Explanation-Friendly Query Answering Under Uncertainty

Maria Vanina Martinez

Universidad de Buenos Aires, Argentina

In collaboration with:

Gerardo I. Simari

Universidad Nacional del Sur, Argentina

Consejo Nacional de Investigaciones Científicas y Técnicas (CONICET)

September 23, 2019 – Bolzano, Italy

Quality AI Systems

- What kind of AI systems are we building?
- What kind of AI systems do we want to build?
- Three pillars to understand and produce quality AI systems:
  - Bias
  - Transparency
  - Explicability

Quality AI Systems

- What kind of software systems are we building?
- What kind of software systems do we want to build?
- Three pillars to understand and produce quality software systems:
  - Bias
  - Transparency
  - Explicability

Quality AI Systems

- *Bias*: judgment based on *preconceived* notions or prejudices.
  - In the data, in the model, in the algorithms…
  - Data used to train systems may come from biased/non-representative samples (collection, human labelling, etc.)
  - Function-based systems learn patterns from our data, they may perpetuate inherent cultural bias.
  - Knowledge-based models may also carry out bias…
  - “Good” (e.g., coming from expertise) bias vs “bad” bias.

Quality AI Systems

- Transparency
  - *Auditable* systems ⇒ norms/standards that guarantee levels of "*quality*".
  - Make sure the reasoning/computational process *makes decisions* that can be *traced back*.
  - Clear *assignments of responsibilities*.

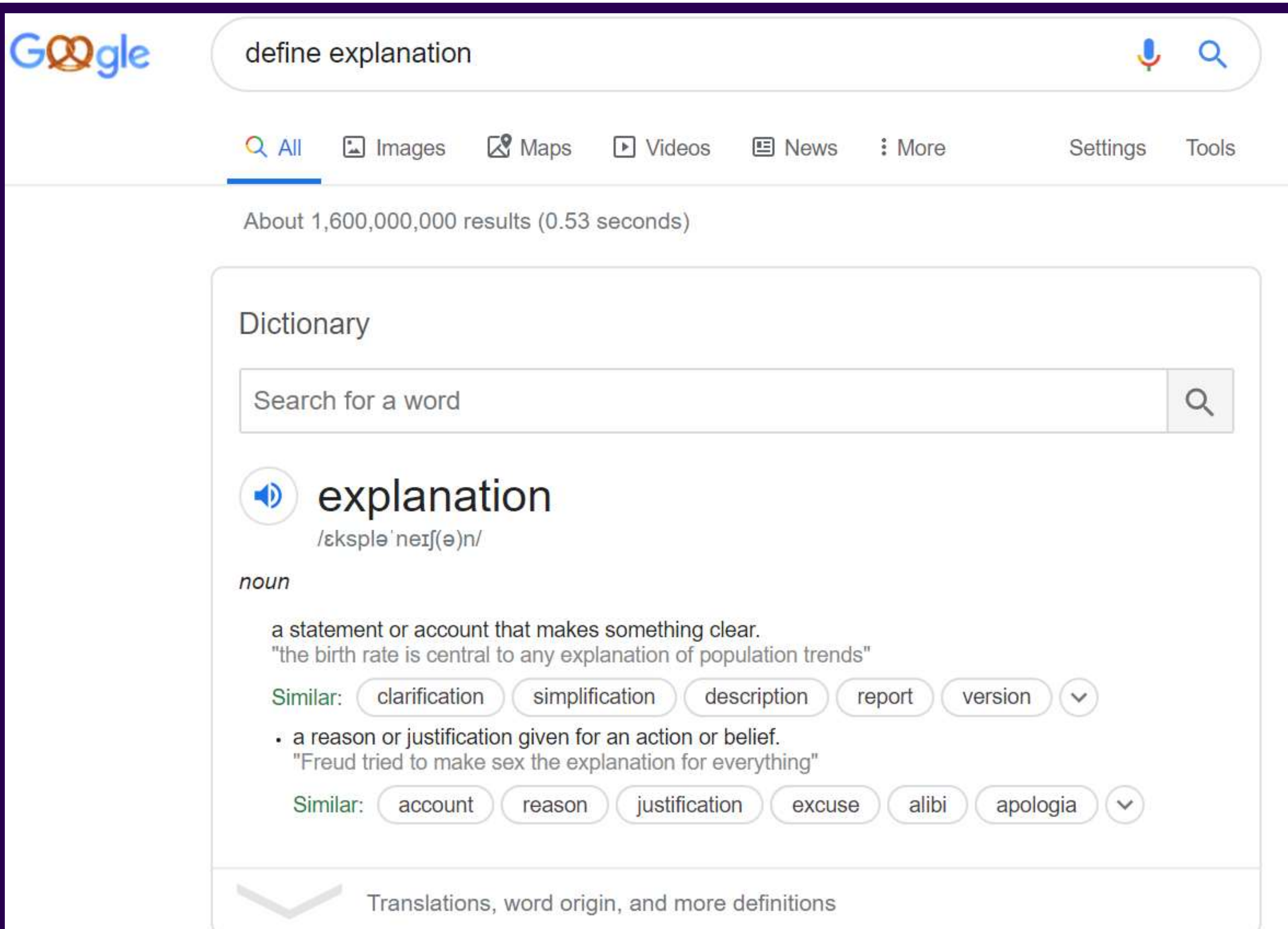
Quality AI Systems

- Explicability
  - Many AI systems are simply *black-boxes*.
  - *Interpretability* is not enough: the extent to which you can predict a model’s result without necessarily trying to understand why or how.
  - Most systems (even those based on explicit knowledge like symbolic AI) are *not designed* to be *questioned* about the decisions they make or how the reasoning process works.

Explanations…Why?

- Explicability
  - Provide some level of transparency (some internal aspects of the system are exposed)
  - To ensure algorithmic fairness
  - Identify *potential bias*/problems in the training data or model
  - To ensure that the algorithms *perform* as *expected*
  - Human-computer interaction: explanations may help in building *trust*

Explanations… What?



Explanations… What?

- The notion of *explanation*, and the related notions of *explainability* and *interpretability*, have been studied for quite some time in philosophy and related disciplines in the social sciences [MILLER2019].

- Explanations are usually consumed by humans:
  - A (human) user would like to know why a certain weather forecast is likely to be true.
  - A (human) user would like the bank employee to explain why they are being denied a loan at the bank.

Explanations… How?

- What is the *form* of an explanation in the setting of a (intelligent) *computational system*?
- What is a *good* or *adequate* explanation in this setting?
- In general terms, we can´t know… it depends on many aspects:
  - The type of system and results: analysis, decision making, actions over the real world.
  - Type of audience: does the user know the system´s mechanics or is it used as a black box? What purpose does the explanation serve for the user? Is the system audited?

Explanations for Decision Making

- In general terms, explanations for conclusions from a reasoning system are typically aimed to:
  - Clarify: ensure the user that the reasoning process is correct.
  - Teach: transfer the knowledge of a certain mechanism so the user can replicate the reasoning process in other situations and contexts.
  - Persuade: convince the user that the conclusion returned is the best in the presence of all valid possibilities.

Static vs Dynamic Explanations

- Static explanations [MOULIN2002,Southwick91]: all the necessary knowledge for the explanation is *available* from the beginning.
  - The explanation is made by means of *a knowledge structure* that justifies the conclusion.
  - More evidence can be provided about how the reasoning *process works* to explain intermediate conclusions.
- This type of explanations are called *fixed* or *based on justifications* (e.g., [Falappa2002,Garcia2013]).

Static vs Dynamic Explanations

- *Dynamic explanations*: they are based on both the *knowledge within the system* and the *knowledge from the user*.
  - The user can *ask* for *additional information* and question the reasoning process itself.
  - This can be done by means of *questions* that guide the explanation itself.
- This type of explanations involve an *interactive mechanism*, usually based on some kind of *controlled dialogue*.

This talk today…

Two *knowledge-based frameworks* to handle *uncertainty* (in Datalog+/- ontologies):

- Probabilistic reasoning
- Inconsistency-tolerant semantics for query answering
- How can we use the *knowledge* contained in the models to *explain* their behavior and results?

Uncertainty

- *Uncertainty* appears everywhere in the Web:
  - Inherent uncertainty: *inherent* to a particular domain (e.g, weather forecast)
  - Uncertainty coming from *automatic processing of data* (e.g., automatic integration of schemas or datasets)
  - Uncertainty coming form the presence of *inconsistency* and *incompleteness*
- At the moment, browsers and other Web technologies do not manage uncertainty in a *principled* way.

Uncertainty

- Goal: fill the gap by developing of tools that can be applied to perform different tasks in the Web; for instance, in *semantic search*.
- One way to do this is by integrating *ontology languages* with *databases technologies* and *probabilistic models*.
- In this class we will cover:
  - Some *probabilistic models* that can be useful to model Web content.
  - *Algorithms* for query answering: classic (exact probability), threshold, and *ranking*.
  - *Scalable but expressive fragments* of the language/model.

Example

- Consider the problem of entity extraction over the following text snippet:

Probabilistic Models

- Probabilistic Graphical Models (*PGMs*) are graph-based structures that are use to represent knowledge about a uncertain domain.
- Representation:
  - Nodes: random *variables*
  - Arcs: probabilistic *dependencies* among variables; if there is no arc between two variables then it means that the variables are conditionally independent.

Probabilistic Models

Some well known and used types of PGMs:

- Bayes Nets (*BNs*)
- Markov Networks / *Markov Random Fields* (*MRFs*)
- Markov Logic Networks (*MLNs*)
- Markov Chains (*MCs*)
- ….

Probabilistic Models

Some well known and used types of PGMs:

- Bayes Nets (*BNs*)
- Markov Networks / *Markov Random Fields* (*MRFs*)
- Markov Logic Networks (*MLNs*)
- Markov Chains (*MCs*)
- ….

Probabilistic Models:

Markov Networks

Markov Networks (MRFs)

A Markov Network (or *Markov Random Field*, MRF) is a non directed graph where:

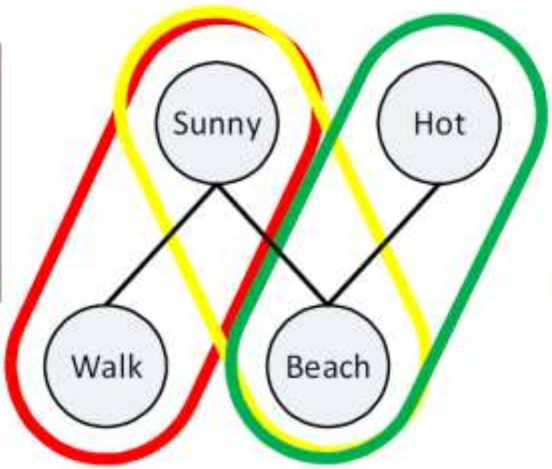
- every *node* represents a discrete random variable;
- *arcs* correspond to a notion of direct probabilistic *interaction*; this interaction is parameterized with *potential functions* (there is a potential function for every maximal clique);
- *potentials*: non-negative real functions over the variables in each clique (the state of the clique);
- a node is *conditionally independent* from the rest of the nodes in the graph given the values of its immediate neighbors (the *Markov blanket* of the node).

Example

Variables:

- Sunny (the day is sunny)
- Hot (the day is hot)
- Beach (we go to the beach)
- Walk (we go for a walk)

Clique 1		
S	W	Ø(H,B)
t	f	2
t	t	1.7
f	t	0.3
f	f	3.1



Clique 2		
H	B	Ø(H,B)
t	f	1.6
t	t	3
f	t	0.4
f	f	2.5

Clique 3		
B	S	Ø(H,B)
t	f	2.8
t	t	1.7
f	t	0.2
f	f	2.8

Markov Networks (MRFs)

The *joint distribution* of variables $X = \{X_1, X_2, \dots, X_n\}$ can be defined as follows:

$$P(X = x) = \frac{1}{Z} \prod_i \phi_i(x_{\{i\}})$$

where ϕ_i is the potential function and $x_{\{i\}}$ is the state of the i-th maximal clique.

Z is a *normalizing constant* so the sum of all probabilities adds up to 1:

$$Z = \sum_{x \in X} \prod_i \phi_i(x_{\{i\}})$$

Example

We can calculate the probability that it is *sunny* and *hot*, and that we go to the *beach* but *don't take a walk*:

$$P(s \wedge h \wedge b \wedge \neg w) = \frac{1}{Z}(2 \times 3 \times 1.7) = \frac{10.2}{Z}$$

Markov Networks (MRFs)

- Problem: expressing a value for each state of each clique is *exponential* in the size of the model.
- We can obtain a more *compact* representation by means of functions called *features*.
- For instance, the *log-linear* model defines:

$$P(X = x) = \frac{1}{Z} e^{\sum_i w_i f_i(x)}$$

where the i vary over the set of cliques:

$$Z = \sum_{x \in X} e^{\sum_i w_i f_i(x)}$$

Markov Networks (MRFs)

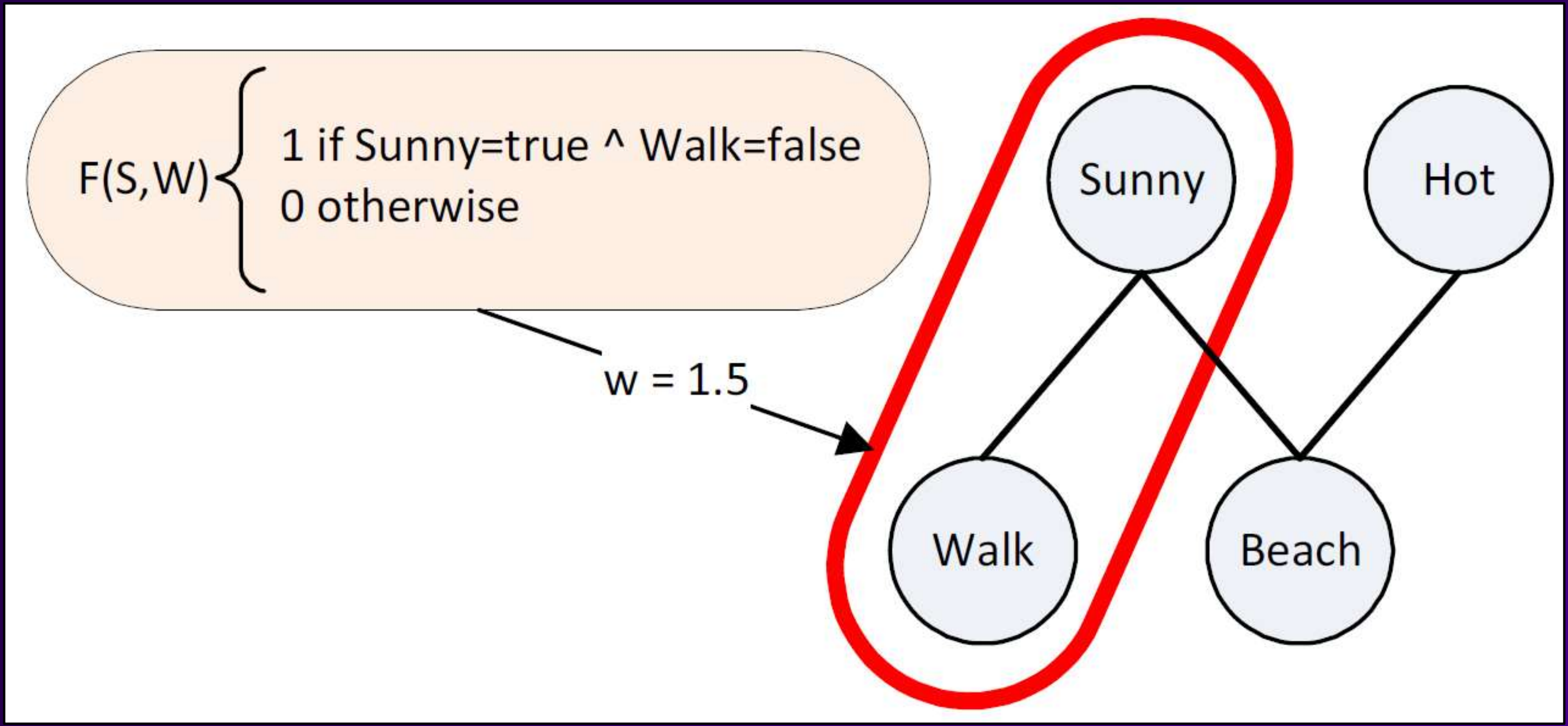
- *Features* $f_i(x)$ (also real functions of the state) replace the *potentials*.
- Each $f_i(x)$ has associated a *weight* w_i
- Here we consider *binary* features: $f_i(x) \in \{0,1\}$.
- The more direct translation from the previous form to this one is:

  a feature corresponding to each possible state $x_{\{i\}}$

  of each clique, with weight $\ln \phi_i(x_{\{i\}})$.

