Digital Knowledge: From Facts to Rules and Back

Daria Stepanova
D5: Databases and Information Systems
Max Planck Institute for Informatics

03.05.2017

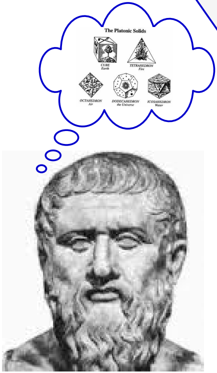
max planck institut
informatik

What is Knowledge?

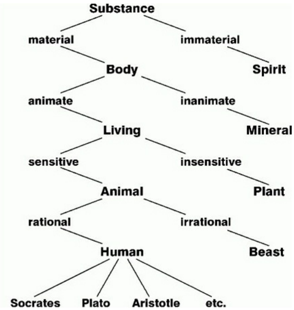
Plato: *"Knowledge is justified true belief"*

350 BC

Personal



External



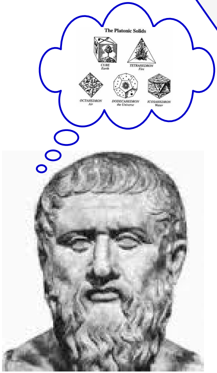
Tree of Knowledge

What is Knowledge?

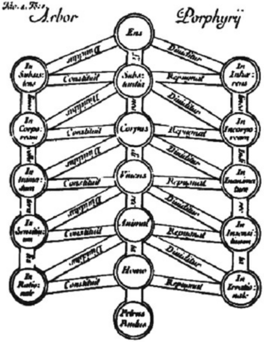
Plato: *"Knowledge is justified true belief"*

350 BC

Personal



External



What is Digital Knowledge?

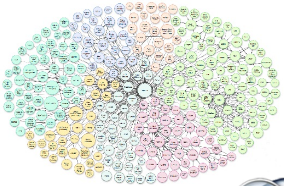
“Digital knowledge is semantically enriched machine processable data”

2017

Personal

External

yago
select knowledge

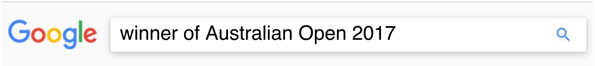


WIKIDATA

NELL

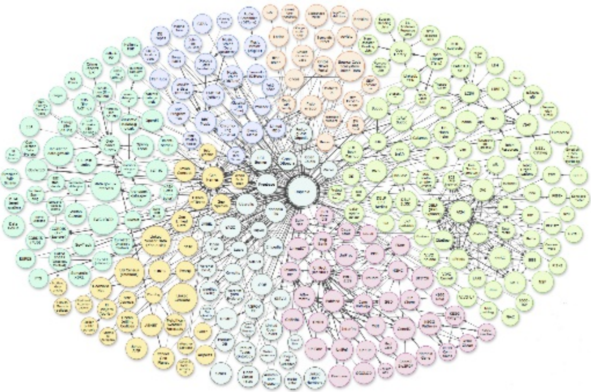
Knowledge Graph

Semantic Web Search



Semantic Web Search

Google $\exists X \ \mathrm{winnerOf}(X, \mathrm{AustralianOpen2017})$



Roger Federer

Tennis player

rogerfederer.com

Roger Federer is a Swiss professional tennis player who is currently ranked world No. 10 by the Association of Tennis Professionals. Many players and analysts have called him the greatest tennis player of all time. Wikipedia

Born: August 8, 1981 (age 35 years), Basel, Switzerland

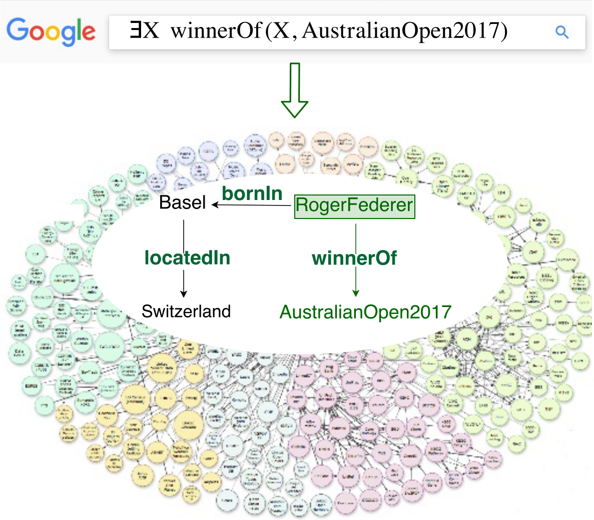
Height: 1.85 m

Weight: 85 kg

Spouse: Mirka Federer (m. 2009)

Children: Lenny Federer, Myla Rose Federer, Charlene Riva Federer, Leo Federer

Semantic Web Search



Roger Federer

Tennis player

rogerfederer.com

Roger Federer is a Swiss professional tennis player who is currently ranked world No. 10 by the Association of Tennis Professionals. Many players and analysts have called him the greatest tennis player of all time. Wikipedia

Born: August 8, 1981 (age 35 years), Basel, Switzerland

Height: 1.85 m

Weight: 85 kg

Spouse: Mirka Federer (m. 2009)

Children: Lenny Federer, Myla Rose Federer, Charlene Riva Federer, Leo Federer

Knowledge Graphs

https://en.wikipedia.org/wiki/Roger_Federer 83% Search

Main page
Contents
Featured content
Current events
Random article
Donate to Wikipedia
Wikipedia store

Interaction
Help
About Wikipedia
Community portal
Recent changes
Contact page

Tools
What links here
Related changes
Upload file
Special pages
Permanent link
Page information
Wikidata item
Cite this page

Print/export
Create a book
Download as PDF
Printable version

In other projects
Wikimedia Commons
Wikinews
Wikiquote

Languages
Acèh
Afrikaans
Alemannisch
አማርኛ
Ænglisc
العربية
Aragonés
Asturianu
Aymar aru
Azərbaycanca
বাংলা
Bân-lâm-gú
Беларуская
Беларуская (тарашкевіца)
Bikol Central
Български
Boarisch
Bosanski
Brezhoneg
Буряад
Català
Чӑвашла

"Federer" redirects here. For other uses, see Federer (disambiguation).

Roger Federer (born 8 August 1981) is a Swiss professional tennis player. Many players and analysts have called him the greatest tennis player of all time.[a] Federer turned professional in 1998 and was continuously ranked in the top 10 from October 2002 to November 2016.[19] He is currently ranked world No. 4 by the Association of Tennis Professionals (ATP).[20]

Federer has won 18 Grand Slam singles titles, the most in history for a male tennis player, and held the No. 1 spot in the ATP rankings for a total of 302 weeks. In majors, Federer has won seven Wimbledon titles, five Australian Open titles, five US Open titles and one French Open title. He is among the eight men to capture a career Grand Slam. He has reached a record 28 men's singles Grand Slam finals, including 10 in a row from the 2005 Wimbledon Championships to the 2007 US Open.

Federer's ATP tournament records include winning a record six ATP World Tour Finals and playing in the finals at all nine ATP Masters 1000 tournaments. He also won the Olympic gold medal in doubles with his compatriot Stan Wawrinka at the 2008 Summer Olympic Games and the Olympic silver medal in singles at the 2012 Summer Olympic Games. Representing Switzerland, he was a part of the 2014 winning Davis Cup team. He was named the Laureus World Sportsman of the Year for a record four consecutive years from 2005 to 2008.

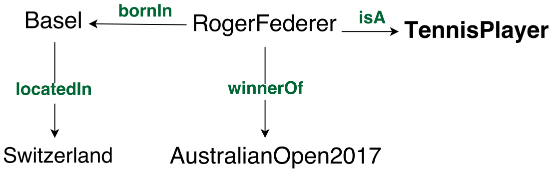
Contents [hide]

Roger Federer

Federer at 2009 Wimbledon where he broke the Grand Slam record

Country (sports)	Switzerland
Residence	Bottmingen, Switzerland[1]
Born	8 August 1981 (age 35) Basel, Switzerland
Height	1.85 m (6 ft 1 in)[2]
Turned pro	1998
Plays	Right-handed (one-handed backhand)
Prize money	US$ 103,990,195
Official website	rogerfederer.com
Singles	
Career record	1099–246 (81.71% in Grand Slam and ATP World Tour main draw matches, in Summer Olympics and in Davis Cup)
Career titles	91 (3rd in the Open Era)
Highest ranking	No. **1** (2 February 2004)
Current ranking	No. 4 (3 April 2017)[3]
Grand Slam Singles results	
Australian Open	**W** (2004, 2006, 2007, 2010, 2017)

Knowledge Graphs



KGs are huge collections of positive unary and binary facts

tennisPlayer(rogerFederer)
bornIn(rogerFederer, basel)

Country (sports)	Switzerland
Residence	Bottmingen, Switzerland[1]
Born	8 August 1981 (age 35) Basel, Switzerland
Height	1.85 m (6 ft 1 in)[2]
Turned pro	1998
Plays	Right-handed (one-handed backhand)
Prize money	US$ 103,990,195
Official website	rogerfederer.com
Singles	
Career record	1099–246 (81.71% in Grand Slam and ATP World Tour main draw matches, in Summer Olympics and in Davis Cup)
Career titles	91 (3rd in the Open Era)
Highest ranking	No. **1** (2 February 2004)
Current ranking	No. 4 (3 April 2017)[3]
Grand Slam Singles results	
Australian Open	**W** (2004, 2006, 2007, 2010, 2017)

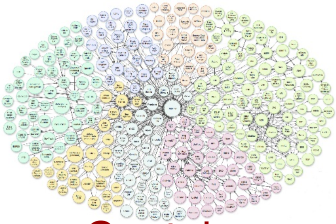
Problem: Inconsistency

Knowledge Graph

Information Extraction

Web pages

Difficult

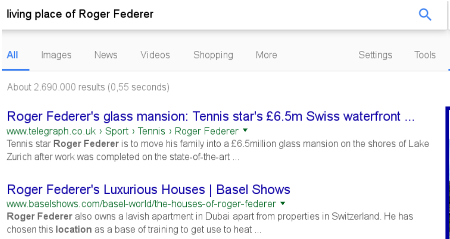


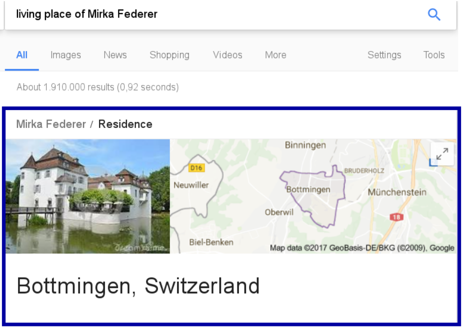
Noisy

Contains mistakes

Problem: Incompleteness

Google KG misses Roger's living place, but contains his wife's Mirka's..





