 2** based on $SROIQ$
A more elaborated example for Stormtrooper in \textit{SROIQ}

- Stormtroopers are soldiers of Galactic Empire having at least one blaster equipped and that have fought at least 10 Ewoks\textsuperscript{1}

\[
\text{Stormtrooper} \equiv \exists \text{hasEquip.Blaster} \sqcap \geq 10 \text{hasFought.Ewok}
\]

- \textit{FN-2199} is equipped with a blaster pistol and a baton, and has fought at least 10 Ewoks (but also humans and other races/species)

\[
\text{FN-2199} : \exists \text{hasEquip.Blaster} \sqcap \exists \text{hasEquip.Baton}\sqcap \\
\geq 10 \text{hasFought.Ewok} \sqcap \neg (\forall \text{hasFought.Ewok})
\]

- I am quite happy now, since, in this KB, \textit{FN-2199} is a Stormtrooper!

\textsuperscript{1}Qualified number restriction
OWL 2: retaining tractability

OWL 2 profiles, identified as maximal OWL 2 sublanguages still implementable in \textit{PTime}, each is more restrictive than OWL DL:

- **OWL QL** disallow $\exists$ (fragment of Datalog, tractable, conjunctive queries answered in LogSpace)
- **OWL EL** disallow inverse roles, (tractable, used in ontology with huge DBs as SNOMED CT)
- **OWL RL** subclass axioms understood as rule-like implications with head (superclass) and body (subclass), with restrictions on subclasses and superclasses (e.g., no existentials in the heads; standard reasoning is P-Time complete)
Probabilistic Description Logics

- Semantic Web
  - Incompleteness or uncertainty are intrinsic of much information on the World Wide Web
  - Most common approaches: probability theory, Fuzzy Logic

- Probabilistic extensions of DL based on Bayesian networks [Koller et al., 1997, Ding and Peng, 2004]

- Statistical terminological knowledge in $\mathcal{PCL}$ [Jaeger, 1994]

- PR-OWL, a framework for building probabilistic ontologies [Carvalho et al., 2010, Laskey and da Costa, 2005]

- P-$\mathcal{SHIQ(D)}$ [Lukasiewicz, 2008], semantically based on the notion of probabilistic lexicographic entailment

- Prob-$\mathcal{ALC}$ [Lutz and Schröder, 2010] and $\mathcal{CRALC}$ [Ochoa-Luna et al., 2011], extending $\mathcal{ALC}$ with epistemic and statistical axioms respectively

- $\mathcal{BEL}$ [Ceylan et al., 2015] exploiting Bayesian networks to extend the $\mathcal{EL}$
Uncertainty Representation in LP

- Have a look at Logic Programming (LP) field
- Uncertain relationships among entities characterize many complex domains

Most common approach: probability theory → Distribution Semantics [Sato, 1995].
  - It underlies many languages (ICL, PRISM, ProbLog, LPADs),...
  - They define a probability distribution over logic programs, called worlds
  - The distribution is extended to a joint distribution over worlds and queries
  - The probability of a query is obtained from this distribution by summing out worlds
Inference for PLP under DS

Computing the probability of a query

- **Knowledge compilation:**
  - compile the program to an intermediate representation
    - **Binary Decision Diagrams** (ProbLog [De Raedt et al., 2007], cplint\(^2\) [Riguzzi, 2007, Riguzzi, 2009], PITA [Riguzzi and Swift, 2011])
    - deterministic, Decomposable Negation Normal Form circuits (d-DNNF) (ProbLog\(^3\) [Fierens et al., 2015])
    - **Sentential Decision Diagrams** (ProbLog2)
  - and then compute the probability by weighted model counting

- **Bayesian Network based:**
  - Convert to BN
  - Use BN inference algorithms (CVE [Meert et al., 2010])

- **Lifted inference** [Poole, 2003]

\(^2\)http://cplint.ml.unife.it/
\(^3\)https://dtai.cs.kuleuven.be/problog/editor.html
Knowledge Compilation