Example

Coming back to our running example, we can define a simple feature for the clique $\{$*Sunny*, *Walk*$\}$ in the following way:



Probabilistic Models:

Markov Logic Networks

(or Markov Logic)

Markov Logic Networks (MLNs)

An MLN is a finite set of *pairs* (F_i, w_i), where:

- F_i is a *formula* in FOL
- w_i is a real number (the *weight* of the formula)

Together with a finite set of *constants* $C = \{c_1, c_2, \dots, c_n\}$, it defines an *MRF* $M_{L,C}$ in the following way:

- $M_{L,C}$ contains a binary node for every possible *basic instance* of an *atom* in L. The value of the node is 1 if the atom is true, and 0 otherwise.

- $M_{L,C}$ contains a *feature* for each *basic instance* of *formulas* F_i in L. The value of the feature is 1 if the formula is true, or 0 otherwise, the weight is the value w_i associated with F_i in L.

Markov Logic Networks (MLNs)

Observations:

- Basic atoms generate the *node* in the network.
- There is an *arc* between two nodes if and only if the basic atoms *appear together* in at least one basic instance of a formula in L.
- The *formulas* generate *cliques* in the network.

Example

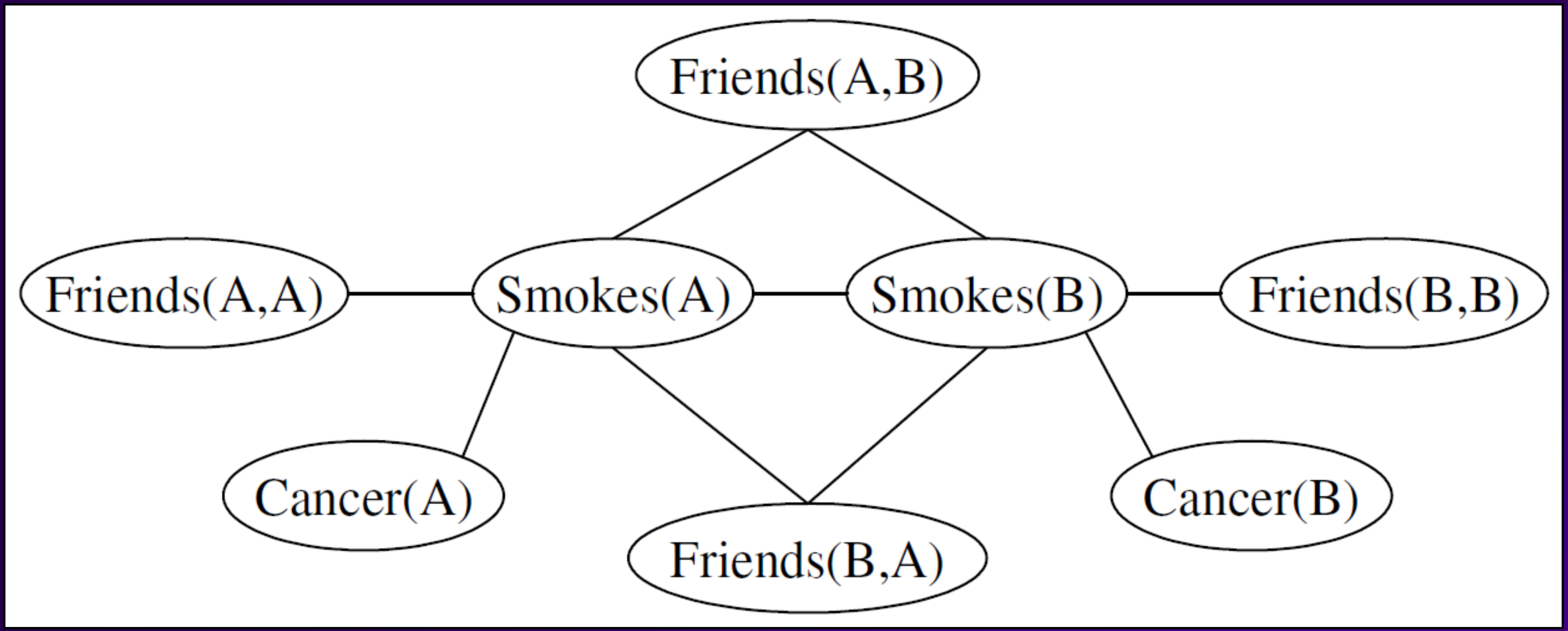
- Consider the MLN defined by the pairs:
  - $(\forall x\ Sm(x) \Rightarrow Ca(x), 1.5) \leadsto$ Smoking causes cancer
  - $(\forall x\ \forall y\ Fr(x,y) \Rightarrow \big(Sm(x) \Leftrightarrow Sm(y)\big), 1.1) \leadsto$ if two people are friends, then either both smoke or neither does.

  Let´s take the constants : $\{Anna, Bob\}$.

- $M_{L,C}$ can be now be used to infer the *probability* of $Anna$ and Bob being friends given their smoking habits; the probability of Bob having cancer given his friendship with $Anna$, etc.

Example

The following graph corresponds to the *induced MRF*:



Formulas: $\forall x\ Sm(x) \Rightarrow Ca(x)$, $\forall x\ \forall y\ Fr(x,y) \Rightarrow \big(Sm(x) \Leftrightarrow Sm(y)\big)$

Markov Logic Networks (MLNs)

The probability *distribution* represented by the MLN is the following:

$$P(X = x) = \frac{1}{Z} e^{\sum_i w_i n_i(x)}$$

where $n_i(x)$ is the *number* of basic instances of F_i that are satisfied by x, and Z is the normalization constant.

Another Example

- Let´s define an MLN with the following *pair*:

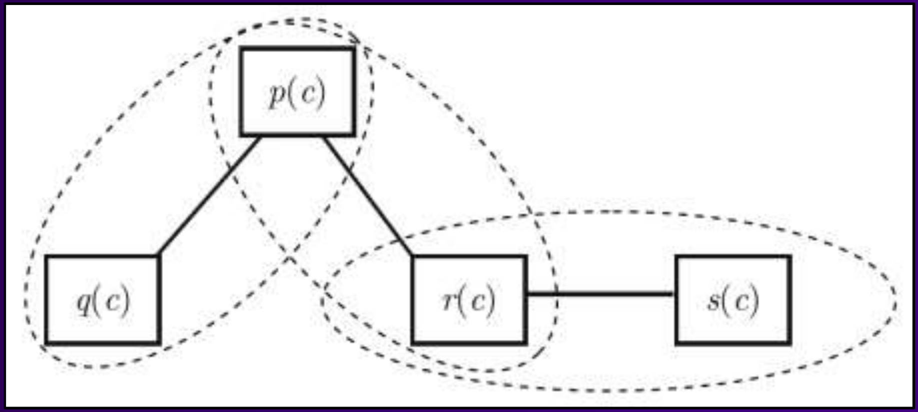
$$\psi_1 : (\, p(X) \Rightarrow q(X)\, , 1.2)$$

$$\psi_2 : (\, p(X) \Rightarrow r(X)\, , 2)$$

$$\psi_3 : (\, s(X) \Rightarrow r(X)\, , 3)$$

and the set of *constants* $\{c\}$.

- The set of *basic atoms* $\{p(c), q(c), r(c), s(c)\}$, and the graph:



Another Example

We have then $2^4 = 16$ possible *value assignments* for the variables in the MRF. The probabilities are:

λ_i	$p(c)$	$q(c)$	$r(c)$	$s(c)$	Satisfies	Potential	Probability
1	*false*	*false*	*false*	*false*	ψ_1, ψ_2, ψ_3	$1.2 + 2 + 3 = 6.2$	$e^{6.2}/Z \approx 0.127$
2	*false*	*false*	*false*	*true*	ψ_1, ψ_2	$1.2 + 2 = 3.2$	$e^{3.2}/Z \approx 0.006$
3	*false*	*false*	*true*	*false*	ψ_1, ψ_2, ψ_3	$1.2 + 2 + 3 = 6.2$	$e^{6.2}/Z \approx 0.127$
4	*false*	*false*	*true*	*true*	ψ_1, ψ_2, ψ_3	$1.2 + 2 + 3 = 6.2$	$e^{6.2}/Z \approx 0.127$
5	*false*	*true*	*false*	*false*	ψ_1, ψ_2	$1.2 + 2 = 3.2$	$e^{3.2}/Z \approx 0.006$
6	*false*	*true*	*false*	*true*	ψ_1, ψ_2	$1.2 + 2 = 3.2$	$e^{3.2}/Z \approx 0.006$
7	*false*	*true*	*true*	*false*	ψ_1, ψ_2, ψ_3	$1.2 + 2 + 3 = 6.2$	$e^{6.2}/Z \approx 0.127$
8	*false*	*true*	*true*	*true*	ψ_1, ψ_2, ψ_3	$1.2 + 2 + 3 = 6.2$	$e^{6.2}/Z \approx 0.127$
9	*true*	*false*	*false*	*false*		0	$e^{0}/Z \approx 0$
10	*true*	*false*	*false*	*true*		0	$e^{0}/Z \approx 0$
11	*true*	*false*	*true*	*false*	ψ_2, ψ_3	$2 + 3 = 5$	$e^{5}/Z \approx 0.038$
12	*true*	*false*	*true*	*true*	ψ_2, ψ_3	$2 + 3 = 5$	$e^{5}/Z \approx 0.038$
13	*true*	*true*	*false*	*false*	ψ_1, ψ_3	$1.2 + 3 = 4.2$	$e^{4.2}/Z \approx 0.017$
14	*true*	*true*	*false*	*true*	ψ_1	1.2	$e^{1.2}/Z \approx 0$
15	*true*	*true*	*true*	*false*	ψ_1, ψ_2, ψ_3	$1.2 + 2 + 3 = 6.2$	$e^{6.2}/Z \approx 0.127$
16	*true*	*true*	*true*	*true*	ψ_1, ψ_2, ψ_3	$1.2 + 2 + 3 = 6.2$	$e^{6.2}/Z \approx 0.127$

Another Example

- The normalization factor Z is computed as follows:

$$Z = 7e^{6.2} + 3e^{3.2} + 2e^{0} + 2e^{5} + e^{4.2} + e^{1.2} \approx 3891.673$$

- To compute the probability of formula $\mathrm{p}(c) \wedge q(c)$, we have to *sum* the probabilities of all the *worlds* that satisfy it, that is 13, 14, 15, and 16:

$$\frac{e^{4.2} + e^{1.2} + e^{6.2} + e^{6.2}}{Z} \approx \frac{1055.5}{3891.673} \approx 0.271$$

Probabilistic Datalog+/− Ontologies

Probabilistic Datalog+/–

- *Goal*: to combine "classic" Datalog+/- with probabilistic models (in this class we use as example MLNs).
- The basic idea is to *annotate* formulas with sets of probabilistic events:
  - Annotations means that the given formula only applies whenever the event occurs.
  - The probability distribution associated to the events is described by means of an MLN (or any other probabilistic model).
- We are going to see different types of queries, as different kinds of explanations may be needed for different queries: *ranking queries, conjunctive queries,* and *threshold queries*.

Datalog+/−

- A database (instance) D over $\mathcal{R}$ is a set of atoms with predicates from $\mathcal{R}$ and arguments from Δ.

$$D = \{emp(bob), manager(bob), directs(bob, hr), emp(ann), supervises(bob,ann), manager(ann), works\_in(ann,hr), works\_in(bob,hr), works\_in(bob,finance)\}$$

- A conjunctive query (CQ) over $\mathcal{R}$ has the form:
$Q(\mathbf{X}) = \exists \mathbf{Y}\ \Phi(\mathbf{X},\mathbf{Y})$, where Φ is a conjunction of atoms.

$$Q(X) = manager(X) \wedge directs(X,hr) \qquad X = \ldots$$

- A boolean conjunctive query (CQ) over $\mathcal{R}$ has the form:
$Q() = \exists \mathbf{X},\mathbf{Y}\ \Phi(\mathbf{X},\mathbf{Y})$, where Φ is a conjunction of atoms.

$$Q() = \exists X\ manager(X) \wedge directs(X,hr) \qquad Yes\ /\ No$$

Datalog+/–

- Answers to queries are defined via homomorphisms, which are mappings $\mu\colon \Delta \cup \Delta_N \cup \mathcal{V} \rightarrow \Delta \cup \Delta_N \cup \mathcal{V}$ s.t.:
  - $c \in \Delta$ implies $\mu(c) = c$
  - $c \in \Delta_N$ implies $\mu(c) \in \Delta \cup \Delta_N$
  - μ is extended to atoms, sets of atoms, and conjunctions.
- The set of answers $Q(D)$ is the set of tuples t over Δ s.t. $\exists\mu\colon \mathbf{X} \cup \mathbf{Y} \rightarrow \Delta \cup \Delta_N$ s.t. $\mu\big(\Phi(\mathbf{X},\mathbf{Y})\big) \subseteq D$, and $\mu(\mathbf{X}) = t$.

  For $Q(X) = \mathit{manager}(X) \wedge \mathit{directs}(X,hr)$, the set of all answers over D is $Q(D) = \{bob\}$.

The answer to $Q() = \exists X\, \mathit{manager}(X) \wedge \mathit{directs}(X,hr)$ is Yes.