Motivation

Important problems of KGs:

1. Inconsistency
2. Incompleteness

In this talk: Reasoning on top of KGs to address these issues

1. Deduction: detecting and repairing inconsistencies
2. Induction: learning common-sense rules and completing KGs

Overview

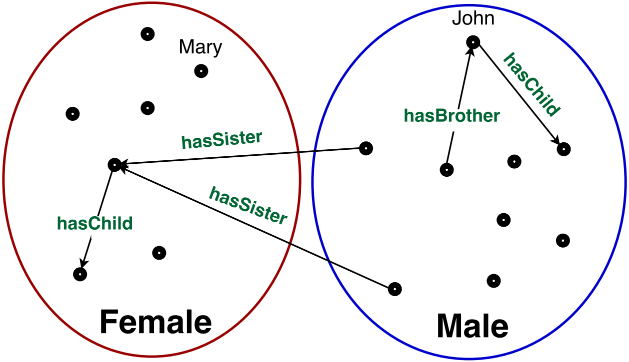
- ✓ Motivation
- **Ontologies and Rules**
- Inconsistencies in DL-programs
- Nonmonotonic Rule Mining
- Further and Future Work

History of Knowledge Representation

- **1950's:** First Order Logic (FOL) for KR (undecidable)
  (e.g. [McCarthy, 1959])
- **1970's:** Network-shaped structures for KR (no formal semantics)
  (e.g. semantic networks [Robinson, 1965], frames [Minsky, 1985])
- **1979:** Encoding of network-shaped structures into FOL [Hayes, 1979]
- **1980's:** Description Logics (DL) for KR
  - Decidable fragments of FOL
  - Theories encoded in DLs are called ontologies
  - Many DLs with different expressiveness and computational features
  - Particularly suited for conceptual reasoning

Description Logic Ontologies

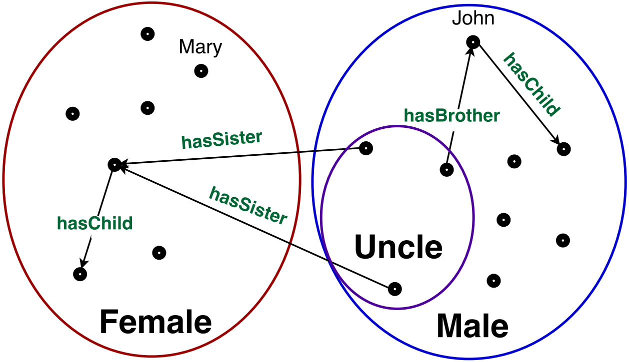
Open World Assumption (OWA): what is not derived is unknown



Inclusions: $Female \sqsubseteq \neg Male, hasSister \sqsubseteq hasSibling, hasBrother \sqsubseteq hasSibling$

Description Logic Ontologies

Open World Assumption (OWA): what is not derived is unknown



Inclusions: $Female \sqsubseteq \neg Male, hasSister \sqsubseteq hasSibling, hasBrother \sqsubseteq hasSibling$

Complex axioms: $Uncle \equiv Male \sqcap \exists hasSibling.\exists hasChild$

What can not be said in DLs?

- Exceptions from theories (due to monotonicity)

What can not be said in DLs?

- Exceptions from theories (due to monotonicity)

$\mathit{WithBeard} \sqsubseteq \mathit{Male}$	*People with beards are male*
$\mathit{Female} \sqsubseteq \neg \mathit{Male}$	*Female are not male*
$\mathit{WithBeard}(c)$	*C has a beard*

What can not be said in DLs?

- Exceptions from theories (due to monotonicity)

$WithBeard \sqsubseteq Male$	*People with beards are male*
$Female \sqsubseteq \neg Male$	*Female are not male*
$WithBeard(c)$	*C has a beard*
$Male(c)$	*C is male*

What can not be said in DLs?

- Exceptions from theories (due to monotonicity)

$\mathit{WithBeard} \sqsubseteq \mathit{Male}$	*People with beards are male*
$\mathit{Female} \sqsubseteq \neg \mathit{Male}$	*Female are not male*
$\mathit{WithBeard}(c)$	*C has a beard*
$\mathit{Female}(c)$	*C is female*
$\mathit{Male}(c)$	*C is male*
$\neg \mathit{Male}(c)$	*C is not male*

Monotonicity: the more we add, the more we get!

History of Knowledge Representation

- **1970's:** Logic programming
  (e.g. Prolog)

- **1980's:** Nonmonotonic logics
  (e.g. circumscription [McCarthy, 1980], default logic [Reiter, 1980])

- **1988:** Nonmonotonic rules under answer set semantics (ASP)
  [Gelfond and Lifschitz, 1988]
  - Logic programs with model-based semantics
  - Disjunctive datalog with default negation *not*

Not is not $\neg$!

Default negation *not*

At a rail road crossing cross the road if **no train is known** to approach

$walk \leftarrow at(X), crossing(X), \textbf{not}\ train\_approaches(X)$

Classical negation $\neg$

At a rail road crossing cross the road if **no train** approaches

$walk \leftarrow at(X), crossing(X), \neg train\_approaches(X)$

Nonmonotonic Rules

Closed World Assumption (CWA): what is not derived is false

Rule: $\underbrace{a_1 \vee \ldots \vee a_k}_{\text{head}} \leftarrow \underbrace{b_1, \ldots, b_m, \, not \; b_{m+1}, \ldots, \, not \; b_n}_{\text{body}}$

Informal semantics: If $b_1, \ldots, b_m$ are true and none of $b_{m+1}, \ldots, b_n$ is known, then at least one among $a_1, \ldots, a_k$ must be true

Default negation: unless a child is adopted one of his parents must be female

$$female(Y) \vee female(Z) \leftarrow hasParent(X, Y), hasParent(X, Z), Y \neq Z, not \; adopted(X)$$

Constraint: ensure that no one is a parent of himself

$$\perp \leftarrow parent(X, Y), parent(Y, X)$$

Answer Set Programs

Evaluation of ASP programs is model-based

Answer set program (ASP) is a set of nonmonotonic rules

(1) $hasParent(john, pat)$ (2) $hasParent(john, alex)$ (3) $male(alex)$

(4) $female(Y) \leftarrow hasParent(X, Y), hasParent(X, Z),$
$Y \neq Z, male(Z), not\ \ adopted(X)$

Answer Set Programs

Evaluation of ASP programs is model-based

1. Grounding: substitute all variables with constants in all possible ways

Answer set program (ASP) is a set of nonmonotonic rules

(1) $hasParent(john, pat)$ (2) $hasParent(john, alex)$ (3) $male(alex)$

$$(4)\ female(Y) \leftarrow hasParent(X, Y), hasParent(X, Z),\\ Y \neq Z, male(Z), not\ \ adopted(X)$$

Answer Set Programs

Evaluation of ASP programs is model-based

1. Grounding: substitute all variables with constants in all possible ways

Answer set program (ASP) is a set of nonmonotonic rules

(1) $hasParent(john, pat)$ (2) $hasParent(john, alex)$ (3) $male(alex)$

(4) $female(pat) \leftarrow hasParent(john, pat), hasParent(john, alex),$
$male(alex), not\ \ adopted(john)$

Answer Set Programs

Evaluation of ASP programs is model-based

1. Grounding: substitute all variables with constants in all possible ways
2. Solving: compute a minimal model (answer set) I satisfying all rules

Answer set program (ASP) is a set of nonmonotonic rules

(1) $hasParent(john, pat)$ (2) $hasParent(john, alex)$ (3) $male(alex)$

$$(4)\ female(pat) \leftarrow hasParent(john, pat), hasParent(john, alex),$$
$$male(alex), not\ \ adopted(john)$$

$I = \{hasParent(john, pat), hasParent(john, alex), male(alex), female(pat)\}$

CWA: $adopted(john)$ can not be derived, thus it is false

Answer Set Programs

Evaluation of ASP programs is model-based

1. Grounding: substitute all variables with constants in all possible ways
2. Solving: compute a minimal model (answer set) I satisfying all rules

Answer set program (ASP) is a set of nonmonotonic rules

$$(1)\ hasParent(john, pat) \quad (2)\ hasParent(john, alex) \quad (3)\ male(alex)$$

$$(4)\ female(pat) \leftarrow hasParent(john, pat), hasParent(john, alex), male(alex), not\ \ adopted(john)$$

$$(5)\ adopted(john)$$

$$I = \{hasParent(john, pat), hasParent(john, alex), male(alex), \overset{adopted(john)}{\cancel{female(pat)}}\}$$

Nonmonotonicity: adding facts might lead to loss of consequences!

Combining Ontologies and Rules

DL Ontologies	**Rules**
Open-World Assumption	Closed-World Assumption
Monotonic	Nonmonotonic
Conceptual reasoning	Defaults and exceptions
...	**...**

Combining Ontologies and Rules

Hybrid Knowledge Bases

MKNF, DL-safe rules, **DL-programs**...

