Integración de Bases de Conocimiento

Clase 5: Manejo de Inconsistencia

Profesores: Maria Vanina Martinez y Ricardo Rodriguez

El problema de la *integración de conocimiento* tiene como desafío lidiar con la heterogeneidad de las bases de conocimiento con el objetivo de proveer una visión unificada.

Decision Making

Model of the World

Observado y registrado como

Data

News

Social Networks

Databases

Blogs/Video

- Heterogéneo
- Multilingual
- Inconsistente/ Incompleto
  - Falta de información
  - Datos incorrectos
  - Inconsistencia temporal

Integración de conocimiento

- La integración de conocimiento debe lidiar inevitablemente con problemas de *incertidumbre* y/o *inconsistencia*.
- En un sistema de información, se espera que la resolución (o no) de esos problemas sea *(semi-)automática*:
  - Esto es no sólo deseable sino imprescindible en sistemas que manejan *grandes cantidades* de datos (por ej., provenientes de la Web).
  - Estos mecanismos o métodos deben además tener una correspondencia con los métodos (o resultados) que un ser *humano* utilizaría al enfrentar la tarea.

Sin embargo…

- Durante mucho tiempo se sostuvo la suposición de que existe una *única* forma*, correcta desde el punto de vista epistémico,* de solucionar o manejar estos problemas.
- Muchas propuestas tienden a *ocultarle* a los usuarios tanto los problemas como la manera en que se resuelven.
- Los usuarios tienen que trabajar con decisiones tomadas por *terceros*, tal vez ajenos al dominio de experticia.

En muchas aplicaciones es necesario que el usuario aplique su conocimiento sobre el dominio de aplicación para poder sobreponerse a los problemas de integración.

En esta clase… Inconsistencia

Consideremos el siguiente escenario:

- La recaudadora nacional de impuestos dispone de una base de datos que registra los ingresos de todos los empleados en relación de dependencia $\Rightarrow$ resulta de la *integración* de varias bases de datos de *distinta fuente*.
- Distintas entidades tienen acceso y usan estos registros:
  - La misma recaudadora los utiliza para computar los *impuestos*.
  - Los bancos para determinar si otorga *créditos* y de qué monto.
- Existen varios registros que muestran *distintos valores* de sueldo para un cierto empleado Juan en el mismo mes.
- Sin embargo, se supone que un empleado debería tener *un sólo* registro por mes.

En esta clase… Inconsistencia

Cómo interpretaría cada usuario esta potencial inconsistencia?

- La oficina de sueldos de la empresa donde trabaja el empleado podría simplemente *ignorar* la situación.
- El administrativo de un banco que está considerando a Juan para un crédito (siendo cauteloso) podría *tomar el valor más bajo* que figura en los registros.
- La agencia de recaudación seguramente querrá guardar *todos* estos registros como evidencia para una potencial investigación de una situación sospechosa.

Tal vez la *suposición* (restricción de integridad) de que un empleado sólo debe tener un registro de sueldo por mes, sea el residuo de una normativa vieja que ya *no modela correctamente el mundo real*.

Otro ejemplo

- Un analista de ciencias sociales está compilando información acerca de una organización etno-política.
- Su trabajo es recolectar información de interés sobre el grupo y para eso consulta *diferentes fuentes*; por ejemplo, periódicos, artículos en la Web, Facebook, YouTube, etc.
- "*¿Qué tipo de liderazgo tiene la organización*? Es una de las tantas variables de su registro que debe determinar.

$Fuente_1$: *"El lider de la agrupación es el Sr X."*

$Fuente_2$: *"El grupo no tiene lider, dado que el Sr. X ha desaparecido."*

$Fuente_3$ *citando al Sr. X en una entrevista: "Yo no soy el lider de nada; el grupo se organiza por sí mismo, mi papel es solo el de un vocero."*

Otro ejemplo: cont.

- ¿Cómo *decide* el analista el valor de la variable?
- La respuesta debería depender, más allá de las fuentes, de:
  - cuánto sabe el analista sobre el grupo,
  - cuan crítica es la obtención de una respuesta,
  - el tipo de toma de decisiones que se hará en base a esa respuesta,
  - el riesgo asociado con una respuesta aproximada o con una respuesta incierta,
  - etc…

Otro ejemplo: cont.

- ¿Cómo *decide* el analista el valor de la variable?
  - ¿Qué pasaría si los datos se obtienen automáticamente de la Web y se necesita hacer un pre-procesamiento de toda esa información?
  - ¿Qué herramientas podemos proveerle a un analista de este estilo para ayudarle a entender los datos?

Inconsistencia

- La inconsistencia en sistemas que manipulan conocimiento es un problema que no puede ignorarse y a veces es necesario *convivir* con la información conflictiva.
- Nos enfocamos en datos provenientes (o accesibles por medio de) la Web.
- Desafío: interpretar la constantemente creciente cantidad de datos heterogéneos y dinámicos provenientes de dominios y fuentes diversas.
- Meta: manejar la inconsistencia con *semánticas razonables* y *métodos computacionalmente eficientes*.

En esta clase…

En esta primera parte veremos:

- Noción de inconsistencia en lenguajes ontológicos
- Consistent Query Answering para Datalog+-
- Resultados de complejidad
- Nuevas propuestas

Datalog+/−

- Asumimos:
  - Un universo infinito de constantes Δ
  - Un conjunto infinito de valores nulos (etiquetados) Δ_N
  - Un conjunto infinito de variables $\mathcal{V}$
  - Un esquema relacional $\mathcal{R}$, un conjunto finito de nombre de relaciones (o símbolos predicativos).
- Diferentes constantes representan diferentes valores; diferentes nulos pueden representar el mismo valor.
- Usamos $\mathbf{X}$ para denotar la secuencia $X_1, \ldots, X_n$, $n \geq 0$.
- Una (instancia de) base de datos D sobre $\mathcal{R}$ es un conjunto de átomos con predicados en $\mathcal{R}$ y argumentos en Δ.

Datalog+/−

- Una consulta conjuntiva (CQ) sobre $\mathcal{R}$ tiene la forma $Q(\mathbf{X}) = \exists \mathbf{Y}\ \Phi(\mathbf{X},\mathbf{Y})$, Φ es una conjunción de átomos.
- Una consulta conjuntiva Booleana (BCQ) sobre $\mathcal{R}$ tiene la forma $Q() = \exists \mathbf{Y}\ \Phi(\mathbf{X},\mathbf{Y})$, Φ es una conjunción de átomos.
- Las *respuestas* a una consulta se definen vía homomorfismos, mapeos μ: $\Delta \cup \Delta_N \cup \mathcal{V} \to \Delta \cup \Delta_N \cup \mathcal{V}$:
  - si $c \in \Delta$ entonces $\mu(c) = c$
  - si $c \in \Delta_N$ entonces $\mu(c) \in \Delta \cup \Delta_N$
  - μ se extiende a (conjuntos de) átomos y conjunciones.
- Conjunto de respuestas $Q(D)$: conjunto de tuplas t sobre Δ tal que $\exists\ \mu$: $\mathbf{X} \cup \mathbf{Y} \to \Delta \cup \Delta_N$ tal que $\mu\big(\Phi(\mathbf{X},\mathbf{Y})\big) \subseteq D$, y $\mu(\mathbf{X}) = t$.

Datalog+/−

- Tuple-generating Dependencies (TGDs) son restricciones de la forma σ: $\forall \mathbf{X} \forall \mathbf{Y}\ \Phi(\mathbf{X},\mathbf{Y}) \rightarrow \exists \mathbf{Z}\ \Psi(\mathbf{X},\mathbf{Z})$ donde Φ y Ψ son conjunciones atómicas sobre $\mathcal{R}$:
  - $\forall \mathbf{X} \forall \mathbf{Y}\ \Phi(\mathbf{X},\mathbf{Y})$ se denomina el cuerpo de σ ($body(\sigma)$)
  - $\exists \mathbf{Z}\ \Psi(\mathbf{X},\mathbf{Z})$ se denomina la cabeza de σ ($head(\sigma)$)
- Dada una BD D y un conjunto Σ de TGDs, el conjunto de modelos $mods(D, \Sigma)$ es el conjunto de todos los B tal que:
  - $D \subseteq B$
  - cada $\sigma \in \Sigma$ es satisfecho en B (clásicamente).
- El conjunto de respuestas para una CQ Q en D y Σ, $ans(Q,D,\Sigma)$, es el conjunto de todas las tuplas a tal que $a \in Q(B)$ para todo $B \in mods(D, \Sigma)$.