Datalog+/−

- Tuple-generating Dependencies (TGDs) are constraints of the form $\forall \mathbf{X} \forall \mathbf{Y}\ \Phi(\mathbf{X},\mathbf{Y}) \rightarrow \exists \mathbf{Z}\ \Psi(\mathbf{X},\mathbf{Z})$ where Φ and Ψ are atomic conjunctions over $\mathcal{R}$ called the *body* and *head* of the TGD, respectively.

- Example TGDs:

$$manager(M) \rightarrow emp(M)$$

$$manager(M) \rightarrow \exists P\ directs(M,P)$$

$$emp(E) \wedge directs(E,P) \rightarrow$$
$$\exists E'\ emp(E') \wedge supervises(E,E') \wedge works\_in(E',P)$$

Datalog+/–

- Given a DB D and a set Σ of TGDs, the set of models $mods(D, \Sigma)$ is the set of all B s.t.:
  - $D \subseteq B$
  - every $\sigma \in \Sigma$ is satisfied in B.
- The set of answers for a CQ Q to D and Σ, $ans(Q,D,\Sigma)$, is the set of all tuples a s.t. $a \in Q(B)$ for all $B \in mods(D, \Sigma)$.
- Answers can be computed via the chase, a procedure for repairing a DB relative to a set of dependencies.

The Chase

- (Informal) TGD Chase rule:
  - a TGD σ is applicable in a DB D if $body(\sigma)$ maps to atoms in D
  - if not already in D, the application of σ on D adds an atom with "fresh" nulls corresponding to each existentially quantified variable in $head(\sigma)$.
- The (possibly infinite) chase is a universal model: there exists a homomorphism from $chase(D, \Sigma)$ onto every $B \in mods(D, \Sigma)$.
- Therefore we have that $D \cup \Sigma \models Q$ iff $chase(D, \Sigma) \models Q$.
- If Σ consists of certain restricted sets of TGDs, CQs can be evaluated on a fragment of constant depth $k \cdot |Q|$, which is PTIME in the data complexity.

Different types of complexity

Depending on the part of the ontology that we consider *fixed*, we have different types of complexity:

- Combined: *nothing* is fixed.
- ba-combined: the *arity* of the relational symbols are considered fixed.
- Data: the *schema* and the *query* are considered fixed.
- Query: the *schema* and *database instance* are considered fixed.

Chase

Input: Database instance D, set of TGDs Σ

Output: A model of $D \cup \Sigma$

D

Person(john)

Σ

$\forall P\ person(P) \rightarrow \exists F\ father(F,P) \qquad \forall F \forall P\ father(F,P) \rightarrow person(F)$

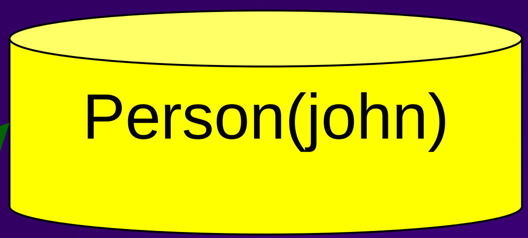
$chase(D,\Sigma) = D \cup\ ?$

Chase

Input: Database instance D, set of TGDs Σ

Output: A model of $D \cup \Sigma$

D



Σ

$\forall P\ person(P) \rightarrow \exists F\ father(F,P) \qquad \forall F \forall P\ father(F,P) \rightarrow person(F)$

$chase(D,\Sigma) = D \cup \{father(z_1,john)\}$

Chase

<u>Input</u>: Database instance D, set of TGDs Σ

<u>Output</u>: A model of $D \cup \Sigma$

D

Person(john)

Σ

$\forall P\ person(P) \rightarrow \exists F\ father(F,P)$ $\quad$ $\forall F \forall P\ father(F,P) \rightarrow person(F)$

$chase(D,\Sigma) = D \cup \{father(z_1,john),\ person(z_1)$

Chase

Input: Database instance D, set of TGDs Σ

Output: A model of $D \cup \Sigma$

D

Person(john)

Σ

$\forall P\ person(P) \rightarrow \exists F\ father(F,P)$ $\quad$ $\forall F \forall P\ father(F,P) \rightarrow person(F)$

$chase(D,\Sigma) = D \cup \{father(z_1,john),\ person(z_1),\ father(z_2,z_1)\}$

Chase

Input: Database instance D, set of TGDs Σ

Output: A model of $D \cup \Sigma$

D

Person(john)

Σ

$\forall P\ person(P) \rightarrow \exists F\ father(F,P)$ $\quad$ $\forall F \forall P\ father(F,P) \rightarrow person(F)$

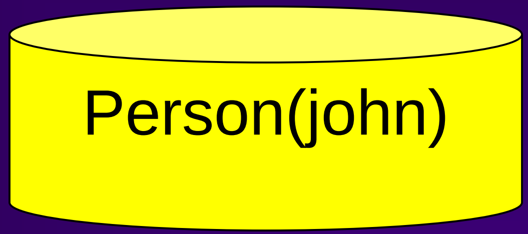
$$chase(D,\Sigma) = D \cup \{father(z_1,john),\ person(z_1),\ father(z_2,z_1),\ldots\}$$

Chase

Input: Database instance D, set of TGDs Σ

Output: A model of $D \cup \Sigma$

D

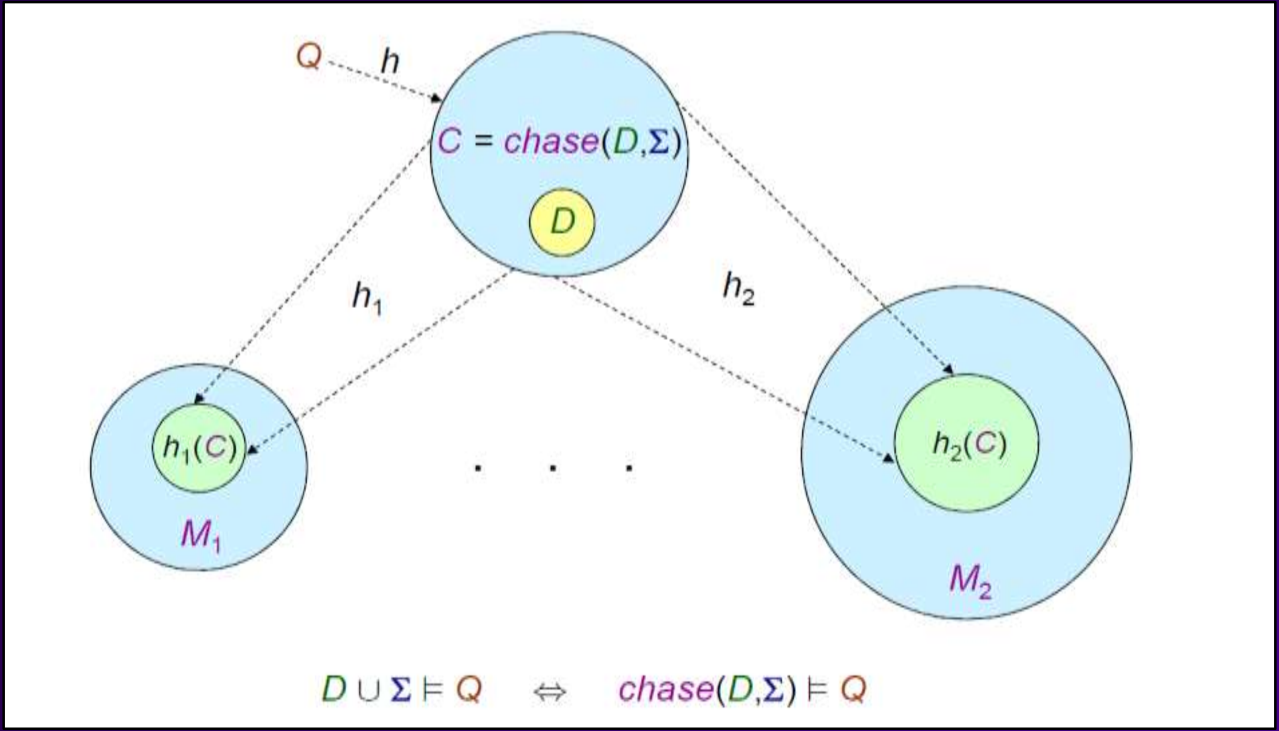


$chase(D,\Sigma) = D \cup \{father(z_1,john),\ person(z_1),\ father(z_2,z_1),\ldots\}$

INFINITE INSTANCE

Query Answering vía the chase

- The possible infinite chase is a *universal model*: there exists a homomorphism from $chase(D, \Sigma)$ to every $B \in mods(D, \Sigma)$.
- Therefore, we have that $D \cup \Sigma \vDash Q$ iff $chase(D, \Sigma) \vDash Q$.



Negative Constraints and EGDs

- Negative constraints (NCs) are formulas of the form $\forall \mathbf{X}\ \Phi(\mathbf{X}) \rightarrow \bot$, where $\Phi(\mathbf{X})$ is a conjunction of atoms.
- NCs are easy to check, since we can simply verify that the CQ $\Phi(\mathbf{X})$ has an empty set of answers.
- Equality Generating Dependencies (EGDs) are of the form $\forall \mathbf{X}\ \Phi(\mathbf{X}) \rightarrow X_i = X_j$, where Φ is a conjunction of atoms and X_i, X_j are variables from $\mathbf{X}$.
- The Chase w.r.t. both TGDs and EGDs is easily extended.
- Here, we assume that EGDs are separable, which intuitively means that EGDs and TGDs are independent of each other.

Guarded Datalog+/–

- A TGD is guarded if there exits an atom in the body that contains all variables that appear in the body.

$$\forall \mathrm{X} \forall \mathrm{Y} \forall \mathrm{Z} \ \underbrace{R(\mathrm{X},\mathrm{Y},\mathrm{Z})}_{guard}, \ S(\mathrm{Y}), \ P(\mathrm{X},\mathrm{Z}) \rightarrow \exists \mathrm{W} \ \ Q(\mathrm{X},\mathrm{W})$$

- The chase has a *finite treewidth* $\Rightarrow$ query answering is decidable
- Query answering is PTIME-complete in data complexity.
- Extends ELH DL (same data complexity).

Guarded Datalog+/–

EL TBox	Datalog$^{\pm}$ Representation
$A \sqsubseteq B$	$\forall X\, A(X) \rightarrow B(X)$
$A \sqcap B \sqsubseteq C$	$\forall X\, A(X), B(X) \rightarrow C(X)$
$\exists R.A \sqsubseteq B$	$\forall X\, R(X,Y),\, A(Y) \rightarrow B(X)$
$A \sqsubseteq \exists R.B$	$\forall X\, A(X) \rightarrow \exists Y\, R(X,Y),\, B(Y)$
$R \sqsubseteq P$	$\forall X \forall Y\, R(X,Y) \rightarrow P(X,Y)$

Linear Datalog+/–

- A TGD is *linear* if there is only one atom in the body.

$$\forall \mathbf{X} \forall \mathbf{Y} \;\; R(\mathbf{X},\mathbf{Y}) \rightarrow \exists \mathbf{Z} \;\; Q(\mathbf{X},\mathbf{Z})$$

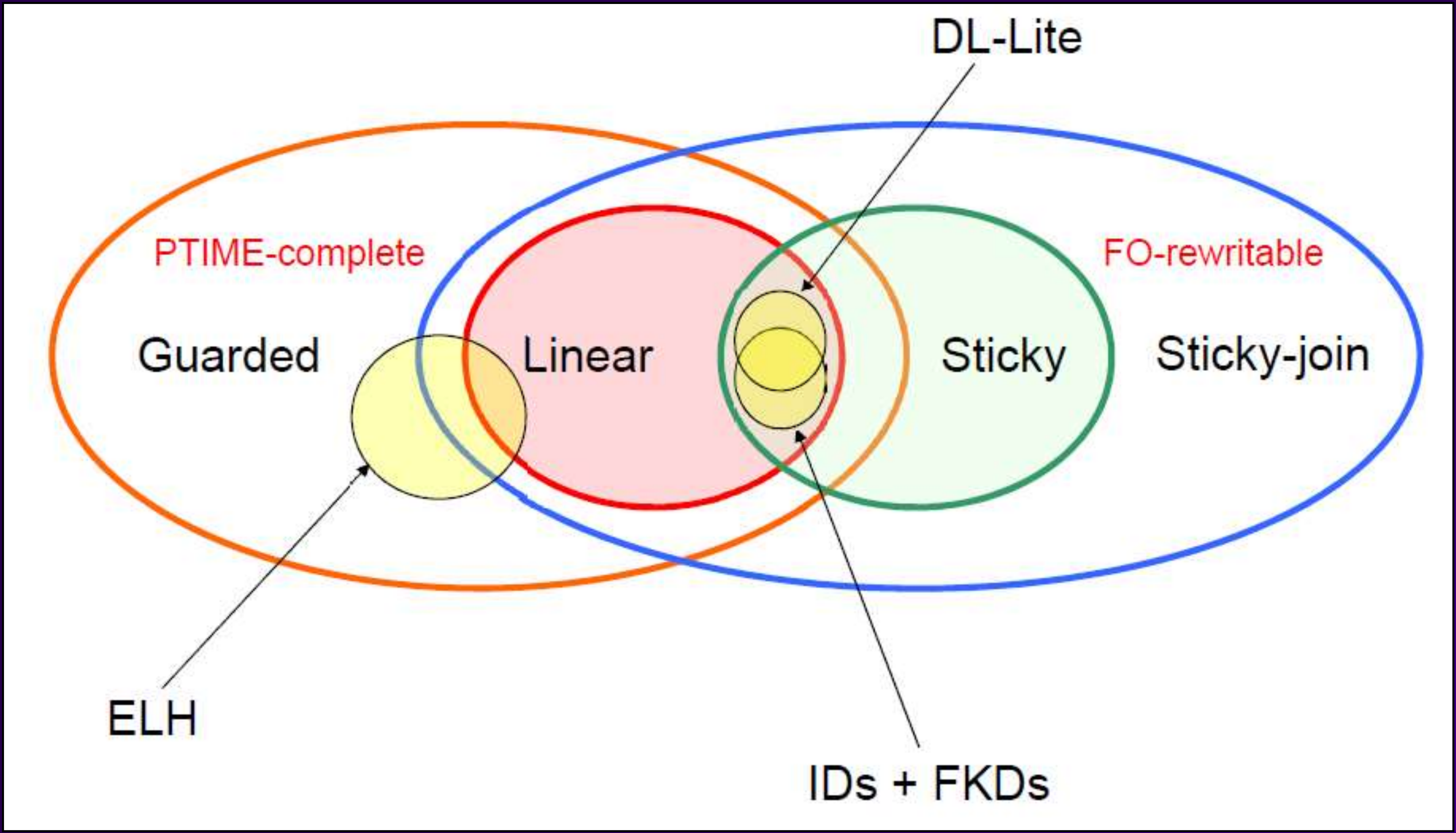
$guard$

- Linear TGDs are (trivially) *guarded.*
- Query answering is in AC_0 in data complexity (*FO rewritablity*).
- Extends the family of *DL-Lite* DLs (same data complexity).

