Knowledge Representation for the Semantic Web

Lecture 8: Answer Set Programming III

Daria Stepanova

D5: Databases and Information Systems
Max Planck Institute for Informatics

WS 2017/18

DLs vs ASP

What of DLs can be expressed directly in ASP?

- ABox: Factual knowledge about class membership and property values can be translated to ASP “as is”:

DL syntax	ASP syntax
$john \in Person$	$person(john)$
$(john, bob) \in hasChild$	$hasChild(john, bob)$

What of DLs can be expressed directly in ASP?

- ABox: Factual knowledge about class membership and property values can be translated to ASP "as is":

DL syntax	ASP syntax
$john \in Person$	$person(john)$
$(john, bob) \in hasChild$	$hasChild(john, bob)$

- RBox/TBox: A subset of OWL can be straightforwardly translated to ASP, here only a subset is given:

DL syntax	ASP syntax
$r \circ r \sqsubseteq r$ (owl:transitiveProperty)	
$r \equiv r^-$ (owl:symmetricProperty)	
$r \equiv s^-$ (owl:symmetricProperty)	
$C_1 \sqcap \ldots \sqcap C_n \sqsubseteq A$	
$\exists r.C \sqsubseteq A$ (owl:someValuesFrom, lhs)	
$\geq 1\, r \sqsubseteq A$ (owl:minCardinality 1, lhs)	
$A \sqsubseteq \forall r.C$ (owl:allValuesFrom, rhs)	
$A \sqsubseteq C_1 \sqcup \ldots \sqcup C_n$ (owl:unionOf, rhs)	
$C_1 \sqcup \ldots \sqcup C_n \sqsubseteq A$ (owl:unionOf, lhs)	

What of DLs can be expressed directly in ASP?

- ABox: Factual knowledge about class membership and property values can be translated to ASP "as is":

DL syntax	ASP syntax
$john \in Person$	$person(john)$
$(john, bob) \in hasChild$	$hasChild(john, bob)$

- RBox/TBox: A subset of OWL can be straightforwardly translated to ASP, here only a subset is given:

DL syntax	ASP syntax
$r \circ r \sqsubseteq r$ (owl:transitiveProperty)	$r(X, Z) \leftarrow r(X, Y), r(Y, Z).$
$r \equiv r^-$ (owl:symmetricProperty)	
$r \equiv s^-$ (owl:symmetricProperty)	
$C_1 \sqcap \ldots \sqcap C_n \sqsubseteq A$	
$\exists r.C \sqsubseteq A$ (owl:someValuesFrom, lhs)	
$\geq 1\, r \sqsubseteq A$ (owl:minCardinality 1, lhs)	
$A \sqsubseteq \forall r.C$ (owl:allValuesFrom, rhs)	
$A \sqsubseteq C_1 \sqcup \ldots \sqcup C_n$ (owl:unionOf, rhs)	
$C_1 \sqcup \ldots \sqcup C_n \sqsubseteq A$ (owl:unionOf, lhs)	

What of DLs can be expressed directly in ASP?

- ABox: Factual knowledge about class membership and property values can be translated to ASP "as is":

DL syntax	ASP syntax
$john \in Person$	$person(john)$
$(john, bob) \in hasChild$	$hasChild(john, bob)$

- RBox/TBox: A subset of OWL can be straightforwardly translated to ASP, here only a subset is given:

DL syntax	ASP syntax
$r \circ r \sqsubseteq r$ (owl:transitiveProperty)	$r(X, Z) \leftarrow r(X, Y), r(Y, Z).$
$r \equiv r^-$ (owl:symmetricProperty)	$r(Y, X) \leftarrow r(X, Y).$
$r \equiv s^-$ (owl:symmetricProperty)	
$C_1 \sqcap \ldots \sqcap C_n \sqsubseteq A$	
$\exists r.C \sqsubseteq A$ (owl:someValuesFrom, lhs)	
$\geq 1\, r \sqsubseteq A$ (owl:minCardinality 1, lhs)	
$A \sqsubseteq \forall r.C$ (owl:allValuesFrom, rhs)	
$A \sqsubseteq C_1 \sqcup \ldots \sqcup C_n$ (owl:unionOf, rhs)	
$C_1 \sqcup \ldots \sqcup C_n \sqsubseteq A$ (owl:unionOf, lhs)	

What of DLs can be expressed directly in ASP?

- ABox: Factual knowledge about class membership and property values can be translated to ASP "as is":

DL syntax	ASP syntax
$john \in Person$	$person(john)$
$(john, bob) \in hasChild$	$hasChild(john, bob)$

- RBox/TBox: A subset of OWL can be straightforwardly translated to ASP, here only a subset is given:

DL syntax	ASP syntax
$r \circ r \sqsubseteq r$ (owl:transitiveProperty)	$r(X, Z) \leftarrow r(X, Y), r(Y, Z).$
$r \equiv r^-$ (owl:symmetricProperty)	$r(Y, X) \leftarrow r(X, Y).$
$r \equiv s^-$ (owl:symmetricProperty)	$s(Y, X) \leftarrow r(X, Y).$
$C_1 \sqcap \ldots \sqcap C_n \sqsubseteq A$	
$\exists r.C \sqsubseteq A$ (owl:someValuesFrom, lhs)	
$\geq 1\, r \sqsubseteq A$ (owl:minCardinality 1, lhs)	
$A \sqsubseteq \forall r.C$ (owl:allValuesFrom, rhs)	
$A \sqsubseteq C_1 \sqcup \ldots \sqcup C_n$ (owl:unionOf, rhs)	
$C_1 \sqcup \ldots \sqcup C_n \sqsubseteq A$ (owl:unionOf, lhs)	

What of DLs can be expressed directly in ASP?

- ABox: Factual knowledge about class membership and property values can be translated to ASP "as is":

DL syntax	ASP syntax
$john \in Person$	$person(john)$
$(john, bob) \in hasChild$	$hasChild(john, bob)$

- RBox/TBox: A subset of OWL can be straightforwardly translated to ASP, here only a subset is given:

DL syntax	ASP syntax
$r \circ r \sqsubseteq r$ (owl:transitiveProperty)	$r(X, Z) \leftarrow r(X, Y), r(Y, Z).$
$r \equiv r^-$ (owl:symmetricProperty)	$r(Y, X) \leftarrow r(X, Y).$
$r \equiv s^-$ (owl:symmetricProperty)	$s(Y, X) \leftarrow r(X, Y).$
$C_1 \sqcap \ldots \sqcap C_n \sqsubseteq A$	$a(X) \leftarrow c_1(X), \ldots, c_n(X).$
$\exists r.C \sqsubseteq A$ (owl:someValuesFrom, lhs)	
$\geq 1\, r \sqsubseteq A$ (owl:minCardinality 1, lhs)	
$A \sqsubseteq \forall r.C$ (owl:allValuesFrom, rhs)	
$A \sqsubseteq C_1 \sqcup \ldots \sqcup C_n$ (owl:unionOf, rhs)	
$C_1 \sqcup \ldots \sqcup C_n \sqsubseteq A$ (owl:unionOf, lhs)	

What of DLs can be expressed directly in ASP?

- ABox: Factual knowledge about class membership and property values can be translated to ASP "as is":

DL syntax	ASP syntax
$john \in Person$	$person(john)$
$(john, bob) \in hasChild$	$hasChild(john, bob)$

- RBox/TBox: A subset of OWL can be straightforwardly translated to ASP, here only a subset is given:

DL syntax	ASP syntax
$r \circ r \sqsubseteq r$ (owl:transitiveProperty)	$r(X,Z) \leftarrow r(X,Y), r(Y,Z).$
$r \equiv r^-$ (owl:symmetricProperty)	$r(Y,X) \leftarrow r(X,Y).$
$r \equiv s^-$ (owl:symmetricProperty)	$s(Y,X) \leftarrow r(X,Y).$
$C_1 \sqcap \ldots \sqcap C_n \sqsubseteq A$	$a(X) \leftarrow c_1(X), \ldots, c_n(X).$
$\exists r.C \sqsubseteq A$ (owl:someValuesFrom, lhs)	$a(X) \leftarrow r(X,Y), c(Y).$
$\geq 1\, r \sqsubseteq A$ (owl:minCardinality 1, lhs)	
$A \sqsubseteq \forall r.C$ (owl:allValuesFrom, rhs)	
$A \sqsubseteq C_1 \sqcup \ldots \sqcup C_n$ (owl:unionOf, rhs)	
$C_1 \sqcup \ldots \sqcup C_n \sqsubseteq A$ (owl:unionOf, lhs)	

What of DLs can be expressed directly in ASP?

- ABox: Factual knowledge about class membership and property values can be translated to ASP "as is":

DL syntax	ASP syntax
$john \in Person$	$person(john)$
$(john, bob) \in hasChild$	$hasChild(john, bob)$

- RBox/TBox: A subset of OWL can be straightforwardly translated to ASP, here only a subset is given:

DL syntax	ASP syntax
$r \circ r \sqsubseteq r$ (owl:transitiveProperty)	$r(X,Z) \leftarrow r(X,Y), r(Y,Z).$
$r \equiv r^-$ (owl:symmetricProperty)	$r(Y,X) \leftarrow r(X,Y).$
$r \equiv s^-$ (owl:symmetricProperty)	$s(Y,X) \leftarrow r(X,Y).$
$C_1 \sqcap \ldots \sqcap C_n \sqsubseteq A$	$a(X) \leftarrow c_1(X), \ldots, c_n(X).$
$\exists r.C \sqsubseteq A$ (owl:someValuesFrom, lhs)	$a(X) \leftarrow r(X,Y), c(Y).$
$\geq 1\, r \sqsubseteq A$ (owl:minCardinality 1, lhs)	$a(X) \leftarrow r(X,Y).$
$A \sqsubseteq \forall r.C$ (owl:allValuesFrom, rhs)	
$A \sqsubseteq C_1 \sqcup \ldots \sqcup C_n$ (owl:unionOf, rhs)	
$C_1 \sqcup \ldots \sqcup C_n \sqsubseteq A$ (owl:unionOf, lhs)	

What of DLs can be expressed directly in ASP?

- ABox: Factual knowledge about class membership and property values can be translated to ASP "as is":

DL syntax	ASP syntax
$john \in Person$	$person(john)$
$(john, bob) \in hasChild$	$hasChild(john, bob)$

- RBox/TBox: A subset of OWL can be straightforwardly translated to ASP, here only a subset is given:

DL syntax	ASP syntax
$r \circ r \sqsubseteq r$ (owl:transitiveProperty)	$r(X,Z) \leftarrow r(X,Y), r(Y,Z).$
$r \equiv r^-$ (owl:symmetricProperty)	$r(Y,X) \leftarrow r(X,Y).$
$r \equiv s^-$ (owl:symmetricProperty)	$s(Y,X) \leftarrow r(X,Y).$
$C_1 \sqcap \ldots \sqcap C_n \sqsubseteq A$	$a(X) \leftarrow c_1(X), \ldots, c_n(X).$
$\exists r.C \sqsubseteq A$ (owl:someValuesFrom, lhs)	$a(X) \leftarrow r(X,Y), c(Y).$
$\geq 1\, r \sqsubseteq A$ (owl:minCardinality 1, lhs)	$a(X) \leftarrow r(X,Y).$
$A \sqsubseteq \forall r.C$ (owl:allValuesFrom, rhs)	$c(Y) \leftarrow r(X,Y), a(X).$
$A \sqsubseteq C_1 \sqcup \ldots \sqcup C_n$ (owl:unionOf, rhs)	
$C_1 \sqcup \ldots \sqcup C_n \sqsubseteq A$ (owl:unionOf, lhs)	

What of DLs can be expressed directly in ASP?