- Assigns boolean random variables to the probabilistic rules
- Given a query $Q$, compute its explanations, assignments to the random variables that are sufficient for entailing the query
- Let $K$ be the set of all possible explanations
- Build the DNF boolean formula:

$$F(Q) = \bigvee_{\kappa \in K} \bigwedge_{X \in \kappa} X \bigwedge_{\overline{X} \in \kappa} \overline{X}$$

- Then build a BDD / d-DNNF / SDD representing $F(Q)$, and compute the probability for $Q$ from it, by Shannon expansion
**DISPONTE: Distribution Semantics for Probabilistic ONTologies**

- **Idea:** annotate axioms of an ontology with a probability, under the assumption that the axioms are independent of each other
  
  \[ 0.6 \bowtie \text{Cat} \sqsubseteq \text{Pet} \]

- The probability value specifies a degree of belief in the corresponding axiom.

- A probabilistic ontology defines thus a distribution over theories (worlds) obtained by including an axiom in a world with the probability given by the annotation
Example - people + pets ontology

- fluffy is a Cat with probability 0.4 and tom is a Cat with probability 0.3; Cats are Pets with probability 0.6. Everyone who has a pet animal ($\exists \text{hasAnimal. Pet}$) is a NatureLover; kevin has two animals, fluffy and tom.

\[
\begin{align*}
0.4 & : \text{fluffy} : \text{Cat} \\
0.3 & : \text{tom} : \text{Cat} \\
0.6 & : \text{Cat} \sqsubseteq \text{Pet}
\end{align*}
\]

\[
\begin{align*}
\exists \text{hasAnimal. Pet} \sqsubseteq \text{NatureLover} \\
(\text{kevin, fluffy}) : \text{hasAnimal} \\
(\text{kevin, tom}) : \text{hasAnimal}
\end{align*}
\]

- $Q = \text{kevin} : \text{NatureLover}$ has 8 worlds (true in 3 of them):

\[
\{ (1), (3) \}, \{ (2), (3) \}, \{ (1), (2), (3) \}
\]

and $P(Q) = 0.4 \times 0.6 \times (1 - 0.3) + 0.3 \times 0.6 \times (1 - 0.4)
\+
0.4 \times 0.6 \times 0.4 = 0.348$
Inference and Query answering

The probability of a query $Q$ can be computed accordingly with the distribution semantics by first finding the explanations for $Q$ in the $KB$

**Explanation**: set of boolean variables corresponding to a set of axioms of $KB$ that is sufficient for entailing $Q$

All the explanations for $Q$ must be found, corresponding to all (minimal) ways of proving $Q$

- Probability of $Q \rightarrow$ probability of the DNF formula

$$F(Q) = \bigvee_{\kappa \in K} \bigwedge_{X \in \kappa} X$$

where $K$ is the set of explanations and $X$ is a Boolean random variable associated to axiom $\kappa$

- We exploit **Binary Decision Diagrams** (BDDs) for efficiently computing the probability of a DNF formula
Example - people+pets ontology cont.

Explanations\(^4\): \{ (1), (3) \}, \{ (2), (3) \}

\[
P(Q) = 0.4 \times 0.6 \times (1 - 0.3) + 0.3 \times 0.6 = 0.348
\]

We associate the random variables \(X_1\) with (1), \(X_2\) with (2) and \(X_3\) with (3).

\[
f(X) = (X_1 \land X_3) \lor (X_2 \land X_3)
\]

- The probability is:
  \[
P(n_3) = 0.6 \cdot 1 + 0.4 \cdot 0 = 0.6
  \]
  \[
P(n_2) = 0.3 \cdot 0.6 + 0.7 \cdot 0 = 0.18
  \]
  \[
P(n_1) = 0.4 \cdot 0.6 + 0.6 \cdot 0.18 = 0.348
  \]
- So \(P(Q = \text{kevin} : \text{NatureLover}) = P(n_1) = 0.348\).

