(Probabilistic) Description Logics

Evelina Lamma

Dipartimento di Ingegneria
Università di Ferrara

evelina.lamma@unife.it

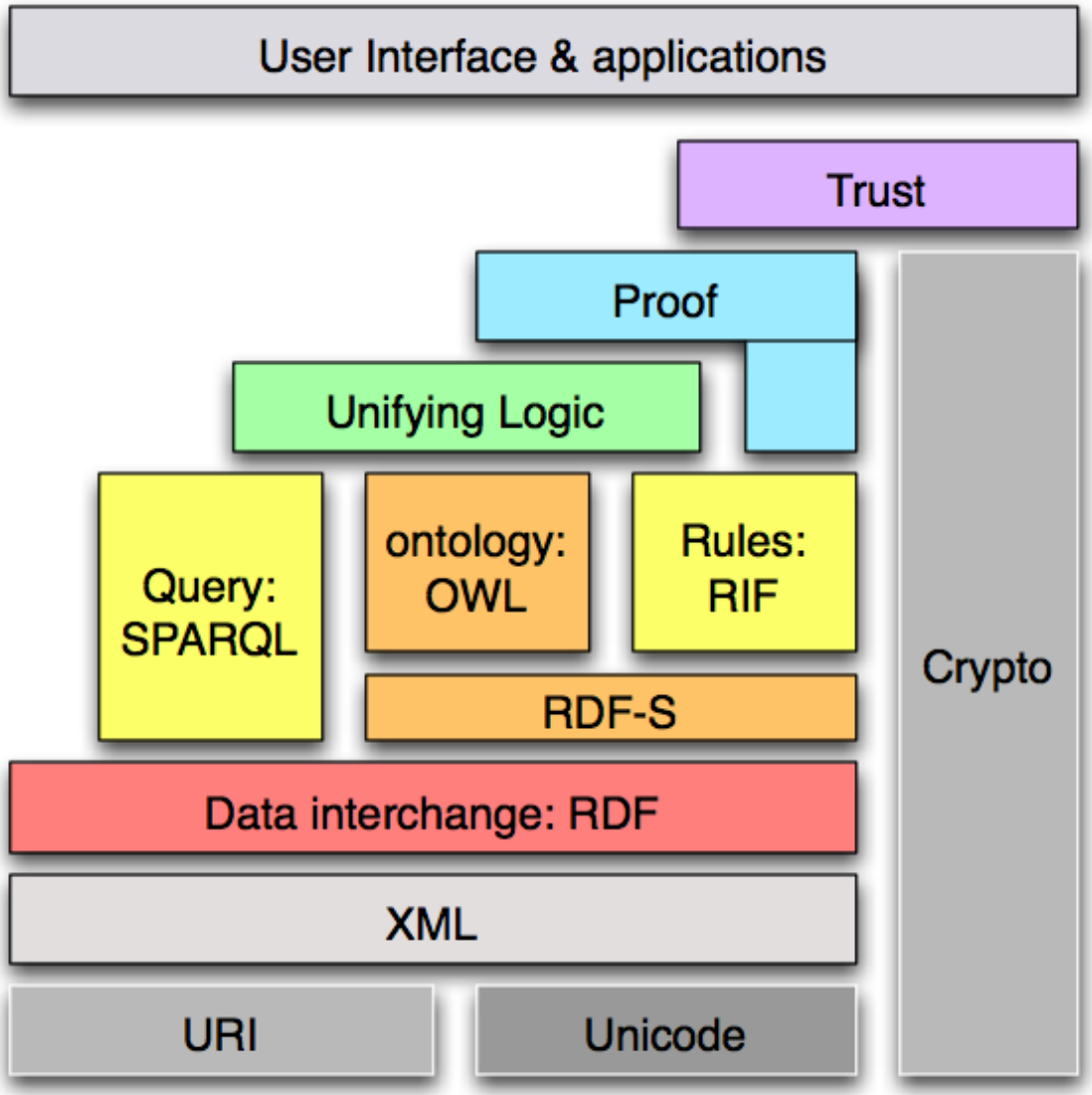
A special thank

This talk would have not been possible without the work by:

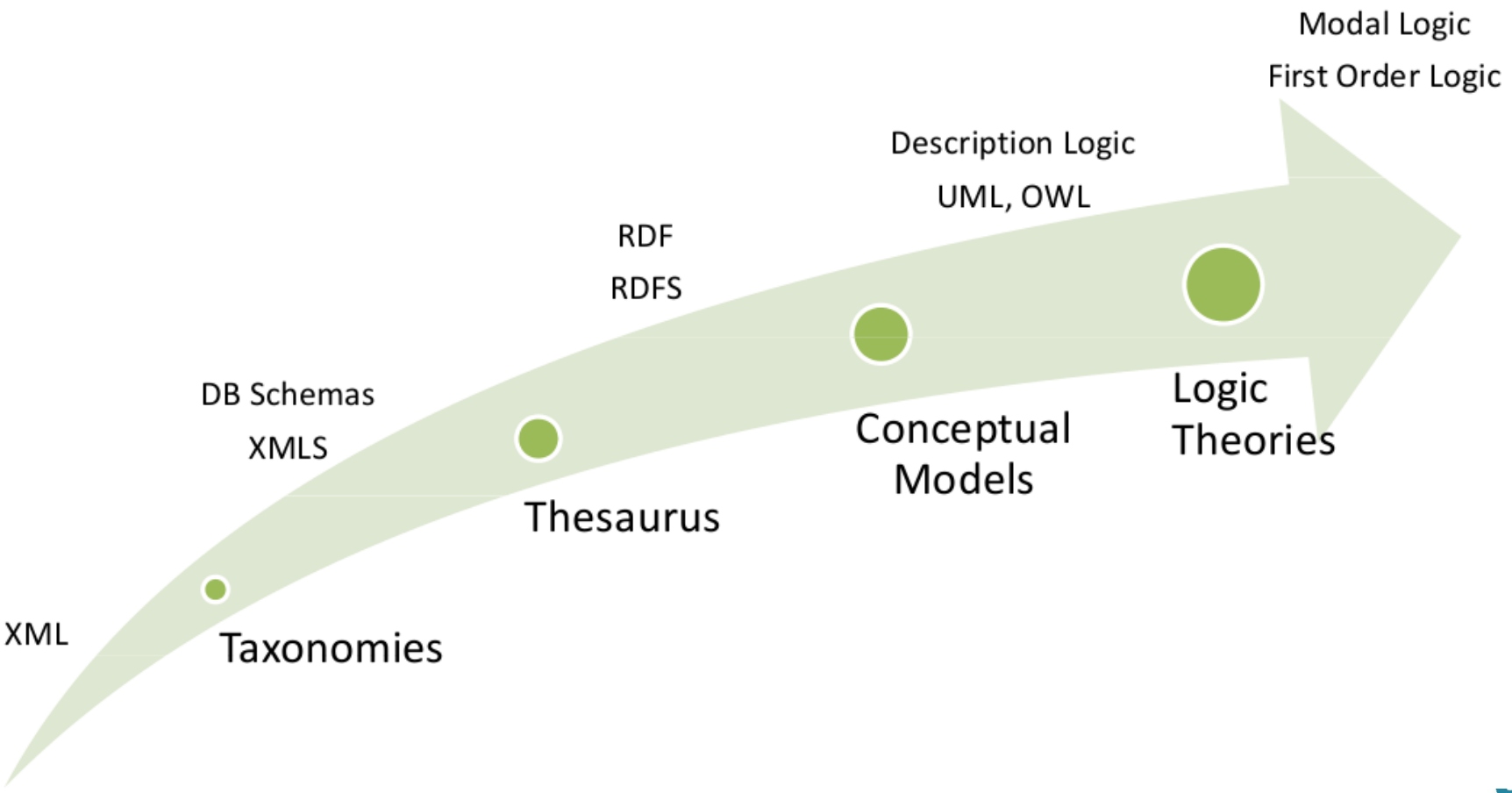
- Fabrizio Riguzzi
- Elena Bellodi
- Riccardo Zese
- Giuseppe Cota

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

Semantic Web



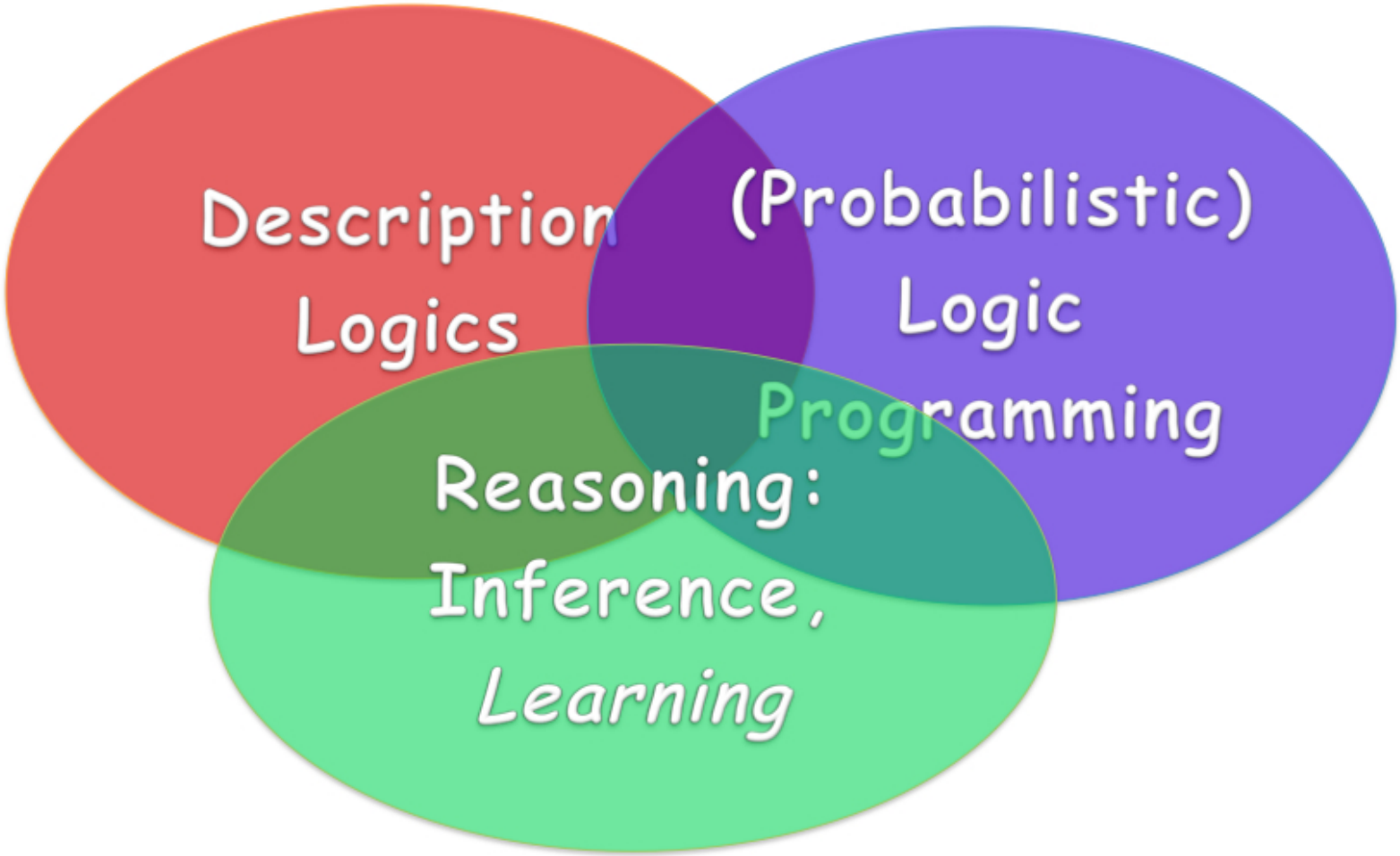
Semantic Models



CL-aim

- The Semantic Web aims at making information available in a form that is understandable and automatically manageable by machines.
- Ontologies are engineering artifacts used to this purpose, and OWL the W3C language for them
- Description Logics are the family of logic-based languages for modeling (and reasoning upon) ontologies and the Semantic Web.
- Logic Programming languages and systems have been extensively used?