- ABox: Factual knowledge about class membership and property values can be translated to ASP "as is":

DL syntax	ASP syntax
$john \in Person$	$person(john)$
$(john, bob) \in hasChild$	$hasChild(john, bob)$

- RBox/TBox: A subset of OWL can be straightforwardly translated to ASP, here only a subset is given:

DL syntax	ASP syntax
$r \circ r \sqsubseteq r$ (owl:transitiveProperty)	$r(X,Z) \leftarrow r(X,Y), r(Y,Z).$
$r \equiv r^-$ (owl:symmetricProperty)	$r(Y,X) \leftarrow r(X,Y).$
$r \equiv s^-$ (owl:symmetricProperty)	$s(Y,X) \leftarrow r(X,Y).$
$C_1 \sqcap \ldots \sqcap C_n \sqsubseteq A$	$a(X) \leftarrow c_1(X), \ldots, c_n(X).$
$\exists r.C \sqsubseteq A$ (owl:someValuesFrom, lhs)	$a(X) \leftarrow r(X,Y), c(Y).$
$\geq 1\, r \sqsubseteq A$ (owl:minCardinality 1, lhs)	$a(X) \leftarrow r(X,Y).$
$A \sqsubseteq \forall r.C$ (owl:allValuesFrom, rhs)	$c(Y) \leftarrow r(X,Y), a(X).$
$A \sqsubseteq C_1 \sqcup \ldots \sqcup C_n$ (owl:unionOf, rhs)	$c_1(X) \vee \ldots \vee c_n(X) \leftarrow a(X).$
$C_1 \sqcup \ldots \sqcup C_n \sqsubseteq A$ (owl:unionOf, lhs)	

What of DLs can be expressed directly in ASP?

- ABox: Factual knowledge about class membership and property values can be translated to ASP "as is":

DL syntax	ASP syntax
$john \in Person$	$person(john)$
$(john, bob) \in hasChild$	$hasChild(john, bob)$

- RBox/TBox: A subset of OWL can be straightforwardly translated to ASP, here only a subset is given:

DL syntax	ASP syntax
$r \circ r \sqsubseteq r$ (owl:transitiveProperty)	$r(X,Z) \leftarrow r(X,Y), r(Y,Z).$
$r \equiv r^-$ (owl:symmetricProperty)	$r(Y,X) \leftarrow r(X,Y).$
$r \equiv s^-$ (owl:symmetricProperty)	$s(Y,X) \leftarrow r(X,Y).$
$C_1 \sqcap \ldots \sqcap C_n \sqsubseteq A$	$a(X) \leftarrow c_1(X), \ldots, c_n(X).$
$\exists r.C \sqsubseteq A$ (owl:someValuesFrom, lhs)	$a(X) \leftarrow r(X,Y), c(Y).$
$\geq 1\, r \sqsubseteq A$ (owl:minCardinality 1, lhs)	$a(X) \leftarrow r(X,Y).$
$A \sqsubseteq \forall r.C$ (owl:allValuesFrom, rhs)	$c(Y) \leftarrow r(X,Y), a(X).$
$A \sqsubseteq C_1 \sqcup \ldots \sqcup C_n$ (owl:unionOf, rhs)	$c_1(X) \vee \ldots \vee c_n(X) \leftarrow a(X).$
$C_1 \sqcup \ldots \sqcup C_n \sqsubseteq A$ (owl:unionOf, lhs)	$a(X) \leftarrow C_1(X). \quad \ldots \quad A(X) \leftarrow C_n(X).$

What of DLs cannot be directly expressed in ASP?

$A \equiv \{o_1, \ldots, o_n\}$ (owlOneOf)	Cannot be directly translated
$A \sqsubseteq \exists r.C$	Impossible to express as there, no existentials in the heads
$\forall r.C \sqsubseteq A$	One might guess: $a(X) \leftarrow not\ no\_rc(X).$ $no\_rc(X) \leftarrow r(X,Y), -c(Y).$ but does not work.. **Exercise: Why?**

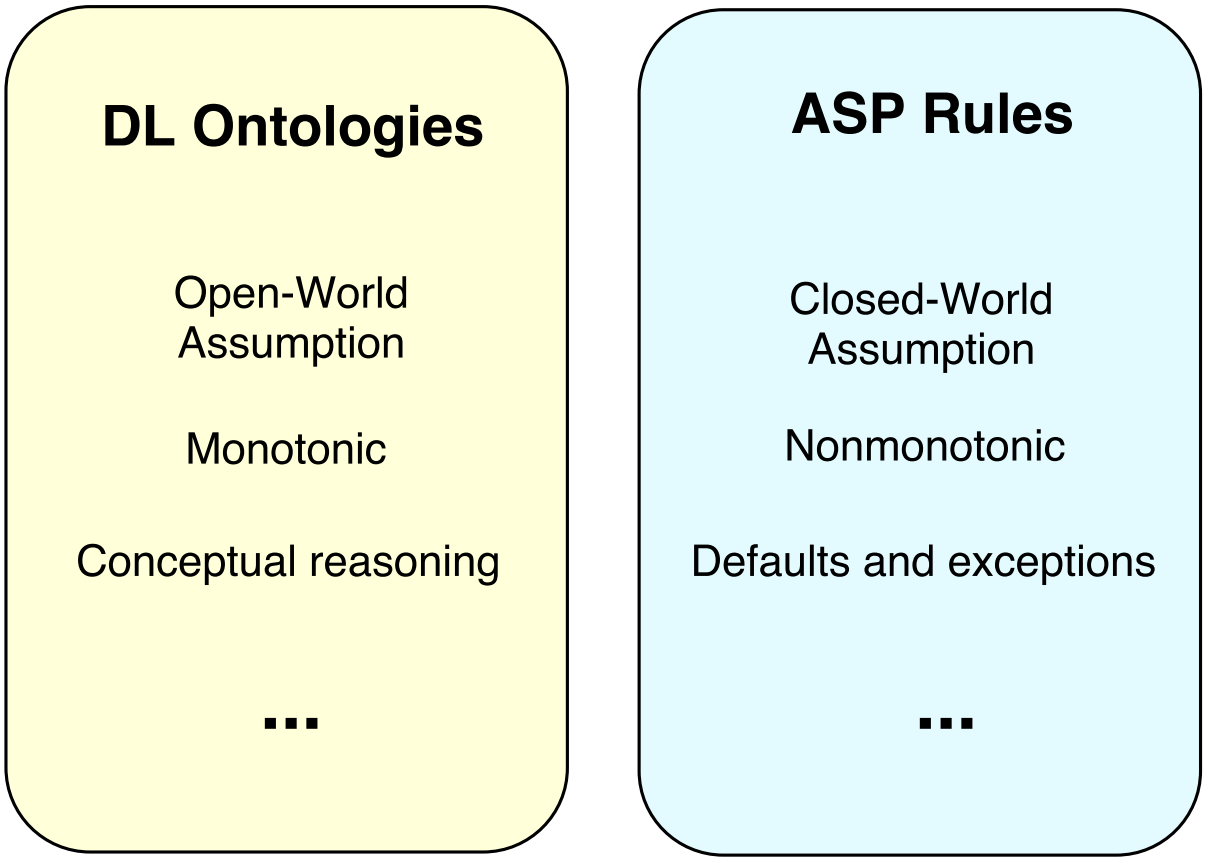
Main difference between DLs and ASP

- $\neg$ in DLs is different from *not* in LP
  - $\neg$: classical negation, monotonicity, open world assumption
  - *not* : default negation, nonmonotonicity, closed world assumption

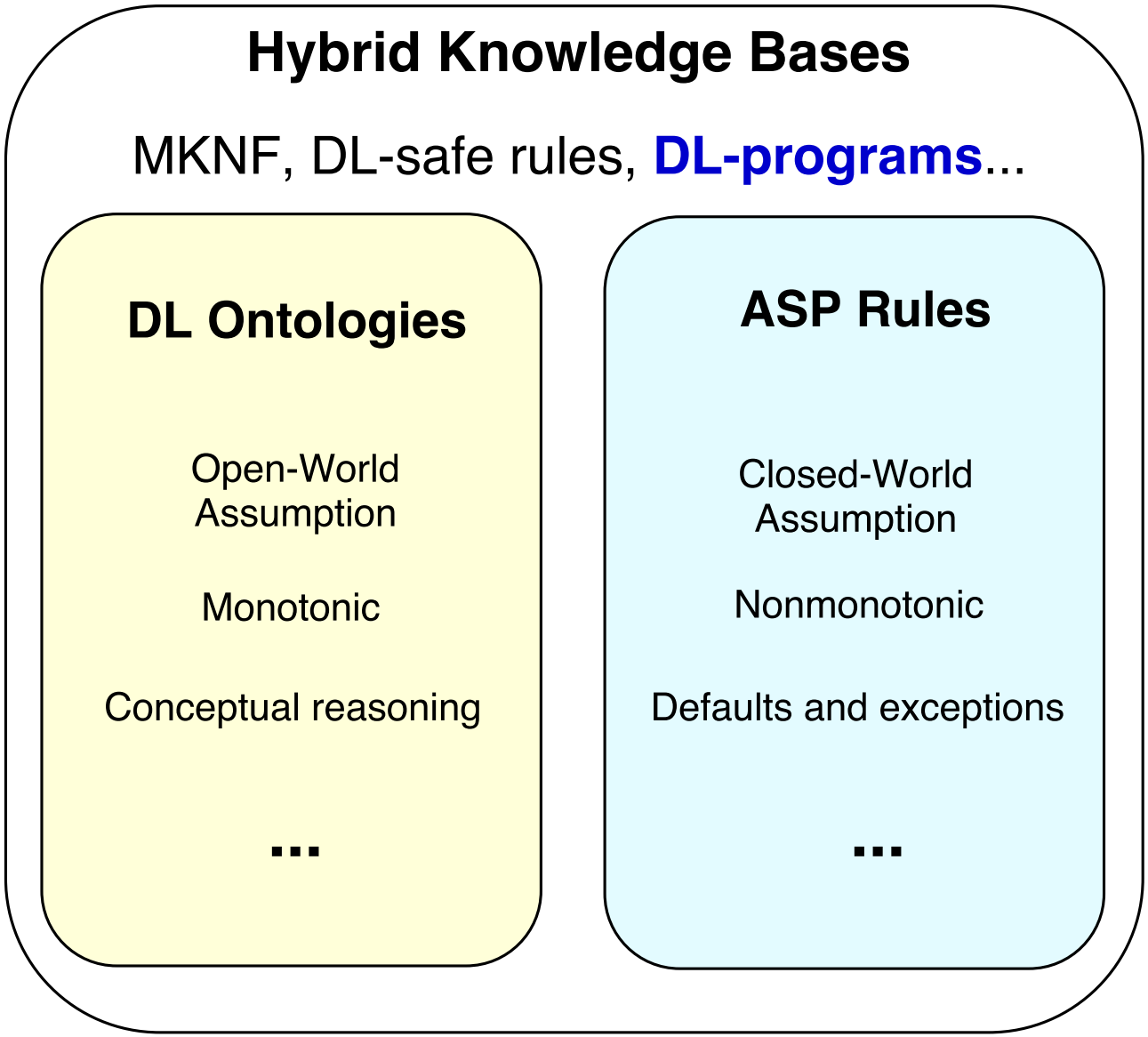
DL ontology $\mathcal{K}$	ASP Program P
$Child \sqsubseteq Person$	$person(X) \leftarrow child(X)$
$\neg Child \sqsubseteq Adult$	$adult(X) \leftarrow not\ child(X)$
$Person(john)$	$person(john)$
$\mathcal{K} \not\models Adult(john)$	P infers $adult(john)$

- DLs are strong in subsumption checking, LPs in expressing relations
- DLs allow complex expressions in heads (rhs of $\sqsubseteq$), while in LPs use of variables in rule bodies is more flexible
- . . .