\(^4\)Certain axioms are irrelevant for computing query probability
**PCL** [Jaeger, 1994]

- Allows the definition of
  - $P(C|D) = [p_l, p_u]$ called *probabilistic terminological axioms* defining statistical information
  - $P(C(a)) = p$ defining probabilistic assertions

- Inference can be done by:
  1. naive method for computing optimal bound using cross-entropy minimization
  2. linear optimization problem
**P-SHIQ(D) [Lukasiewicz, 2008]**

- Allows the definition of
  - $(C | D)[l, u]$ (as in [Jaeger, 1994])
  - $(\exists R. \{a\} | C)[l, u]$ (an arbitrary instance of a concept $C$ is $R$-related to the individual $a$ with probability in the interval $[l, u]$)
  - $(C | \{a\})[l, u]$ and $(\exists R. \{b\} | \{a\})[l, u]$ defining probabilistic assertions (correspond with DISPONTE)

- Terminological knowledge is interpreted statistically (as in [Jaeger, 1994])

- Assertional knowledge is interpreted in an epistemic way.

PR-OWL
[Carvalho et al., 2010, Laskey and da Costa, 2005]

- An upper ontology that provides a framework for building probabilistic ontologies. It allows to use the first-order probabilistic logic MEBN [Laskey and da Costa, 2005] for representing uncertainty in ontologies.

- DISPONTE differs from PR-OWL because it allows the reuse of inference technology from DLs by minimally extending DL.
Prob-\(\text{ALC}\) [Lutz and Schröder, 2010]

- Allows the definition of
  - \(P \geq n \mathcal{C}\) indicating set of individuals belonging to \(\mathcal{C}\) with probability greater than \(n\)
  - \(\exists P \geq n \mathcal{R}. \mathcal{C}\) set of individuals \(a\) connected to at least another individual \(b\) of \(\mathcal{C}\) by role \(R\) such that the probability of \(R(a, b)\) is greater than \(n\)
  - \(P \geq n \mathcal{C}(a)\) and \(P \geq n \mathcal{R}(a, b)\) defining probabilistic assertions

- Considers only epistemic probabilities (as DISPONTE)

- Complementary to DISPONTE \(\text{ALC}\) as it allows new concept and assertional expressions while DISPONTE allows probabilistic axioms.
CRALC [Ochoa-Luna et al., 2011]

- Allows the definition of
  - \( P(C|D) = \alpha \), for any element \( x \) in the domain, the probability that \( x \) is in \( C \) given that it is in \( D \) is \( \alpha \)
  - \( P(R) = \beta \), for each couple of elements \( x \) and \( y \) in the domain, the probability that \( x \) is linked to \( y \) by the role \( R \) is \( \beta \)

- Considers only statistical probabilities

- Adopts an interpretation-based semantics based on probability measures over the space of interpretations with a fixed domain.

- Inference performed by a first order loopy belief propagation algorithm on ground direct acyclic graphs which represents the KBs w.r.t. queries.
BEL [Ceylan et al., 2015]

- Extends $\mathcal{EL}$ DL by exploiting Bayesian network with variables $V$.
- Allows the definition of $E : X = x$ where $E$ is a DL axiom and $X = x$ is an annotation with $X \subseteq V$ and $x$ a set of values for these variables.
- Bayesian network assigns a probability to every assignment of $V$, called a world.
- The probability of a query $Q = E : X = x$ is given by the sum of the probabilities of the worlds where $X = x$ is satisfied and where $E$ is a logical consequence of the theory composed of the axioms whose annotation is true in the world.
- DISPONTE is a special case of these semantics where every axiom $E_i : X_i = x_i$ is such that $X_i$ is a single Boolean variable and the Bayesian network has no edges, i.e., all the variables are independent.
Reasoning on a $KB$: approaches

A variety of reasoning techniques (at least for $\mathcal{ALC}$):

- Resolution-based approaches
- Automata based approaches
- Structural approaches
- Tableau based approaches

Most DL reasoners use a tableau algorithm for doing inference, and most of them are implemented in a procedural language (e.g., Pellet, RacerPro, FaCT++)
Tableau-based approaches

They usually solve the $KB$ (in)consistency problem, i.e., determine whether a given $KB$ has a model, reducing reasoning tasks to this problem:

- **Concept unsatisfiability**
  \[ KB \models C \equiv \bot \iff KB \cup \{C(x)\} \text{ has no model} \]

- **Subsumption**
  \[ KB \models C \sqsubseteq D \iff KB \cup \{(C \sqcap \neg D)(x)\} \text{ has no model} \]

- **Instance checking**
  \[ KB \models C(a) \iff KB \cup \{\neg C(a)\} \text{ has no model} \]
The Tableau Algorithm

- Starts from the ABox $\mathcal{A}$, represented as a graph (the tableau), where
  - Each node represents an individual $a$ and is labeled with the set of concepts it belongs to;
  - Each edge between two individuals $a$ and $b$ is labeled with the set of roles to which the couple $(a, b)$ belongs.