Topics


Description
Logics
(Probabilistic)
Logic
Programming
Reasoning:
Inference,
Learning

Outline

Description Logics [Baader et al., 1994]

- (Decidable) subsets of First Order Logic, choosen as the logic-based grounding OWL
- Open World Assumption
- Class concepts (unary predicates), roles (binary predicates), and concept-forming operators
- $KB = \langle Abox, Tbox \rangle$

$fluffy : Cat$	$Cat(fluffy)$
$tom : Cat$	$Cat(tom)$
$(kevin, fluffy) : hasAnimal$	$hasAnimal(kevin, fluffy)$
$(kevin, tom) : hasAnimal$	$hasAnimal(kevin, tom)$
$Cat \sqsubseteq Pet$	$\forall x.Cat(x) \rightarrow Pet(x)$
$\exists hasAnimal.Pet \sqsubseteq NatureLover$	$\forall x.\exists y.hasAnimal(x, y) \wedge Pet(y) \rightarrow NatureLover(x)$

Existential quantifiers in rules' head:

$NatureLover \sqsubseteq \exists hasAnimal.Pet$	$\forall x.NatureLover(x) \rightarrow \exists y.hasAnimal(x, y) \wedge Pet(y)$

Description Logics

- Some other concept-forming operators, and their mapping into Fist Order Logic:

class conjunction	
$C \sqcap D$	$c(X) \wedge d(X)$
class disjunction	
$C \sqcup D$	$c(X) \vee d(X)$
existential restriction	
$\exists R.C$	$\exists Y(r(X,Y) \wedge c(Y))$
class inclusion/subsumption	
$C \sqsubseteq D$	$c(X) \rightarrow d(X)$
$C \equiv D$	$c(X) \leftrightarrow d(X)$
role chains	
$R_1 \circ \ldots R_n \subseteq R$	$r_1(X,X_1) \wedge \cdots \wedge r_n(X_{n-1},X_n) \rightarrow r(X,X_n)$
inverse roles	
$R \equiv S^-$	$r(X,Y) \leftrightarrow s(Y,X)$

- 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 it might to be *undecidable*
  - ...
- `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

KB_{LP}: A simple example in $\mathcal{ALC}$

- The class of Logic Programmers is that of (people) having at least one LP publication, and also whose publications are all LP ones

$$LPer \equiv \exists\, hasPaper.LP\_Publication \sqcap \forall\, hasPaper.LP\_Publication$$

- The class of LP publications is the union of publications in LP conferences or journals

$$LP\_Publication \equiv LP\_Conference \sqcup LP\_Journal$$

- Evelina has a publication in the LP area, but not all her publications are in that area

$$evelina : \exists\, hasPaper.LP\_Publication \sqcap \neg\, (\forall\, hasPaper.LP\_Publication)$$

Usual (set-based) notion of model of a KB and entailment

- We can check whether $KB_{LP} \models evelina : LPer$, and it is not

OWL: Web Ontology Language

- Developed by the W3C Working group, for representing ontologies

Three different sublanguages:

- *OWL-Lite*, based on $\mathcal{SHIF}$, good for thesauri or hierarchies
- *OWL DL*, based on $\mathcal{SHOIN}$, more expressive, computational complete, decidable, availability of practical reasoning algorithms
- *OWL 1.1*, based on $\mathcal{SROIQ}$

A more elaborated example for LPers in $\mathcal{SROIQ}$

- LPers are those (people) having **at least 10 LP publications** [1]

$$LPer \equiv \geq 10\ hasPaper.LP\_Publication$$

- LP publications are LP conference or LP journal publications

$$LP\_Publication \equiv LP\_Conference \sqcup LP\_Journal$$

- Evelina has at least 10 publications in the LP area, and not all her publications are in that area

$$evelina : \geq 10\ hasPaper.LP\_Publication \sqcap \neg (\forall\ hasPaper.LP\_Publication)$$

- I am quite happy now, since, in this theory, I am a LPer!

[1] Qualified number restriction

OWL 2: retaining tractability

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

- OWL QL disallow $\exists$ (fragment of Datalog, tractable, conjunctive queries answered in LogSpace, in AC_0 class)
- OWL EL disallow inverse roles, (tractable, used in ontology with huge DBMS 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)

We have had a nice invited talk by Pascal Hitzler at ICLP2013 about OWL 2

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:

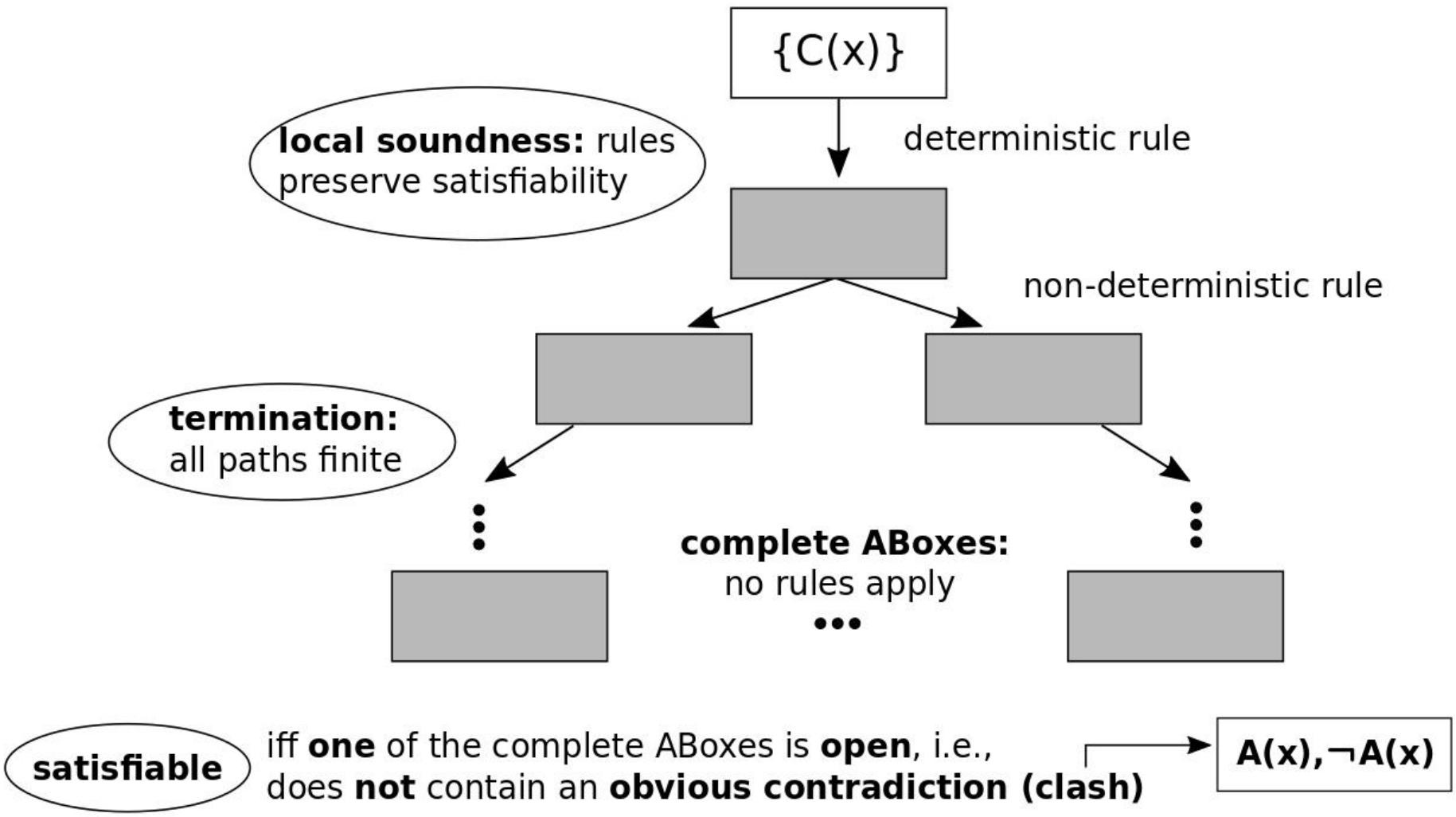
- Concept unsatisfiability
  $\text{KB} \models C \equiv \bot \Leftrightarrow \text{KB} \cup \{C(x)\}$ has no model
- Subsumption
  $\text{KB} \models C \sqsubseteq D \Leftrightarrow \text{KB} \cup \{(C \sqcap \neg D)(x)\}$ has no model
- Instance checking
  $\text{KB} \models C(a) \Leftrightarrow \text{KB} \cup \{\neg C(a)\}$ has no model

Typically it is assumed that the *KB* is in a negation normal form (NNF)

The Tableau Algorithm

- Starts from the Abox $\mathcal{A}$, represented as a graph (the tableau), where individuals are nodes, concepts their labels, and roles arcs between nodes
- 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 looping branches, and ensure termination)

The Tableau Algorithm: OR-search tree



Expansion Rules

$\rightarrow$ *unfold*:	**if** $A \in \mathcal{L}(a)$, A atomic and $(A \sqsubseteq D) \in KB$, **then**
	if $D \notin \mathcal{L}(a)$, **then**
	$\mathcal{L}(a) := \mathcal{L}(a) \cup \{D\}$
	$\tau(D, a) := \tau(A, a) \cup \{(A \sqsubseteq D, a)\}$