Linear Datalog+/–

DL-Lite TBox	Datalog$^{\pm}$ Representation
$A \sqsubseteq B$	$\forall X\, A(X) \rightarrow B(X)$
$A \sqsubseteq \exists R$	$\forall X\, A(X) \rightarrow \exists Y\, R(X,Y)$
$\exists R \sqsubseteq A$	$\forall X \forall Y\, R(X,Y) \rightarrow A(X)$
$R \sqsubseteq P$	$\forall X \forall Y\, R(X,Y) \rightarrow P(X,Y)$

Datalog+/– Overview



Datalog+/– Overview

	Data	Fixed Σ	Combined
Guarded	PTIME-complete	NP-complete	2EXPTIME-complete
Linear	in AC_0	NP-complete	PSPACE-complete
Sticky	in AC_0	NP-complete	EXPTIME-complete
Sticky-join	in AC_0	NP-complete	EXPTIME-complete

- Same complexity if we consider NCs and non-conflicting EGDs.
- Same complexity for finite models.

Example

- Consider the example at the beginning modeled as an MLN:

$$\phi_1: ann(S_1,I_1,num) \wedge ann(S_2,I_2,X) \wedge overlap(I_1,I_2) \quad : 3$$

$$\phi_2: ann(S_1,I_1,shop) \wedge ann(S_2,I_2,mag) \wedge overlap\,(I_1,I_2) \quad : 1$$

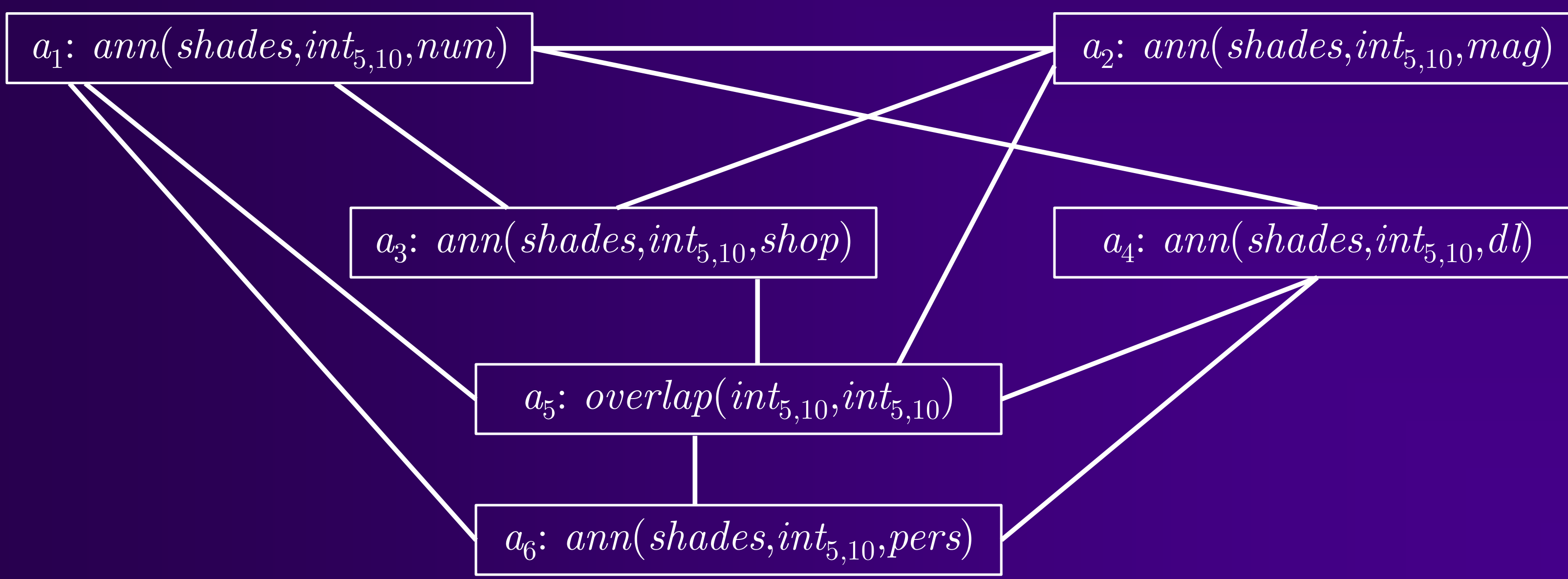
$$\phi_3: ann(S_1,I_1,dl) \wedge ann(S_2,I_2,pers) \wedge overlap(I_1,I_2) \quad : 0.25$$

Example

- Consider the example at the beginning modeled as an MLN:

$$\phi_1\text{: } ann(S_1,I_1,num) \wedge ann(S_2,I_2,X) \wedge overlap(I_1,I_2) \quad : 3$$
$$\phi_2\text{: } ann(S_1,I_1,shop) \wedge ann(S_2,I_2,mag) \wedge overlap\ (I_1,I_2) \quad : 1$$
$$\phi_3\text{: } ann(S_1,I_1,dl) \wedge ann(S_2,I_2,pers) \wedge overlap(I_1,I_2) \quad : 0.25$$

- Graph representation:



Example

- Computing probabilities w.r.t. this MLN:

λ_i	a_1	a_2	a_3	a_4	a_5	a_6	SAT	Probability
1	False	False	False	False	False	False	–	$e^0 \;/\; Z$
2	False	False	False	True	True	True	ϕ_3	$e^{0.25} \;/\; Z$
3	True	False	False	True	True	True	ϕ_1, ϕ_3	$e^{3+0.25} \;/\; Z$
4	True	False	True	True	True	True	ϕ_1, ϕ_3	$e^{3+0.25} \;/\; Z$
5	False	True	False	False	True	False	–	$e^0 \;/\; Z$
6	False	True	True	False	True	True	ϕ_2	$e^1 \;/\; Z$
7	False	True	True	True	True	True	ϕ_2, ϕ_3	$e^{1+0.25} \;/\; Z$
8	True	True	True	True	True	True	ϕ_1, ϕ_2, ϕ_3	$e^{3+1+0.25} \;/\; Z$

. . . (64 possible settings for the binary random variables)

Probabilistic Datalog+/– Ontologies

- A probabilistic Datalog+/- ontology consists of a classical Datalog+/- ontology O along with an MLN M.

  Notation: $KB = (O, M)$

- Formulas in O are annotated with a set of pairs $\langle X_i = x_i \rangle$, with $x_i \in \{true, false\}$ (we also use 0 and 1, respectively).
- Variables that don't appear in the annotation are unconstrained.
- Possible world: a set of pairs $\langle X_i = x_i \rangle$ where each $X_i \in \mathrm{X}$ has a corresponding pair.
- Intuition: given a possible world, a subset of the formulas in O is induced.

Probabilistic Datalog+/– Ontologies

- A probabilistic Datalog+/- ontology consists of a classical Datalog+/- ontology O along with an MLN M.

  Notation: $KB = (O, M)$

- Formulas in O are annotated with a set of pairs $\langle X_i = x_i \rangle$, with $x_i \in \{true, false\}$ (we also use 0 and 1, respectively).

- In tightly coupled ontologies, we allow annotations to contain variables, which can also appear in the formulas:

  Example: $number(X)$: $\{ann(X,I,num) = true\}$

- Though this increases expressivity, it causes the number of worlds to depend on the size of the database.

Example Revisited

The following formulas were adapted from the previous examples to give rise to a probabilistic Datalog+/- ontology:

$$book(X) \rightarrow editorialProd(X) \quad : \{\}$$

$$magazine(X) \rightarrow editorialProd(X) \quad : \{\}$$

$$author(X) \rightarrow person(X,P) \quad : \{\}$$

$$descLogic(X) \wedge author(X) \rightarrow \bot \quad : \{ann(\mathrm{X},I_1,dl) = 1 \wedge ann(\mathrm{X},I_2,pers) = 1 \; overlap(I_1,I_2) = 0\}$$

$$shop(X) \wedge editorialProd(X) \rightarrow \bot \quad : \{ann(\mathrm{X},I_1,shop) = 1 \wedge ann(\mathrm{X},I_2,mag) = 1 \; overlap(I_1,I_2) = 0\}$$

$$number(X) \wedge date(X) \rightarrow \bot \quad : \{ann(\mathrm{X},I_1,num) = 1 \wedge ann(\mathrm{X},I_1,date) = 1 \; overlap(I_1,I_2) = 0\}$$

Formulas with an empty annotation always hold.

Queries

There are three kinds of queries that have been proposed in this model:

1) Threshold queries: all ground atoms that have probability at least p, where p is specified as an input of the query.

> Answer to threshold query $Q = (\Phi,\ p)$ (with $p \in [0{,}1]$): set of all ground atoms a with $Pr_{\Phi}(a) \geq p$.

Example: *Refer to the lecture notes.* Consider probabilistic ontology $\Phi = (O,\ M)$ from Example 4, and threshold query $Q = (\Phi,\ 0.15)$. See Figure 5 for the computation of the probabilities.

We have that $Pr_{\Phi}(a(x_1)) \approx 0.191$ and $Pr_{\Phi}(d(x_3)) \approx 0.135$.

Therefore, $a(x_1)$ belongs to the output, while $d(x_3)$ does not.

Queries

There are three kinds of queries that have been proposed in this model:

2) Ranking queries: the ranking of atomic consequences based on their probability values.

> Answer to ranking query $Q = rank(\Phi)$: tuple $ans(Q) = a_1, \ldots, a_n$ such that $\{a_1, \ldots, a_n\}$ are all of the atomic consequences of O_λ for any $\lambda \in Worlds(M)$, and $i < j \Rightarrow Pr_\Phi(a_i) > Pr_\Phi(a_j)$.

Example: *Refer to the lecture notes.*

The answer to query $rank(\Phi)$ is: $\langle a(x_1),\ c(x_1),\ d(x_3),\ b(x_2),\ d(x_2)\rangle$

Queries

There are three kinds of queries that have been proposed in this model:

3) Probabilistic Conjunctive Queries: answers are computed classically and accompanied by the probability value with which it is entailed by Φ.

Example: *Refer to the lecture notes.*

The answer to query $Q(X) = a(X) \wedge c(X)$ is $(x_1, 0.191)$

UBA Buenos Aires CONICET

Summary of Probabilistic Datalog+/-

- Uncertainty in rules is expressed by means of annotations that refer to an underlying Markov Logic Network.
- The goal is to develop a language and algorithms capable of managing uncertainty in a principled and scalable way.
- Scalability in the framework rests on two pillars:
  - We combine scalable rule-based approaches from the DB literature with annotations reflecting uncertainty;
  - Many possibilities for heuristic algorithms; MLNs are flexible, and sampling techniques may be leveraged.

Explaining Probabilistic Uncertainty

- We now explore the question:

  What constitutes an *explanation* for a query to a probabilistic Datalog+/- KB?

- In general, the answer to this question will depend heavily on *whom* the explanation is intended for.
- How can we use the explicit knowledge from the model to explain answers and query answering?
- We now analyze some *basic building blocks* and discuss some approaches that unless stated otherwise apply to all three kinds of queries.

Annotated chase

The chase *data structure* used to answer queries can be *annotated* to keep track of the probabilistic *events* that must hold; two ways of doing this are:

1) Annotate each node with a *Boolean array* of size $|Worlds(M)|$; during the execution of the chase procedure, annotations are *propagated* as inferences are made. This is best for cases in which:

- The *number of worlds* is not excessively large, since the space used by the chase structure will grow by a factor of $|Worlds(M)|$;
- when a *sampling-based* approach is used to approximate: the size of each array can be reduced to (a function of) the number of samples.
- for *tractable probabilistic models*, this representation can be used to clearly obtain either the exact or approximate probability mass associated with each node of interest.

Annotated chase

2) Annotate each node with a *logical formula* expressing the *conditions* that must hold for the node to be inferrable.

- More *compact* than the array-based method: *size* of formulas are *bounded* by the length of the *derivation* and the length of the original *annotations* in the probabilistic ontology.
- On the other hand, extracting the *specific worlds* that make up the probabilistic mass associated with a given atom (or set of atoms for a query) is essentially equivalent to solving a *#SAT problem*.
- For *tractable probabilistic models* there is a greater chance of performing feasible computations, though the *structure* of the resulting logical formula depends greatly on how *rules interact*.

Probabilities of atomic formulas

The *annotated chase* yields several tools that facilitate the provision of an explanation for the probability of an *atom*:

- Different derivation paths leading to the *same result* (can be summarized).
- *Example branches*, e.g. highlighting well-separated ones to show variety.
- Common aspects of worlds that make up *most of the probability mass* (e.g., atoms in the probabilistic model that appear in most derivations).

To provide a *balanced* explanation, we can also focus on the cases in which the atom in question is *not derived*.

All of these elements are available *independently* of the specific probabilistic model used in the KB.

Probabilities of more complex queries

Probabilistic conjunctive queries

- The basic building blocks described for atomic queries can be leveraged for this *more complex* case.
- Depending on the kind of annotated chase graph used the probability of a set of atoms that must be true *at once* can be derived from that of each individual member.
- Opportunities for explanations of why a query is *derived* or *not derived* may also include selecting one or more elements of the conjunction that are responsible for *lowering* the resulting probability of the query.

Probabilities of more complex queries

Ranking queries

- Fundamental component of the answer: *relationship* between the probabilities of atoms.
- The most important question to answer regarding *explanations* of such results is thus:

  For a given pair of atoms (a, b) such that a is ranked above b, why is it $a > b$ and not $b > a$?

- The basic elements discussed above can be used to shed light on this aspect.

Probabilities of more complex queries

Ranking queries (cont.)

- Sampling-based methods yield *probability intervals* instead of point probabilities.
- The *width* of the resulting interval will be a function of the *number* and *probability mass* of the worlds taken into account vs. those left out.
- Explanations can involve *examples* or *summaries* of how the probability mass gets to a *minimum* (lower bound) and, conversely, why the *maximum* (upper bound) is *not higher*.