Chase

- El Chase es un procedimiento para reparar una BD en relación a un conjunto de dependencias (TGDs).
- (Informalmente) Regla de aplicación de TGD:
  - una TGD σ es aplicable a una BD D si $body(\sigma)$ mapea a átomos en D
  - la aplicación de σ sobre D agrega (si ya no existe) un átomo con nulos "frescos" correspondientes a cada una de las variables existenciales cuantificadas en $head(\sigma)$.

Negative Constraints y EGDs

- *Negative* *constraints* (NCs) son fórmulas de la forma ✂**X** ☞(**X**) ✹ ♈, donde ☞(**X**) es a conjunción of átomos.
- NCs son fáciles de verificar: podemos verificar que la CQ ☞(**X**) tiene un conjunto vacío de respuestas en *D* y 💧.
- *Equality Generating Dependencies* (EGDs) son de la forma ✂**X** ☞(**X**) ✹ $X_i = X_j$, donde ☞ es una conjunción of átomos y X_i, X_j son variables que aparecen en **X**.
- EGDs pueden verse como NCs: ✂**X** ☞(**X**), X_i 🕒 X_j ✹ ♈
- Se asume un conjunto de EGDs separables; intuitivamente significa que las EGDs y TGDs son independientes entre sí.

Inconsistencia en Datalog+/−

- La noción de inconsistencia en la que nos enfocamos es la inconsistencia lógica, es decir una teoría lógica es inconsistente si no tiene modelos.
  - Dada una ontología Datalog+/− (D, Σ), decimos que (D, Σ) es *inconsistente* ssi $mods(D'',\Sigma) \neq \varnothing$ (a veces lo notaremos como $(D, \Sigma) \vDash \perp$).
- En Datalog+/− la inconsistencia surge de la violación de las restricciones de integridad.
  - $chase(D,\Sigma) \vDash body(\nu)$, para alguna $\nu \in \Sigma_E \cup \Sigma_{NC}$

Inconsistencia en Datalog+/−

- ***Importante***: asumimos que las TGDs son *correctas*; es decir, capturan correctamente la semántica del dominio.
- Esta suposición implica que:
  - el conjunto de TGDs es *siempre* satisfacible; la aplicación de las TGDs no generan inconsistencias
  - Debe ser el caso entonces que los conflictos se generan a partir de los datos $\Rightarrow$ la instancia de base de datos o ABox en DLs es la parte que debe ser *reparada o modificada* para restaurar la consistencia.
- Claramente…¡no es la única opción! Muchos trabajos actuales se enfocan en otras posibilidades.

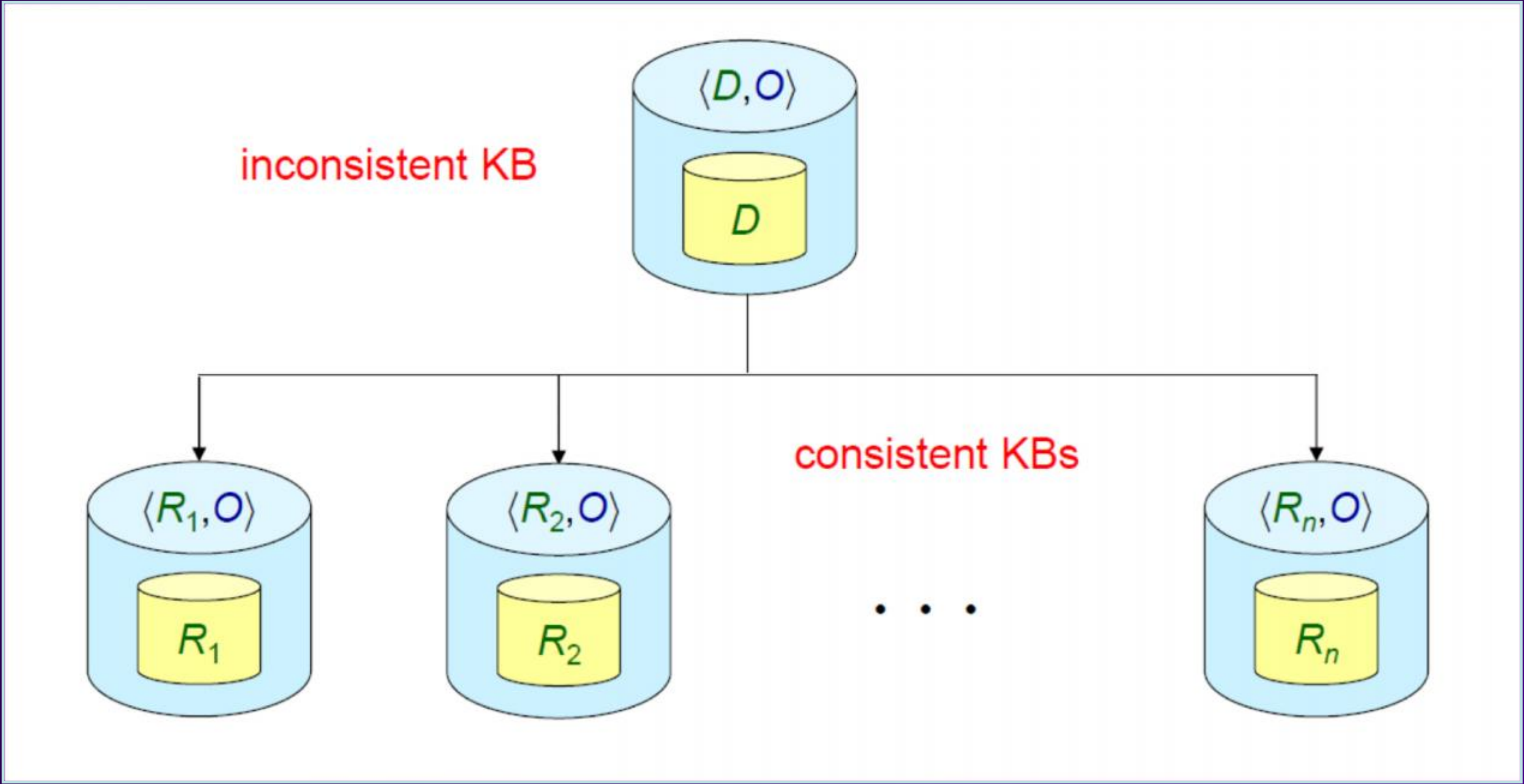
Repairs

- Definición mas general:

  Dada una ontología Datalog+/− (D, Σ), un *repair* para (D, Σ) es otra ontología (D', Σ'), tal que $mods(D'', \Sigma) \neq \emptyset$ y (D', Σ') es "*lo más cercana posible*" a (D, Σ).

- La noción de *cercanía* cambia según el poder expresivo del lenguaje y de las suposiciones del dominio de aplicación.
- Asumiendo que sólo los *datos* producen conflictos:
- Un *data* (ABox) *repair* para (D, Σ) es una base de datos D' tal que:
  - (1) $D' \subseteq D$,
  - (2) $mods(D', \Sigma) \neq \emptyset$, y
  - (3) no existe $D'' \subseteq D$ tal que $D'' \subseteq D'$ y $mods(D'', \Sigma) \neq \emptyset$.