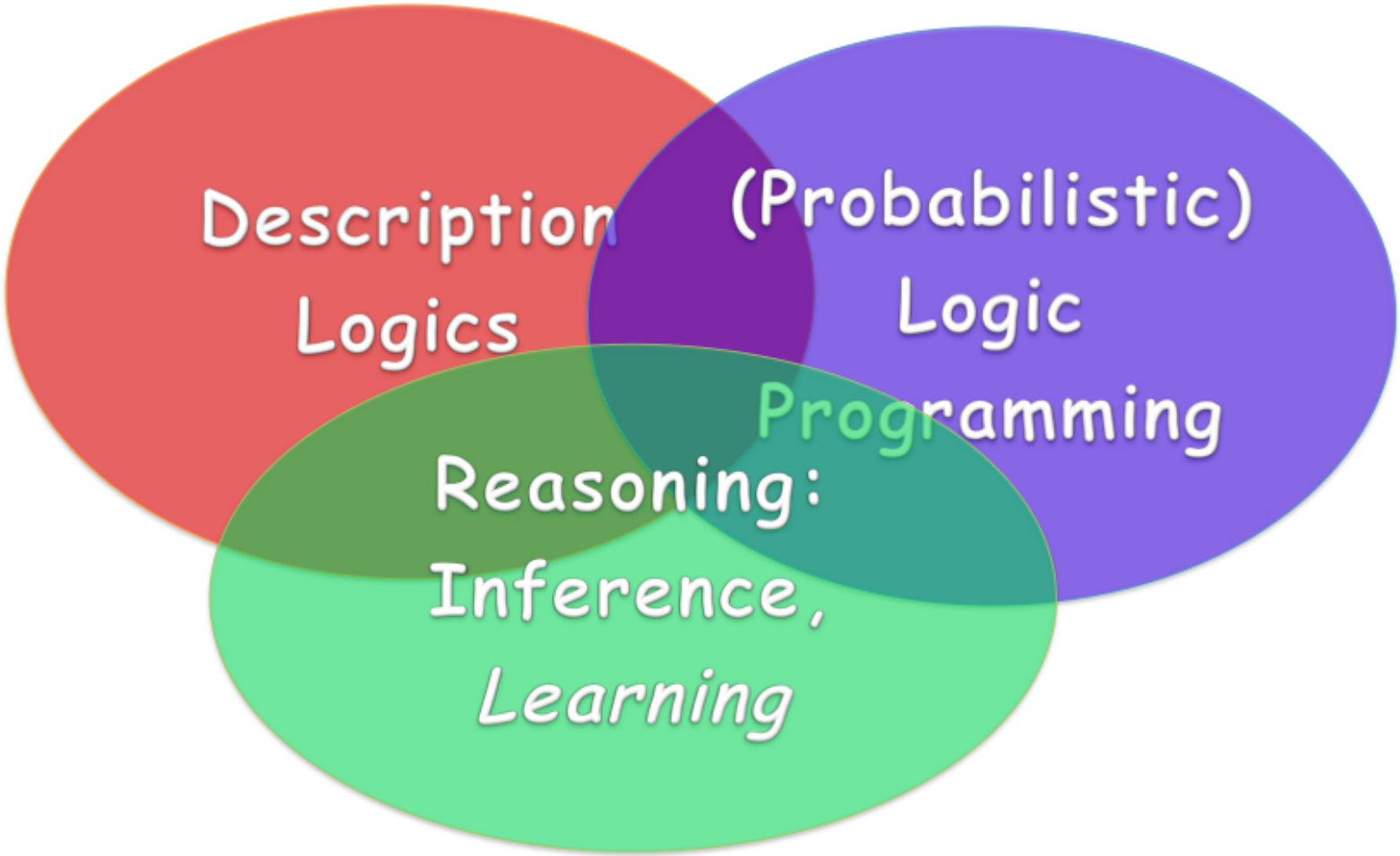
$\rightarrow \sqcup$:	**if** $(C_1 \sqcup C_2) \in \mathcal{L}(a)$ and a is not blocked, **then**
	if $\{C_1, C_2\} \cap \mathcal{L}(a) = \emptyset$, **then**
	Generate graphs $G_i := G$ for each $i \in \{1, 2\}$,
	$\mathcal{L}(a) := \mathcal{L}(a) \cup \{C_i\}$ for each $i \in \{1, 2\}$
	$\tau(C_i,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

- Output: explanation, as set of axioms of the *KB* entailing the query (thanks to the tracing function τ)
- Often, we also want to find all the possible explanations for a query (axiom pinpointing)

Reasoning: why not LP?

- 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., [Beckert & Posegga, JAR 1995], [Meissner, 2004], TRILL implemented in Prolog [Zese et al, URWS2011-13]
  - LP-based inference methods based on resolution (e.g., in the DLog system by [Lukacsy & Szeredi, TPLP 2009])
  - Bottom-up inference methods, e.g., based on Answer Set Programming (e.g., ontoDLP, and its reasoner ontoDLV [Ricca et al., JLC 2009]), or the chase algorithm for Datalog$^{\pm}$ [Calí et al, PODS-DL-IEEE LiCS 2009]

Probabilistic Logic Programming



Probabilistic Logic Programming (PLP)

- Many approaches proposed in the areas of Logic Programming, Uncertainty in Artificial Intelligence, Machine Learning, Databases, Knowledge Representation
- A notable one is Distribution Semantics [Sato, ICLP95]
- A probabilistic logic program defines a probability distribution over normal logic programs (called instances or possible worlds or simply worlds)
- The distribution is extended to a joint distribution over worlds and interpretations (or queries)
- The probability of a query is obtained from this distribution

PLP Languages under the Distribution Semantics

- Probabilistic Logic Programs [Dantsin RCLP91]
- Probabilistic Horn Abduction [Poole NGC93], Independent Choice Logic (ICL) [Poole AI97]
- PRISM [Sato ICLP95]
- Logic Programs with Annotated Disjunctions (LPADs) [Vennekens et al. ICLP04]
- ProbLog [De Raedt et al. IJCAI07]
- They differ in the way they define the distribution over logic programs

Logic Programs with Annotated Disjunctions (LPAD)

$$sneezing(X) : 0.7 \vee null : 0.3 \leftarrow flu(X).$$
$$sneezing(X) : 0.8 \vee null : 0.2 \leftarrow hay\_fever(X).$$
$$flu(bob).$$
$$hay\_fever(bob).$$

- The distribution is over the head of rules
- *null* does not appear in the body of any rule
- Worlds obtained by selecting one atom from the head of every grounding of each clause

A sample LPAD program

- 4 worlds

$sneezing(bob) \leftarrow flu(bob).$
$sneezing(bob) \leftarrow hay\_fever(bob).$
$flu(bob).$
$hay\_fever(bob).$
$P(w_1) = 0.7 \times 0.8$

$null \leftarrow flu(bob).$
$sneezing(bob) \leftarrow hay\_fever(bob).$
$flu(bob).$
$hay\_fever(bob).$
$P(w_2) = 0.3 \times 0.8$

$sneezing(bob) \leftarrow flu(bob).$
$null \leftarrow hay\_fever(bob).$
$flu(bob).$
$hay\_fever(bob).$
$P(w_3) = 0.7 \times 0.2$

$null \leftarrow flu(bob).$
$null \leftarrow hay\_fever(bob).$
$flu(bob).$
$hay\_fever(bob).$
$P(w_4) = 0.3 \times 0.2$

$$P(Q) = \sum_{w \in W_{\mathcal{T}}} P(Q, w) = \sum_{w \in W_{\mathcal{T}}} P(Q|w)P(w) = \sum_{w \in W_{\mathcal{T}} : w \models Q} P(w)$$