- Proves the satisfiability of an axiom by refutation, e.g.:
  - To test a class assertion axiom $C(a)$, it adds $\neg C$ to the label of $a$.
  - To test the unsatisfiability of a concept $C$, it adds a new anonymous node $x$ to the tableau and adds $\neg C$ to the label of $x$.

- Applies expansion rules (one for each language construct, possibly non-deterministic), until all constraints are satisfied or a contradiction (clash) is detected

- Uses a blocking system (to avoid loops, and ensure termination)
The Tableau Algorithm

\( \text{tom : Cat} \)

\((\text{kevin, tom}) : \text{hasAnimal}\)

\(\text{Cat} \sqsubseteq \text{Pet}\)

\(\exists \text{hasAnimal}.\text{Pet} \sqsubseteq \text{NatureLover}\)

\(\text{hasPet} \sqsubseteq \text{hasAnimal}\)

\(Q = \text{kevin : NatureLover}\)

\(\text{kevin} : \neg \text{NatureLover}\)

\(\text{hasAnimal}\)

\(\text{tom} : \text{Cat}\)

\(\Rightarrow\)

\(\text{kevin} : \text{NatureLover}\)

\(\neg \text{NatureLover}\)

\(\text{hasAnimal}\)

\(\text{hasPet}\)

\(\text{tom} : \text{Cat}\)

\(\text{Pet}\)
### Expansion Rules

- **→ unfold:** if \( A \in \mathcal{L}(a) \), \( A \) atomic and \((A \sqsubseteq D) \in KB\), then
  - if \( D \notin \mathcal{L}(a) \), then
    \[
    \begin{align*}
    \mathcal{L}(a) &:= \mathcal{L}(a) \cup \{D\} \\
    \tau(D, a) &:= \tau(A, a) \cup \{(A \sqsubseteq D, a)\}
    \end{align*}
    \]
  - otherwise

- **→ ⊔:** if \((C_1 \sqcup C_2) \in \mathcal{L}(a)\) and \( a \) is not blocked, then
  - if \( \{C_1, C_2\} \cap \mathcal{L}(a) = \emptyset \), then
    - Generate graphs \( G_i := G \) for each \( i \in \{1, 2\} \),
    - \( \mathcal{L}(a) := \mathcal{L}(a) \cup \{C_i\} \) for each \( i \in \{1, 2\} \)
    - \( \tau(C_i, a) := \tau((C_1 \sqcup C_2), a) \)

- **Output:** explanation, as set of axioms of the \( KB \) entailing the query (thanks to the tracing function \( \tau \))
- **Often,** we also want to find **all the possible explanations** for a query (**axiom pinpointing**)
The Tableau Algorithm: OR-search tree

- **Local soundness:** rules preserve satisfiability
- **Termination:** all paths finite
- **Complete ABoxes:** no rules apply

Satisfiable iff **one** of the complete ABoxes is **open**, i.e., does **not** contain an **obvious contradiction** (clash).
Despite the availability of many DL reasoners, the number of probabilistic reasoners is quite small

- BUNDLE
- TRILL
- TRILL$^P$
- TORNADO
- PRONTO [Klinov, 2008] which follows P-$SHIQ(D)$ semantics
- BORN [Ceylan et al., 2015] following $BEL$
**BUNDLE**

Binary decision diagrams for Uncertain reasoning on Description Logic theories

- BUNDLE performs inference over DISPONTE OWL knowledge bases
- It exploits the underlying ontology reasoner Pellet [Sirin et al., 2007] able to return all explanations for a query
- Explanations for a query in the form of a set of sets of axioms
- BUNDLE performs a double loop over the set of explanations and over the set of axioms in each explanation, in which it builds a BDD representing the set of explanations
- JavaBDD library for the manipulation of BDDs
- [https://sites.google.com/a/unife.it/ml/bundle](https://sites.google.com/a/unife.it/ml/bundle)
Why not Prolog?