DL Ontologies

Open-World Assumption

Monotonic

Conceptual reasoning

...

Rules

Closed-World Assumption

Nonmonotonic

Defaults and exceptions

...

Overview

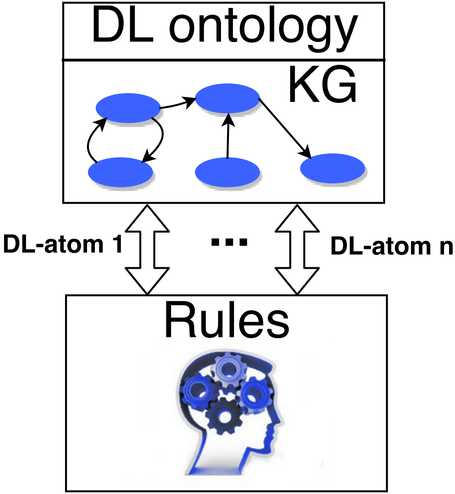
- ✓ Motivation
- ✓ Ontologies and Rules
- **Inconsistencies in DL-programs**
- Nonmonotonic Rule Mining
- Further and Future Work

T. Eiter

M. Fink

DL-programs

DL-programs: loose coupling of ontologies and rules [Eiter *et al.*, 2008]



DL-program

DL ontology

Logical part

(1) $Child \sqsubseteq \exists hasParent$
(2) $Female \sqsubseteq \neg Male$
(3) $Adopted \sqsubseteq Child$

Data part (KG)

(4) $Male(pat)$
(5) $Male(john)$
(6) $hasParent(john, pat)$

Rules

(7) $isChildOf(john, alex)$ (8) $boy(tim)$

(9) $hasFather(john, pat) \leftarrow$  , 

DL-program

DL ontology

Logical part

(1) $Child \sqsubseteq \exists hasParent$
(2) $Female \sqsubseteq \neg Male$
(3) $Adopted \sqsubseteq Child$

Data part (KG)

(4) $Male(pat)$
(5) $Male(john)$
(6) $hasParent(john, pat)$

Rules

(7) $isChildOf(john, alex)$ (8) $boy(tim)$

(9) $hasFather(john, pat) \leftarrow \mathrm{DL}[; hasParent](john, pat),$ 

DL-program

DL ontology

Logical part

(1) $Child \sqsubseteq \exists hasParent$
(2) $Female \sqsubseteq \neg Male$
(3) $Adopted \sqsubseteq Child$

Data part (KG)

(4) $Male(pat)$
(5) $Male(john)$
(6) $hasParent(john, pat)$

Rules

(7) $isChildOf(john, alex)$ (8) $boy(tim)$

(9) $hasFather(john, pat) \leftarrow \mathrm{DL}[; hasParent](john, pat)$ $\checkmark$, 

DL-program

DL ontology

Logical part

(1) $Child \sqsubseteq \exists hasParent$
(2) $Female \sqsubseteq \neg Male$
(3) $Adopted \sqsubseteq Child$

Data part (KG)

(4) $Male(pat)$
(5) $Male(john)$
(6) $hasParent(john, pat)$

Rules

(7) $isChildOf(john, alex)$ (8) $boy(tim)$
(9) $hasFather(john, pat) \leftarrow \text{DL}[; hasParent](john, pat)$ ✓,
$\text{DL}[Male \uplus boy; Male](pat)$

DL-program

DL ontology

Logical part

(1) $Child \sqsubseteq \exists hasParent$
(2) $Female \sqsubseteq \neg Male$
(3) $Adopted \sqsubseteq Child$

Data part (KG)

(4) $Male(pat)$
(5) $Male(john)$
(6) $hasParent(john, pat)$

Rules

(7) $isChildOf(john, alex)$ (8) $boy(tim)$

(9) $hasFather(john, pat) \leftarrow \mathrm{DL}[;\, hasParent](john, pat)$ ✓,
$\mathrm{DL}[Male \uplus boy;\, Male](pat)$

DL-program

DL ontology

Logical part

(1) $Child \sqsubseteq \exists hasParent$
(2) $Female \sqsubseteq \neg Male$
(3) $Adopted \sqsubseteq Child$

Data part (KG)

(4) $Male(pat)$ $\quad$ $Male(tim)$
(5) $Male(john)$
(6) $hasParent(john, pat)$

Rules

(7) $isChildOf(john, alex)$ $\quad$ (8) $boy(tim)$
(9) $hasFather(john, pat) \leftarrow \mathrm{DL}[; hasParent](john, pat)$ ✓,
$\mathrm{DL}[Male \uplus boy; Male](pat)$

DL-program

DL ontology

Logical part

(1) $Child \sqsubseteq \exists hasParent$
(2) $Female \sqsubseteq \neg Male$
(3) $Adopted \sqsubseteq Child$

Data part (KG)

(4) $Male(pat)$ $Male(tim)$
(5) $Male(john)$
(6) $hasParent(john, pat)$

Rules

(7) $isChildOf(john, alex)$ (8) $boy(tim)$

(9) $hasFather(john, pat) \leftarrow \mathrm{DL}[; hasParent](john, pat)$ ✓,
$\mathrm{DL}[Male \uplus boy; Male](pat)$ ✓

DL-program

DL ontology

Logical part

(1) $Child \sqsubseteq \exists hasParent$
(2) $Female \sqsubseteq \neg Male$
(3) $Adopted \sqsubseteq Child$

Data part (KG)

(4) $Male(pat)$
(5) $Male(john)$
(6) $hasParent(john, pat)$

Rules

(7) $isChildOf(john, alex)$ (8) $boy(tim)$

(9) $hasFather(john, pat) \leftarrow \mathrm{DL}[;\, hasParent](john, pat)$ ✓,
$\mathrm{DL}[Male \uplus boy;\, Male](pat)$ ✓

Answer set: $I = \{isChildOf(john, alex), boy(tim), hasFather(john, pat)\}$

DL-program

DL ontology

Logical part

(1) $Child \sqsubseteq \exists hasParent$
(2) $Female \sqsubseteq \neg Male$
(3) $Adopted \sqsubseteq Child$

Data part (KG)

(4) $Male(pat)$
(5) $Male(john)$
(6) $hasParent(john, pat)$

Rules

(7) $isChildOf(john, alex)$ (8) $boy(tim)$

(9) $hasFather(john, pat) \leftarrow \mathrm{DL}[; hasParent](john, pat),$
$\mathrm{DL}[Male \uplus boy; Male](pat)$

(10) $\perp \leftarrow hasFather(john, pat), isChildOf(john, alex),$
$not\ \mathrm{DL}[; Adopted](john),$
$not\ \mathrm{DL}[Child \uplus boy; \neg Male](alex)$

DL-program

DL ontology

Logical part

(1) $Child \sqsubseteq \exists hasParent$
(2) $Female \sqsubseteq \neg Male$
(3) $Adopted \sqsubseteq Child$

Data part (KG)

(4) $Male(pat)$
(5) $Male(john)$
(6) $hasParent(john, pat)$

Rules

(7) $isChildOf(john, alex)$ (8) $boy(tim)$

(9) $hasFather(john, pat) \leftarrow \mathrm{DL}[; hasParent](john, pat),$
$\mathrm{DL}[Male \uplus boy; Male](pat)$

(10) $\perp \leftarrow hasFather(john, pat), isChildOf(john, alex),$
$not\ \mathrm{DL}[; Adopted](john),$
$not\ \mathrm{DL}[Child \uplus boy; \neg Male](alex)$

DL-program

DL ontology

Logical part

(1) $Child \sqsubseteq \exists hasParent$
(2) $Female \sqsubseteq \neg Male$
(3) $Adopted \sqsubseteq Child$

Data part (KG)

(4) $Male(pat)$
(5) $Male(john)$
(6) $hasParent(john, pat)$

Rules

(7) $isChildOf(john, alex)$ (8) $boy(tim)$

(9) $hasFather(john, pat) \leftarrow \text{DL}[; hasParent](john, pat),$
$\text{DL}[Male \uplus boy; Male](pat)$

(10) $\bot \leftarrow hasFather(john, pat), isChildOf(john, alex),$
$not\ \text{DL}[; Adopted](john),$
$not\ \text{DL}[Child \uplus boy; \neg Male](alex)$

DL-program

DL ontology

Logical part

(1) $Child \sqsubseteq \exists hasParent$
(2) $Female \sqsubseteq \neg Male$
(3) $Adopted \sqsubseteq Child$

Data part (KG)

(4) $Male(pat)$
(5) $Male(john)$
(6) $hasParent(john, pat)$

Rules

(7) $isChildOf(john, alex)$ (8) $boy(tim)$

(9) $hasFather(john, pat) \leftarrow \mathrm{DL}[;hasParent](john, pat),$
$\mathrm{DL}[Male \uplus boy; Male](pat)$

(10) $\bot \leftarrow hasFather(john, pat), isChildOf(john, alex),$
$not\ \mathrm{DL}[;Adopted](john),$
$not\ \mathrm{DL}[Child \uplus boy; \neg Male](alex)$

DL-program

DL ontology

Logical part

(1) $Child \sqsubseteq \exists hasParent$

(2) $Female \sqsubseteq \neg Male$

(3) $Adopted \sqsubseteq Child$

Data part (KG)

(4) $Male(pat)$

(5) $Male(john)$

(6) $hasParent(john, pat)$

Rules

(7) $isChildOf(john, alex)$ (8) $boy(tim)$

(9) $hasFather(john, pat) \leftarrow \mathrm{DL}[; hasParent](john, pat), \mathrm{DL}[Male \uplus boy; Male](pat)$

(10) $\perp \leftarrow hasFather(john, pat), isChildOf(john, alex), not\ \mathrm{DL}[; Adopted](john), not\ \mathrm{DL}[Child \uplus boy; \neg Male](alex)$

Inconsistent DL-program

DL ontology

Logical part

(1) $Child \sqsubseteq \exists hasParent$
(2) $Female \sqsubseteq \neg Male$
(3) $Adopted \sqsubseteq Child$

Data part (KG)

(4) $Male(pat)$
(5) $Male(john)$
(6) $hasParent(john, pat)$

Rules

(7) $isChildOf(john, alex)$ (8) $boy(tim)$

(9) $hasFather(john, pat) \leftarrow \mathrm{DL}[; hasParent](john, pat),$
$\mathrm{DL}[Male \uplus boy; Male](pat)$

(10) $\perp \leftarrow hasFather(john, pat), isChildOf(john, alex),$
$not\ \mathrm{DL}[; Adopted](john),$
$not\ \mathrm{DL}[Child \uplus boy; \neg Male](alex)$

Inconsistent DL-program: no answer sets!