- $sneezing(bob)$ is true in 3 worlds
- $P(sneezing(bob)) = 0.7 \times 0.8 + 0.3 \times 0.8 + 0.7 \times 0.2 = 0.94$

Reasoning Tasks in PLP

- Inference: we want to compute the probability of a query given the model and, possibly, some evidence
- Weight learning: we know the structural part of the model (the logic formulas) but not the numeric part (the weights) and we want to infer the weights from data
- Structure learning: we want to infer both the structure and the weights of the model from data

There are many applications (see Luc De Raedt's talk)

Inference for PLP under DS

- Computing the probability of a query (no evidence)
- Knowledge compilation:
  - compile the program to an intermediate representation
    - Binary Decision Diagrams (ProbLog [De Raedt et al. IJCAI07], `cplint` [2] [Riguzzi AIIA07,Riguzzi LJIGPL09], PITA [Riguzzi & Swift ICLP10])
    - deterministic, Decomposable Negation Normal Form circuit (d-DNNF) (ProbLog2 [3] [Fierens et al. TPLP15])
    - 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. ILP09])
- Lifted inference [Poole IJCAI03]

[2] `http://cplint.lamping.unife.it/`

[3] `https://dtai.cs.kuleuven.be/problog/editor.html`

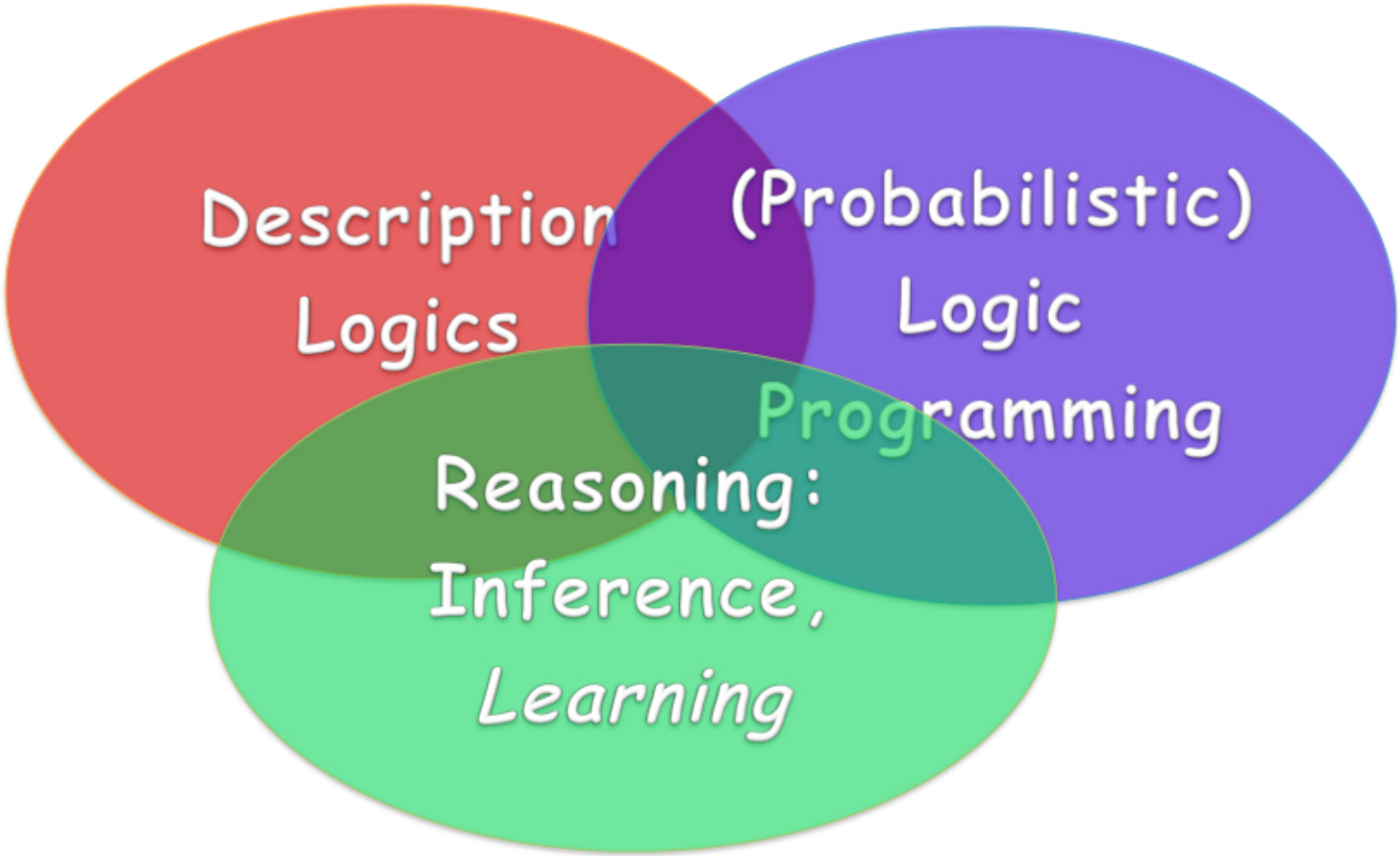
Knowledge Compilation

- Assign 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

Probabilistic Description Logics



Probabilistic Description Logics

- A probabilistic DL in [Koller et al, AAAI97], and a probabilistic extension of OWL in [Ding & Peng, IEEE Hawai Conf. 2004], both based on Bayesian networks
- Statistical terminological knowledge in [Jaeger, KRR94]
- PR-OWL, a framework for building probabilistic ontologies [Laskey & Da Costa, UIA 2005]
- pdl-programs [Lukasiewicz, JAR 2007], combining DLs with Poole's ICL, equipped with answer-set and well-founded semantics
- $P-\mathcal{SHIF}(D)$ and $P-\mathcal{SHOIN}(D)$ [Lukasiewicz, AIJ 2007], semantically based on the notion of probabilistic lexicographic entailment, with algorithms for inference
- Pronto [Klinov, European SW Conf. 2008], a probabilistic reasoner for $P-\mathcal{SHIQ}(D)$ [Lukasiewicz, AIJ 2007]
- CR$\mathcal{ALC}$ [Luna et al., Mexican Conf. on AI 2011] , extending $\mathcal{ALC}$ with statistical axioms