DLs vs ASP


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

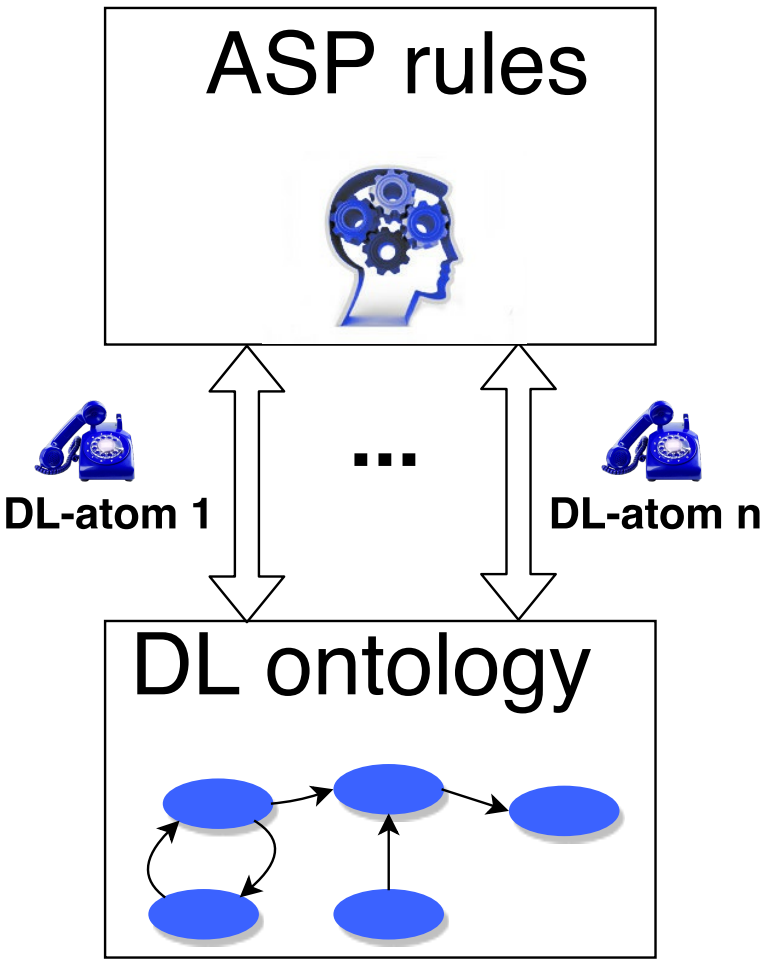
DLs vs ASP


Hybrid Knowledge Bases
MKNF, DL-safe rules, DL-programs...
DL Ontologies
Open-World Assumption
Monotonic
Conceptual reasoning
...
ASP Rules
Closed-World Assumption
Nonmonotonic
Defaults and exceptions
...

DL-programs

DL-programs

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



DL-program

DL ontology

TBox

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

ABox (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

TBox

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

ABox (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

TBox

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

ABox (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

TBox

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

ABox (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

TBox

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

ABox (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

TBox

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

ABox (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

TBox

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

ABox (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

TBox

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

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

Example: Semantic Route Planning

- Personalized semantic route planning
- Requirements:
  - Find shortest trip visiting predefined locations
  - If the shortest trip is beyond a certain length, add lunch location satisfying user's preferences
- DL ontology: restaurant classification
- ASP rules: optimal path with user constraints

Other ASP Extensions

Extensions of ASP

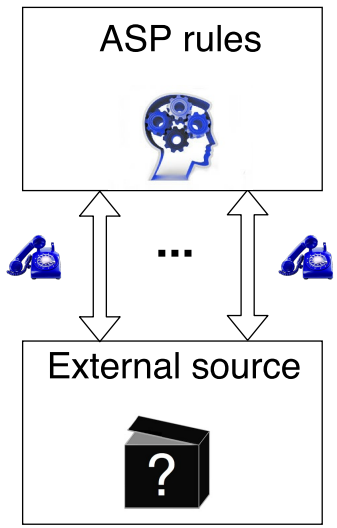
Language extensions like aggregates, complex formula syntax are within same semantic / computational framework

Need:

- interoperability with other logics
- interfacing with programming languages, e.g. *C++*, Python
- access to general *external* sources of information, e.g. WordNet

Approaches:

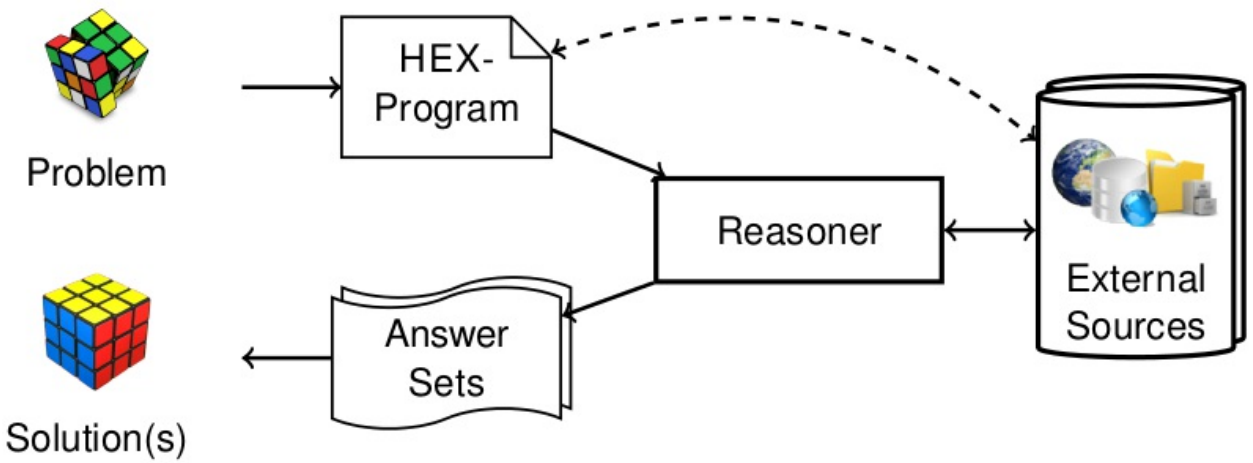
- ASP + concrete theories, e.g.,
  - ASP + DL ontologies (DL-programs[1])
  - constraint ASP
- ASP + abstract theories, e.g.,
  - HEX-programs[2]



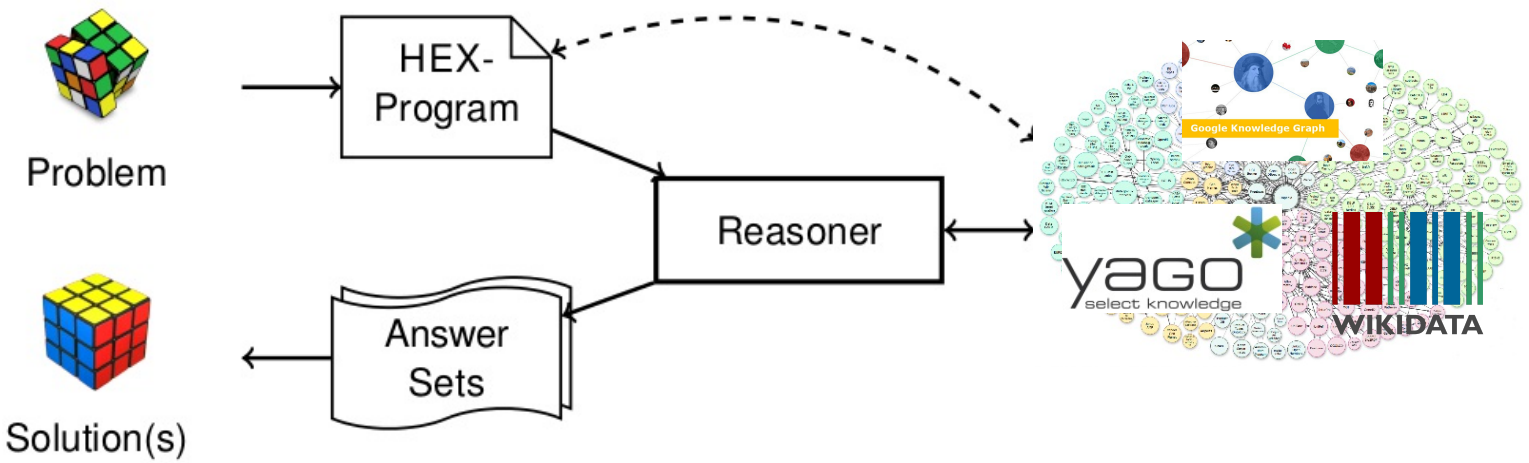
[1] https://github.com/hexhex/dlliteplugin

[2] http://www.kr.tuwien.ac.at/research/systems/dlvhex/

External Information Access



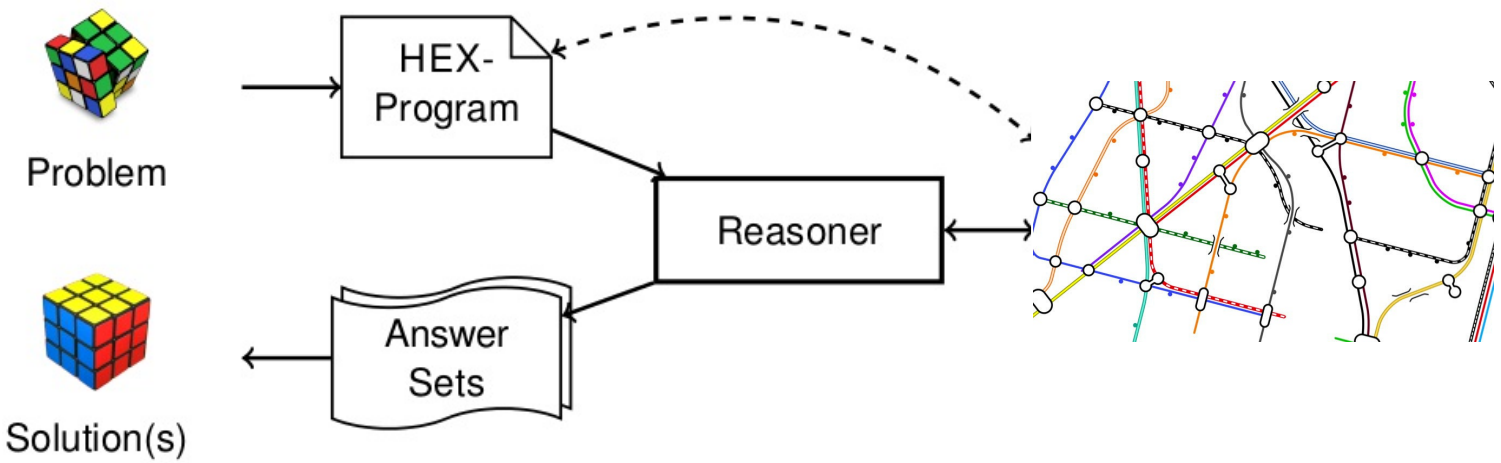
External Information Access



Examples:

- import external knowledge graph triples into the program
  $triple(S, P, O) \leftarrow \&rdf[\text{"http://}\langle Nick \rangle\text{.livejournal.com/data/foaf"}](S, P, O).$

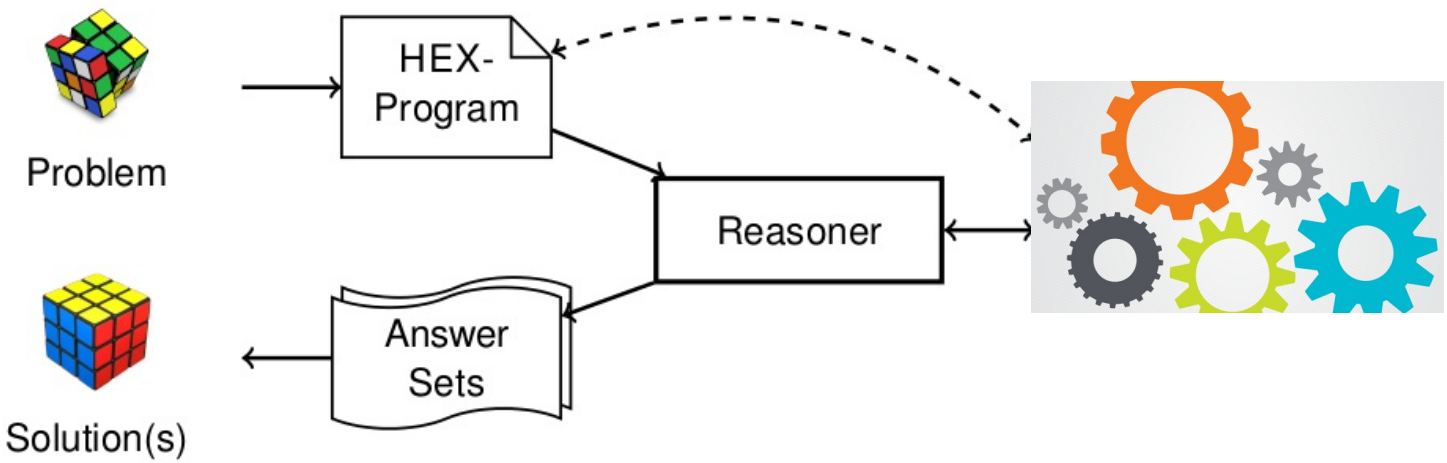
External Information Access



Examples:

- import external knowledge graph triples into the program
  $triple(S, P, O) \leftarrow \&rdf[\text{"http://}\langle Nick \rangle\text{.livejournal.com/data/foaf"}](S, P, O).$
- access external graph
  $reachable(X) \leftarrow \&reachable[conn, a](X).$

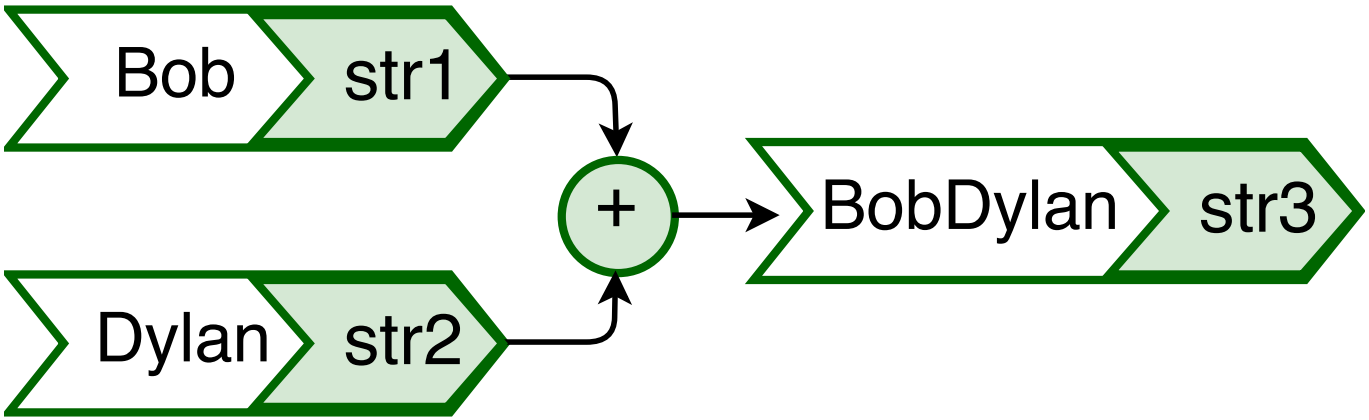
External Information Access



Examples:

- import external knowledge graph triples into the program
  $triple(S, P, O) \leftarrow \&rdf[\text{"http://}\langle Nick \rangle\text{.livejournal.com/data/foaf"}](S, P, O).$
- access external graph
  $reachable(X) \leftarrow \&reachable[conn, a](X).$
- perform auxiliary / data structure computations
  $fullname(Z) \leftarrow \&concat[X, Y](Z), firstname(X), lastname(Y).$

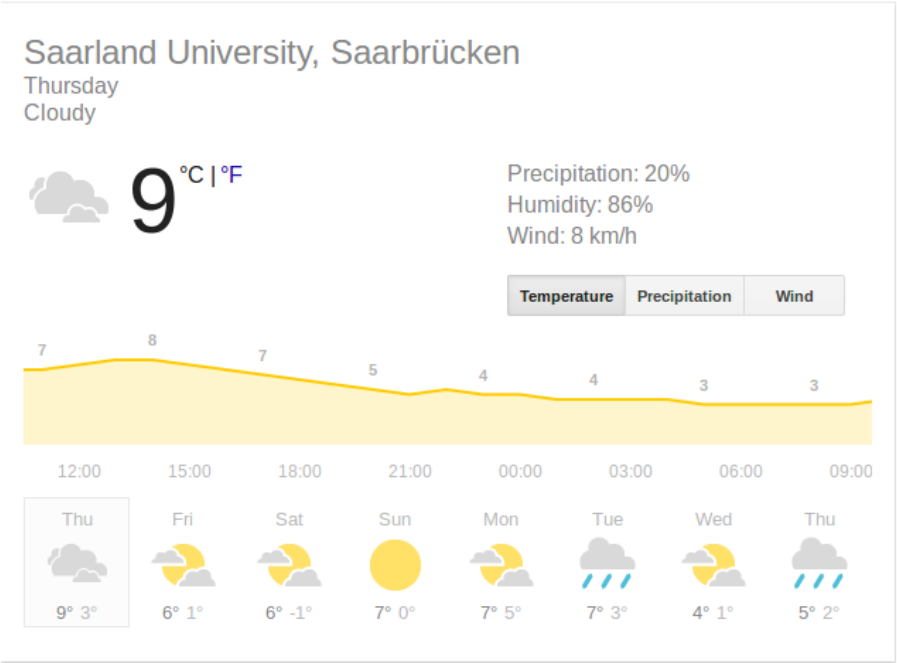
Examples: External Atoms



Concatenate two strings:

- *&concat*$[X, Y](Z)$: intuitively, concatenate two strings
  - *&concat*$[bob, dylan](bobdylan)$ is true
  - *&concat*$[bob, dylan](Z)$ is true for $Z = bobdylan$
  - *&concat*$[bob, Y](bobdylan)$ is true for $Y = dylan$

Examples: External Atoms (cont'd)



Query a web-based weather report:

- *&weatherreport*[*dateLocationPredicate*](*WeatherConditions*)
  - input *dateLocationPredicate* is a binary predicate with tuples (d, l) of dates d and locations l (facts *dateLocationPredicate*(d, l))
  - output *WeatherConditions* are (one by one) all weather conditions that occur at some input date & location

Example: City Trip

Plan to visit Paris and London, under the condition the weather isn't bad:

% Define bad weather conditions

(1) $badweather(rain)$. (2) $badweather(snow)$.

% Decide where to go on which day

(3) $goto(1, paris) \vee goto(1, london)$.

(4) $goto(2, paris) \vee goto(2, london)$.

% Rule out invalid trips

(5) $\leftarrow$ $\&weatherreport[goto](W), badweather(W)$.

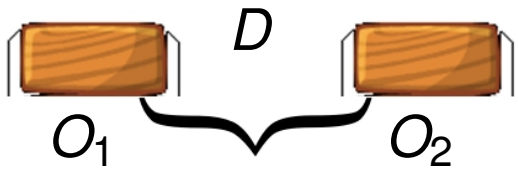
Example: AI Agent for Angrybirds Game

- AngryBirds is a game, whose goal is to kill pigs with birds
- AI competition to automate playing (`angrybirds.org`)
- **Approach:** design an agent based on declarative logic programming[a]
- **Challenge:** plan optimal shots under consideration of some physics
- **Means:** HEX-programs

[a] Joint project with TU Wien and University of Calabria

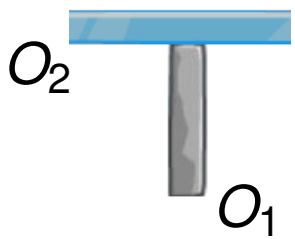
External Atoms Examples

- HEX-program Π for shot computation
- Examples of external atoms:
  - $\&distance[O_1, O_2](D)$ is true iff distance between O_1 and O_2 is D
  - $\&canpush[ngobject](O_1, O_2)$ is true iff O_1 can push O_2 given additional info in the extention of *ngobject*



- $Rule_1$ estimates the likelihood that object O_2 falls when O_1 is hit

$$Rule_1: \quad pushDamage(O_2, P_1, P) \leftarrow pushDamage(O_1, \_, P_1), P_1 > 0$$
$$\&canpush[ngobject](O_1, O_2),$$
$$pushability(O_2, P_2), P = P_1 * P_2/100.$$



Architecture of Angry-HEX

- We use the provided framework (browser plugin, vision module, . . .)
- Agent builds on Tactics and Strategy, both are realized declaratively

- **Tactics:** reasoning about the next shot is done in a HEX-program Π
  - **Input**: scene info from the vision module (facts of Π)
  - **Output**: desired target (models of Π)

- **Strategy:** next level to played is computed in an ASP program Π'
  - **Input:** info about the number of times levels were played, best scores achieved, scores of our agent (facts of Π')
  - **Output:** next optimal level to be played (models of Π')

HEX **Encoding for Tactics**

- **Physics simulation results** are accessed via external atoms:
  - decide which O' intersect with trajectory of a bird after hitting O
  - decide whether O_1 falls whenever O_2 falls ...

- **Tactics in details:**
  - Consider each shootable target
  - Compute the estimated damage on each non-target object
  - Rank the targets (=Answer Sets) using weak constraints
  - Consider history: never play a level in the same way again!

ASP Encoding for Strategy

- **Decides which level to play next** based on info about:
  - number of times each level was played
  - best scores
  - our agent's scores . . .

- **Strategy in details:**
  - First play each level once
  - Second play levels in which our score maximally differs from the best one
  - Third play levels in which we played best and the difference to the second best score is minimal