DL-program Repair

DL ontology

Logical part

(1) $Child \sqsubseteq \exists hasParent$
(2) $Female \sqsubseteq \neg Male$
(3) $Adopted \sqsubseteq Child$

Data part (KG)

(4) $Male(pat)$
(5) $Male(john)$
(6) $hasParent(john, pat)$

$Female(alex)$

Rules

(7) $isChildOf(john, alex)$ (8) $boy(tim)$

(9) $hasFather(john, pat) \leftarrow \text{DL}[;hasParent](john, pat),$
$\text{DL}[Male \uplus boy; Male](pat)$

(10) $\perp \leftarrow hasFather(john, pat), isChildOf(john, alex),$
$not\ \text{DL}[;Adopted](john),$
$not\ \text{DL}[Child \uplus boy; \neg Male](alex)$

Repair answer set: $I = \{isChildOf(john, alex), boy(tim), hasFather(john, pat)\}$

DL-program Repair

DL ontology

Logical part

(1) $Child \sqsubseteq \exists hasParent$
(2) $Female \sqsubseteq \neg Male$
(3) $Adopted \sqsubseteq Child$

Data part (KG)

(4) $\cancel{Male(pat)}$ $Female(pat)$
(5) $Male(john)$
(6) $hasParent(john, pat)$

Rules

(7) $isChildOf(john, alex)$ (8) $boy(tim)$

(9) $hasFather(john, pat) \leftarrow \text{DL}[; hasParent](john, pat),$
$\text{DL}[Male \uplus boy; Male](pat)$

(10) $\bot \leftarrow hasFather(john, pat), isChildOf(john, alex),$
$not\ \text{DL}[; Adopted](john),$
$not\ \text{DL}[Child \uplus boy; \neg Male](alex)$

Repair answer set: $I = \{isChildOf(john, alex), boy(tim)\}$

DL-program Repair

DL ontology

Logical part

(1) $Child \sqsubseteq \exists hasParent$
(2) $Female \sqsubseteq \neg Male$
(3) $Adopted \sqsubseteq Child$

Data part (KG)

(4) $Male(pat)$
(5) $Male(john)$

Rules

(7) $isChildOf(john, alex)$ (8) $boy(tim)$

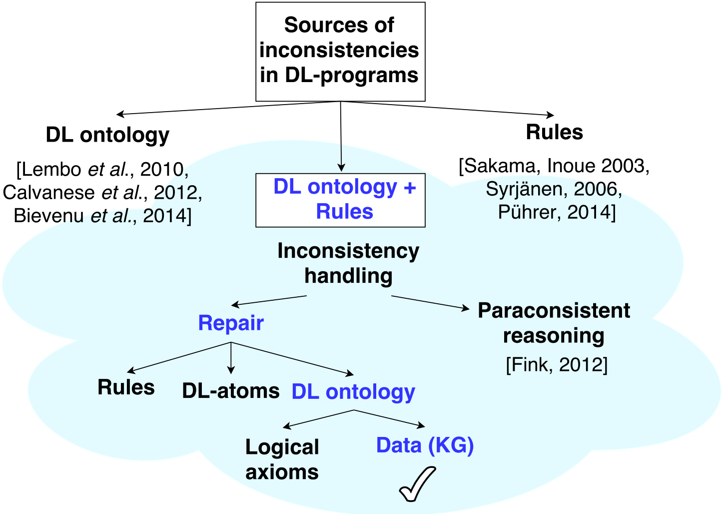
(9) $hasFather(john, pat) \leftarrow \mathrm{DL}[; hasParent](john, pat), \mathrm{DL}[Male \uplus boy; Male](pat)$

(10) $\bot \leftarrow hasFather(john, pat), isChildOf(john, alex), \mathit{not}\ \mathrm{DL}[; Adopted](john), \mathit{not}\ \mathrm{DL}[Child \uplus boy; \neg Male](alex)$

Repair answer set: $I = \{isChildOf(john, alex), boy(tim)\}$

Inconsistency Handling in DL-programs

Goal: develop techniques for handling inconsistencies in DL-programs
Approach: repair ontology data part (KG) to regain consistency



Complexity of Repair Answer Sets

INSTANCE: A ground DL-program $\Pi = \langle O, P \rangle$.
QUESTION: Does there exist a repair answer set for Π?

Theorem

Deciding repair and standard answer set existence have the same complexity if instance query-answering in O is polynomial (DL-Lite$_\mathcal{A}, \mathcal{EL}$).

Π	FLP semantics	weak semantics
normal	Σ_2^P-complete	NP-complete
disjunctive	Σ_2^P-complete	Σ_2^P-complete

T. Eiter, M. Fink, D. Stepanova. Data Repair of Inconsistent DL-programs. *IJCAI2013*
T. Eiter, M. Fink, D. Stepanova. Data Repair of Inconsistent Nonmonotonic Description Logic Programs. *JAI2016*

Ontology Repair Problem

INSTANCE: Ontology O, $D_{true} = \{\langle update, query \rangle\}$, $D_{false} = \{\langle update, query \rangle\}$
QUESTION: Does there exist O data part, for which queries under their updates from D_{true} are true and from D_{false} are false?

Theorem

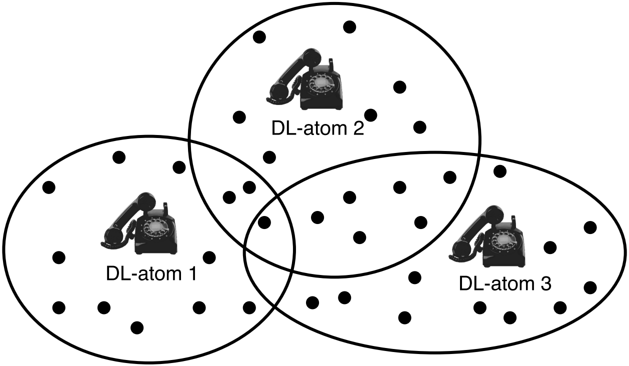
The Ontology Repair Problem is NP*-complete even if* $O = \emptyset$*.*

Tractable cases:

- Deletion repair
- Bounded addition
- Bounded change
- ...

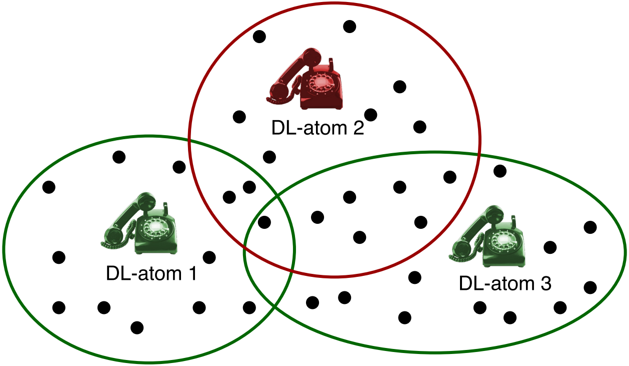
Optimized DL-program Repair

- For each DL-atom compute minimal sets of facts (●), whose presence in ontology ensures DL-atom's query entailment (small for some DLs)



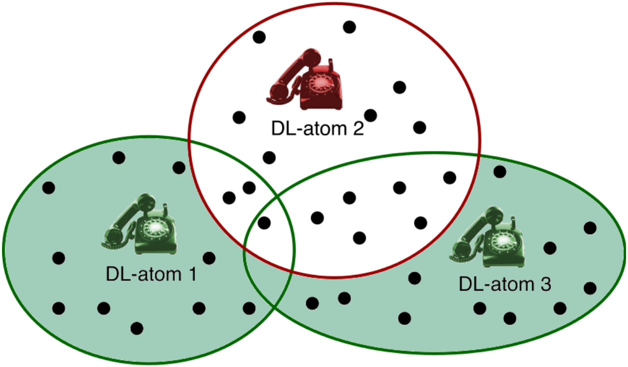
Optimized DL-program Repair

- For each DL-atom compute minimal sets of facts (●), whose presence in ontology ensures DL-atom's query entailment (small for some DLs)
- Guess values of DL-atoms under which the program has an answer set



Optimized DL-program Repair

- For each DL-atom compute minimal sets of facts (●), whose presence in ontology ensures DL-atom's query entailment (small for some DLs)
- Guess values of DL-atoms under which the program has an answer set
- Solve ontology repair problem as a variant of a hitting set problem



T. Eiter, M. Fink, D. Stepanova. Towards Practical Deletion Repair of Inconsistent DL-programs. *ECAI2014*
T. Eiter, M. Fink, D. Stepanova. Computing Repairs for Inconsistent DL-programs over $\mathcal{EL}$ Ontologies. *JELIA2014, JAIR2016*

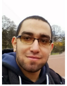
Example Benchmark

- Ontology: MyITS[1]
  - personalized route planning with semantic information
  - logical axioms (406), (building features located inside private areas are not publicly accessible, covered bus stops are those with roofs)
  - KG (4195 facts), Cork city map with leisure areas, bus stops,..
- Rules: check that public stations don't lack public access, using CWA on private areas
- Inconsistency: wrong GPS coordinates result in roofed bus stops being located inside private areas
- Repair: found within 12 seconds