Repairs



Consistent Query Answering

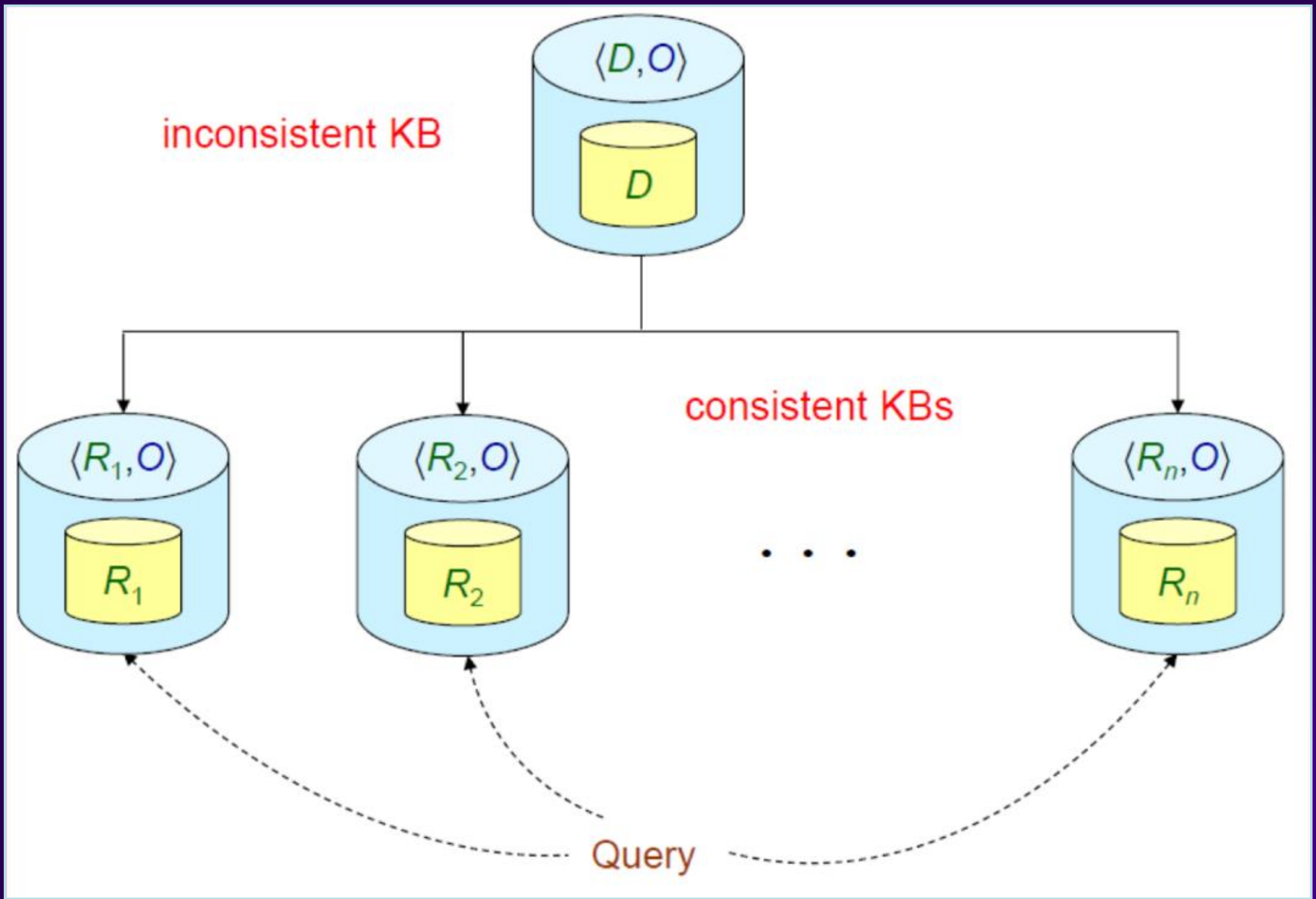
Resumen CQA

- Repasaremos algunas semánticas para query answering tolerantes a la inconsistencia (algunas desarrolladas para *RDBMSs* y otras específicamente para *OBDA*).
- *Consistent Query Answering* [ABC99], conocida como semántica *AR* para lenguajes ontológicos [LemboRR10]
- *Aproximaciones* a *consistent answers* (certain answers):
  - *IAR*, *CAR*, *ICAR* [LemboRR11] y *ICR* [BienvenuAAAI12]
  - k-*defeater* y k-*support* [BRIJCAI13]
- *Lazy Answers* [LMSECAI12]

Semántica AR [LemboRR10]

- La *inconsistencia* en Datalog+/− surge de la violación de las restricciones de integridad.
- Meta: poder manejar la inconsistencia usando procesos de razonamiento con semánticas *razonables* y métodos computacionalmente *eficientes*.
- La semántica AR, inspirada en *consistent answers* (CQA) para RDBMS es la más aceptada para query answering en ontologías potencialmente inconsistentes.
- Dada $KB = (D,\boldsymbol{\Sigma})$ y una CQ Q, decimos que $KB \models_{AR} Q$ ssi $(R,\ \Sigma) \models\ Q$ para cada $R \in Rep(KB)$.
- Decidir si $KB \models_{AR} Q$ (aun para queries atómicas) es coNP-completo para linear Datalog+/−.

Semántica AR [LemboRR10]



Semántica AR: Ejemplo

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

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

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

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

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

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

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

Semántica AR [LemboRR10]

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

Tenemos 3 data repairs:

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

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

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

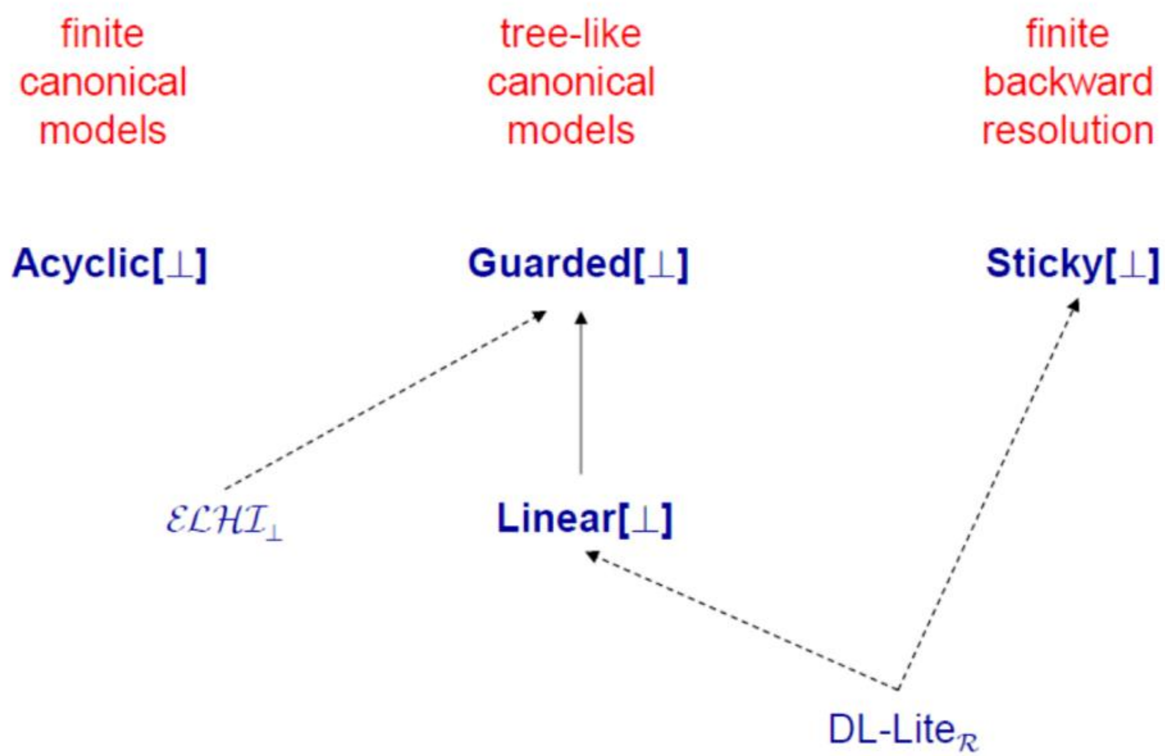
$$KB \vDash_{AR} \exists X\ teamMember(X) \wedge player(X)$$

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

Resultados de complejidad

para la semántica AR

Semántica AR: Complejidad



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

Semántica AR: Complejidad

	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

Resultado de complejidad genérico

métrica de complejidad

clase de TGDs

clase de complejidad

Complejidad **M** de *query answering clásico* bajo **L** es **C**-completo

$\Downarrow$

Complejidad **M** consistent *query answering* bajo **L**[$\perp$] es:

$\Pi_{p,2}$-complete	if	$\mathbb{C}$ = NP
$\mathbb{C}$-complete	if	$\mathbb{C} \supseteq$ PSPACE $\mathbb{C}$ is deterministic

Resultado de complejidad genérico: Cota superior

Algoritmo *Adivinar y Verificar* (*para el complemento del problema*)

Input: $D,\ O \in \mathrm{L}[\perp],\ Q$:

1. Adivinar $R \subseteq D$ (un posible repair)
2. Verificar que R es un repair y que $(R,O) \nvDash \mathrm{Q}$

Nuestro problema está en $\mathrm{coNP}^{\mathbf{C}}$; entonces:

Si $\mathbf{C} = \mathrm{NP} \Rightarrow \mathrm{coNP}^{\mathrm{NP}} = \mathrm{co}\Sigma_{\mathrm{p},2} = \prod_{\mathrm{p},2}$

Si $\mathbf{C} \supseteq \mathrm{PSPACE}$ y deterministico $\Rightarrow \mathrm{coNP}^{\mathbf{C}} = \mathrm{co}\mathbf{C} = \mathbf{C}$

Semántica AR: Complejidad

	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

Semántica AR: Complejidad

Un resultado fuerte de $\prod_{\mathrm{p},2}$-hardness

Consistent query answering con *una sola restricción* de la forma $\forall \mathrm{X} \forall \mathrm{Y} \forall \mathrm{Z}\ p(X,Y,Z) \wedge p(W,X,Z) \rightarrow \perp$

– La base de datos y la query utilizan sólo predicados binarios y ternarios (reducción a partir de satisfabilidad de formulas 2QBF)

$$\Downarrow$$

Para cada clase L de TGDs, la complejidad de consistente query answering asumiendo ontología fija bajo $\mathbf{L}[\perp]$ es

$\prod_{\mathrm{p},2}$-hard

De QA clásico a CQA

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

De QA clásico a CQA

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

$\Pi_{p,2}$-complete if $\mathbb{C}$ = NP

$\mathbb{C}$-complete if $\mathbb{C} \supseteq$ PSPACE
$\mathbb{C}$ is deterministic

UNS Bahía Blanca CONICET

De QA clásico a CQA

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

Conjetura: complejidad (ba-)combined para Acyclic[⊥] es NEXPTIME-completo.