- The reasoners based on procedural languages have to implement also a backtracking algorithm to find all the possible explanations.
  - Example: BUNDLE (Java-based) uses a hitting set algorithm that repeatedly removes an axiom from the KB and then computes again a new explanation.

- Reasoners written in Prolog can exploit Prolog’s backtracking facilities for performing the search.
Reasoning: use of Logic Programming

- Some tableau expansion rules are non-deterministic (e.g., \(\sqcup\))
- Reasoners implement a search strategy in an or-branching space
- The algorithm has to explore all the non-deterministic choices
  - LP-based implementations of the tableau (e.g., [Meissner, 2004], [Herchenröder, 2006], [Beckert and Posegga, 1995])
  - LP-based inference methods based on resolution (e.g., in the DLog system by [Lukácsy and Szeredi, 2009])
  - Bottom-up inference methods, e.g., based on Answer Set Programming (e.g., ontoDLP, and its reasoner ontoDLV [Ricca et al., 2009]), or the chase algorithm for Datalog\(\pm\) [Calì et al., 2009]
Inference Algorithms
Reasoners

TRILL - Tableau Reasoner for Description Logics in ProLog

- Implements the *tableau algorithm* using *Prolog*
  - Prolog’s search strategy is exploited for the non-determinism of the reasoning process
- Solves the axiom pinpointing problem since we are interested in the set of explanations that entail a query
- Applies all the possible expansion rules, first the non-deterministic ones then the deterministic ones
- Computes the probability of the query from the set of explanations

https://sites.google.com/a/unife.it/ml/trill
TRILL$^P$ - Tableau Reasoner for description Logics in proLog powered by Pinpointing formula

- TRILL$^P$ solves the axiom pinpointing problem by computing a pinpointing formula [Baader and Peñaloza, 2010]
  1. The pinpointing formula is a monotone Boolean formula on the variables associated to axioms that compactly encodes the set of all explanations
  2. Like $F(Q)$ when finding all explanations, except it may not be in DNF: $((F_1 \lor F_3) \land F_2)$ instead of $((F_1 \land F_2) \lor (F_3 \land F_2))$
  3. Convert it into a BDD, so we can compute the probability as in BUNDLE

- https://sites.google.com/a/unife.it/ml/trill
TORNADO - Trill powered by pinpointing formulas and binary decision diagrams

- TORNADO solves the axiom pinpointing problem by computing a BDD representing the *pinpointing formula*.
- The use of BDDs instead of explanations or pinpointing formulas allows TORNADO being usually faster and avoiding some exponential blow-ups.

https://sites.google.com/a/unife.it/ml/trill
TRILL-on-SWISH is an interface for the reasoners TRILL, TRILL$^P$ and TORNADO.

- It allows users to write a $KB$ in the RDF/XML format or in Prolog syntax directly in the web page or load it from a URL, and specify queries that are answered by TRILL running on the server.

- Available at http://trill.ml.unife.it
PRONTO [Klinov, 2008]

A reasoner for P-$SHIQ(D)$.

- Based on Pellet, implemented by same authors.
- Exploits a linear program solver such as GLPK\(^5\).

- Performs probabilistic lexicographic entailment by means of solving Probabilistic Satisfiability problems and tight logical entailments.
- Pellet is used to help the generation of linear programs given as input to the linear program solver.

\(^5\)https://www.gnu.org/software/glpk/
BORN [Ceylan et al., 2015]

Answers probabilistic subsumption queries w.r.t. $\mathcal{BEL}$ KBs.
- Exploits ProbLog for managing the probabilistic part of the KB.
- Can perform inference on KBs of low expressivity ($\mathcal{EL}$ DL).
Probabilistic Learning

Problem

**Input:** Background knowledge as a (probabilistic) KB $B$, a set of positive and negative examples $E^+$ and $E^-$, and possibly a language bias $L$.

**Output:** A probabilistic KB $P$ such that the probability of positive examples according to $P \cup B$ is maximized and the probability of negative examples is minimized.