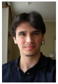
[1] http://www.kr.tuwien.ac.at/research/projects/myits/

Overview

- ✓ Motivation
- ✓ Ontologies and Rules
- ✓ Inconsistencies in DL-programs
- **Nonmonotonic Rule Mining**
- Further and Future Work



M. Gad-Elrab



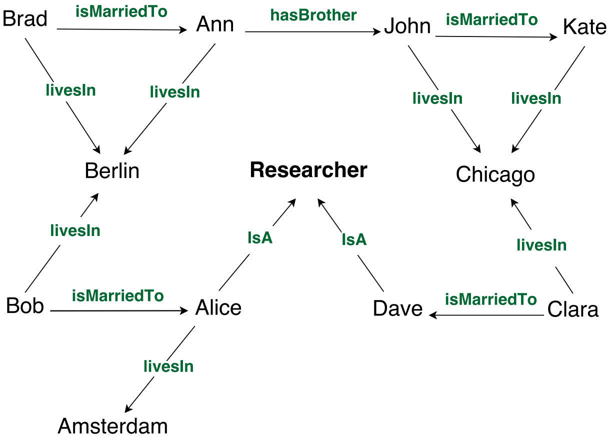
J. Urbani

G. Weikum

D. H. Tran

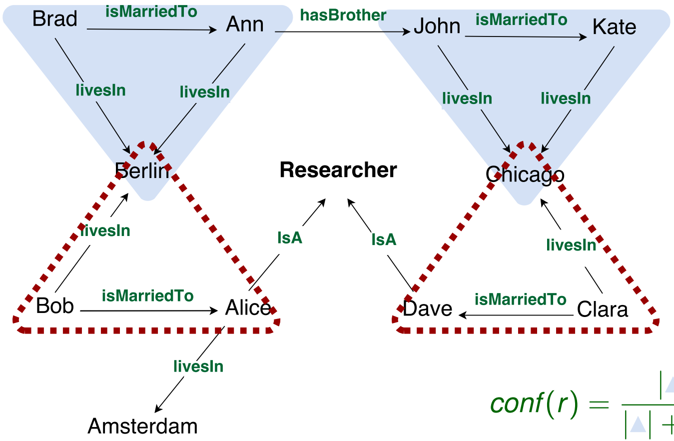
F. A. Lisi

Horn Rule Mining



Horn Rule Mining

Horn rule mining for KG completion [Galárraga *et al.*, 2015]

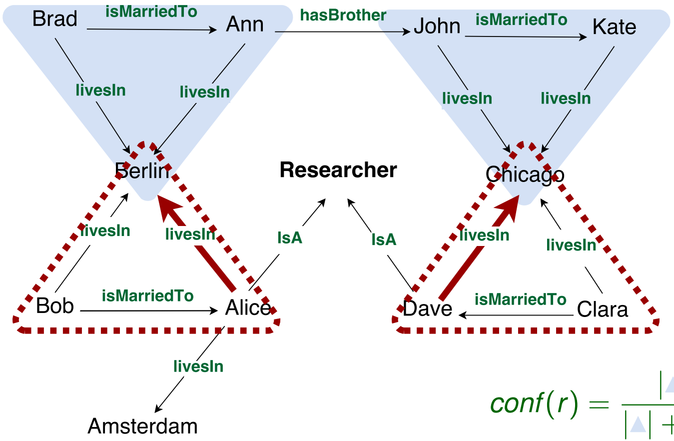


$$conf(r) = \frac{|\blacktriangle|}{|\blacktriangle| + |\triangle|} = \frac{2}{4}$$

$$r: \ livesIn(X, Z) \leftarrow isMarriedTo(Y, X), livesIn(Y, Z)$$

Horn Rule Mining

Horn rule mining for KG completion [Galárraga *et al.*, 2015]

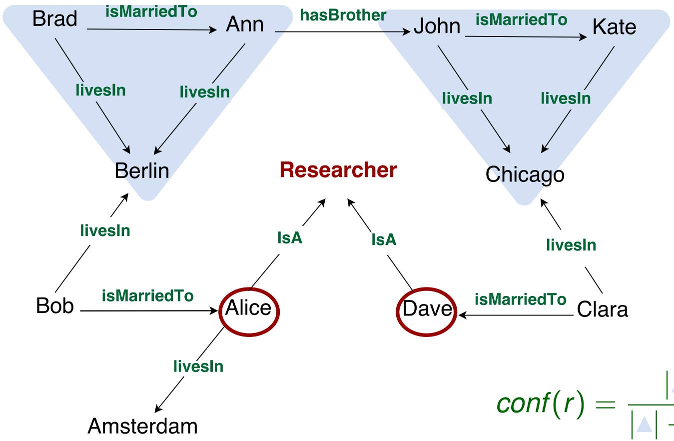


$$conf(r) = \frac{|\triangle|}{|\triangle| + |\triangle|} = \frac{2}{4}$$

$$r: \; livesIn(X, Z) \leftarrow isMarriedTo(Y, X), livesIn(Y, Z)$$

Nonmonotonic Rule Mining

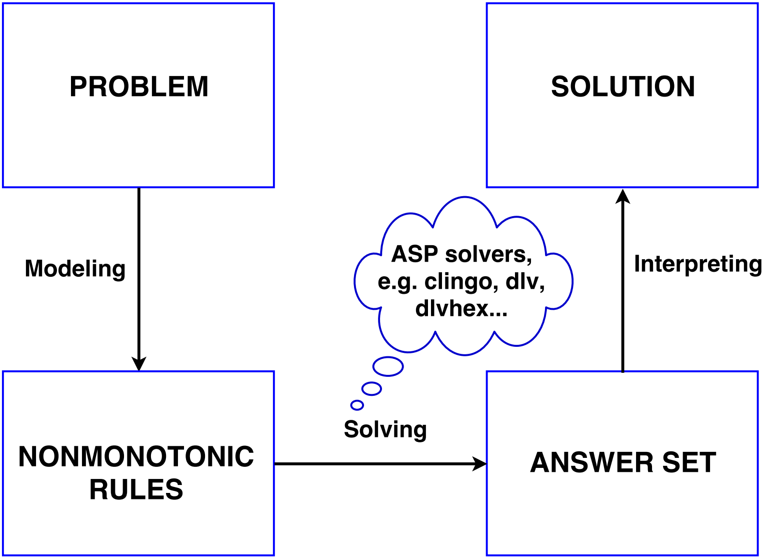
Nonmonotonic rule mining from KGs: OWA is a challenge!



$$conf(r) = \frac{|\blacktriangle|}{|\blacktriangle| + |\triangle|} = 1$$

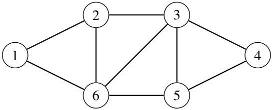
$$r: \; livesIn(X, Z) \leftarrow isMarriedTo(Y, X), livesIn(Y, Z), not \;\; researcher(X)$$

Declarative Programming Paradigm



Declarative Programming Example

Graph 3-colorability



Modeling

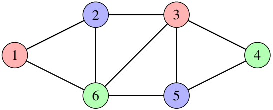
$node(1 \ldots 6)$; $\quad edge(1, 2)$; $\quad \ldots$
$col(V, red) \leftarrow not \;\; col(V, blue), not \;\; col(V, green), node(V)$;
$col(V, green) \leftarrow not \;\; col(V, blue), not \;\; col(V, red), node(V)$;
$col(V, blue) \leftarrow not \;\; col(V, green), not \;\; col(V, red), node(V)$;
$\perp \leftarrow col(V, C), col(V, C'), C \neq C'$;
$\perp \leftarrow col(V, C), col(V', C), edge(V, V')$

NONMONOTONIC RULES

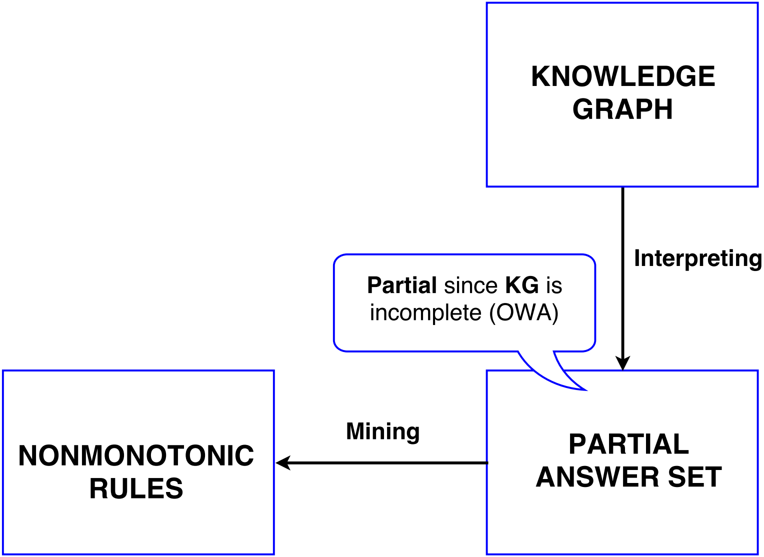
Solving

$node(1 \ldots 6)$; $\quad edge(1, 2); \ldots$
$col(1, red), col(2, blue)$,
$col(3, red), col(4, green)$,
$col(6, green), col(5, blue)$

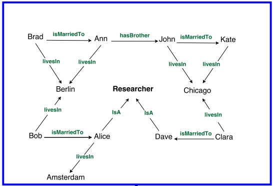
Interpreting



Nonmonotonic Rule Mining



Nonmonotonic Rule Mining



Interpreting

$isMarriedTo(brad, ann);$
$isMarriedTo(john, kate);$
$isMarriedTo(bob, alice);$
$isMarriedTo(clara, dave);$
$livesIn(brad, berlin;$
$\ldots$
$researcher(alice);$
$researcher(dave)$