References

1. T. Lukasiewicz, M. V. Martinez, G. Orsi, and G.I. Simari: "Heuristic Ranking in Tightly Coupled Probabilistic Description Logics". Proc. of UAI 2012.
2. G. Gottlob, T. Lukasiewicz, M.V. Martinez, and G.I. Simari: "Query Answering Under Uncertainty in Datalog+/- Ontologies". AMAI, 2013.
3. T. Lukasiewicz, M.V. Martinez, and G.I. Simari: "Exact and Approximate Query Answering in Tightly Coupled Probabilistic Datalog+/-". *In Preparation*.
4. Project PrOQAW (EPSRC): `http://www.cs.ox.ac.uk/projects/PrOQAW/`
5. Project DIADEM (ERC): `http://diadem.cs.ox.ac.uk/`
6. Nyaya Ontological Query Answering System: `http://mais.dia.uniroma3.it/Nyaya/`
7. ProbCog MLN Toolbox: `http://ias.cs.tum.edu/research/probcog`

Inconsistency Tolerant Reasoning with

Datalog+/− Ontologies

Inconsistency

- The presence of inconsistency in systems that manipulate knowledge cannot be ignored and sometimes it is not clear how to get rid of it.
- We need to live with conflicting information.
- Challenge: to interpret the constantly increasing amount of heterogeneous and dynamic data that come from disparate sources and domains.
- Goal: manage the inconsistency at query answering time by means of *reasonable semantics* and *computationally efficient methods.*

In the rest of this class…

- The notion of inconsistency in ontological languages such as Datalog+/-
- Consistent Query Answering for Datalog+/-
- Approximate Consistent Query Answering
- Going beyond repairs (novel approaches)
- Explanations for Inconsistency-Tolerant Semantics

Inconsistency in Datalog+/−

- We focus in the notion of logical inconsistency, that is a logic theory is inconsistent iff it has no models.
  - Given a Datalog+/− ontology (D, Σ), we say that (D, Σ) is *inconsistent* iff $mods(D,\Sigma) = \varnothing$ (sometimes we will write it as $(D, \Sigma) \vDash \bot$).
- In Datalog+/−, inconsistency appears as the results of the violation of the integrity constraints (NCs and EGDs).
  - $chase(D,\Sigma) \vDash body(\nu)$, for some $\nu \in \Sigma_E \cup \Sigma_{NC}$

Inconsistency in Datalog+/–

- ***Important***: we assume that TGDs are *correct*; that is, they correctly capture the semantics of the domain.
- This assumption implies:
  - The set of TGDs is *always* satisfiable; given Σ, there always exists a database instance D such that $mods(D,\Sigma) \neq \varnothing$.
  - Conflicts arise because the data is *wrong* $\Rightarrow$ the database instance is the part of the ontology that needs to be changed or *repaired* if we want to restore consistency.
- This is not the only option! Other works in the literature consider alternative assumptions (e.g., repair the set of TGDs, or TGDs and data together).

Repairs

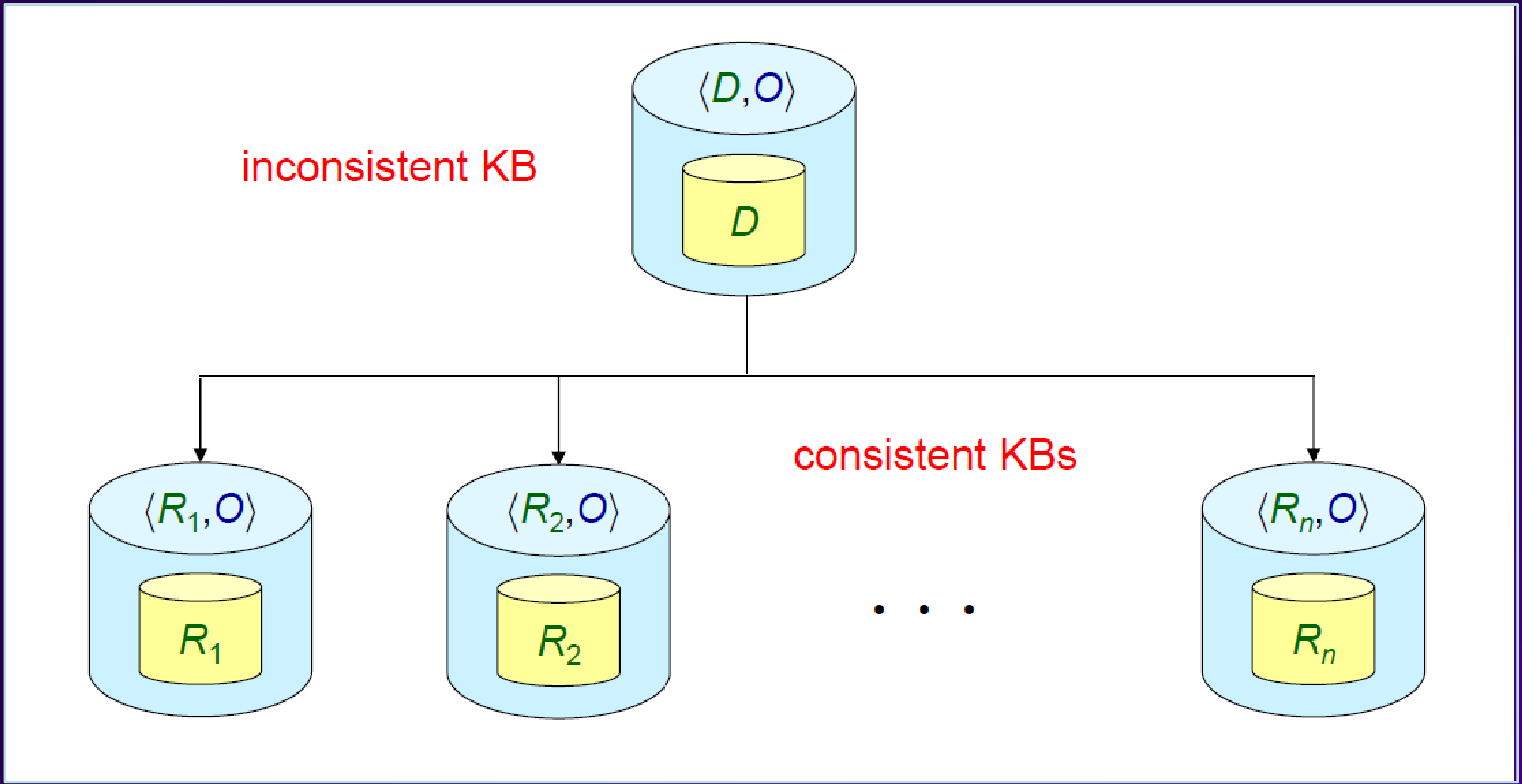
- General definition:

  Given a Datalog+/− ontology (D, Σ), a *repair* for/of (D, Σ) is another ontology (D', Σ), such that $mods(D,\Sigma) \neq \varnothing$ and (D', Σ) is "*as close as possible to*" a (D, Σ).

- The notion of *closeness* changes depending on the expressive power of the language and different assumptions over the application domain.

- A *data* (ABox) *repair* for (D, Σ) is a database instance D' such that:
  - (1) $D' \subseteq D$,
  - (2) $mods(D',\Sigma) \neq \varnothing$, and
  - (3) there is no $D'' \subseteq D$ such that $D' \subseteq D''$ and $mods(D'',\Sigma) \neq \varnothing$.

Repairs



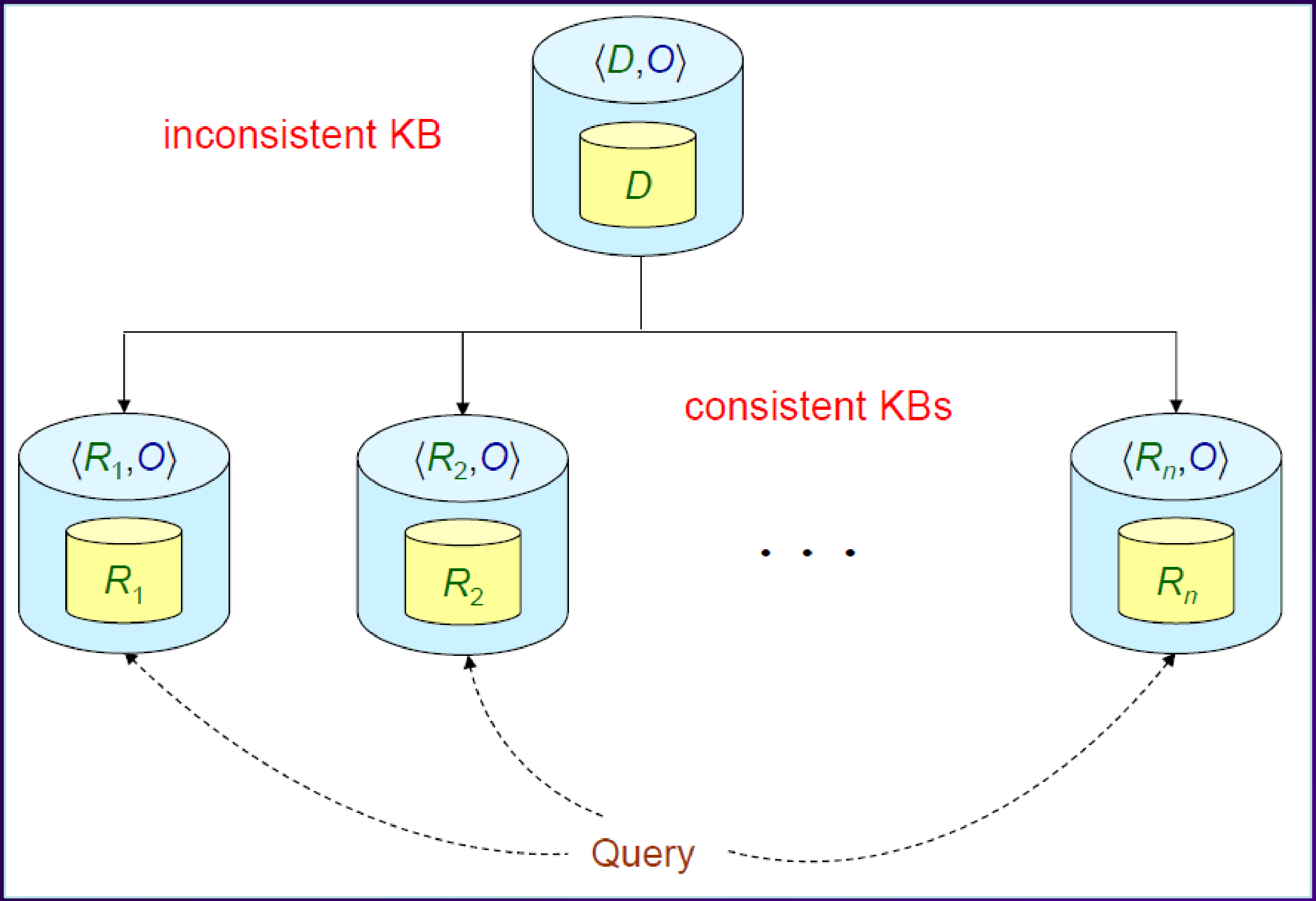
Consistent Query Answering

- We review some inconsistency-tolerant semantics for query answering (some originally designed for *RDBMSs* and others defined specifically for *OBDA*).
- *Consistent Query Answering* [ABC99], adapted as ***AR*** *semantics for Description Logics and rule-based ontological languages* [LemboRR10]
- *Approximations to AR*:
  - *IAR*, *CAR*, *ICAR* [LemboRR11] and *ICR* [BienvenuAAAI12]
  - k*-defeater* and k*-support* [BRIJCAI13]
- *Lazy Answers* [LMSECAI12]

AR Semantics[LemboRR10]

- AR semantics is inspired on *consistent answers* (CQA) for RDBMS.
- It is based on the notion of data repairs, the idea is not to fix the database instance but to consider on-the-fly all possible ways of repairing it.
- Given $KB = (D, \Sigma)$ and a CQ Q, we say that $KB \models_{AR} Q$ iff $(R, \Sigma) \models Q$ for every repair $R \in Rep(KB)$.
- It is a *cautions approach*, similar to the notion of certain answers.
- To decide if $KB \models_{AR} Q$ (even for atomic queries) is coNP-complete for linear Datalog+/−.

AR Semantics[LemboRR10]



AR Semantics: Example

$$D = \{player(lio),\ striker(lio),\ coach(pep),\ coach(lio),$$
$$midfielder(pep)\}$$

$$\Sigma_T = \{player(X) \rightarrow teamMember(X),\ striker(X) \rightarrow player(X),$$

$$coach(X) \rightarrow teamMember(X),\ striker(X) \rightarrow plays(X, forward),$$

$$midfielder(X) \rightarrow plays(X, midfield),\ midfielder(X) \rightarrow player(X)\}$$

$$chase(D,\Sigma) = D \cup \{teamMember(lio),\ teamMember(pep),$$
$$plays(lio, forward),\ plays(pep, midfield),\ player(pep)\}$$

$$\Sigma_{NC} = \{player(X) \wedge coach\ (\mathrm{X}) \rightarrow \bot\}$$

$$\Sigma_E = \{coach(X) \wedge coach(\mathrm{Y}) \rightarrow \mathrm{X} = \mathrm{Y}\}$$

AR Semantics [LemboRR10]

$$D = \{player(lio),\ striker(lio),\ coach(pep),\ coach(lio),\ midfielder(pep)\}$$

We have 3 data repairs:

$$R_1 = \{player(lio),\ striker(lio),\ coach(pep)\}$$

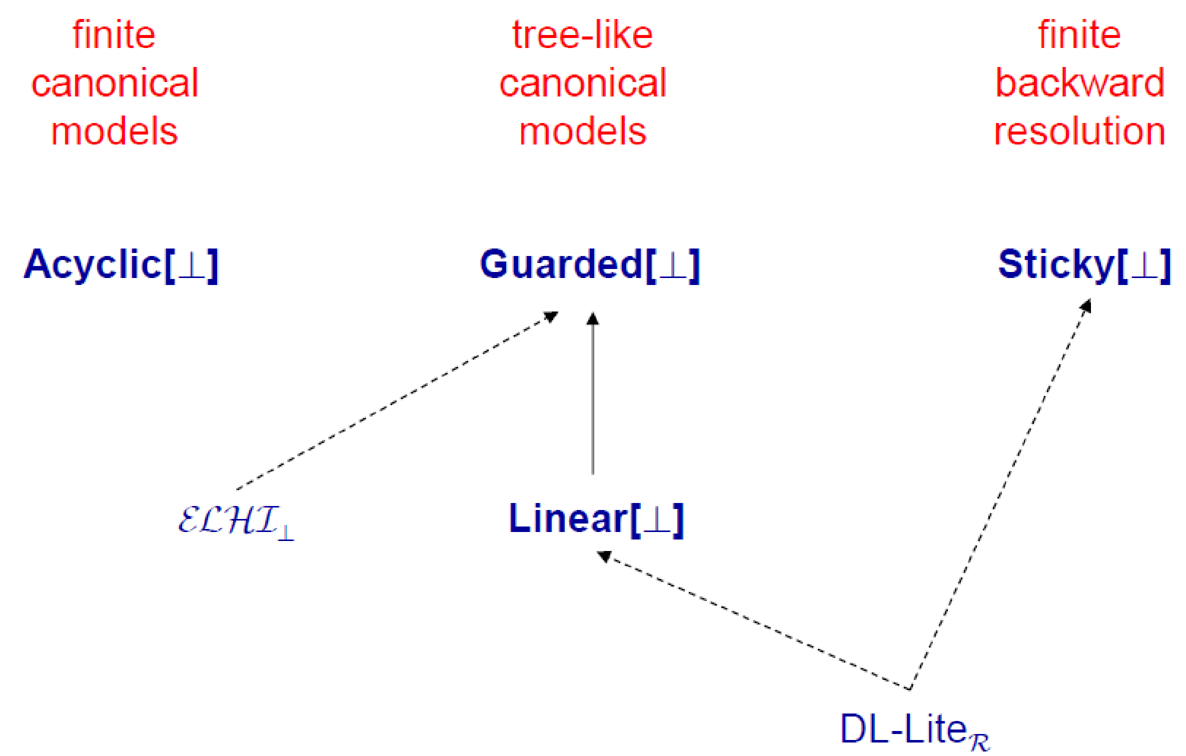
$$R_2 = \{player(lio),\ striker(lio),\ midfielder(pep)\}$$

$$R_3 = \{\ coach(lio),\ midfielder(pep)\ \}$$

$$KB \vDash_{AR} \exists X\ teamMember(X) \wedge player(X)$$

$$KB \nvDash_{AR} \exists X\ plays(pep, X)$$

AR Semantics: Complexity



plus negative constraints - $\forall \mathbf{X}\ (\varphi(\mathbf{X}) \rightarrow \perp)$

AR Semantics: Complexity

	Combined	Bounded Arity	Fixed Ontology	Data
Acyclic[⊥]	NEXP - P^{NE}	NEXP - P^{NE}	$\Pi_{p,2}$	coNP
Guarded[⊥]	2EXPTIME	EXPTIME	$\Pi_{p,2}$	coNP
Linear[⊥]	PSPACE	$\Pi_{p,2}$	$\Pi_{p,2}$	coNP
Sticky[⊥]	EXPTIME	$\Pi_{p,2}$	$\Pi_{p,2}$	coNP

From classic QA to AR

	Combined	Bounded Arity	Fixed Ontology
Acyclic[⊥]	NEXPTIME	NEXPTIME	NP
Guarded[⊥]	2EXPTIME	EXPTIME	NP
Linear[⊥]	PSPACE	NP	NP
Sticky[⊥]	EXPTIME	NP	NP

Complexity AR (No ∃ in the head)

	Combined	Bounded Arity	Fixed Ontology
Acyclic[⊥]	PSPACE	$\Pi_{p,2}$	$\Pi_{p,2}$
Guarded[⊥]	EXPTIME	$\Pi_{p,2}$	$\Pi_{p,2}$
Linear[⊥]	PSPACE	$\Pi_{p,2}$	$\Pi_{p,2}$
Sticky[⊥]	EXPTIME	$\Pi_{p,2}$	$\Pi_{p,2}$

Consistent Query Answering:

Approximations to AR

Approximations to AR

Goal: manage the inconsistency by means of *reasonable semantics* and *computationally efficient methods.*

- We could argue if AR is a reasonable/meaningful semantics or not: too cautious?
- Complexity wise… we saw AR is not likely to work in practice.