Two variants:

1. **parameter learning:** learning the parameters of a fixed probabilistic logic program $B$

2. **structure learning:** both the structure and the parameters of a probabilistic logic program $B$
Parameter Learning

Parameter learning for languages following the distribution semantics has been performed by using the Expectation Maximization (EM) algorithm or by gradient descent.

- **Gradient Descent**
  Computes the gradient of the target function and iteratively modify the parameters moving in the direction of the gradient.

LeProbLog [Gutmann et al., 2008]: use of a dynamic programming algorithm for computing the gradient exploiting BDDs.
Expectation Maximization

- **EM**
  Estimates the probability of models containing random variables that are not observed in the data (hidden).
  Used in many systems: PRISM [Sato and Kameya, 2001], LFI-ProbLog [Fierens et al., 2015], EMBLEM [Bellodi and Riguzzi, 2013], RIB [Riguzzi and Di Mauro, 2012]).

**Why EM?**

- We can not use relative frequency.
  1. For each example we can find more than one explanation with more than one probabilistic axiom.
  2. We have to associate a variable to each (probabilistic) axiom.
- We know only the value of a Boolean function that uses the variables, thus they are hidden.
**Expectation Maximization**

- **Expectation:** for each example $Q$, computes $\mathbb{E}[c_{i0}|Q]$ and $\mathbb{E}[c_{i1}|Q]$ for all axioms $E_i$ where $c_{ix}$ is the number of times variable $X_i$ takes value $x$ for $x \in \{0, 1\}$:

$$\mathbb{E}[c_{ix}|Q] = P(X_i = x|Q) = \frac{P(X_i = x, Q)}{P(Q)}$$

Then it sums up the contributions of the different examples

$$\mathbb{E}[c_{ix}] = \sum_Q \mathbb{E}[c_{ix}|Q]$$

- **Maximization:** computes $p_i$ for all axioms $E_i$:

$$p_i = \frac{\mathbb{E}[c_{i1}]}{\mathbb{E}[c_{i0}] + \mathbb{E}[c_{i1}]}$$

- The EM algorithm is guaranteed to find a local maximum of the probability of the examples.
Some learning systems for DL KBs:

- EDGE
- LEAP
- CR\textit{ALC} [Ochoa-Luna et al., 2011], both parameter and structure learning
- GoldMiner [Völker and Niepert, 2011, Fleischhacker and Völker, 2011], both parameter and structure learning
EDGE: Parameter Learning in DISPONTE

EDGE (Em over bDds for description Logics parameter learning)

1. learns the parameters (weights) of a probabilistic KB by taking as input a DL KB and a number of positive and negative examples
2. EDGE builds the BDD corresponding with the explanations of each example using BUNDLE
3. Then, it learns the probabilities associated with axioms by traversing the BDDs during the execution of EM algorithm

\[ \text{EDGE}^{\text{MR}} \] (EDGE powered by MapReduce) parallelizes EDGE:

- The examples are divided in chunks and associated to different processes, since each example is independent of the others
- The learning phase is spread on all the workers as well
EDGE

1. Expectation takes as input a list of BDDs and computes $P(X_i = x|Q)$ using two passes over the BDD.
   - Expected counts of random variables: $E[c_{i0}|Q]$ and $E[c_{i1}|Q]$ are computed by traversing twice the BDDs

2. Maximization computes the parameters values for the next EM iteration by relative frequency.
LEAP: Structure Learning in DISPONTE

- **LEAP** (LEArning Probabilistic description logics)
  1. Learns the structure of a probabilistic KB by taking as input a DL KB and a number of positive and negative examples
  2. Exploits CELOE [Lehmann et al., 2011], a top-down algorithm which learns (acyclic) concept expressions, given a set of positive and negative examples
  3. Descriptions $C$ for one or more Target classes are created, one probabilistic subsumption axiom at a time of the form $p :: C \sqsubseteq \text{Target}$ is added to the ontology $\mathcal{K}$;
  4. EDGE is run on the extended theory to compute the log-likelihood of the data $LL$ and the updated parameters
    - If $LL$ is better than the current best $LL_0$, the new axiom is kept in the knowledge base, otherwise the new axiom is discarded.