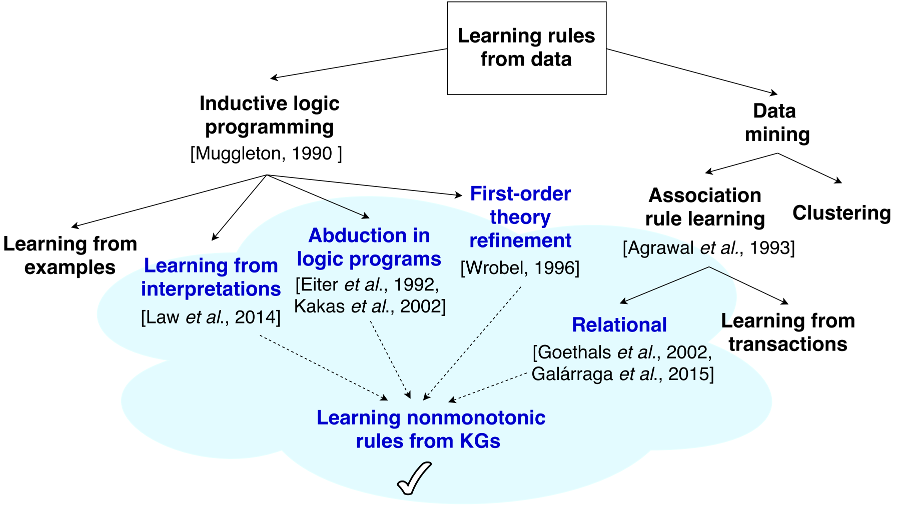
Mining

$$livesIn(Y, Z) \leftarrow isMarried(X, Y), \; livesIn(X, Y), \; not \;\; researcher(Y)$$

Nonmonotonic Rule Mining from KGs

Goal: learn nonmonotonic rules from KG

Approach: revise association rules learned using data mining methods

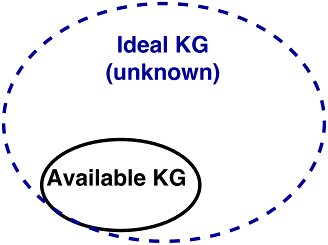


Horn Theory Revision

Quality-based Horn Theory Revision

Given:

- Available KG

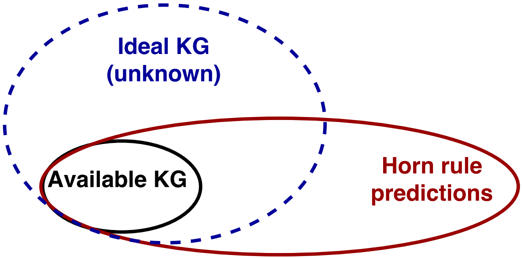


Horn Theory Revision

Quality-based Horn Theory Revision

Given:

- Available KG
- Horn rule set

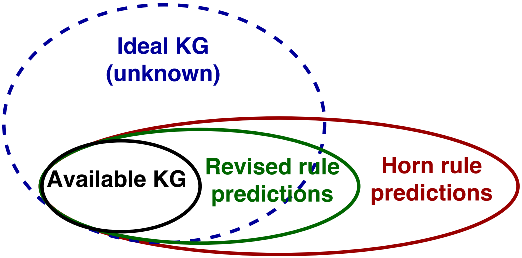


Horn Theory Revision

Quality-based Horn Theory Revision

Given:

- Available KG
- Horn rule set



Find:

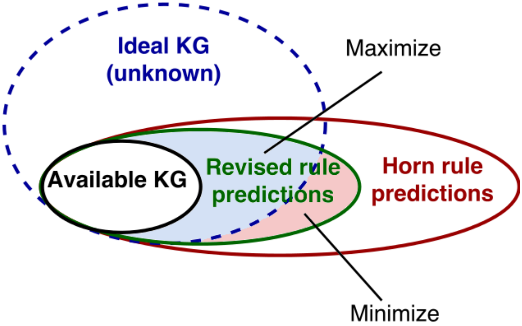
- Nonmonotonic revision of Horn rule set

Horn Theory Revision

Quality-based Horn Theory Revision

Given:

- Available KG
- Horn rule set



Find:

- Nonmonotonic revision of Horn rule set
  with better predictive quality

Avoid Data Overfitting

How to distinguish exceptions from noise?

$r1: \ livesIn(X, Z) \leftarrow isMarriedTo(Y, X), livesIn(Y, Z), not \ \ researcher(X)$

Avoid Data Overfitting

How to distinguish exceptions from noise?

$r1: \ livesIn(X,Z) \leftarrow isMarriedTo(Y,X), livesIn(Y,Z), not \ \ researcher(X)$

$not\_livesIn(X,Z) \leftarrow isMarriedTo(Y,X), livesIn(Y,Z), researcher(X)$

Avoid Data Overfitting

How to distinguish exceptions from noise?

$r1: \; livesIn(X, Z) \leftarrow isMarriedTo(Y, X), livesIn(Y, Z), not \;\; researcher(X)$
$\quad not\_livesIn(X, Z) \leftarrow isMarriedTo(Y, X), livesIn(Y, Z), researcher(X)$

$r2: \; livesIn(X, Z) \leftarrow bornIn(X, Z), not \;\; moved(X)$
$\quad not\_livesIn(X, Z) \leftarrow bornIn(X, Z), moved(X)$

Avoid Data Overfitting

How to distinguish exceptions from noise?

$r1: \ livesIn(X,Z) \leftarrow isMarriedTo(Y,X), livesIn(Y,Z), not\ \ researcher(X)$
$\quad not\_livesIn(X,Z) \leftarrow isMarriedTo(Y,X), livesIn(Y,Z), researcher(X)$

$r2: \ livesIn(X,Z) \leftarrow bornIn(X,Z), not\ \ moved(X)$
$\quad not\_livesIn(X,Z) \leftarrow bornIn(X,Z), moved(X)$

$\{livesIn(c,d), not\_livesIn(c,d)\}$ are conflicting predictions

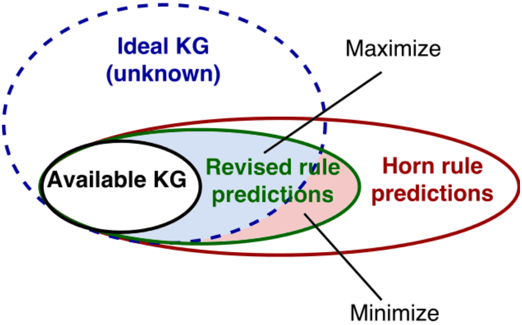
Intuition: Rules with good exceptions should make few conflicting predictions

Horn Theory Revision

Quality-based Horn Theory Revision

Given:

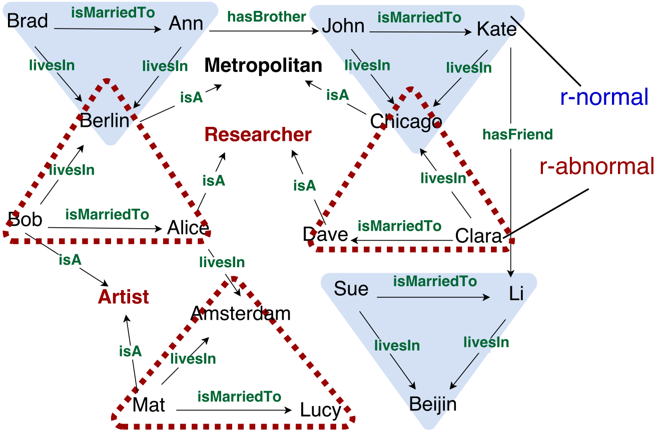
- Available KG
- Horn rule set



Find:

- Nonmonotonic revision of Horn rules, such that
  - number of conflicting predictions is **minimal**
  - average conviction is **maximal**

Exception Candidates



$$r: livesIn(X, Z) \leftarrow isMarriedTo(Y, X),\, livesIn(Y, Z) \quad \begin{Bmatrix} not\ \ researcher(X) \\ not\ \ artist(Y) \end{Bmatrix}$$

Exception Ranking

$$\begin{array}{ll} \textit{rule1} & \{\underline{\mathbf{e_1}}, e_2, e_3, \dots\} \\ \textit{rule2} & \{e_1, \underline{\mathbf{e_2}}, e_3, \dots\} \\ \textit{rule3} & \{\underline{\mathbf{e_1}}, e_2, e_3, \dots\} \end{array}$$

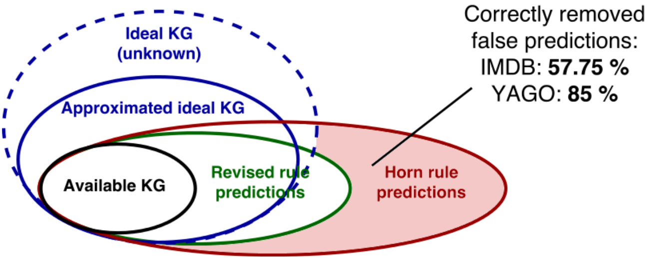
Finding globally best revision is expensive, exponentially many candidates!