- Given two semantics X and Y, and $KB = (D,\Sigma)$ we say X *is a sound approximation to* Y iff for every query Q, if $KB \models_X Q$ then $KB \models_Y Q$.
- We say X *is a complete approximation to* Y iff for every query Q, if $KB \models_Y Q$ then $KB \models_X Q$.

IAR Semantics [LemboRR10]

Goal: manage the inconsistency by means of *reasonable semantics* and *computationally efficient methods.*

- We could argue if AR is a reasonable/meaningful semantics.
- Complexity wise… we saw AR is not likely to work in practice.

- Given $KB = (D,\Sigma)$ and a CQ Q, we say $KB \vDash_{IAR} Q$ iff $(\bigcap_{R \in Rep(D,\Sigma)} R,\ \Sigma) \vDash\ Q$.
  - *Sound approximation to* AR.
  - P-TIME complete for guarded Datalog+/− for UCQs
  - AC0 (FO rewritable) for linear Datalog+/-.

IAR Semantics [LemboRR10]

$$D = \{player(lio),\ striker(lio),\ coach(pep),\ coach(lio),\ midfielder(pep)\}$$

We have 3 data repairs:

$$R_1 = \{player(lio),\ striker(lio),\ coach(pep)\}$$

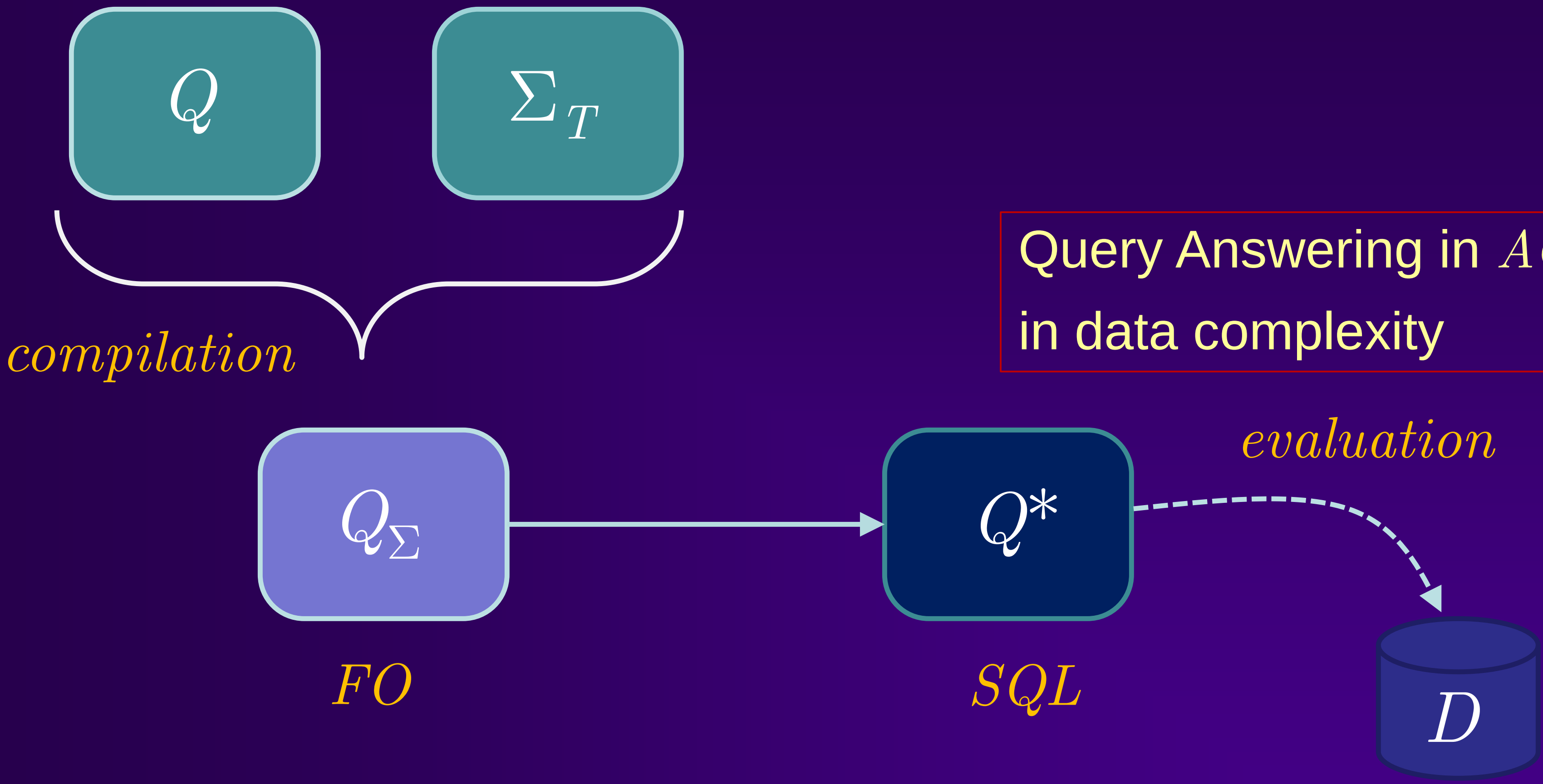
$$R_2 = \{player(lio),\ striker(lio),\ midfielder(pep)\}$$

$$R_3 = \{\ coach(lio),\ midfielder(pep)\ \}$$

$$R_1 \cap R_2 \cap R_3 = \{\}$$

$$KB \not\models_{IAR} \exists X\ player(X)$$

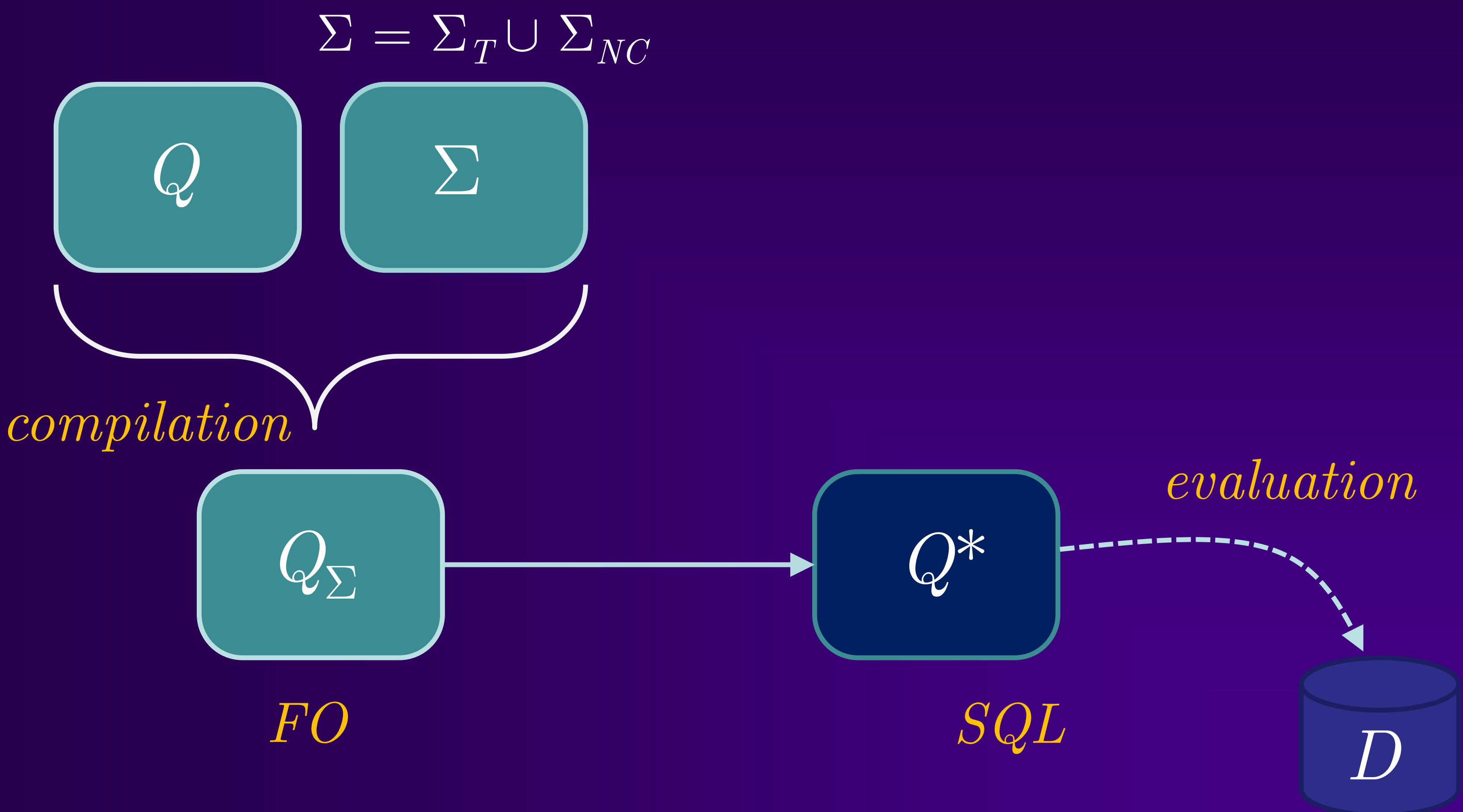
FO rewritable TGDs



Query Answering in AC_0 in data complexity

$$\forall D\ (D \cup \Sigma \models Q) \Leftrightarrow D \models Q^*$$

FO rewriting: IAR Semantics



$$\forall D \,(D \cup \Sigma \models_{IAR} Q) \Leftrightarrow D \models Q^*$$

CAR Semantics [LemboRR10]

- AR is not independent of the KB *syntactic form*: two logically equivalent KBs that are inconsistent may have a different set of repairs.

- Consistent Closure: $CLC(D,\Sigma) = \{\alpha \mid \alpha \in HB(\mathcal{L}_{\mathcal{R}})$ s.t. $\exists S \subseteq D$ and $mods(S, \Sigma) \neq \varnothing$ and $(S,\Sigma) \models \alpha\}$

- A closed (AR-) repair of (D, Σ) is a database instance D' such that: (1) $D' \subseteq CLC(D,\Sigma)$, (2) $mods(D',\Sigma) \neq \varnothing$, and (3) there is no $D'' \subseteq CLC(D,\Sigma)$ such that $mods(D'',\Sigma) \neq \varnothing$ and:
  - $D'' \cap D \supset D' \cap D$ o,
  - $D'' \cap D = D' \cap D$ and $D'' \supset D'$

- (3) means that a *closed repair* maximally preserves D.

CAR Semantics

- Given $KB = (D,\Sigma)$ and a CQ Q, we say that $KB \vDash_{CAR} Q$ iff $(R,\ \Sigma) \vDash\ Q$ for every repair $R \in\ CRep(D,\Sigma)$.
- CAR *is a complete approximation* to AR.
- Answering atomic queries is in PTIME for linear Datalog+/−
- Coincides with $ICAR$ for $DL\text{-}Lite_A$, it is FO rewritable.
- CONP-complete for UCQs for linear Datalog+/−.
- DP-complete for EL / guarded Datalog+/− (UCQs).

CAR Semantics

$CLC(D,\Sigma) = \{player(lio),\ striker(lio),\ coach(pep),\ teamMember(pep),\ teamMember(lio),\ plays(lio,\ forward),\ midfielder(pep),\ plays(pep,\ mildfielder),\ coach(lio),\ player(pep)\}$

$RC_1 = \{player(lio),\ striker(lio),\ coach(pep),\ teamMember(lio),\ teamMember(pep),\ plays(lio,forward),\ plays(pep,\ mildfilder)\}$

$RC_2 = \{player(lio),\ striker(lio),\ midfielder(pep),\ teamMember(lio),\ teamMember(pep),\ plays(lio,forward),\ plays(pep,midfield),player(pep)\}$

$RC_3 = \{coach(lio),\ midfielder(pep),\ teamMember(lio),\ player(pep),\ teamMember(pep),\ plays(lio,\ forward),\ plays(pep,midfield)\}$

$$KB \vDash_{CAR} \exists\ X\ plays\ (X,\ midfield)$$

UBA Buenos Aires CONICET

ICAR Semantics [LemboRR11]

- Given $KB = (D,\Sigma)$ an a CQ Q, we say that $KB \models_{ICAR} Q$ iff $(\bigcap_{R \,\in\, CRep(D,\Sigma)} R,\ \Sigma) \models Q$.

$$D_{RC} = \{teamMember(lio),\ teamMember(pep),$$

$$plays(pep,\ mildfilder),\ plays(lio,\ forward)\}$$

- *A sound approximation* of CAR, and a *complete approximation* for IAR, and it is neither sound nor complete w.r.t. AR.

- PTIME for linear Datalog+/− for UCQs (FO rewritable for $DL\text{-}Lite_A$).

- DP-complete for EL / guarded Datalog+/−.