- **LEAP$^{MR}$** (LEAP powered by MapReduce) parallelizes LEAP by exploiting EDGE$^{MR}$
**CR\(_{ALC}\) learner [Ochoa-Luna et al., 2011]**

A parameter and structure learner for \(\text{CR}_{ALC}\).

- Starts from positive and negative examples for a single concept and the general concept \(\top\) in the root of the search tree.
- The space of possible concept definitions is explored by means of a revision operator in the style of Inductive Logic Programming.
- Parameters are learned using an EM algorithm.
  - If the best score in the tree is above a threshold, a deterministic concept definition is returned, otherwise a probabilistic inclusion \(C_i\) is searched on a weighted spanning tree, where the target concept is added as a parent of each vertex and probabilities are learned as \(P(C_i|\text{Parents}(C_i))\).
  - Each definition is scored against the examples
  - The definition with the highest score is retained and the algorithm enters a new refinement iteration.

- top-down procedure for building axioms (CELOE) but LEAP exploits BDD structures to compute the expected counts for EM instead of resorting to inference in a graphical model.
GoldMiner

[Völker and Niepert, 2011, Fleischhacker and Völker, 2011]

A parameter and structure learner.

- Extracts information about individuals, named classes and roles using SPARQL queries
- Builds two transaction tables storing class and property assertions, a row for each individual (pair of individuals) and a column for each named concept and $\exists R.C$ concept (role), the cells contains 1 if the individual (pair) belongs to concept (role).
- APRIORI algorithm [Agrawal and Srikant, 1994] is applied to each table in turn in order to find ARs.
  - Each AR $A \Rightarrow B$ is converted to the axiom $A \sqsubseteq B$.
  - Confidence of AR can be interpreted as the probability of the axiom.
- GoldMiner can be used to obtain a probabilistic knowledge base.
Thank you for listening!

Research group: **MachineLearning@unife**

http://ml.unife.it/
Fast algorithms for mining association rules in large databases.
In *International Conference on Very Large Data Bases*, pages 487–499. Morgan Kaufmann.

Automata-based axiom pinpointing.

leantap: Lean tableau-based deduction.

Expectation Maximization over Binary Decision Diagrams for probabilistic logic programs.

A general datalog-based framework for tractable query answering over ontologies.

PR-OWL 2.0 - bridging the gap to OWL semantics.
The bayesian ontology reasoner is born!


A probabilistic extension to ontology language OWL.

Inference and learning in probabilistic logic programs using weighted boolean formulas.

Inductive learning of disjointness axioms.
Parameter learning in probabilistic databases: A least squares approach.

Lightweight semantic web oriented reasoning in Prolog: Tableaux inference for description logics.
Master's thesis, School of Informatics, University of Edinburgh.

Probabilistic reasoning in terminological logics.

Pronto: A non-monotonic probabilistic description logic reasoner.

P-CLASSIC: A tractable probabilistic description logic.
Of starships and klingons: Bayesian logic for the 23rd century.

Class expression learning for ontology engineering.

Efficient description logic reasoning in prolog: The dlog system.

Expressive probabilistic description logics.

Probabilistic Description Logics for subjective uncertainty.

CP-Logic theory inference with contextual variable elimination and comparison to BDD based inference methods.
References V

An automated deduction system for description logic with alcn language.
*Studia z Automatyki i Informatyki*, 28-29:91–110.

Probabilistic logic.

Learning probabilistic description logics: A framework and algorithms.

First-order probabilistic inference.

OntoDLV: An ASP-based system for enterprise ontologies.

A top down interpreter for LPAD and CP-logic.
In *Proceedings of the 10th Congress of the Italian Association for Artificial Intelligence*, volume 4733 of *LNAI*, pages 109–120. Springer.
Extended semantics and inference for the Independent Choice Logic.

Applying the information bottleneck to statistical relational learning.

The PITA system: Tabling and answer subsumption for reasoning under uncertainty.

A statistical learning method for logic programs with distribution semantics.

Parameter learning of logic programs for symbolic-statistical modeling.

Pellet: A practical OWL-DL reasoner.

Statistical schema induction.