- Naive ranking: for every rule inject exception that results in the highest conviction
- Partial materialization (PM): apply all rules apart from a given one, inject exception that results in the highest average conviction of the rule and its rewriting
- Ordered PM (OPM): same as PM plus ordered rules application
- Weighted OPM: same as OPM plus weights on predictions

M. Gad-Elrab, D. Stepanova, J. Urbani, G. Weikum. Exception-enriched Rule Learning from Knowledge Graphs. *ISWC2016*
D. Tran, D. Stepanova, M. Gad-Elrab, F. Lisi, G. Weikum. Towards Nonmonotonic Relational Learning from KGs. *ILP2016*

Experimental Setup

- Approximated ideal KG: original KG
- Available KG: for every relation randomly remove 20% of facts from approximated ideal KG
- Horn rules: $h(X, Y) \leftarrow p(X, Z), q(Z, Y)$
- Exceptions: $e_1(X), e_2(Y), e_3(X, Y)$
- Predictions are computed using answer set solver DLV



Experimental Setup

- Approximated ideal KG: original KG
- Available KG: for every relation randomly remove 20% of facts from approximated ideal KG
- Horn rules: $h(X, Y) \leftarrow p(X, Z), q(Z, Y)$
- Exceptions: $e_1(X), e_2(Y), e_3(X, Y)$
- Predictions are computed using answer set solver DLV

Examples of revised rules:

Plots of films in a sequel are written by the same writer, unless a film is American

$r_1 : \mathit{writtenBy}(X, Z) \leftarrow \mathit{hasPredecessor}(X, Y), \mathit{writtenBy}(Y, Z), \mathit{not}\ \mathit{american\_film}(X)$

Spouses of film directors appear on the cast, unless they are silent film actors

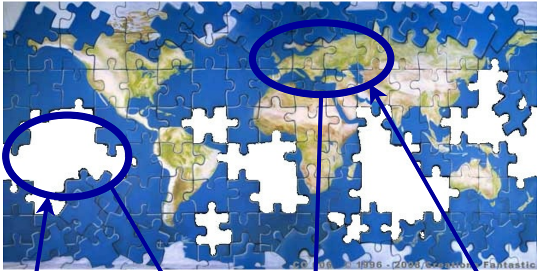
$r_2 : \mathit{actedIn}(X, Z) \leftarrow \mathit{isMarriedTo}(X, Y), \mathit{directed}(Y, Z), \mathit{not}\ \mathit{silent\_film\_actor}(X)$

Overview

- ✓ Motivation
- ✓ Ontologies and Rules
- ✓ Inconsistencies in DL-programs
- ✓ Nonmonotonic Rule Mining
- **Ongoing and Future Work**

Completeness-aware Rule Mining

- Exploit cardinality meta-data [Mirza *et al.*, 2016] in rule mining
  *John has **5** children, Mary is a citizen of **2** countries*



5 missing

do not build here!

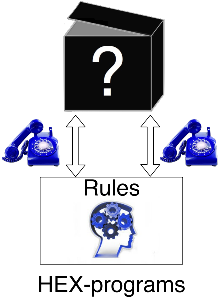
build here!

0 missing

Joint work with T. Pellissier-Tanon, S. Razniewski, P. Mirza, G. Weikum

Ongoing and Future Work

- Make use of logical background knowledge in
  - Rule learning and other data mining tasks[2]
  - Information extraction from text corpora[3]
  - Natural language processing tasks
- Exploit answer set programs with external computations [Eiter *et al.*, 2009] for the above problems



[2] S. Paramonov, D. Stepanova, P. Miettinen. Hybrid Approach to Constraint-based Pattern Mining. Accepted to *RR2017*

[3] Joint work with M. Gad-Elrab, J. Urbani, G. Weikum

Conclusion

Summary:

- Inconsistencies in combination of rules and ontologies
  - Repair semantics and its complexity analysis
  - Optimized algorithms for repair computation and their evaluation (*DL-Lite*$_\mathcal{A}$ and $\mathcal{EL}$ DLs)

- Nonmonotonic rule mining from KGs
  - Quality-based Horn theory revision framework under OWA
  - Approach for computing and ranking exceptions based on cross-talk among rules and its evaluation on real-world KGs

Future Directions:

- Interlinking mining and reasoning in the KG context
- Exploiting logical background knowledge in information extraction and natural language processing tasks

References I

Thomas Eiter, Giovambattista Ianni, Thomas Lukasiewicz, Roman Schindlauer, and Hans Tompits.
Combining answer set programming with description logics for the semantic web.
Artificial Intelligence, 172(12-13):1495–1539, August 2008.

Thomas Eiter, Gerhard Brewka, Minh Dao-Tran, Michael Fink, Giovambattista Ianni, and Thomas Krennwallner.
Combining nonmonotonic knowledge bases with external sources.
In *Frontiers of Combining Systems, 7th International Symposium, FroCoS 2009, Trento, Italy, September 16-18, 2009. Proceedings*, pages 18–42, 2009.

Luis Galárraga, Christina Teflioudi, Katja Hose, and Fabian M. Suchanek.
Fast rule mining in ontological knowledge bases with AMIE+.
VLDB J., 24(6):707–730, 2015.

Michael Gelfond and Vladimir Lifschitz.
The stable model semantics for logic programming.
In *Proceedings of the 5th International Conference and Symposium on Logoc Programming, ICLP 1988*, pages 1070–1080. The MIT Press, 1988.

P. J. Hayes.
The logic of frames.
In *Frame Conceptions and Text Understanding*, pages 46–61. 1979.

John McCarthy.
Programs with common sense.
In *TeddingtonConference*, pages 75–91, 1959.

John McCarthy.
Circumscription - A form of non-monotonic reasoning.
Artif. Intell., 13(1-2):27–39, 1980.

Marvin Minsky.
A framework for representing knowledge.
In *Readings in Knowledge Representation*, pages 245–262. Kaufmann, 1985.

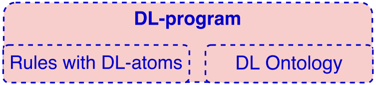
References II

Paramita Mirza, Simon Razniewski, and Werner Nutt.
Expanding wikidata's parenthood information by 178%, or how to mine relation cardinality information.
In *ISWC 2016 Posters & Demos*, 2016.

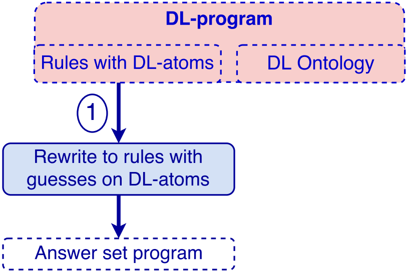
Raymond Reiter.
A logic for default reasoning.
Artif. Intell., 13(1-2):81–132, 1980.