Complejidad CQA sin existenciales

	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	$\Pi_{p,2}$	$\Pi_{p,2}$

Complejidad de Acyclic[$\perp$]

- El algoritmo *Adivinar y Verificar* nos da una cota superior $\mathrm{coNP}^{\mathrm{NEXPTIME}}$
- La clase $\mathrm{NP}^{\mathrm{NEXPTIME}}$ se encuentra a un nivel muy alto de la *jerarquía exponencial fuerte* (SEH)
- SEH colapsa en su segundo nivel Δ_2, entonces tenemos que $\mathrm{NP}^{\mathrm{NEXPTIME}} = \mathrm{P}^{\mathrm{NE}}$
- P^{NE} es una clase determinista y por lo tanto $\mathrm{coP}^{\mathrm{NE}} = \mathrm{P}^{\mathrm{NE}}$
- CQA para Acyclic[$\perp$] está entre $\mathrm{NEXP} - \mathrm{P}^{\mathrm{NE}}$

Consistent Query Answering:

Aproximaciones a *certain answers*

Semántica IAR [LemboRR10]

- Dada $KB = (D,\Sigma)$ y una CQ Q, decimos que $KB \models_{IAR} Q$ ssi $(\bigcap_{R \in Rep(D,\Sigma)} R, \Sigma) \models Q$.
- *Aproximación sana* de AR.
- P-TIME completo para guarded Datalog+/− para UCQs
- AC0 (FO re-escribible) para linear Datalog+/-.

Semántica IAR [LemboRR10]

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

Tenemos 3 data repairs:

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

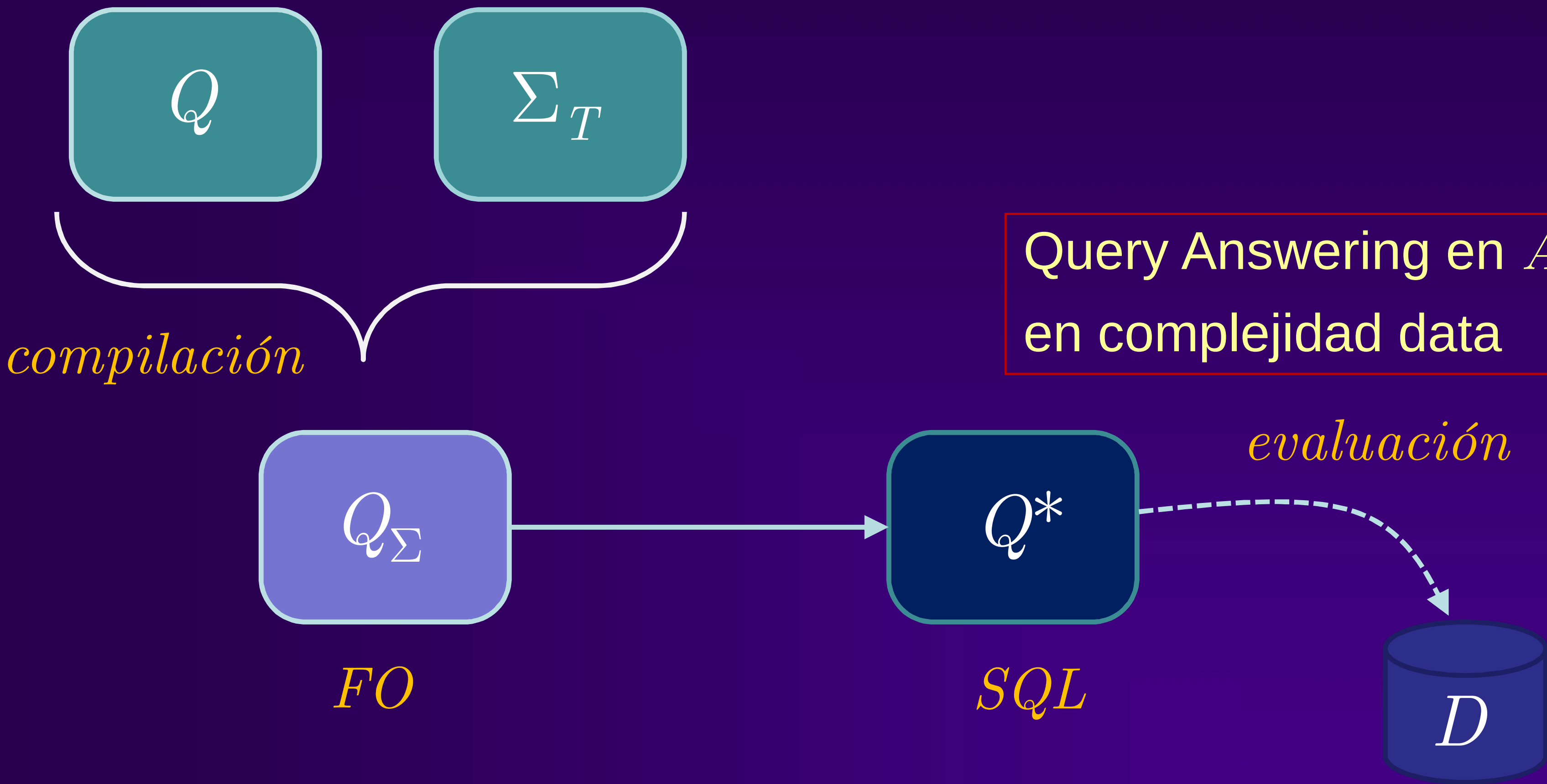
$$R_2 = \{player(lio),\ striker(lio),\ midfielder(pep)\}$$