ICR Semantics [BienvenuAAAI12]

- Let $Cn(D,\Sigma)$ be the *logical closure* of D and Σ.
- Let $KB = (D,\Sigma)$ and a CQ Q, we say that $KB \vDash_{ICR} Q$ iff

$$\left(\bigcap_{R \,\in\, Rep(KB)} Cn(R,\Sigma)\right) \vDash Q.$$

- A sound approximation of AR and $ICAR$; all IAR answers are also ICR answers, but not the wother way around.
- coNP-hard for $DL\text{-}Lite_{Core}$ (more restrictive than $DL\text{-}Lite_{A}$ and linear Datalog+/−), even for atomic queries.
- FO-rewritable for UCQs for very simple ontologies (only concept inclusions and binary NCs).

ICR Semantics

$Cn(R_1,\Sigma) = \{player(lio), striker(lio), coach(pep), plays(lio,forward), teamMember(lio), teamMember(pep)\}$

$Cn(R_2,\Sigma) = \{player(lio), striker(lio), midfielder(pep), teamMember(lio), teamMember(pep), plays(lio,forward), plays(pep,midfield), player(pep)\}$

$Cn(R_3,\Sigma) = \{coach(lio), midfielder(pep), teamMember(lio), teamMember(pep), player(pep), plays(pep,midfield)\}$

The intersection of all closed repairs is:

$$D_{CR} = \{teamMember(lio), teamMember(pep)\}$$

$$KB \nvDash_{ICR} \exists\ X\ teamMember(X) \wedge player(X)$$

ICR vs. ICAR

$Cn(R_1,\Sigma) = \{player(lio),\ striker(lio),\ coach(pep),\ teamMember(lio),$ $teamMember(pep),\ plays(lio,forward)\} \subseteq RC_1$

$Cn(R_2,\Sigma) = \{player(lio),\ striker(lio),\ midfielder(pep),\ teamMember(lio),$ $teamMember(pep),\ plays(lio,forward),\ plays(pep,midfield),\ player(pep)\} =$ RC_2

$Cn(R_3,\Sigma) = \{coach(lio),\ midfielder(pep),\ teamMember(lio),$ $teamMember(pep),\ player(pep),\ plays(pep,midfield)\} \subseteq RC_3$

$$D_{CR} = \{teamMember(lio),\ teamMember(pep)\}$$

$$\subseteq$$

$$D_{RC\,=}\{teamMember(lio),\ teamMember(pep),\ plays(pep,\ mildfilder),\ plays(lio,\ forward)\}$$

$$KB \models_{ICAR}\ plays(lio,\ forward)$$

$$KB \not\models_{ICR}\ plays(lio,\ forward)$$

Consistent Answers:

Alternatives not directly based on data repairs

k-lazy Semantics

A dual perspective of inconsistency

Given a $KB = (D,\Sigma)$:

- Culprits or minimal inconsistent subsets of D w.r.t. Σ.
- Clusters: are sets of culprits that *overlap.*
  - Very informally, clusters group together atoms in D by their "type of inconsistency", that is the atoms that are involved in (some of) the same conflicts.
  - We define an *equivalence relation* w.r.t. this overlap relation.

Example:

$$c_1 = \{player(lio),\ coach(lio)\}$$

$$c_2 = \{striker\ (lio),\ coach(lio)\}$$

$$c_3 = \{coach(pep),\ coach(lio)\}$$

$$c_4 = \{midfielder\ (pep),\ coach(pep)\}$$

$$clusters(KB) = c_1 \cup\ c_2 \cup\ c_3 \cup\ c_4$$

Clusters (another example)

$$D = \Big\{ player(lio),\ plays(lio,\ forward),\ coach(pep),\ coach(lio),$$
$$midfielder(pep),\ striker(lio) \Big\}$$

$$\Sigma_{NC} = \{player(X) \wedge coach(\mathrm{X}) \rightarrow \bot\}$$

Minimal inconsistent subsets of D:

$$c_1 = \{player(lio),\ coach(lio)\},$$

$$c_2 = \{striker(lio),\ coach(lio)\}$$

$$c_4 = \{midfielder(pep),\ coach(pep)\}$$

$$clusters(KB) = \Big\{ \{player(lio),\ striker\ (lio),\ coach(lio)\},$$
$$\{midfielder\ (pep),\ coach(pep)\} \Big\}$$

Incision Functions

- Informally, incision functions allow to *cut inconsistencies* from clusters.
- Given an Ontology $KB = (D,\Sigma)$, and incision function is a function χ such that:
  - $\chi(clusters(KB)) \subseteq \bigcup_{cl \in clusters(KB)} cl$
  - $mods(D - \chi(clusters(KB))) \neq \varnothing$
- Incision functions are generalizations of kernel incision functions used in belief revision for kernel contraction [Hansson94].

Incision Functions and CQA

- Optimal incision function χ_{opt} is *optimal* iff for every subset $B \subset \chi(clusters(KB))$ we have $mods(D - B) = \varnothing$.

Theorem: $R \in Rep(KB)$ iff exists an optimal incision function χ such that $R = D$ - $\chi(clusters(KB))$.

- A repair is the reminder of D after applying an optimal incision function to its clusters.

- Incision function $\chi_{all}(clusters(KB)) = \bigcup_{cl \in clusters(KB)} cl.$

$$KB \vDash_{IAR} Q \text{ iff } (D - \chi_{all}(clusters(KB)), \Sigma) \vDash Q.$$

k-Lazy Semantics

- Alternative semantics based on incisions of size at most k to clusters in D:
  - $\chi_{k\text{-}cut}$ returns all subsets of size at most k of a cluster cl such that cl without each subset is consistent w.r.t. Σ.
  - $\chi_{lazy}(k,\ clusters(KB)) = \bigcup_{cl\,\in\,clusters(KB)} c_{cl},\ c_{cl} \in \chi_{k\text{-}cut}(cl)$
- A k-lazy-repair is any set $R = D - \chi_{lazy}(k,\ clusters(KB))$.
- *k-lazy answers:* $KB \vDash_{LCONS} Q$ iff $(R,\Sigma) \vDash Q$ for every $R \in LRep(k,KB)$.

Example

$$cl\text{:}\{player(lio),\ striker\ (lio),\ coach(lio),\ midfielder\ (pep),\ coach(pep)\}$$

For $k = 1$ we have: $\chi_{1\text{-}cut}(cl) = \{cl\}$ $\mathrm{LR} = \mathrm{D} - cl = \{\}$

For $k = 2$ we have:

$$\chi_{2\text{-}cut}(cl) = \{\{coach(lio),\ coach(pep)\},\ \{coach(lio),\ midfielder\ (pep)\}\}$$

2-lazy repairs:

$$\mathrm{LR}_1 = \mathrm{D} - \{coach(lio),\ coach(pep)\} = \{player(lio),\ striker(lio),\ coach(pep)\}$$

$$\mathrm{LR}_2 = \mathrm{D} - \{coach(lio),\ midfielder\ (pep)\} = \{player(lio),\ striker(lio),\ midfielder(pep)\}$$

$$Q() = \exists\ \mathrm{X}\ player\ (X, forward)$$

$KB \nvDash_{AR}\ Q$ but $KB \vDash_{2\text{-}LCONS}\ \mathrm{Q}$

Example (Cont.)

For $k = 3$ we have:

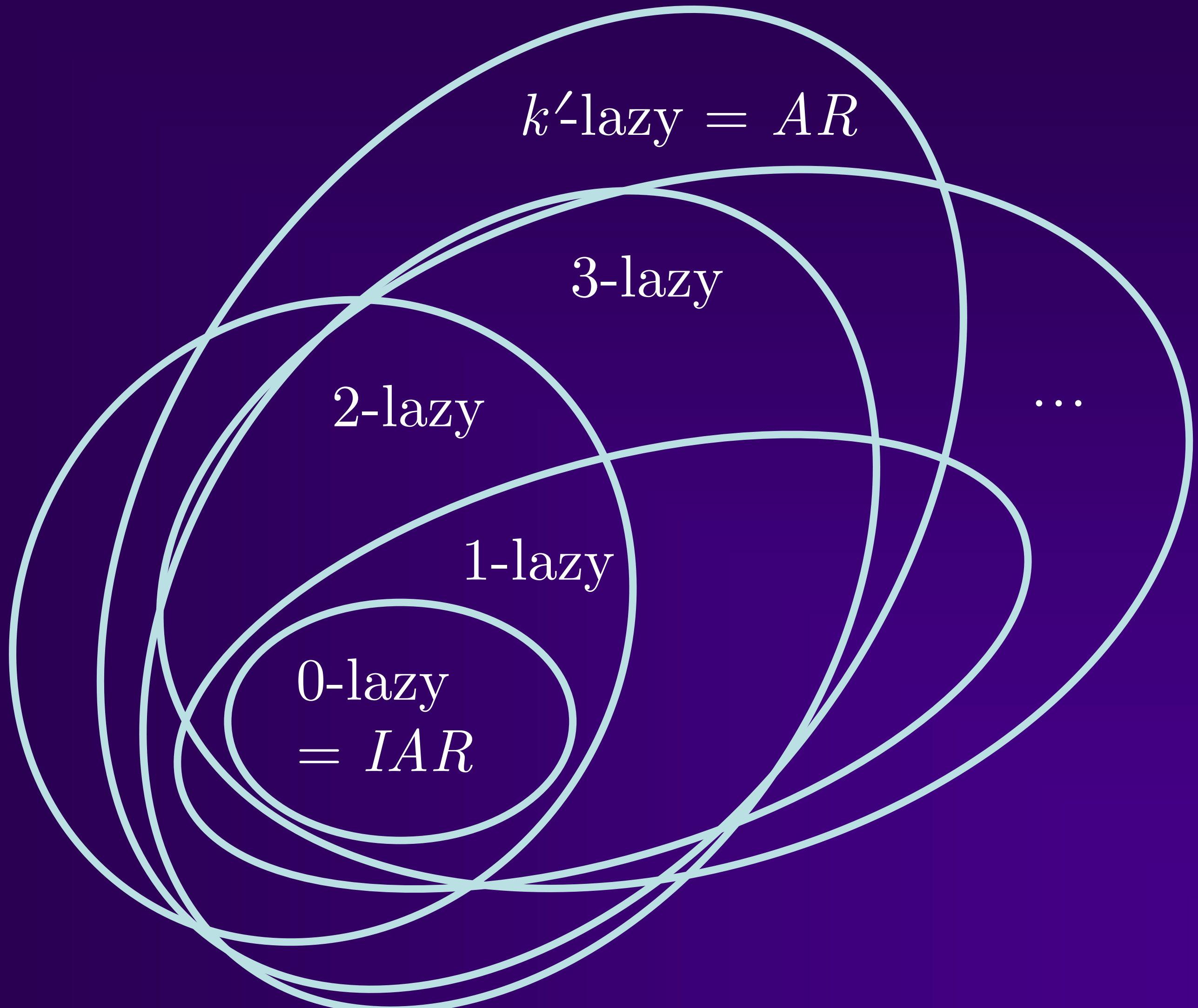
$$\chi_{3\text{-}cut}(cl) = \chi_{2\text{-}cut}(cl) \cup \{\{player(lio),\ striker\ (lio),\ coach(pep)\}\}$$
$$= \{\ coach(lio),\ midfielder(pep)\ \} = \{r_1,\ r_2,\ r_3\}$$

$$LRep(3, KB) = Rep(KB)$$

k-lazy Semantics

- For any $KB = (D,\Sigma)$, and CQ Q we have:
  - $KB \vDash_{IAR}\ Q$ iff $KB \vDash_{0\text{-}LCONS}\ \mathrm{Q}$
  - There exists $k \geq 0$ such that $KB \vDash_{AR}\ Q$ iff $KB \vDash_{k\text{-}LCONS}\ Q$
- The k-lazy incisions are not always minimal, therefore not every k-lazy repair is a repair.
- In general, k-lazy answers are NEITHER sound nor complete with respect to AR nor CAR.
- k-lazy answers are not monotonic in k.

k-lazy



k-lazy Semantics

- To compute de answers under k-lazy is coNP-hard for guarded Datalog +/− ontologies.
- Tractability for linear Datalog+/−:
  - For a set of linear TGDs, the set of clusters can be computed in polynomial time in data complexity.
  - Derivation from a cluster (without the corresponding cuts) are independent of the other clusters: there is no need to look at combinations of cuts across clusters.

k-lazy and union-k-lazy Semantics

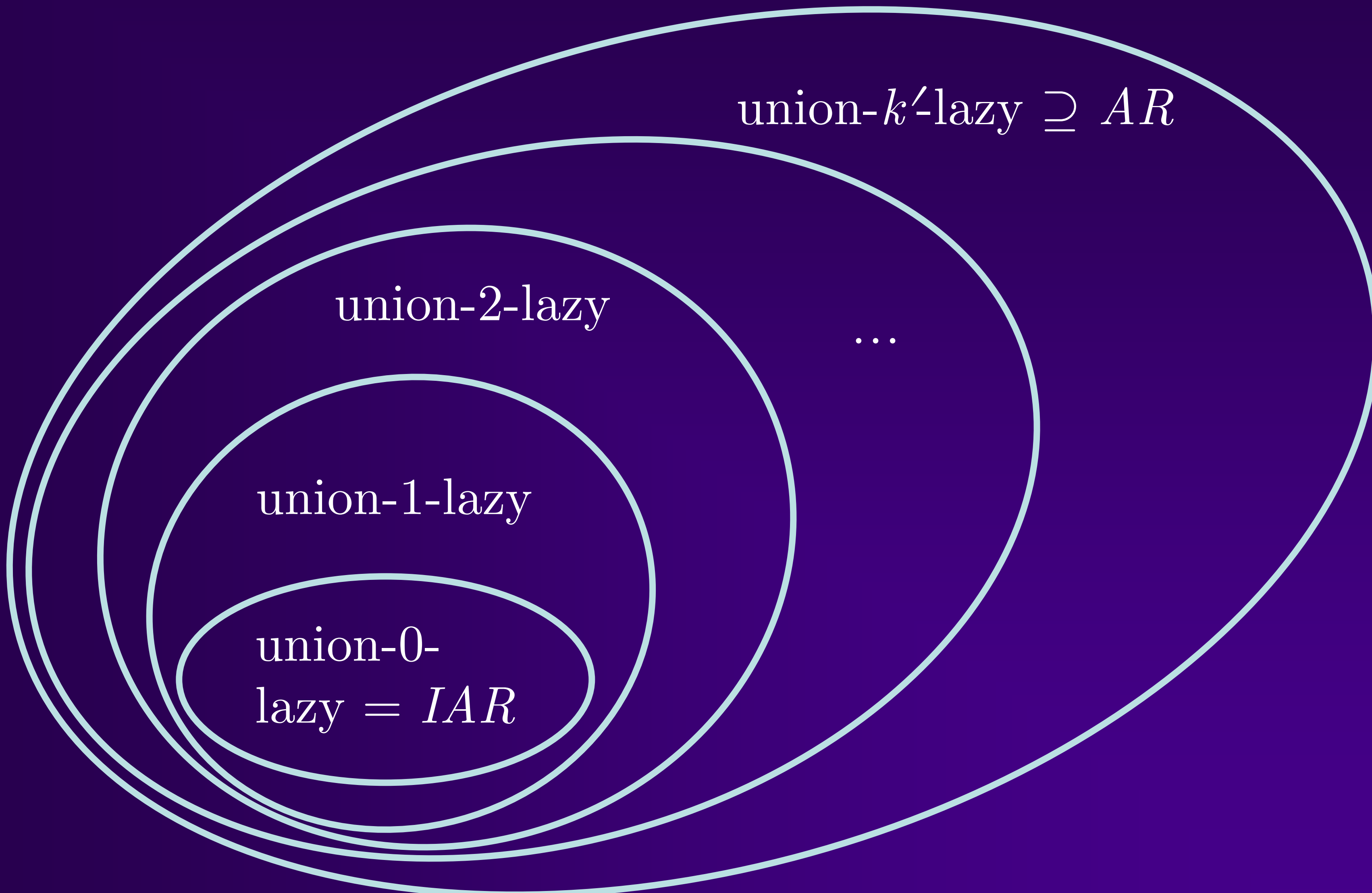
- Given $KB = (D,\Sigma)$, and CQ Q, for any $k \geq 0$, we say that Q is entailed under *union-k-lazy* semantics iff

  $KB \vDash_{k'\text{-}LCONS}\ Q$ for some $0 \leq k' \leq k$.