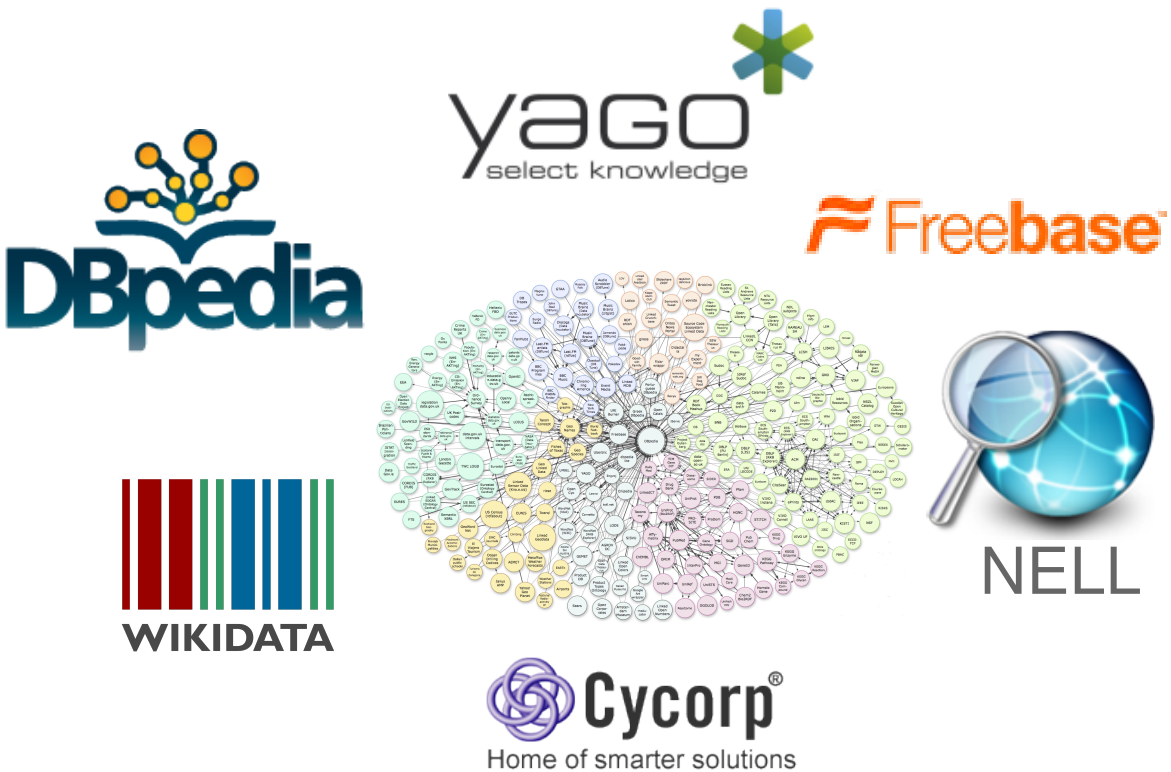
Rule Learning

Motivation

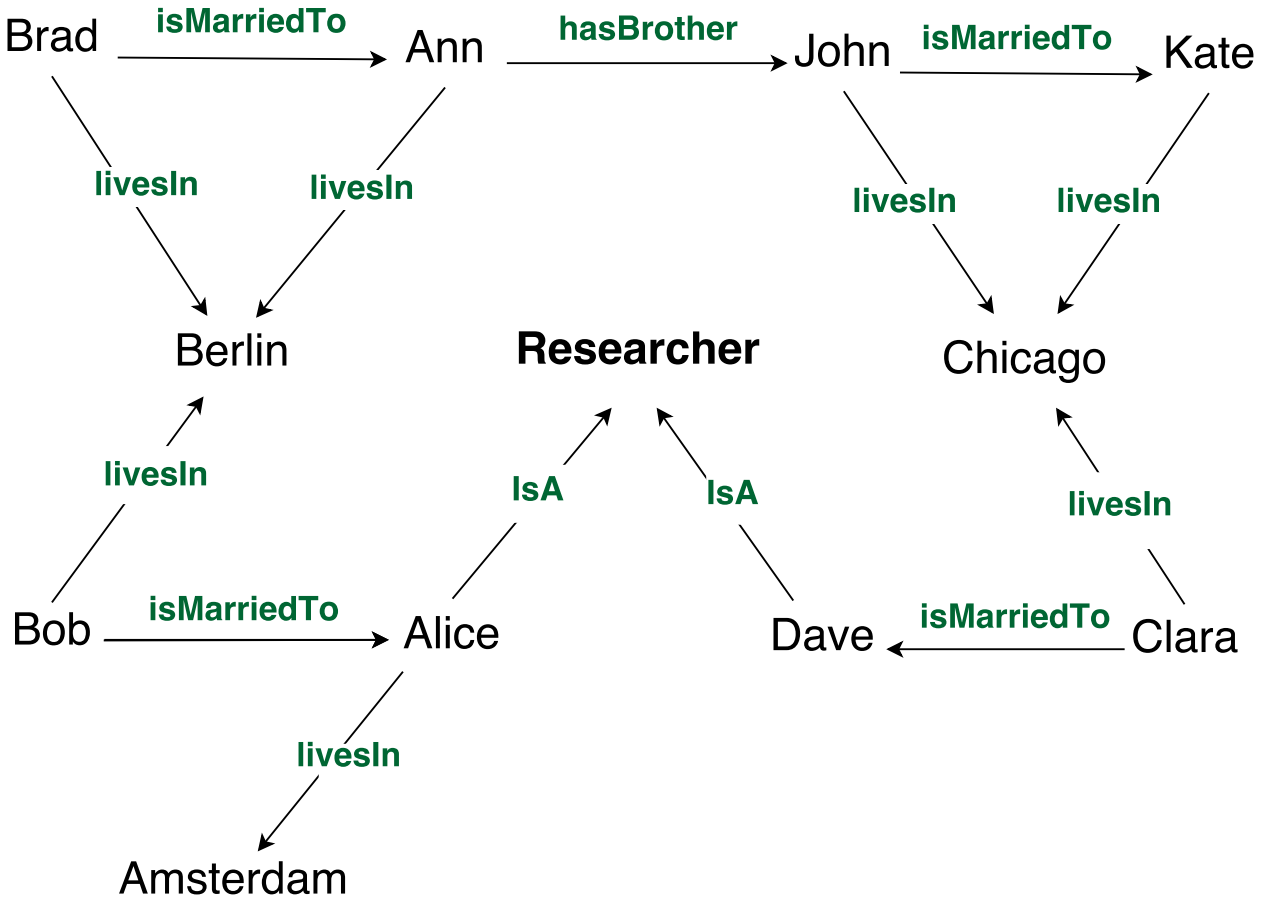
ASP programs are usually constructed by domain experts

Issue: requires a lot of manual efforts!

Question: Can we learn rules from data, e.g., from knowledge graphs?

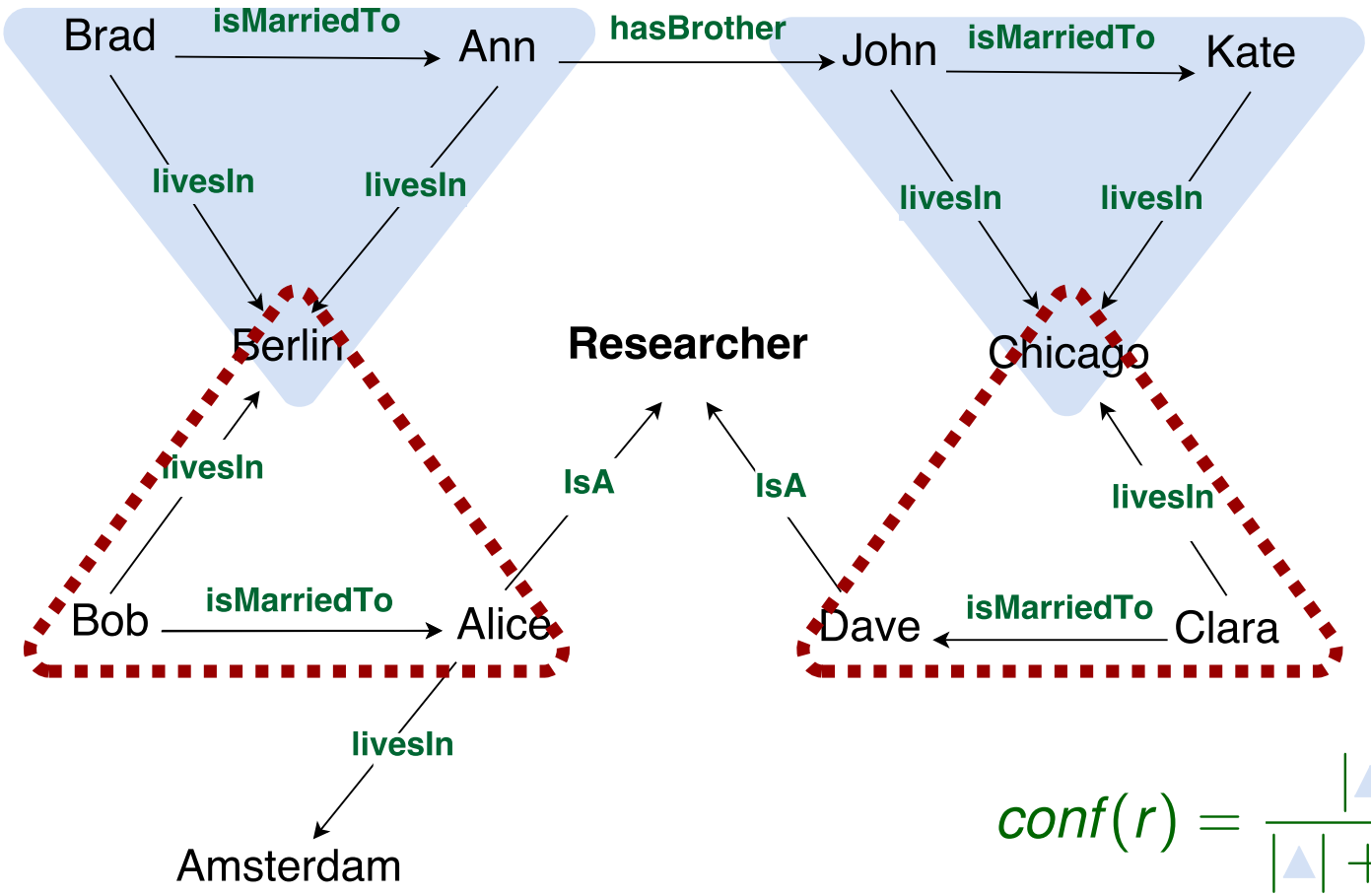


Horn Rule Mining



Horn Rule Mining

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

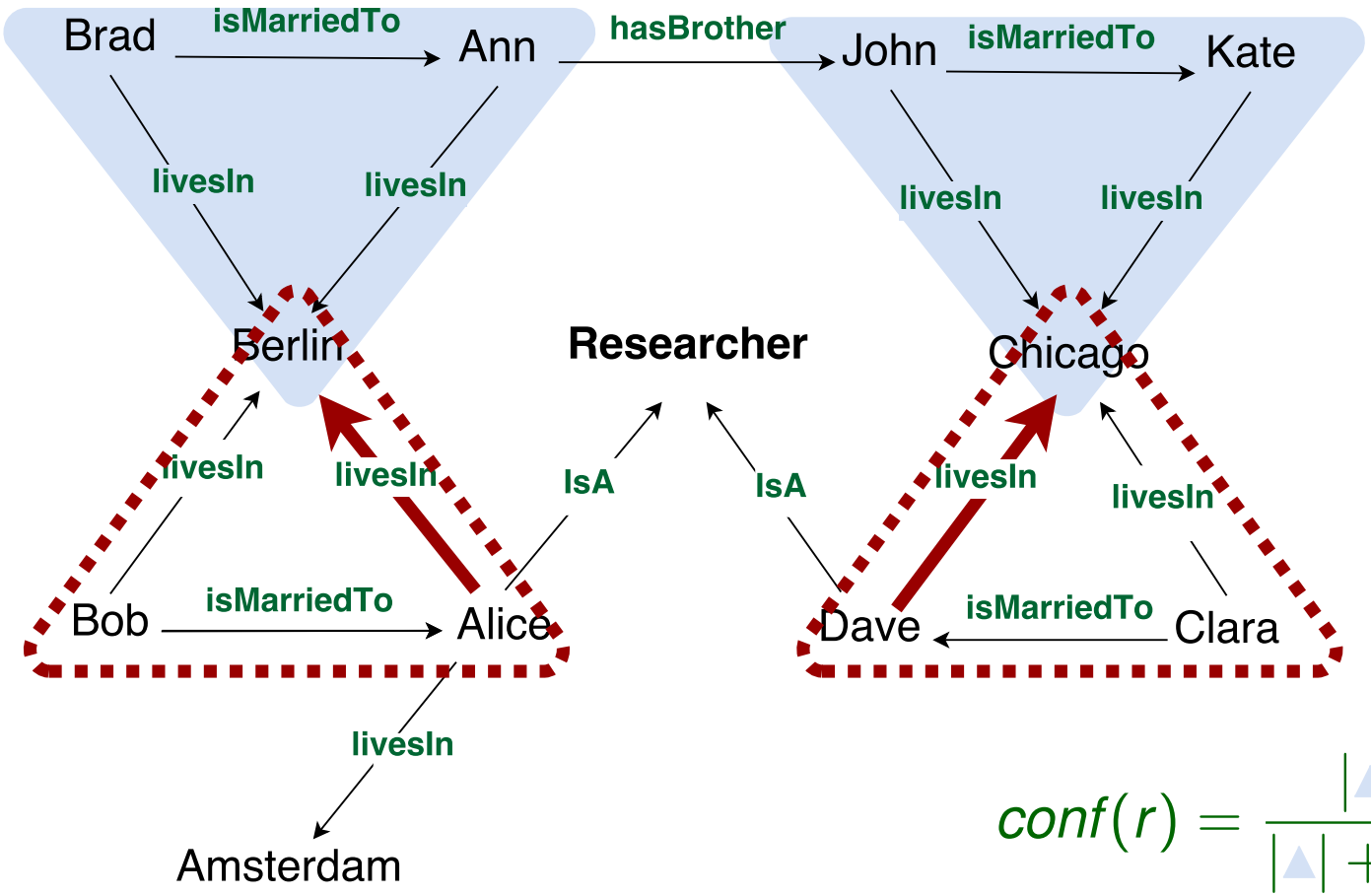


$$conf(r) = \frac{|\blacktriangle|}{|\blacktriangle| + |\text{♻}|} = \frac{2}{4}$$