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

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

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

TGDs FO re-escribibles

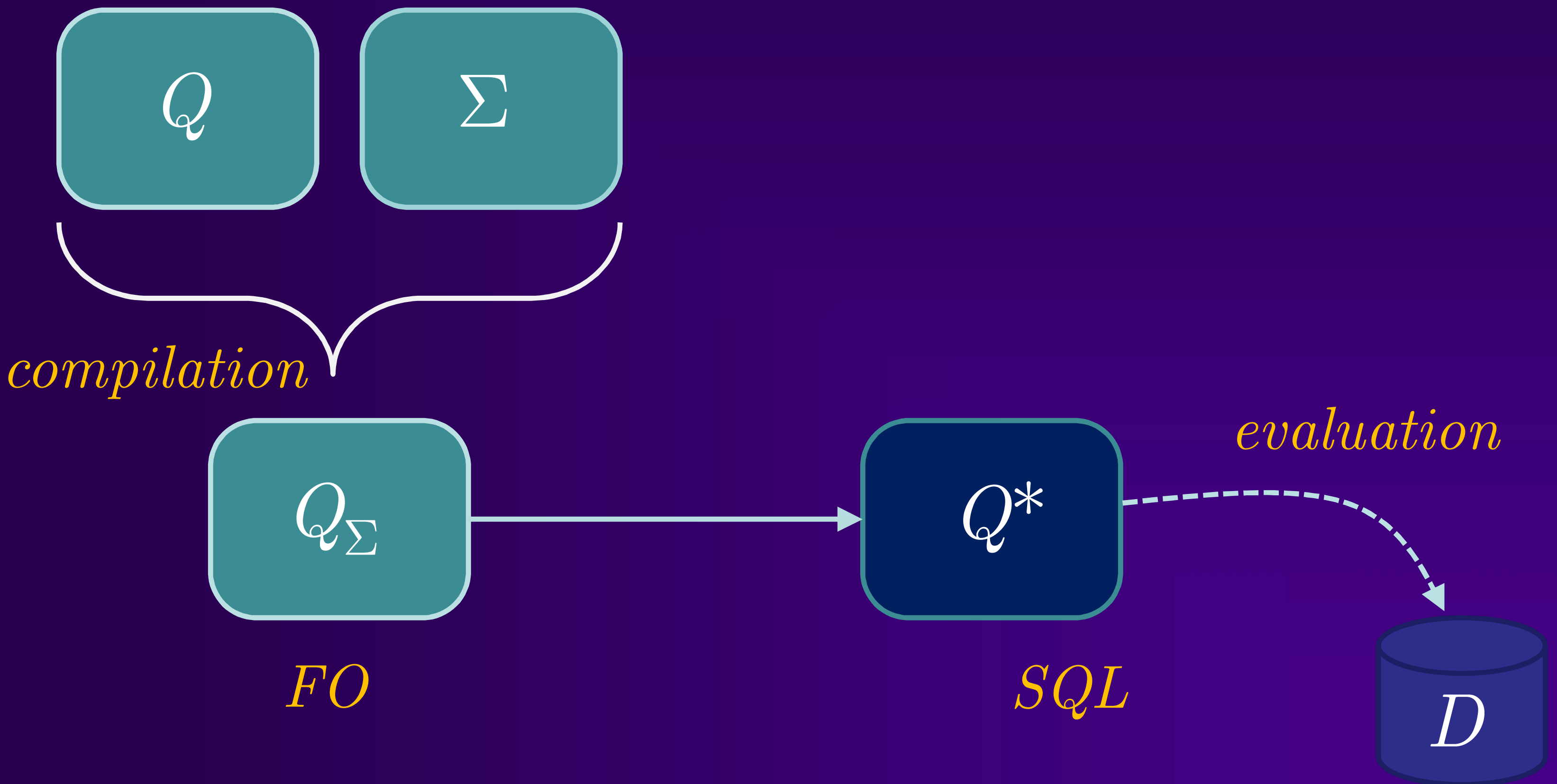


Query Answering en AC_0
en complejidad data

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

Re-escritura de consultas FO: Semántica IAR

$\Sigma = \Sigma_T \cup \Sigma_{NC}$



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

Semántica CAR [LemboRR10]

- AR no es independiente de la *forma* (sintáctica) de la KB: dos KBs inconsistentes que son lógicamente equivalentes pueden tener repairs distintos.
- $CLC(D,\Sigma) = \{\alpha \mid \alpha \in HB(\mathcal{L}_{\mathcal{R}}) \text{ s.t. } \exists S \subseteq D$ y $mods(S, \Sigma) \neq \emptyset$ y $(S,\Sigma) \models \alpha\}$ es la clausura consistente de D y Σ.
- Closed (AR-) repair de (D, Σ): una base de datos D' tal que: (1) $D' \subseteq CLC(D,\Sigma)$, (2) $mods(D',\Sigma) \neq \emptyset$, y (3) no existe $D'' \subseteq CLC(D,\Sigma)$ tal que $mods(D'',\Sigma) \neq \emptyset$ y:
  - $D'' \cap D \supset D' \cap D$ o,
  - $D'' \cap D = D' \cap D$ y $D'' \supset D'$
- Un *closed repair* preserva máximamente a D.

Semántica CAR

- Dada $KB = (D,\boldsymbol{\Sigma})$ y una CQ Q, decimos que $KB \vDash_{CAR} Q$ ssi $(R,\ \Sigma) \vDash\ Q$ para cada $R \in\ CRep(D,\Sigma)$.

- *Aproximación completa* de AR.

- Es PTIME para linear Datalog+/− para atomic queries (coincide con $ICAR$ para $DL\text{-}Lite_A$, FO-rewritable)

- CONP-completo para UCQs para linear Datalog+/−.

- DP-completo para EL / guarded Datalog+/− (UCQs).

Semántica CAR

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

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

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

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

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

Semántica ICAR [LemboRR11]

- Dada $KB = (D,\Sigma)$ y una CQ Q, decimos que $KB \models_{ICAR} Q$ ssi $(\bigcap_{R\, \in\, CRep(D,\Sigma)} R,\, \Sigma) \models Q$.

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

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

- *Aproximación sana* de CAR, y una *aproximación completa* para IAR, y ni sana ni completa para AR.

- PTIME para linear Datalog+/− para UCQs (FO re-escribible para $DL\text{-}Lite_A$).

- DP-completo para EL / guarded Datalog+/−.

Semántica ICR [BienvenuAAAI12]

- Sea $Cn(D,\Sigma)$ la clausura lógica de D y Σ.
- Sea $KB = (D,\Sigma)$ y una CQ Q, decimos que $KB \vDash_{ICR} Q$ ssi $(\bigcap_{R\, \in\, Rep(KB)}\ Cn(R,\Sigma)) \vDash\ Q$.
- Aproximación sana de AR e $ICAR$; las respuestas bajo IAR son respuestas de ICR, pero no vale al revés.
- coNP-hard para $DL\text{-}Lite_{Core}$ (más restringido que $DL\text{-}Lite_{A}$ y linear Datalog+/−); vale aún para queries atómicas.
- FO-rewritable para UCQs para ontologías muy simples (sólo inclusiones de conceptos y NCs binarias).

Semántica ICR

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

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

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

La intersección de todos los closed repairs es:

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

$$KB \not\models_{ICR} \exists X\ teamMember(X) \wedge player(X)$$

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:

Alternativas no basadas en data repairs

Semántica k-lazy

Inconsistencia en Datalog+/−: Otra perspectiva

Dada una ontología $KB = (D,\Sigma)$:

- Culprits o subconjuntos inconsistentes minimales de D.
- Clusters: conjunto de culprits que *comparten* elementos (se superponen)
  - Los clusters agrupan a los átomos por "el *tipo* de inconsistencia", es decir los que son inconsistentes entre sí por el mismo conflicto.
  - Clase de *equivalencia* con respecto a la relación de "superposición".

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

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

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

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

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

Clusters (otro ejemplo)

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

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

Subconjuntos minimales inconsistentes de D:

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

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

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

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

Funciones de incisión: Una perspectiva dual

- Informalmente, una función de incisión permite *cortar inconsistencias* de los clusters.
- Dada una ontología $KB = (D,\Sigma)$, y una función de incisión es una función χ tal que:
  - $\chi(clusters(KB)) \subseteq \bigcup_{cl \in clusters(KB)} cl$
  - $mods(D - \boldsymbol{\chi}(clusters(KB))) \neq \emptyset$
- Las funciones de incisión son generalizaciones de funciones de incisión de *kernel* usadas en el área de revisión de creencias para *kernel contraction* [Hansson94].

Funciones de incisión y CQA

- Una función de incisión χ_{opt} es *optimal* ssi para cada subconjunto $B \subset \boldsymbol{\chi}(clusters(KB))$ tenemos que $mods(D - B) = \varnothing$.
- $R \in Rep(KB)$ ssi existe una función de incisión optimal $\boldsymbol{\chi}$ tal que $R = D - \boldsymbol{\chi}(clusters(KB))$.
- Un repair es lo que queda en D luego de aplicar una función de incisión optimal a sus *clusters*.
- Función de incisión: $\chi_{all}\,(clusters(KB)) = \bigcup_{cl\, \in\, clusters(KB)}\; cl$.

$KB \vDash_{IAR} Q$ ssi $(D$ - $\boldsymbol{\chi}_{all}(clusters(KB)),\, \Sigma) \vDash Q.$

Semántica k-lazy [LMSECAI12]

- Semántica para query answering basada en incisiones a los clusters de tamaño a lo sumo k:
  - $\chi_{k\text{-}cut}$ retorna todos los subconjuntos de un cluster cl de tamaño a los sumo k, tal que sacándole a cl cada subconjunto queda un conjunto consistente con respecto a Σ.
  - $\chi_{lazy}(k,\ clusters(KB)) = \bigcup_{cl\,\in\,clusters(KB)}\ c_{cl},\ c_{cl} \in \chi_{k\text{-}cut}(cl)$
- k-lazy-repair: conjunto $R = D - \chi_{lazy}(k,\ clusters(KB))$.
- *k-lazy answers:* $KB \vDash_{LCONS} Q$ ssi $(R,\Sigma) \vDash Q$ para cada $R \in LRep(k,KB)$.

Ejemplo

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

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

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

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

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

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

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

Ejemplo: Data Repairs

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

Tenemos 3 data repairs:

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

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

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

Ejemplo

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

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

Para $k = 1$ tenemos: $\chi_{1\text{-}cut}(cl) = \{cl\}$ $\quad\quad \mathrm{LR} = \mathrm{D} - cl = \{\}$

Para $k = 2$ tenemos:

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

2-lazy repairs:

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

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

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

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

Ejemplo (Cont.)