- For any $k \geq 0$, any union-k-lazy answer for Q is also a union-k+1-lazy answer for Q (monotonic in k).

- Given $KB = (D,\Sigma)$ and CQ Q, for any $k \geq 0$, the set of all k-lazy and union-k-lazy answers for Q is *always consistent* w.r.t. Σ.

union-k-lazy


union-k'-lazy $\supseteq AR$
union-2-lazy
$\dots$
union-1-lazy
union-0-lazy $= IAR$

k-lazy and union-k-lazy Semantics

- Semantics based on *incisions* to clusters of at most size k.
- Value k is the "*allowed budget*" that an agent has for a reasoning task:
  - If k is enough to solve the conflicts in the cluster then we consider all possible ways to fix them.
  - If not, remove the whole cluster (our budget is not enough!).
- The value of k bounds the reasoning capabilities of the agent (higher values of k afford more complex reasoning).
- As a consequence, the *union-k-lazy* semantics allows us to perform reasoning in an anytime progression.

k-support and k-defeater

Semantics

k-support Semantics [BienvenuIJCAI13]

- Given a $KB = (D,\Sigma)$ and a CQ Q, a set $S \subseteq D$ is called a *Σ-support* for Q in D if $mods(S, \Sigma) \neq \varnothing$ and $(S, \Sigma) \vDash Q$.

- Given a $KB = (D,\Sigma)$ and a CQ Q, $KB \vDash_{k\text{-}supp} Q$ if there exists $S_1,\dots, S_k$ such that each S_i is a Σ-support for Q *in* D, and for each $D' \in Rep(D,\Sigma)$ there exists some $S_i \subseteq D'$.

Consider $Q() = \exists X\ teamMember(X) \wedge player(X)$

$$S_1 = \{player(lio)\}\ S_2 = \{midfielder(pep)\}$$

$S_1 \subseteq r_1$, $S_1 \subseteq r_2$, and $S_2 \subseteq r_3$

k-support Semantics

- Sound approximation to AR.
- For any $KB = (D,\Sigma)$, and CQ Q we have:
  - $KB \models_{IAR} Q$ iff $KB \models_{1\text{-}supp} Q$
  - $KB \models_{AR} Q$ iff $KB \models_{k\text{-}supp} Q$ for some k
  - For any $k \geq 0$, if $KB \models_{k\text{-}supp} Q$ then $KB \models_{k+1\text{-}supp} Q$
- For a $KB = (D,\Sigma)$ for which query answering is FO-rewritable, the size of Σ-supports are bounded by the query Q, KB is FO-rewritable for k-support semantics for $k \geq 1$.

k-defeater Semantics [BienvenuIJCAI13]

- Given a $KB = (D,\Sigma)$ and a CQ Q, a k-defeater for Q in D is a set $S \subseteq D$ s.t. $|S| \leq k$, $mods(S, \Sigma) \neq \varnothing$, and $mods(S \cup C, \Sigma) = \varnothing$ for every minimal Σ-support C of Q in D.

- Given a $KB = (D,\Sigma)$ and a CQ Q, $KB \vDash_{k\text{-}def} Q$ if there is no $S \subseteq D$ s.t. S is a k-defeater for Q in D.

Consider $Q() = \exists X\ teamMember(X) \wedge player(X)$

$C_1 = \{player(lio)\}$ $C_2 = \{striker(lio)\}$ $C_3 = \{midfielder(pep)\}$

$$KB \vDash_{1\text{-}def} Q$$

k-defeater Semantics

A family of progressively complete approximations to AR, starting from the *brave* semantics.

The method may return answers that are together inconsistent.

- For any $KB = (D,\Sigma)$, and CQ Q we have:
  - $KB \models_{brave} Q$ iff $KB \models_{0\text{-}def} Q$
  - $KB \models_{AR} Q$ iff $KB \models_{k\text{-}sdef} Q$ for every $k \geq 0$
  - For every $k \geq 1$, if $KB \models_{k+1\text{-}def} Q$ then $KB \models_{k\text{-}def} Q$
- If $KB = (D,\Sigma)$ is FO-rewritable s.t. the size of all culprits of D are bound, then KB is FO-rewritable for k-support for every $k \geq 1$.

Some Conclusions

- AR semantics is the *default* semantics for querying inconsistent ontologies.
- The approximation methods aim to provide computationally *tractable* procedures with *cautious* results.
- Other alternatives seek to get more meaningful results incorporating in the semantics other elements that either reflect aspects of the application domain (the budget in k-lazy), or provide more substantial evidence for the answers (support, argumentation, etc.).

Why Explain Consistent Answers

- Inconsistency-tolerant semantics provide a way to reason with logical knowledge bases in the presence of inconsistency, without answers becoming meaningless.
- Inconsistency then remains *transparent* to the user.
- But for *decision making* we may not just want an answer, we may want an *explanation* of why that answer is true, especially if there are conflicts and the answer is not an expected one.
- It seems reasonable then to *provide information that complements the set of answers* in a way that helps the user *understand*.

Explaining Consistent Answers

- For instance, suppose the user asks if the query $q() = \exists X\, p(X)$ is true and they get the answer No.
- A natural question to ask would be: "*Was it the case that there is no possible way to derive* $p(X)$ *from the knowledge base, it is actually false, or was it the case that* q *is derivable from the KB but it is involved in a contradiction and the semantics cannot assure its truth value?*".
- Interesting questions for explanatory purposes may be:
  - "What makes Q true under some semantics S?"
  - "What makes Q false under some semantics S?".

Explaining Positive Answers

- Explanations for positive and negative query answers under the brave, AR, and IAR for DLs [Bienvenu2016].
- An explanation for a query Q is based on *causes* for Q : A cause is *a consistent set of facts from the KB* (D or ABox) *that yield* Q.
- Positive explanations:
  - For brave semantics is *any cause* for Q.
  - For IAR, is *any cause* of Q that *does not participate* in any *contradiction*.

Explaining Positive Answers

- An explanation for a query Q is based on *causes* for Q:
  A cause is *a consistent set of facts from the KB* (D or ABox) *that yield* Q.

- Positive explanations:
  - For AR: *not enough* to provide *one cause* as different repairs may use different causes.
  - An explanation is *a (minimal) disjunction of causes* that *cover all repairs* (every cause belongs to at least one repair and for each repair there is one cause in the set).

Explaining Negative Answers

- Explanations for *negative answers* for Q under *AR* are *minimal subsets* of D s.t. together with any cause for Q yield an inconsistency.
- Explanations for negative answers under *IAR*: we only need to show that *every cause* is *contradicted* by *some consistent subset* of D (no cause can belong to all repairs).
- Most of these problems are polynomial for the case of explanations for positive and negative answers under brave and IAR.
- Explanations in both cases under the AR semantics are intractable.

Explaining k-lazy Answers

- Explanations for *negative answers* for Q under *AR* are *minimal subsets* of D s.t. together with any cause for Q yield an inconsistency (basically incisions).
- Other interesting questions may include:
  - What is the smallest k needed to make Q true under both k-lazy and union-k-lazy semantics?
  - What are the causes that make Q change its truth value from k to k+1 under the k-lazy semantics (either from true to false or the other way around)?

Explaining k-lazy Answers

- Other interesting questions may include:
  - If Q is true under (union-)k-lazy semantics for some k > 0 but it is not a consistent answer, what are the reasons for this behavior?
  - This question actually elaborates on the previous one, as we can try to find for which $k' > k$ the truth value of Q changes, and find the reason by comparing k-cuts against $k'+1$-cuts.

Explaining Answers using Argumentation

- Informally: an argument can be seen as a set of premises (facts) that derives a conclusion by means of a logical theory (in Datalog+/– the application of the TGDs).
- We find arguments for and against a conclusion and analyze which ones survive (different semantics).
- Argumentation provides a natural dialogic structure and mechanism as part of the reasoning process.
- We can examine this structure to understand both why and how conclusions (answers) are reached.

Explaining Answers using Argumentation

- The work in [Arioua2015], proposes explanations as *sets of logical arguments supporting the query*.
- We can think of *causes* of a query as *arguments* that *entail or support* the entailment of the query.
- We can build arguments that *contradict* some sentence, and these can be used as *reasons against* a query or as explanations for negative answers.
- All the examples of explanation proposals mentioned so far can be considered as argument-based explanations: different notions of argument and counterarguments can be constructed as a means for explanations.

Static vs. Dynamic Explanations

- The proposals mentioned above provide arguments for and against conclusions in a static way.
- Dynamical characteristics of argumentation frameworks can be exploited in an interactive explanation mechanism.
- [Arioua2016] dialectical explanations for brave, IAR and ICR:
  - The system aims to make a user understand why a query Q is or is not entailed by the query answering semantics.
  - Arguments for and against the query are identified, analyzed, and weighed among each other.
  - A query is entailed under a specific semantics if and only if the dialectical process ends with a winning argument in favor of the query.

Explaining through Defeasible Reasoning

- *Defeasible reasoning*: allows to model knowledge with contradictions and obtain conclusions that can be challenged in the presence of additional knowledge.
- [Martinez2014] develops a framework for inconsistency-tolerant semantics for Datalog+/– based on defeasible argumentative reasoning:
  - Defeasible TGDs and conclusions (Strict and Defeasible).
  - Argumentation theory within the Datalog+/– query answering process itself: considering reasons for and against potential conclusions and deciding which are the ones that can be obtained (warranted) from the knowledge base.

Explaining through Defeasible Reasoning

- Provides a framework to implement different inconsistency tolerant semantics depending on the *argument comparison criterion*: most of the semantics we saw today can be obtained within this framework.
- It is not necessary to use and compute elements that are outside of the logic, such as repairs, kernels, clusters, incisions, etc., as the query answering engine is inconsistency-tolerant in itself.
- The argumentative process allows to compute the answers and the required explanations at the same time $\Rightarrow$ no extra cost for computing explanations.

References

[ABC99] Marcelo Arenas, Leopoldo Bertossi, and Jan Chomicki. "*Consistent query answers in inconsistent databases*". Proceedings of PODS 1999. ACM, pp. 68–79.

[LemboRR10] Domenico Lembo, Maurizio Lenzerini, Riccardo Rosati, Marco Ruzzi, Domenico Fabio Savo: "*Inconsistency-Tolerant Semantics for Description Logics*". RR 2010: 103–117.

[LemboRR11] Domenico Lembo, Maurizio Lenzerini, Riccardo Rosati, Marco Ruzzi, Domenico Fabio Savo: "*Query Rewriting for Inconsistent DL-Lite Ontologies*". RR 2011: 155–169.

[BienvenuAAAI12] Meghyn Bienvenu: "*On the Complexity of Consistent Query Answering in the Presence of Simple Ontologies*". AAAI 2012.

[BRIJCAI13] Meghyn Bienvenu, Riccardo Rosati: "*Tractable Approximations of Consistent Query Answering for Robust Ontology-based Data Access*". IJCAI 2013.

References

[MILLER2019] Miller, T.: Explanation in artificial intelligence: Insights from the social sciences. Artificial Intelligence (267), 1–38 (2019) .

[MOULIN2002] Bernard Moulin, Hengameh Irandoust, Micheline Bélanger, and Gaëlle Desbordes. Explanation and argumentation capabilities: Towards the creation of more persuasive agents. Artificial Intelligence Review, 17(3):169–222, 2002.

[Southwick91] Richard W Southwick. Explaining reasoning: an overview of explanation in knowledge based systems. The knowledge engineering review, 6(1):1–19, 1991.

[Falappa2002] Marcelo A Falappa, Gabriele Kern-Isberner, and Guillermo R Simari. Explanations, belief revision and defeasible reasoning. Artificial Intelligence, 141(1-2):1–28, 2002.

[Garcia2013] Alejandro J García, Carlos I Chesñevar, Nicolás D Rotstein, and Guillermo R Simari. Formalizing dialectical explanation support for argument-based reasoning in knowledge-based systems. Expert Systems with Applications, 40(8):3233–3247, 2013.

References

[Bienvenu2016] Bienvenu,M.,Bourgaux,C.,Goasdoue,F.:Explaininginconsistency-tolerantqueryansweringoverdescriptionlogicknowledgebases.In:Proc.ofAAAI'16.pp.900–906.AAAIPress (2016)

[Arioua2015] Arioua, A., Tamani, N., Croitoru, M.: Query answering explanation in inconsistent datalog +/- knowledge bases. In: DEXA (2015)

[Arioua2016] Arioua,A., Croitoru,M.: Dialectical Characterization of Consistent Query Explanation with Existential Rules. In: FLAIRS: Florida Artificial Intelligence Research Society (2016)

[Martinez2014] Martinez, M.V., Deagustini, C.A.D., Falappa, M.A., Simari, G.R.: Inconsistency-tolerant reasoning in datalog+- ontologies via an argumentative semantics. In: Advances in Artificial Intelligence – IBERAMIA 2014. pp. 15–27. Springer International Publishing (2014)

References

[LMSECAI12] Thomas Lukasiewicz, Maria Vanina Martinez, Gerardo I. Simari: "*Inconsistency Handling in Datalog+/- Ontologies*". ECAI 2012: 558–563.

[LMSDat12] Thomas Lukasiewicz, Maria Vanina Martinez, Gerardo I. Simari: "*Inconsistency-Tolerant Query Rewriting for Linear Datalog+/-"*. Datalog 2012: pp. 123–134.

[LMPSAAAI15] Thomas Lukasiewicz, Maria Vanina Martinez, Andreas Pieris, Gerardo I. Simari: "*From Classical to Consistent Query Answering under Existential Rules"*. AAAI 2015: 1546–1552.

[Hansson94] Sven Ove Hansson, "Kernel Contraction", Journal of Symbolic Logic 59:845-859, 1994.

Part of the content of this course is based on the research done in collaboration with Thomas Lukasiewicz, Georg Gottlob, V.S. Subrahmanian, Andreas Pieris, and Giorgio Orsi.