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

Horn Rule Mining

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

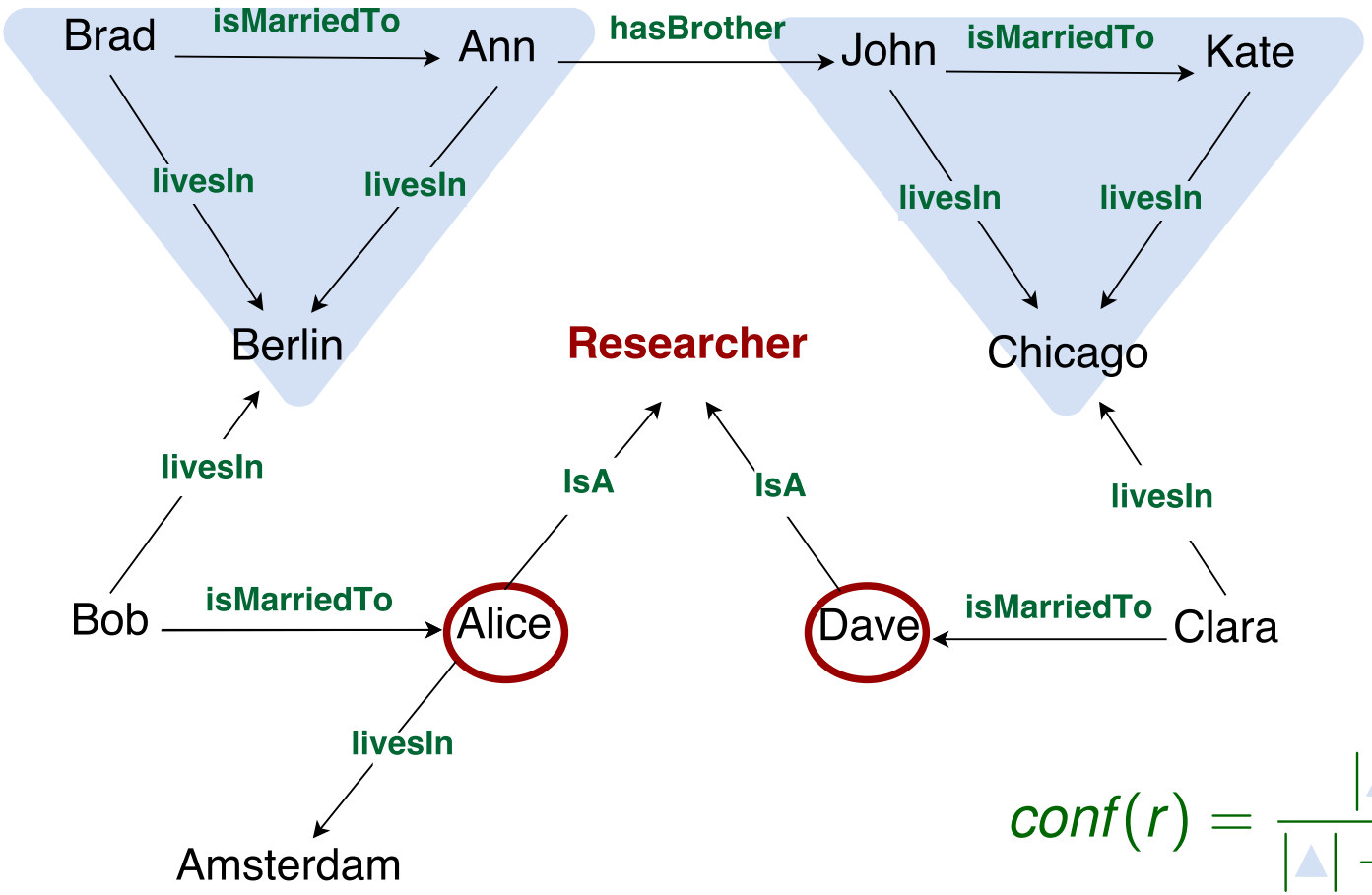


$$conf(r) = \frac{|\blacktriangle|}{|\blacktriangle| + |\text{♻}|} = \frac{2}{4}$$

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

Nonmonotonic Rule Mining

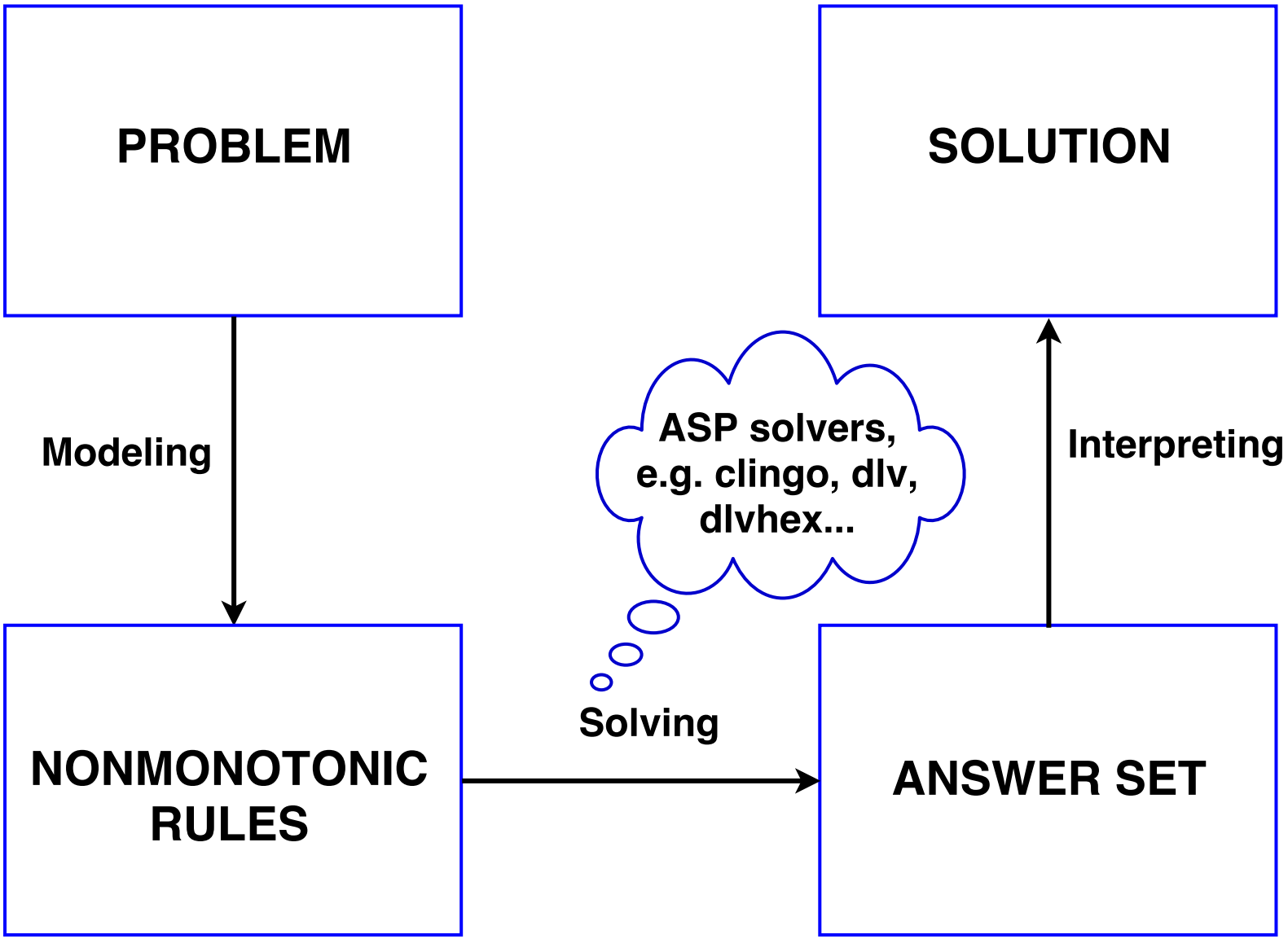
Nonmonotonic rule mining from KGs: OWA is a challenge!



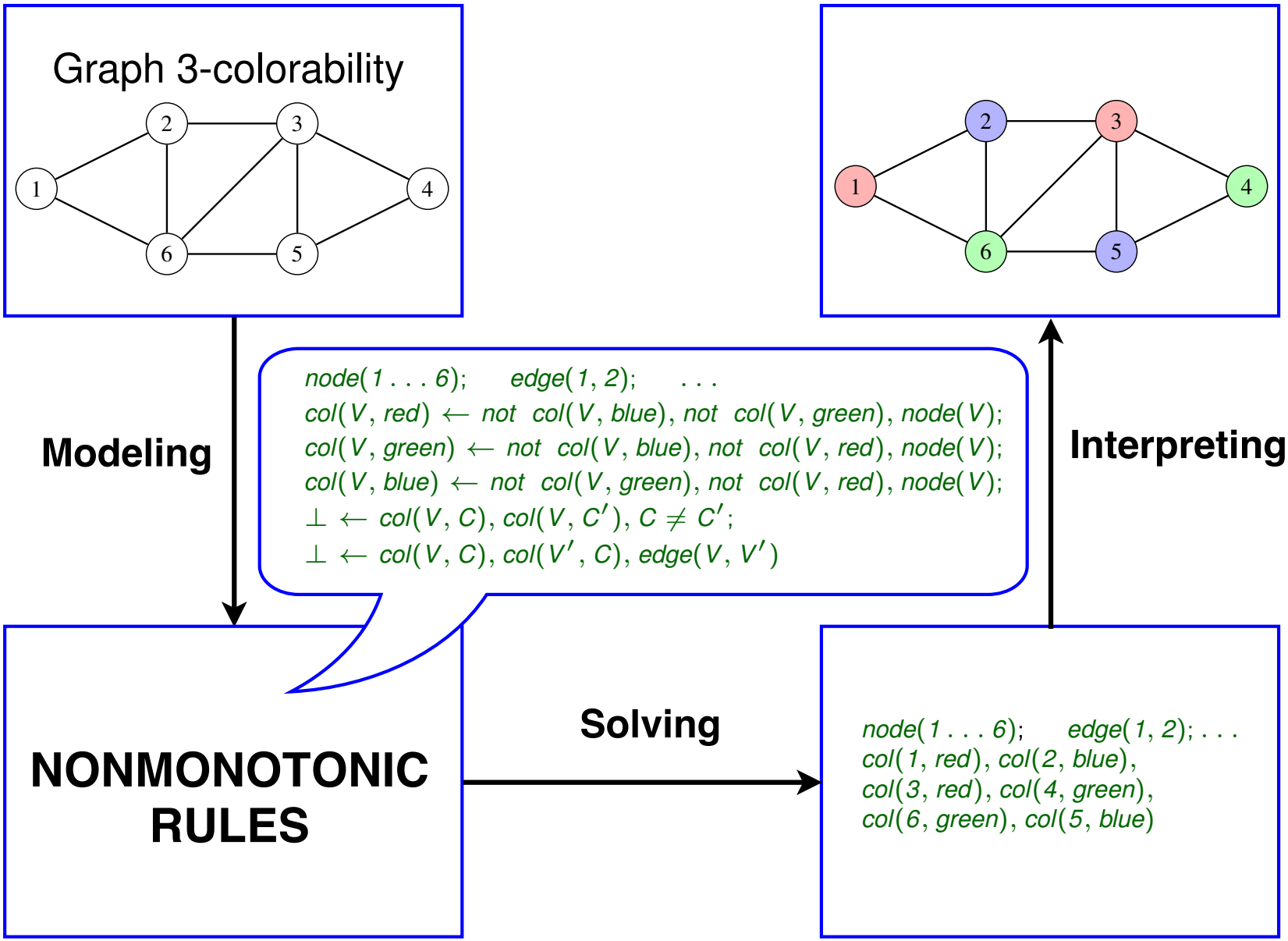
$$conf(r) = \frac{|\blacktriangle|}{|\blacktriangle| + |\text{♻}|} = 1$$