J. Robinson.
A machine-oriented logic based on the resolution principle.
Journal of the ACM, 12(6):23–41, 1965.

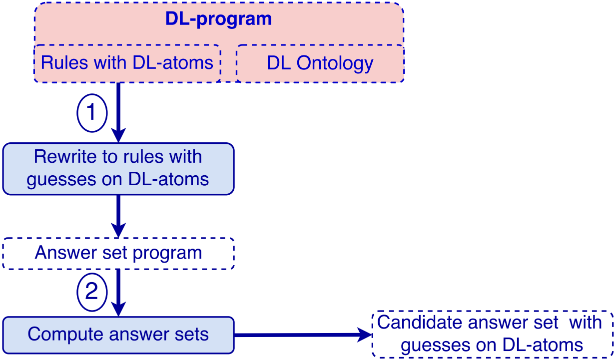
DL-program Repair Algorithm



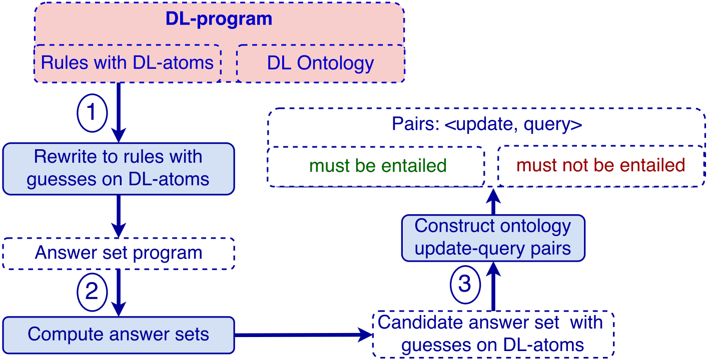
DL-program Repair Algorithm



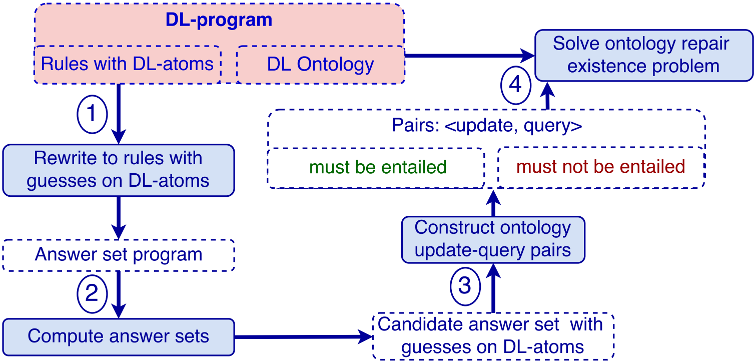
DL-program Repair Algorithm



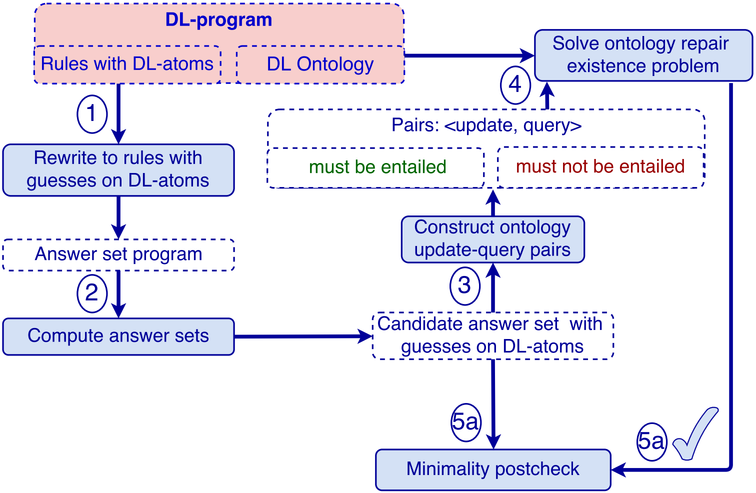
DL-program Repair Algorithm



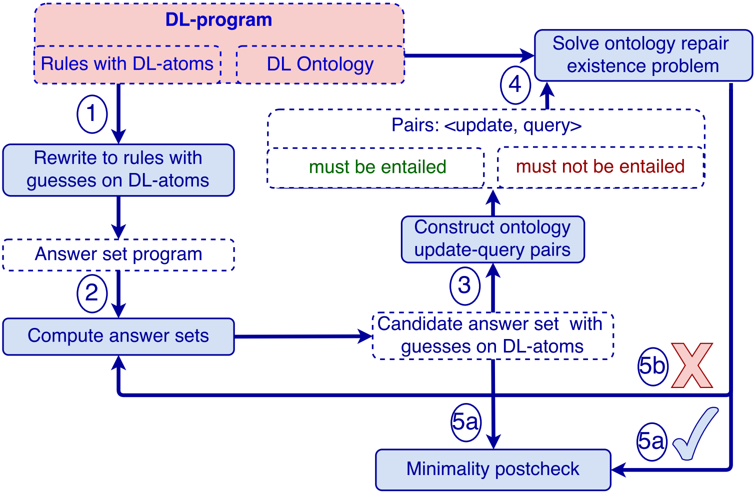
DL-program Repair Algorithm



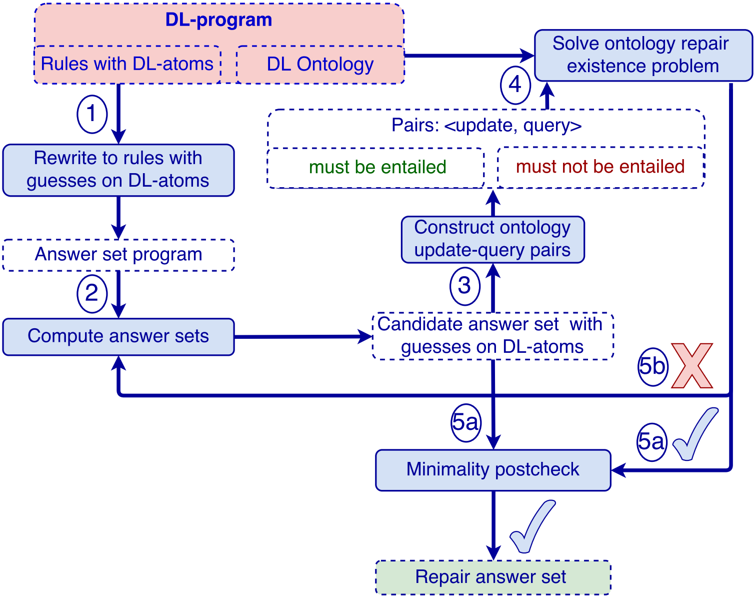
DL-program Repair Algorithm



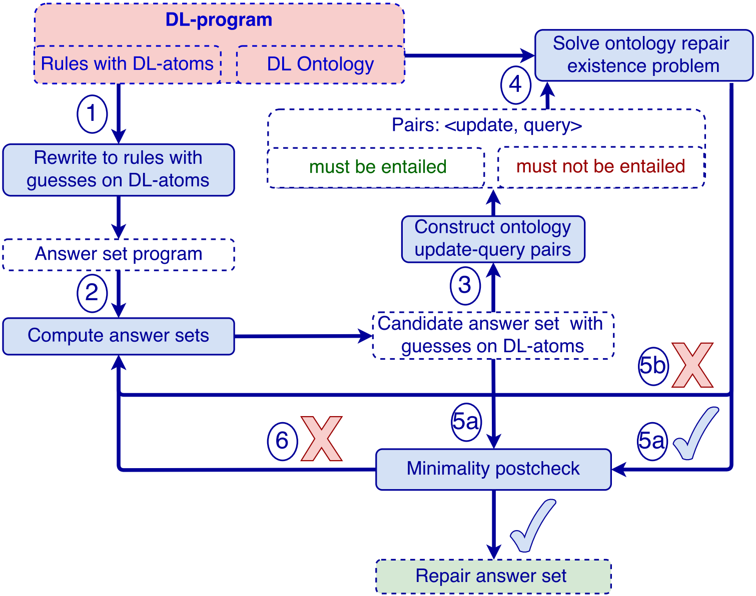
DL-program Repair Algorithm



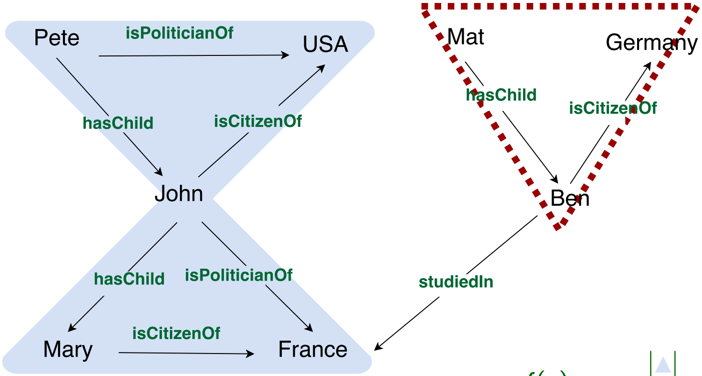
DL-program Repair Algorithm



DL-program Repair Algorithm



Spurious Rules due to Incompleteness

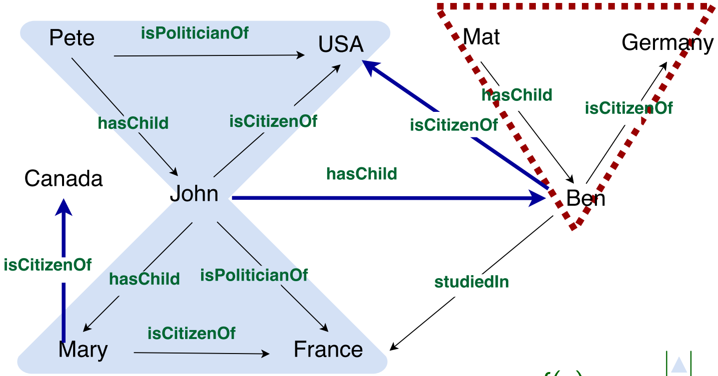


$$conf(r) = \frac{|\blacktriangle|}{|\blacktriangle| + |\triangle|} = \frac{2}{3}$$

$$r: \ isPoliticianOf(X, Z) \leftarrow hasChild(X, Y), isCitizenOf(Y, Z)$$

Spurious Rules due to Incompleteness

In real world:

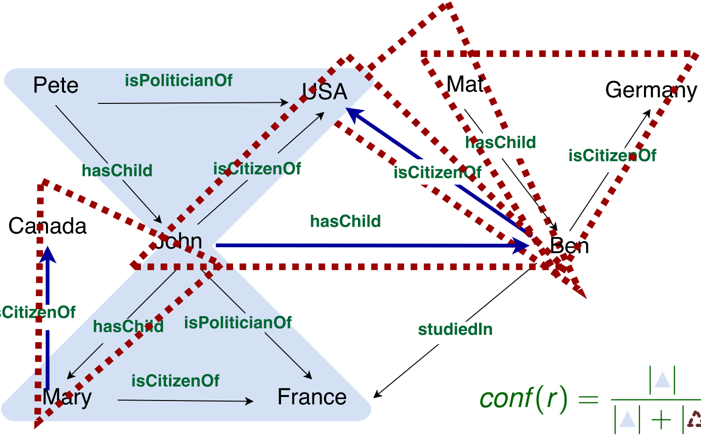


$$\mathit{conf}(r) = \frac{|\blacktriangle|}{|\blacktriangle| + |\triangle|} = \frac{2}{3}$$

$$r: \ \mathit{isPoliticianOf}(X, Z) \leftarrow \mathit{hasChild}(X, Y), \mathit{isCitizenOf}(Y, Z)$$

Spurious Rules due to Incompleteness

In real world:

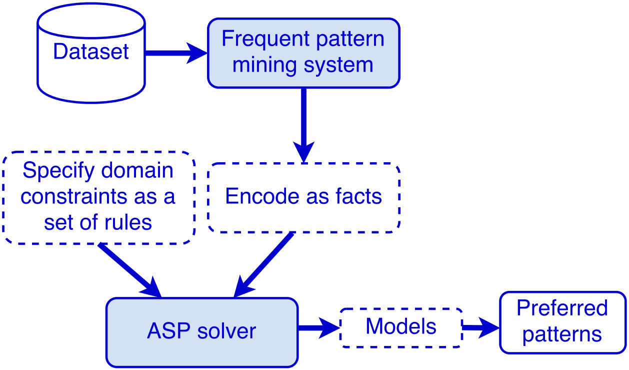


$$conf(r) = \frac{|\triangle|}{|\triangle| + |\triangle|} = \frac{2}{6}$$

$$r : \overbrace{isPoliticianOf(X, Z)}^{complete} \leftarrow \overbrace{hasChild(X, Y), isCitizenOf(Y, Z)}^{incomplete}$$

Hybrid Constraint-based Pattern Mining

- Interlink mining and reasoning
- Use declarative logic programming
  for frequent pattern (itemset/sequence) filtering
- Combine various domain-specific constraints



Sergey Paramonov, Daria Stepanova, Pauli Miettinen. Hybrid Approach to Constraint-based Pattern Mining. *RR2017*

Semantically-enhanced Fact Spotting

KG population problem: some facts are hard to spot in text due to reporting bias $lost(nadal, australianOpen2017)$

Given:

- Fact: $lost(nadal, australianOpen2017)$
- Rule set: $lost(Z, Y) \leftarrow won(X, Y), finalist(Z, Y), X \neq Z$
- KG: $won(federer, australianOpen2017)$
- Text: "... *another **finalist** of **Australian Open in 2017** was **Nadal***"

Find:

- Fact's truth value: $lost(nadal, australianOpen2017)$ is true!

Joint work with Mohamed Gad Elrab, Jacopo Urbani and Gerhard Weikum