Para $k = 3$ tenemos:

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

3-lazy repairs:

$$\mathrm{LR}_1 = \mathrm{D} - \{coach(lio),\ coach(pep)\} = \{player(lio),\ striker(lio),\ coach(pep)\} = R_1$$

$$\mathrm{LR}_2 = \mathrm{D} - \{coach(lio),\ midfielder\ (pep)\} = \{player(lio),\ striker(lio),\ midfielder(pep)\} = R_2$$

$$\mathrm{LR}_3 = \mathrm{D} - \{player(lio),\ striker\ (lio),\ coach(pep)\} = \{player(lio),\ striker(lio),\ midfielder(pep)\} = \{coach(lio),\ midfielder(pep)\} = R_3$$

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

Semántica k-lazy

- Para cualquier $KB = (D,\Sigma)$, y CQ Q tenemos:
  - $KB \models_{IAR}\ Q$ ssi $KB \models_{0\text{-}LCONS}\ \mathrm{Q}$
  - Existe $k \geq 0$ tal que $KB \models_{AR}\ Q$ ssi $KB \models_{k\text{-}LCONS}\ Q$
- Las incisiones k-lazy no son siempre minimales, por lo tanto no todo k-lazy repair es un *data repair*.
- En general, las respuestas bajo la semántica k-lazy no son ni sanas ni completas con respecto a AR ni CAR.
- Las respuestas bajo la semántica k-lazy no son *monotónicas* en k.

Semántica k-lazy

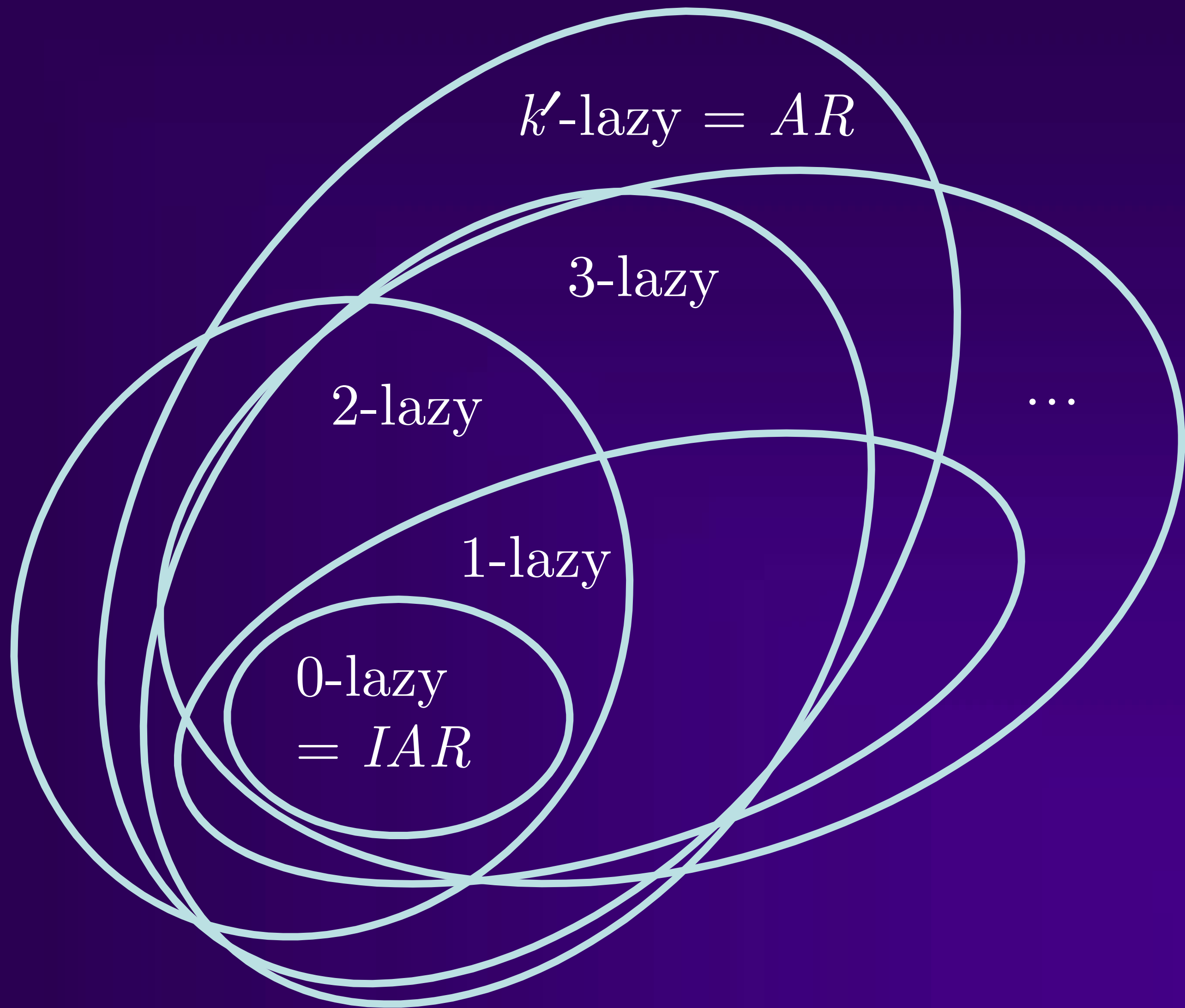
- Computar el conjunto de respuestas bajo k-lazy answers es coNP-hard para ontologías guarded.
- Tratabilidad para linear Datalog+/−:
  - Para un conjunto de linear TGDs, el conjunto de clusters pueden computarse en tiempo polinomial (complejidad data).
  - Las derivaciones a partir de un cluster (menos los cortes correspondientes) son independientes de otros clusters: no hay necesidad de mirar a cada combinación de cortes a través de los clusters.

k-lazy y union-k-lazy

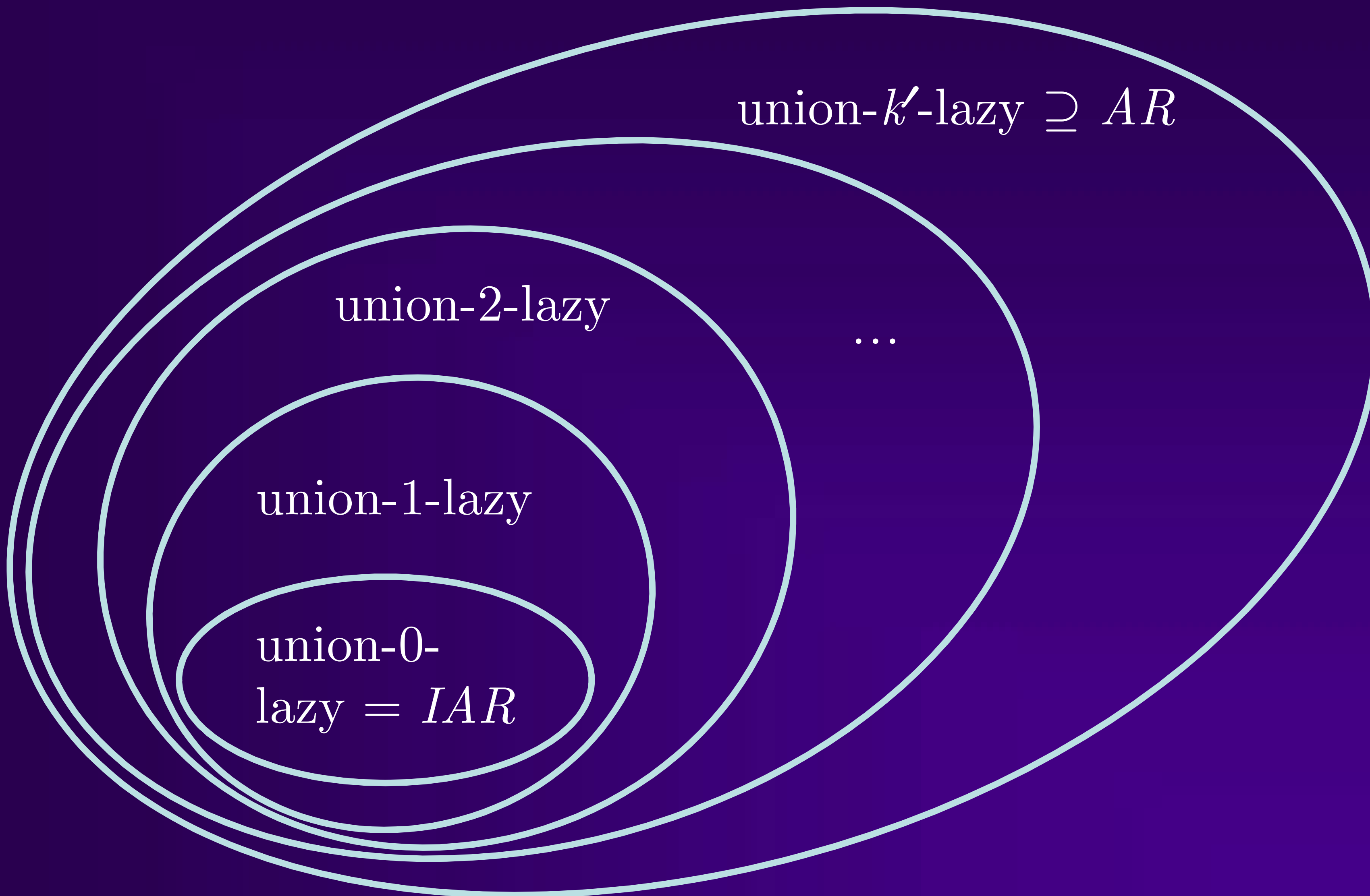
- Dada $KB = (D,\Sigma)$, y CQ Q, para cualquier $k \geq 0$, decimos que Q se infiere bajo la semántica union-k-lazy ssi $KB \vDash_{k'\text{-}LCONS} Q$ para algún $0 \leq k' \leq k$.
- Proposición: Para $k \geq 0$, cualquier respuesta bajo union-k-lazy para Q es también una respuesta bajo union-k+1-lazy.
  - Query answering bajo union-k-lazy es monotónico en k.
- Teorema: Dada $KB = (D,\Sigma)$ y CQ Q, para cualquier $k \geq 0$, el conjunto de todas las respuestas para Q bajo k-lazy y union-k-lazy Q son siempre consistentes con respecto a Σ.