DISPONTE: DIstribution Semantics for Probabilistic ONTologiEs

- Idea: annotate axioms of an ontology with a probability, under the assumption that the axioms are independent each other

$$0.6 :: Cat \sqsubseteq Pet$$

- A probabilistic ontology defines thus a distribution over normal theories (worlds) obtained by including an axiom in a world with a 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; *Cat*s are *Pet*s with probability 0.6.

$$0.4 :: \mathit{fluffy} : \mathit{Cat} \quad (1)$$

$$0.3 :: \mathit{tom} : \mathit{Cat} \quad (2)$$

$$0.6 :: \mathit{Cat} \sqsubseteq \mathit{Pet} \quad (3)$$

- Everyone who has a pet animal ($\mathit{hasAnimal}.\mathit{Pet}$) is a *NatureLover*; *kevin* has two animals, *fluffy* and *tom*.

$$\exists \mathit{hasAnimal}.\mathit{Pet} \sqsubseteq \mathit{NatureLover} \quad (4)$$

$$(\mathit{kevin}, \mathit{fluffy}) : \mathit{hasAnimal} \quad (5)$$

$$(\mathit{kevin}, \mathit{tom}) : \mathit{hasAnimal} \quad (6)$$

- $Q = \mathit{kevin} : \mathit{NatureLover}$ has two explanations [4] :

{ (1), (3) }, { (2), (3) }

[4] Certain axioms are irrelevant for computing query probability

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
Assign a boolean random variable to each axiom
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[5]

- We exploit Binary Decision Diagrams (BDDs) for efficiently computing the probability of a DNF formula

[5] We get a monotone boolean formula, being the underlying DL monotone

Example - `people+pets` ontology

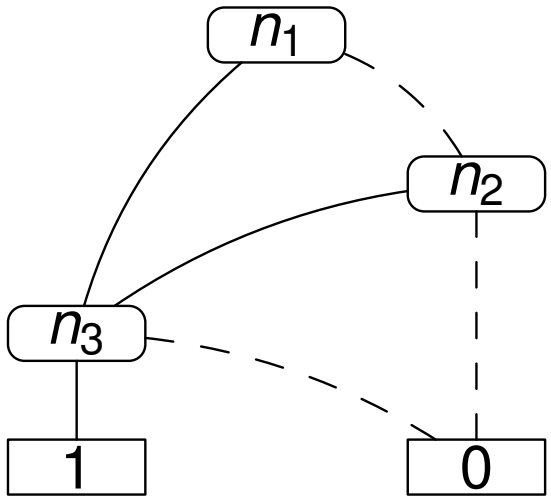
Explanations: { (1), (3) }, { (2), (3) }

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

X_1 0.4

X_2 0.3

X_3 0.6



$$f(\mathbf{X}) = (X_1 \wedge X_3) \vee (X_2 \wedge X_3)$$

- 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 $P(Q = kevin : NatureLover) = P(n_1) = 0.348$.

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. JSW 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`

Why not Prolog?

- The reasoners based on procedural languages have to implement also a backtracking algorithm to find all the possible explanations
  - Example: Pellet (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

TRILL - Tableau Reasoner for descrIption Logics in proLog

- TRILL implements the *tableau algorithm* using *Prolog*
- It resolves the axiom pinpointing problem since we are interested in the set of explanations that entail a query
- Thea2 library for converting OWL ontologies to Prolog
- It applies all the possible expansion rules, first the non-deterministic ones then the deterministic ones
- It returns the set of explanations

TRILL^P - Tableau Reasoner for descrIption Logics in proLog exploiting Pinpointing formula

- TRILL^P solves the axiom pinpointing problem by computing a *pinpointing formula* [Baader & Penaloza, JAR10]
  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:

     $$((F_1 \vee F_3) \wedge F_2) \quad \text{instead of} \quad ((F_1 \wedge F_2) \vee (F_3 \wedge F_2))$$

  3. Convert it into a BDD, so we can compute the probability as in BUNDLE

Experiments

- Comparison between TRILL, TRILL^P and BUNDLE
- We consider four datasets:
  1. BRCA that models the risk factor of breast cancer
  2. An extract of DBPedia
  3. Biopax level 3 that models metabolic pathways
  4. Vicodi that contains information on European history