UNS Bahía Blanca CONICET

k-lazy



union-k-lazy



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

Semánticas k-support y k-defeater

Semántica k-support [BienvenuIJCAI13]

- Dada $KB = (D,\Sigma)$ y una CQ Q, un conjunto $S \subseteq D$ es un Σ-*support* para Q en D si $mods(S, \Sigma) \neq \varnothing$ y $(S, \Sigma) \vDash Q$.

- Dado $KB = (D,\Sigma)$ y una CQ Q, $KB \vDash_{k\text{-}supp} Q$ si existe $S_1,\ldots,$ S_k tal que cada S_i es un Σ-support para Q *en* D, y para cada $D' \in Rep(D,\Sigma)$ existe algún $S_i \subseteq D'$.

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

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

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

Semántica k-support

- Aproximación sana de AR.
- Para cualquier $KB = (D,\Sigma)$, y CQ Q tenemos:
  - $KB \vDash_{IAR} Q$ ssi $KB \vDash_{1\text{-}supp} Q$
  - $KB \vDash_{AR} Q$ ssi $KB \vDash_{k\text{-}supp} Q$ para algún k
  - Para $k \geq 0$, si $KB \vDash_{k\text{-}supp} Q$ entonces $KB \vDash_{k+1\text{-}supp} Q$
- Para ontologías $KB = (D,\Sigma)$ FO-rewritable tal que el tamaño de los Σ-supports están acotados por la query Q, KB es FO-rewritable bajo la semántica k-support para todo $k \geq 1$.

Semántica k-defeater [BienvenuIJCAI13]

- Dada $KB = (D,\Sigma)$ y CQ Q, un k-defeater para Q en D es un conjunto $S \subseteq D$ tal que $|S| \leq k$, $mods(S, \Sigma) \neq \varnothing$, y $mods(S \cup C, \Sigma) = \varnothing$ para cada Σ-support minimal C de Q en D.
- Dada $KB = (D,\Sigma)$ y una CQ Q, $KB \vDash_{k\text{-}def} Q$ si no existe $S \subseteq D$ que sea un k-defeater para Q en D.

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

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

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

Semántica k-defeater

Una familia de aproximaciones progresivamente completas de AR, comenzando desde la semántica brave (osada).

Nota: el conjunto de respuestas puede ser inconsistente.

- Para cualquier $KB = (D,\Sigma)$, y CQ Q tenemos:
  - $KB \models_{brave} Q$ ssi $KB \models_{0\text{-}def} Q$
  - $KB \models_{AR} Q$ ssi $KB \models_{k\text{-}sdef} Q$ para todo $k \geq 0$
  - Para todo $k \geq 1$, si $KB \models_{k+1\text{-}def} Q$ entonces $KB \models_{k\text{-}def} Q$
- Si $KB = (D,\Sigma)$ es FO-rewritable tal que el tamaño de todos los culprits de D están acotados, KB es FO-rewritable bajo k-support para todo $k \geq 1$.

Consideraciones finales

- *Consistent answers* (semántica AR) es la semántica *default* para query answering en ontologías potencialmente inconsistentes.
- Las aproximaciones desarrolladas apuntan a proveer procedimientos *tratables* computacionalmente con resultados *significativos*.
- Otras alternativas como k-lazy intenta separarse del paradigma de *consistent answers* y obtener respuestas que son *consistentes entre ellas*.
- Nuevas alternativas buscan alejarse del paradigma basado en *repairs*.

Referencias

[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.

Referencias

[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.

Parte del contenido de este curso está basado en trabajo de investigación realizado en colaboración con Thomas Lukasiewicz, Georg Gottlob, V.S. Subrahmanian, Avigdor Gal, Andreas Pieris, Giorgio Orsi, Livia Predoiu y Oana Tifrea-Marciuska.

Re-escritura $\models_{IAR}$: Caso sin TGDs

- Para re-escribir una query bajo IAR necesitamos *imponer* las NCs dentro de la reescritura.
- Establecer una *correspondencia* entre la *minimización de negative constraints en la reescritura* de Q y la minimización codificada inherentemente en los culprits.

$$\Sigma_{NC} = \{\upsilon_1\colon p(U,U) \rightarrow \bot,\ \upsilon_2\colon p(X,Y) \wedge q(X) \rightarrow \bot\}$$
$$Q\colon \exists X\ q(X)$$

$$D_1 = \{p(a,a),\ q(a)\} \quad culprits(KB) = \{p(a,a)\}$$
$$D_2 = \{p(a,b),\ q(a)\} \quad culprits(KB) = \{p(a,b), q(a)\}$$

Re-escritura FO $\models_{IAR}$: 1 – Normalización de NCs

Definición: Sea $\upsilon \in \Sigma_{NC}$ y Q una BCQ; entonces, $\sim_\upsilon$ es una *relación de equivalencia* sobre los argumentos del cuerpo de υ y las constantes en υ y Q tal que cada clase de equivalencia contiene al menos una constante.

$$\upsilon_1\text{: } p(U,U) \rightarrow \perp \quad \upsilon_2\text{: } p(X,Y) \wedge q(X) \rightarrow \perp \qquad Q\text{: } q(a)$$

$\sim_{\upsilon 1}$: $\{\{U,\ a\}\}$
$\{\{U\},\ \{a\}\}$

$\sim_{\upsilon 2}$: $\{\{X, Y,\ a\}\}$
$\{\{X\}\ \{Y\}\ \{a\}\}$

Re-escritura FO $\models_{IAR}$: 1 – Normalización de NCs

Definición: Sea $\upsilon \in \Sigma_{NC}$ y Q una BCQ; la normalización de υ con respecto a $\sim_{\upsilon}$, se obtiene *reemplazando cada argumento en el cuerpo de* υ por un *representante* de su clase de equivalencia (una constante si la clase de equivalencia contiene una) y *agregando* al cuerpo la conjunción de todos los $s \neq t$ para cada par de representantes s y t tal que s sea una variable que ocurre en la instancia, y t es o bien una variable que ocurre en la instancia o una constante en Σ_{NC} y Q.

La normalización de υ, $\mathcal{N}(\upsilon, Q)$, es el conjunto de todas las instancias de υ sujetas a las relaciones de equivalencia $\sim_{\upsilon}$. y $\mathcal{N}(\Sigma_{NC}, Q) = \bigcup_{\upsilon \in \Sigma_{NC}} \mathcal{N}(\upsilon, Q)$.

Normalización de NCs

$$\Sigma_{NC} = \{\upsilon_1\text{: } p(U,U) \rightarrow \perp,\ \upsilon_2\text{: } p(X,Y) \wedge q(X) \rightarrow \perp\}$$

$$Q\text{: } \exists X\ q(X)$$

$$\mathcal{N}(\Sigma_{NC}) = \{\upsilon_1\text{': } p(U,U) \rightarrow \perp,\ \upsilon_2\text{': } p(X,X) \wedge q(X) \rightarrow \perp,$$

$$\upsilon_2\text{': } p(X,Y) \wedge q(X) \wedge X \neq Y \rightarrow \perp\}$$

FO Query Rewriting $\models_{IAR}$: 2 – Imposición de NCs

- *Identificar* el conjunto de restricciones que necesitan ser impuestas en la reescritura.