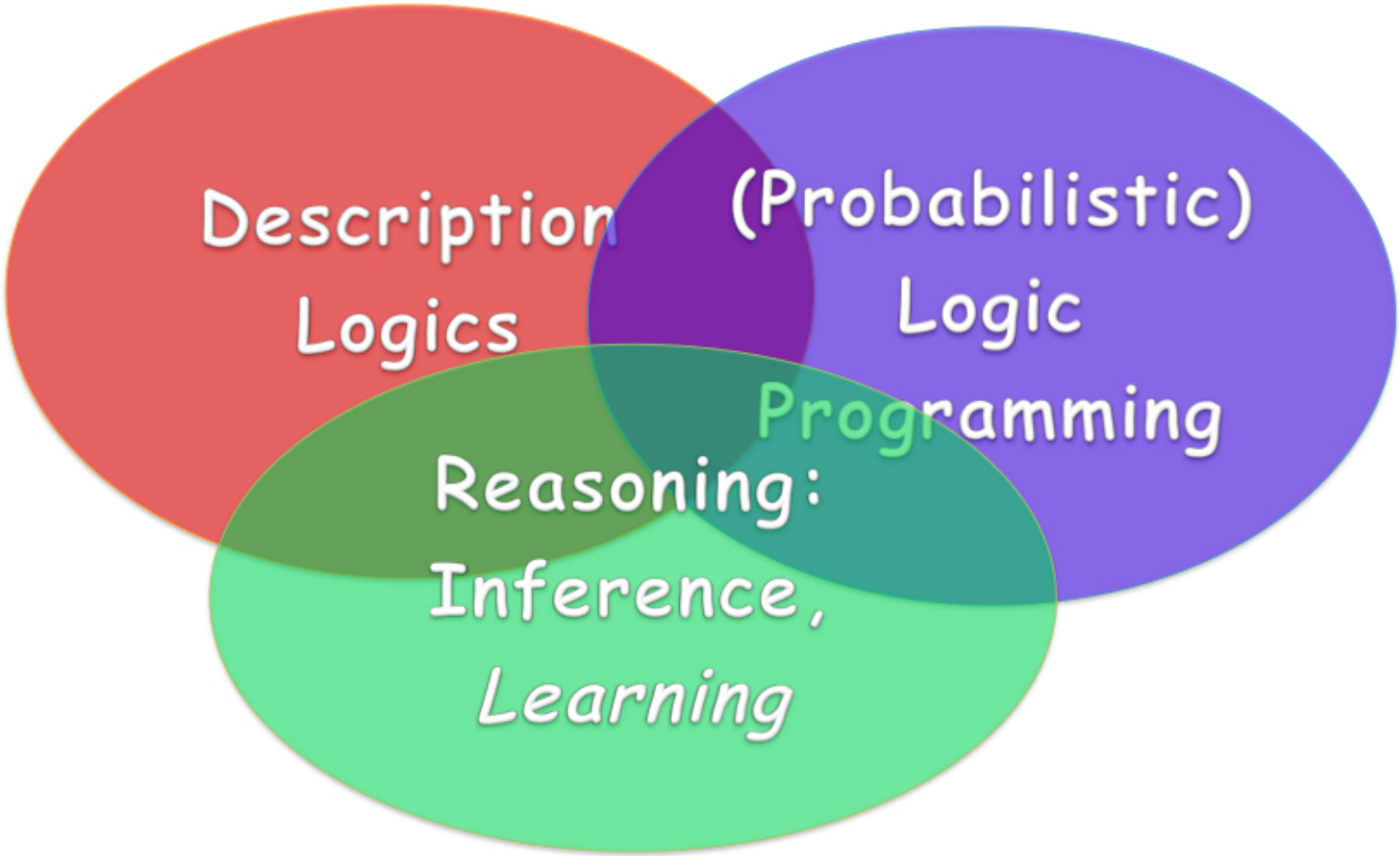
DATASET	TRILL TIME (S)	TRILL^P TIME (S)	BUNDLE TIME (S)
BRCA	27.87	4.74	6.96
DBPedia	51.56	4.67	3.79
Biopax level 3	0.12	0.12	1.85
Vicodi	0.19	0.19	1.12

Table : Average times for computing the probability of queries in seconds

TRILL-on-SWISH: a Web interface

- SWISH [Lager & Wielemaker, TPLP 2014]
  - a recently proposed Web framework for logic programming
  - based on various features and packages of SWI-Prolog
  - allows the user to write Prolog programs and ask queries in the browser
- TRILL-on-SWISH allows users to write a *KB* in the RDF/XML format 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.lamping.unife.it`

Learning Probabilistic Description Logics



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. inspired to EMBLEM [Bellodi & Riguzzi IDA13], an ILP algorithm for learning parameters of PLP programs
  3. EDGE computes the explanations of each example using BUNDLE, then builds the corresponding BDD
  4. Then, EM cycle

- 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. it exploits CELOE [Lehmann et al, JWS 2011], a top-down algorithm which learns (acyclic) concept expressions, given a set of positive and negative examples
  3. then, it exploits EDGE for learning the parameters

EDGE: Experimental results

- Comparison between EDGE-learned ontology and Association Rules (AR) learned as in GoldMiner [Volker & Niepert ESWV11], where rule confidence is taken as probability

educational.data.gov.uk

DBPedia

- Query probability computed using BUNDLE
- 5-fold cross validation

	gov.uk		DBPedia	
	EDGE	ARs	EDGE	ARs
AUCPR[6]	0.9702 ± 0.029	0.8804 ± 0.016	0.9784 ± 0.048	0.5916 ± 0.099
AUCROC[7]	0.9796 ± 0.017	0.9158 ± 0.017	0.9902 ± 0.022	0.4346 ± 0.132

[6] Area Under the Curve Precision Recall

[7] Area Under the Curve Receiver Operating Characteristic

LEAP

- A set of class expressions is generated by using CELOE
- Then EDGE is applied to $\mathcal{K}$ to compute the initial value of the parameters and of the LL.
- Greedy search in the space of theories:
  - For each class expression C, one probabilistic subsumption axiom at a time of the form $p :: C \sqsubseteq \texttt{Target}$ is added to the ontology $\mathcal{K}$;
  - initial probabilty p is the accuracy returned by CELOE for the class expression C
  - 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: Experiments

Carcinogenesis

Original (probabilistic) KB		LEAP	
AUCPR	AUCROC	AUCPR	AUCROC
0.534 ± 0.1082	0.4452 ± 0.0510	0.8006 ± 0.2399	0.798 ± 0.2463

LP-based approaches for DLs

- DLs are very close to logic-based languages
- There is already notable research about LP-based languages and approaches for representation and reasoning on ontologies (e.g., ontoDLP and its reasoner ontoDLV [Ricca et al., JLC 2009], and Datalog$^{\pm}$ [Calí et al, PODS-DL-IEEE LiCS 2009])

- We have borrowed Probabilistic Logic Programming inference (and learning) techniques for (Probabilistic) DLs

- But I am strongly convinced that many (other) LP techniques (e.g., program analysis, and transformation) might be applicable in this domain
- There is one whole area which would greatly benefit from our research and skills!

THANKS FOR LISTENING AND ANY QUESTIONS ?