- Definición: Dada una BCQ Q y un conjunto Σ_{NC}, $\upsilon \in \mathcal{N}(\Sigma_{NC}, Q)$ *necesita ser impuesta* ssi existe $C \subseteq Q$, $C \neq \varnothing$, tal que C unifica con $B \subseteq body(\upsilon)$, y no existe υ' *tal que* $body(\upsilon)$ mapea mediante un homomorfismo en $B' \subseteq body(\upsilon)$.

$$\mathcal{N}(\Sigma_{NC}) = \{\upsilon_1\text{'}: p(U,U) \rightarrow \bot,\ \upsilon_2\text{'}: p(X,X) \wedge q(X) \rightarrow \bot,$$
$$\upsilon_3\text{'}: p(X,Y) \wedge q(X) \wedge X \neq Y \rightarrow \bot\}$$
$$Q: \exists X\ q(X)$$

υ_1', υ_2' no necesitan ser impuestas, pero υ_3' si.

UNS Bahía Blanca CONICET

FO Query Rewriting $\models_{IAR}$: 2 – Imposición de NCs

Algorithm Enforcement(*BCQ* $Q = \exists G$, *normalized negative constraints IC*)

Here, G is a quantifier-free formula, and $\exists G$ is the existential closure of G.

1. $F := G$;
2. **for every** $\upsilon \in IC$ **do**
3. **for every** $C \subseteq Q$, $C \neq \emptyset$, that unifies with $B \subseteq body(\upsilon)$ via mgu $\gamma_{C,B}$ **do**
4. **if** for no $\upsilon' \in IC$, $body(\upsilon')$ maps isomorphically to $B' \subset body(\upsilon)$ **then**
5. $F := F \wedge \neg\exists_{\overline{G}}((\bigwedge_{X \in var(C)} X = \gamma_{C,B}(X)) \wedge \gamma_{C,B}(body(\upsilon)))$
   (where $\exists_{\overline{G}} R$ is the existential closure of R
   relative to all variables in R that are not in G);
6. **return** $\exists F$.

υ_3': $p(X,Y) \wedge q(X) \wedge X \neq Y \rightarrow \bot$

Q: $\exists X\ q(X)$

$F = q(X) \wedge \neg\exists Y\ (p(X,Y) \wedge q(X) \wedge X \neq Y)$

UNS Bahía Blanca CONICET

Re-escritura FO $\models_{IAR}$: 2 – Imposición de NCs

- Proposición: Sea $KB = (D, \Sigma_{NC})$, Q una CQ, y $\Sigma_Q \subseteq \mathcal{M}(\Sigma_{NC}, Q)$ el conjunto de restricciones que necesitan ser impuestas en Q. Entonces, $KB \models_{IAR} Q$ ssi $(D, \Sigma_Q) \models_{IAR} Q$.

- Teorema: $KB \models_{IAR}$ Q ssi $D \models \mathit{enforcement}(Q, \mathcal{M}(\Sigma_{NC}, Q))$.

Re-escritura FO $\models_{IAR}$: Caso general

- Re-escritura de Q bajo IAR cuando $\Sigma = \Sigma_T \cup \Sigma_{NC}$.
- Es posible *reescribir el cuerpo de una NC* en Q, primero relativo a un conjunto de Σ_T y después imponer el nuevo conjunto de NCs (que contienen todas las posibles reescrituras de las NCs).
- Muchos trabajos han desarrollados algoritmos para reescritura FO de diferentes fragmentos de Datalog+/−; en lo que sigue asumimos el uso de un algoritmo *arbitrario* aplicable a Linear Datalog+/−.

Re-escritura FO $\vDash_{IAR}$: Caso general

- Proposición: Sea $KB = (D, \Sigma)$ con $\Sigma = \Sigma_T \cup \Sigma_{NC}$, Q una CQ, y $\Sigma_{Rew} = \{\mathrm{F} \rightarrow \bot \mid \mathrm{F} \in \mathrm{TGD}rewrite(body(\upsilon), \Sigma_T)$ con $\upsilon \in \Sigma_{NC}\}$. Entonces, $culprits(KB) = culprits(KB')$ con $KB' = (D, \Sigma_{Rew})$.

- Proposición: $KB \vDash_{ICons} Q$ ssi $(D, \Sigma_{Rew} \cup \Sigma_T) \vDash Q$.

Re-escritura FO $\models_{IAR}$: Caso general

Algorithm rewriteICons(BCQ Q, set of negative constraints and linear TGDs $\Sigma_{NC} \cup \Sigma_T$)
1. $\Sigma_{Rew} := \{F \rightarrow \bot \mid F \in$ TGD-rewrite$(body(\upsilon), \Sigma_T)$ for some $\upsilon \in \Sigma_{NC}\}$;
2. $Q_{rw} :=$ TGD-rewrite(Q, Σ_T);
3. $out := \emptyset$;
4. **for each** $Q \in Q_{rw}$ **do**
5. $\quad out := out \cup$ Enforcement$(Q, \mathcal{N}(\Sigma_{Rew}))$;
6. **return** out.

Teorema: Sea $KB = (\mathrm{D}, \Sigma)$ con (linear) $\Sigma = \Sigma_T \cup \Sigma_{NC}$, y Q una BCQ. Entonces, $KB \models_{IAR} Q$ ssi $D \models rewriteICons(Q, \Sigma)$.

rewriteICons: Ejemplo

$$\Sigma_T = \{s(X) \to q(X),\ t(X,Y) \to \exists Z\ p(Z,X)\}$$
$$\Sigma_{NC} = \{\upsilon_1\text{: } p(U,U) \to \bot,\ \upsilon_2\text{: } p(X,Y) \wedge q(X) \to \bot\}$$
$$Q\text{: } \exists X\ q(X)$$

1 – Reescribir Σ_{NC} relativo a Σ_T

$$\Sigma_T = \{s(Z) \to q(Z)\}$$
$$\Sigma_{Rew} = \{\upsilon_1\text{: } p(U,U) \to \bot,\ \upsilon_2\text{: } p(X,Y) \wedge q(X) \to \bot,$$
$$\upsilon_3\text{: } p(X,Y) \wedge s(X) \to \bot\}$$

2 – Normalizar Σ_{Rew}

$$\mathcal{N}(\Sigma_{Rew}) = \{\upsilon_1\text{': } p(U,U) \to \bot,\ \upsilon_2\text{': } p(X,X) \wedge q(X) \to \bot,$$
$$\upsilon_2\text{': } p(X,X) \wedge s(X) \to \bot,\ \upsilon_3\text{': } p(X,Y) \wedge q(X) \wedge X \neq Y \to \bot,$$
$$\upsilon_4\text{': } p(X,Y) \wedge s(X) \wedge X \neq Y \to \bot\}$$

rewriteICons: Ejemplo

$$\mathcal{N}(\Sigma_{Rew}) = \{\upsilon_1\text{'}: p(U,U) \rightarrow \perp,\ \upsilon_2\text{'}: p(X,X) \wedge q(X) \rightarrow \perp,$$
$$\upsilon_2\text{'}: p(X,X) \wedge s(X) \rightarrow \perp,\ \upsilon_3\text{'}: p(X,Y) \wedge q(X) \wedge X \neq Y \rightarrow \perp,$$
$$\upsilon_4\text{'}: p(X,Y) \wedge s(X) \wedge X \neq Y \rightarrow \perp\}$$

3 – Reescribir Q relativo a Σ_T

$$Q_{rew} = \{\exists X_1\ q(X_1),\ \exists X_2\ s(X_2)\}$$

4 – Imponer $\mathcal{N}(\Sigma_{Rew})$ en Q_{rew}

$F_1 = q(X_1) \wedge \neg\exists Y\ (p(X_1, Y) \wedge q(X_1) \wedge X_1 \neq Y)$	$F_2 = q(X) \wedge \neg\exists Y\ (p(X, Y) \wedge s(X) \wedge X \neq Y)$

$$\exists X\ [q(X) \wedge \neg\exists Y\ (p(X,Y) \wedge q(X) \wedge X \neq Y)]$$
$$\vee$$
$$\exists X\ [q(X) \wedge \neg\exists Y\ (p(X,Y) \wedge s(X) \wedge X \neq Y)]$$