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

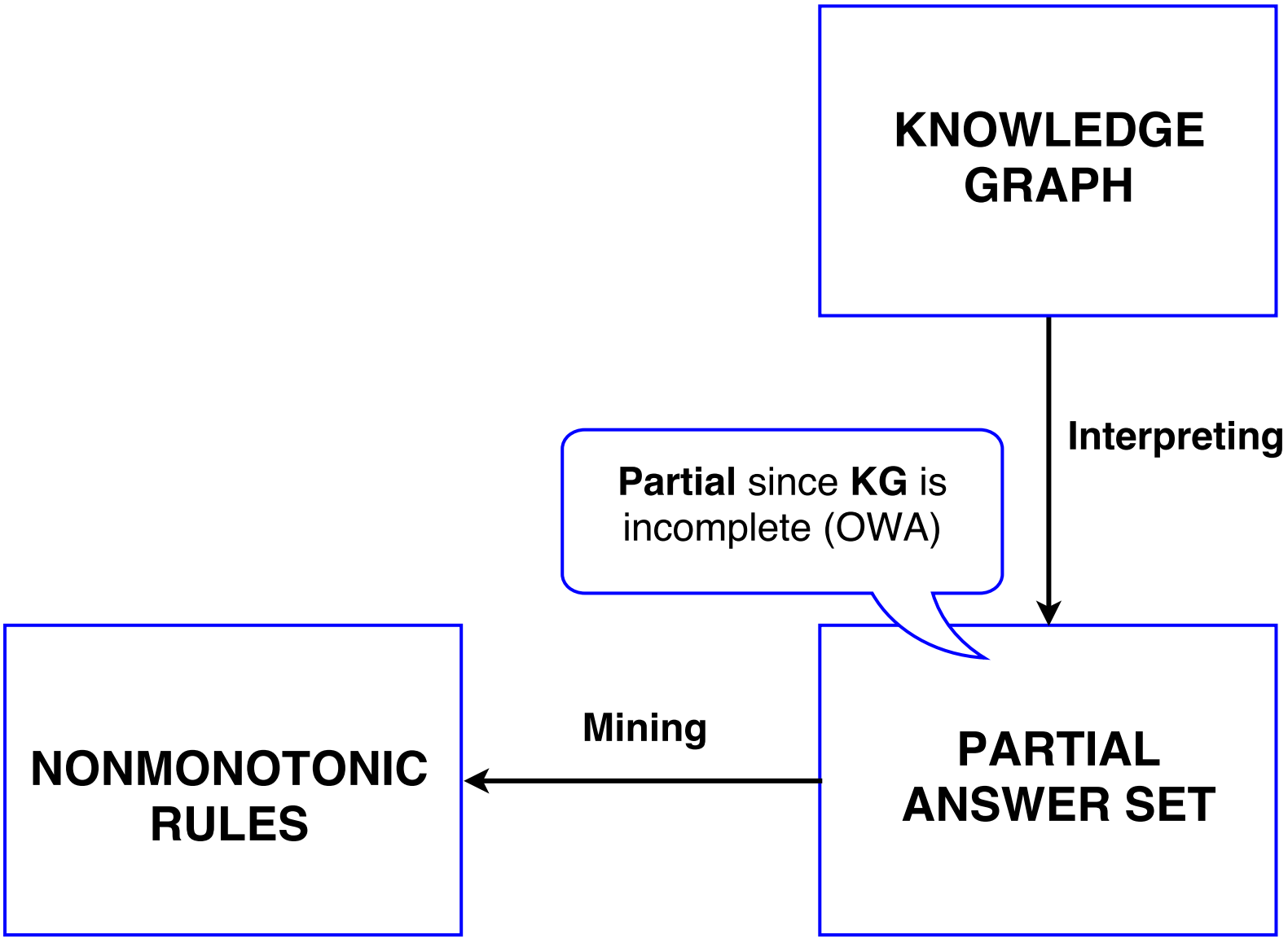
Declarative Programming Paradigm



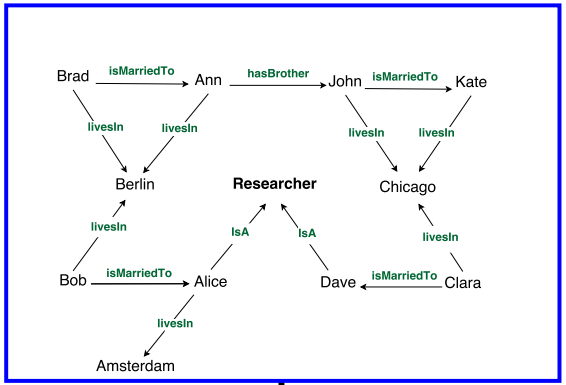
Declarative Programming Example



Nonmonotonic Rule Mining



Nonmonotonic Rule Mining



Interpreting

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

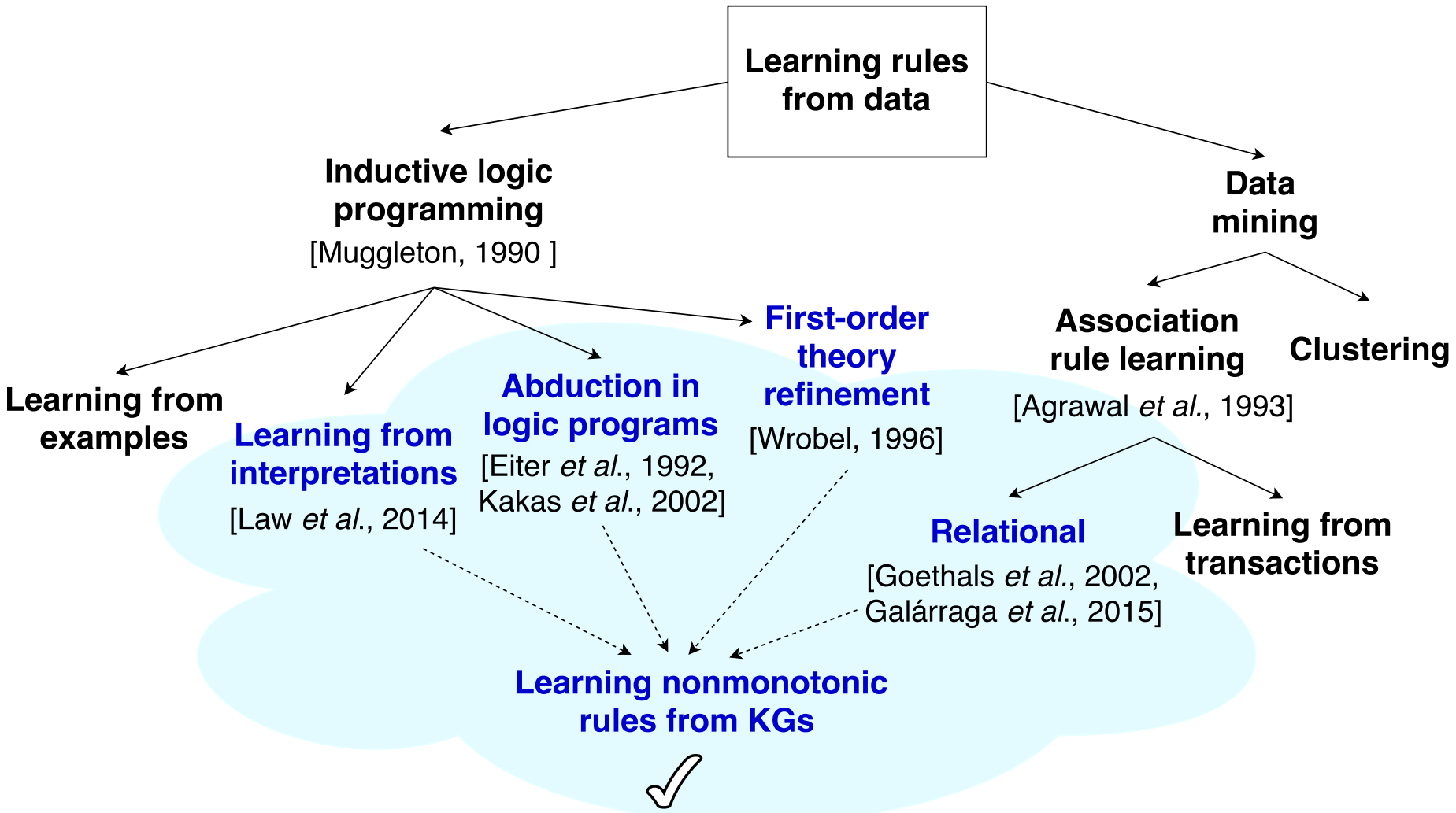
Mining

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

Nonmonotonic Rule Mining from KGs

Goal: learn nonmonotonic rules from KG

Approach: revise association rules learned using data mining methods

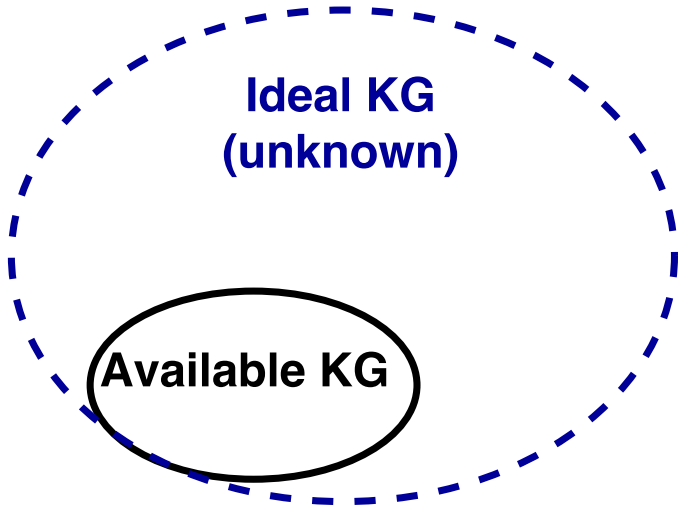


Horn Theory Revision

Quality-based Horn Theory Revision

Given:

- Available KG

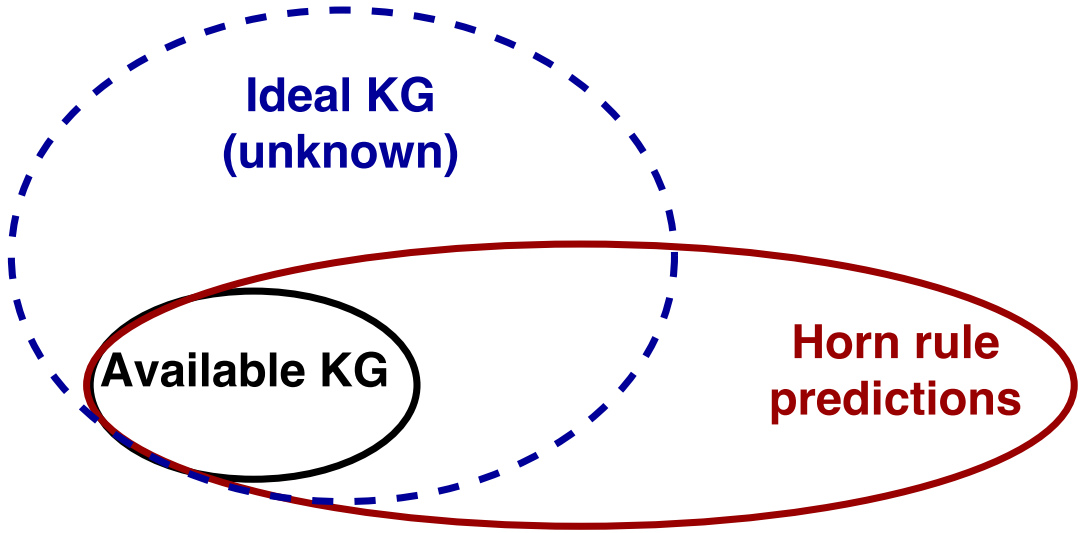


Horn Theory Revision

Quality-based Horn Theory Revision

Given:

- Available KG
- Horn rule set

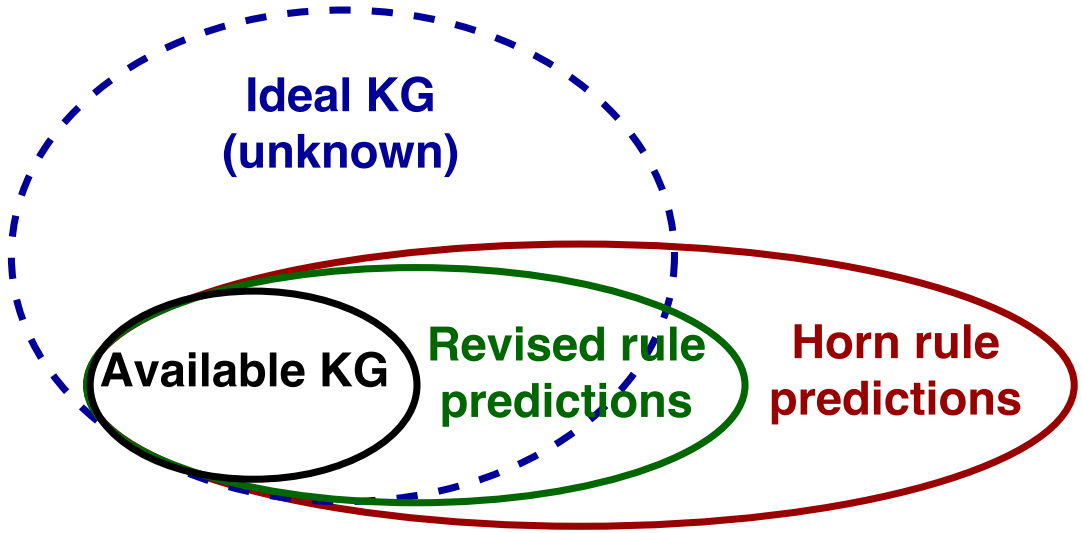


Horn Theory Revision

Quality-based Horn Theory Revision

Given:

- Available KG
- Horn rule set



Find:

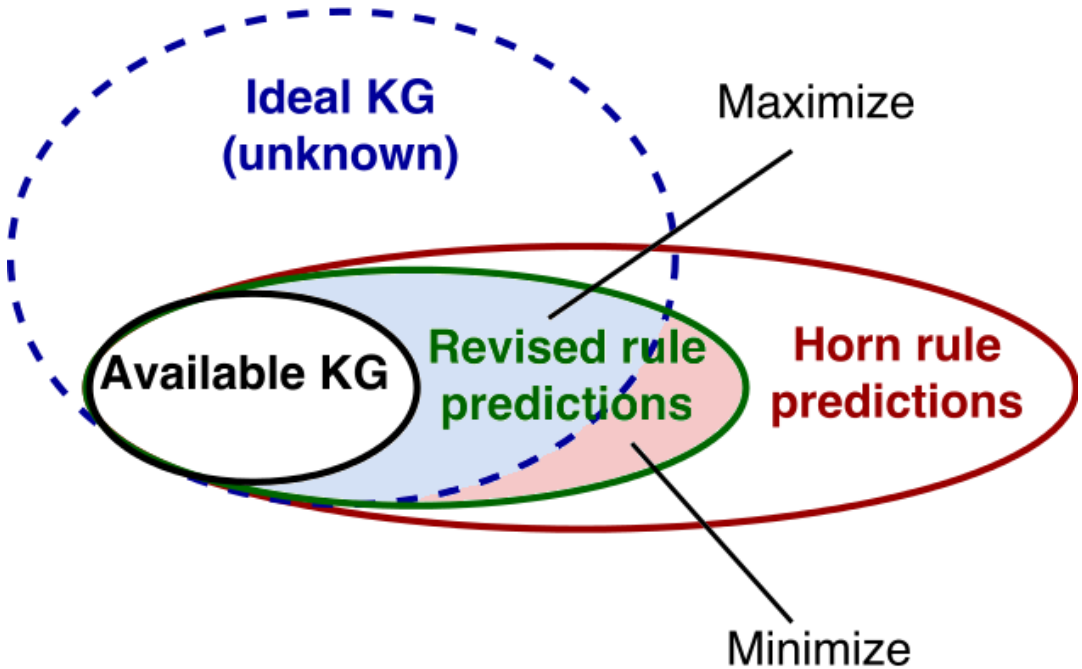
- Nonmonotonic revision of Horn rule set

Horn Theory Revision

Quality-based Horn Theory Revision

Given:

- Available KG
- Horn rule set



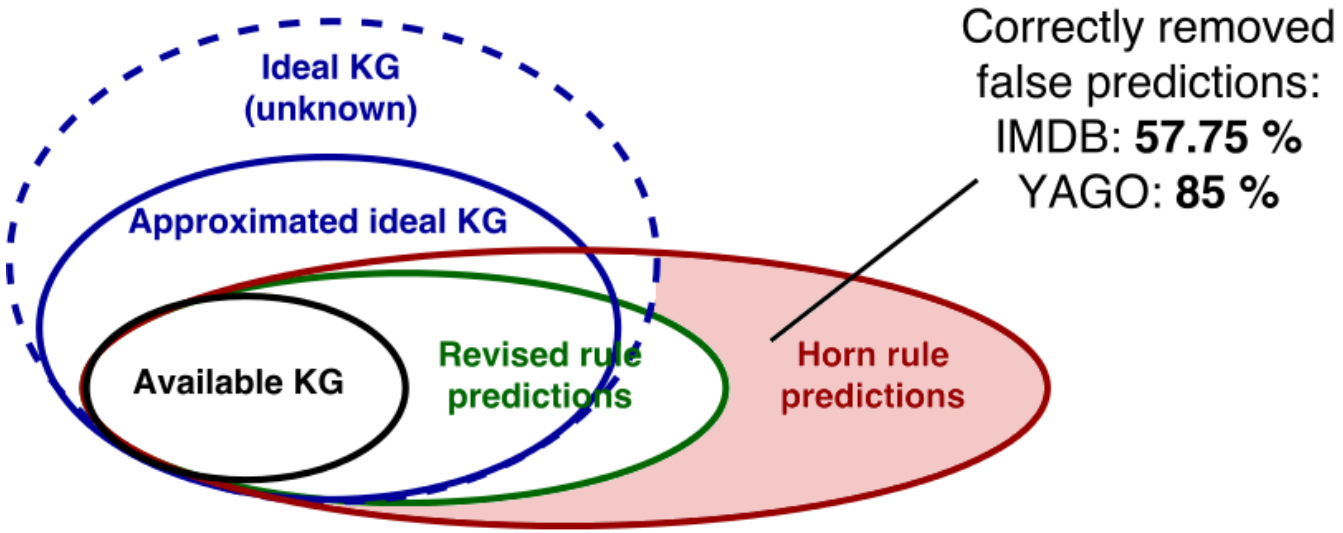
Find:

- Nonmonotonic revision of Horn rule set with better predictive quality

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:

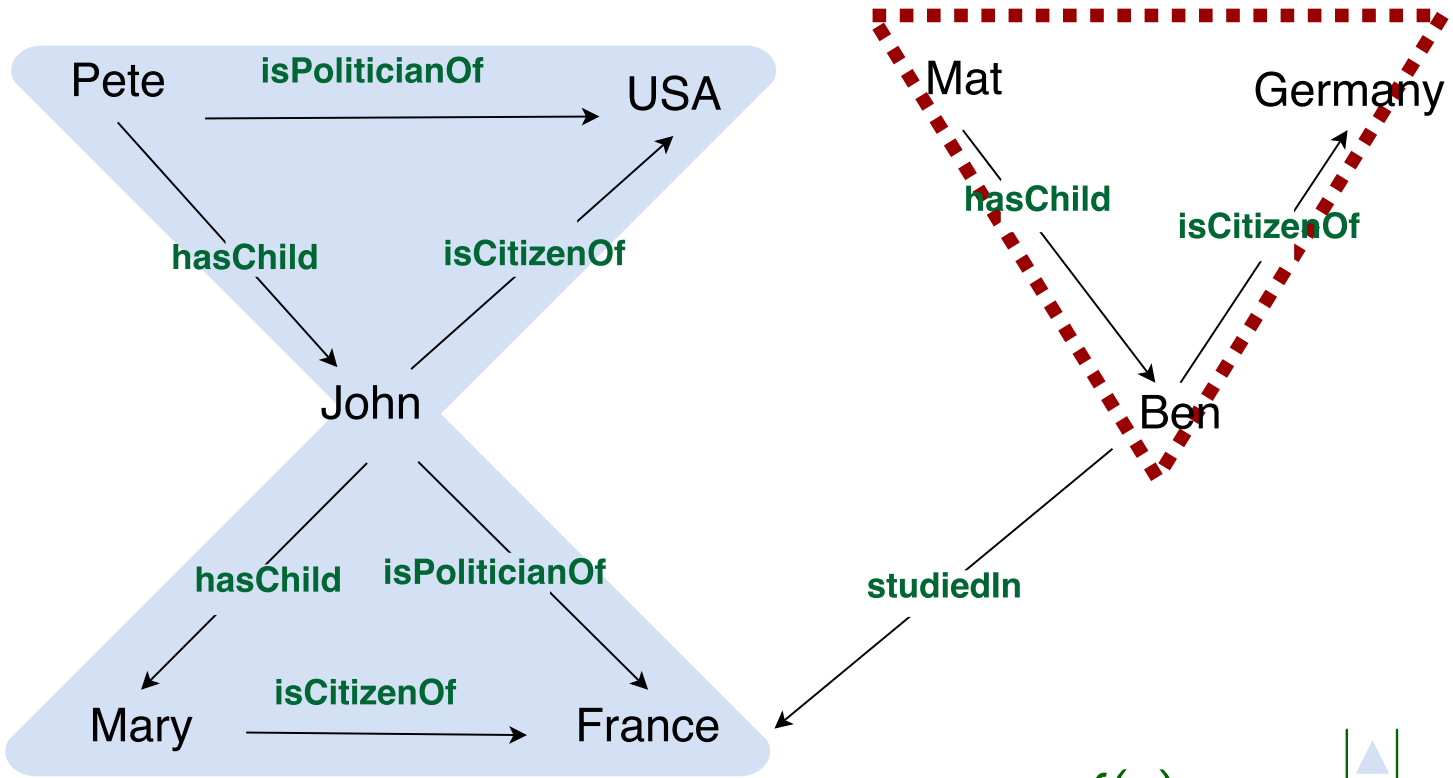
Plots of films in a sequel are written by the same writer, unless a film is American

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

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

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

Spurious Rules due to Incompleteness

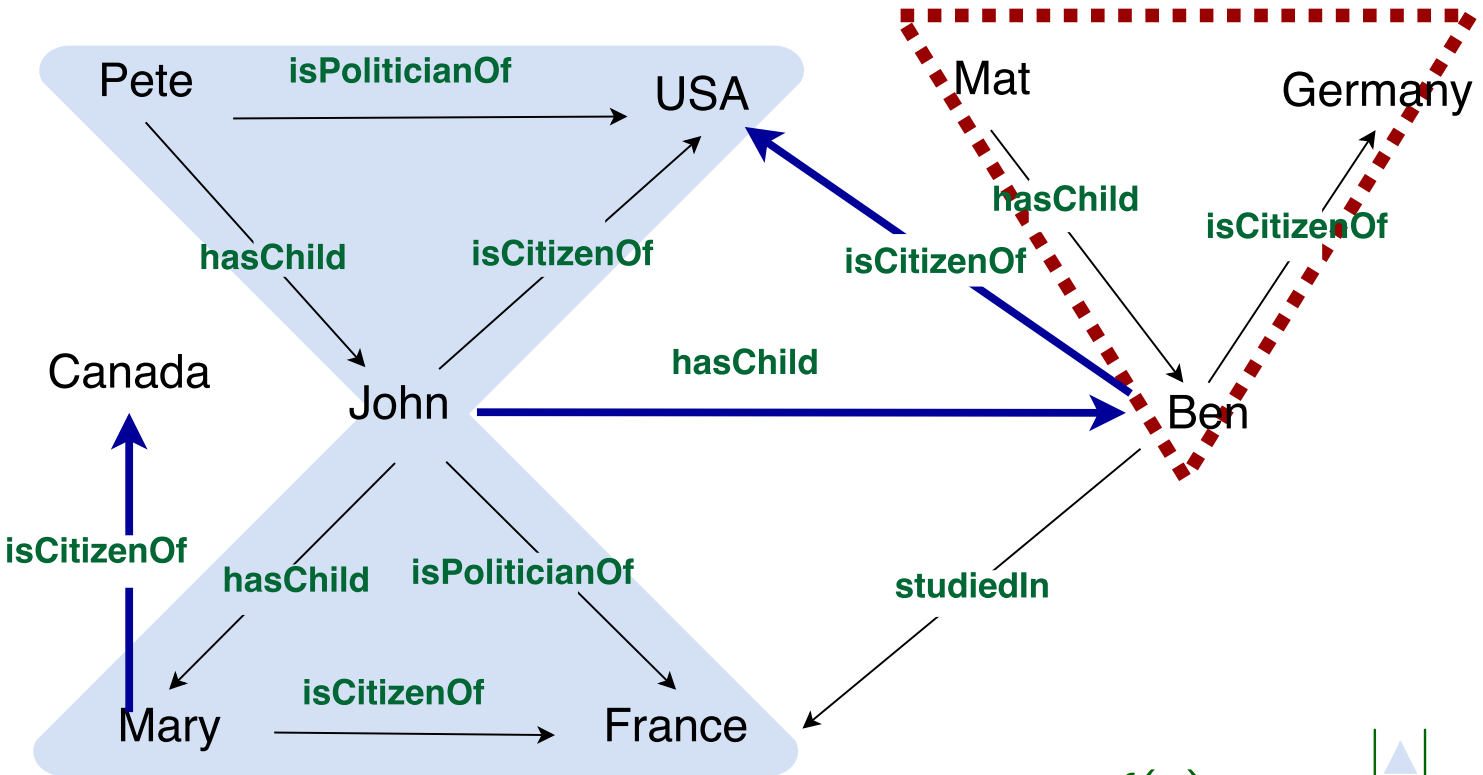


$$conf(r) = \frac{|\blacktriangle|}{|\blacktriangle| + |\text{♻}|} = \frac{2}{3}$$

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

Spurious Rules due to Incompleteness

In real world:

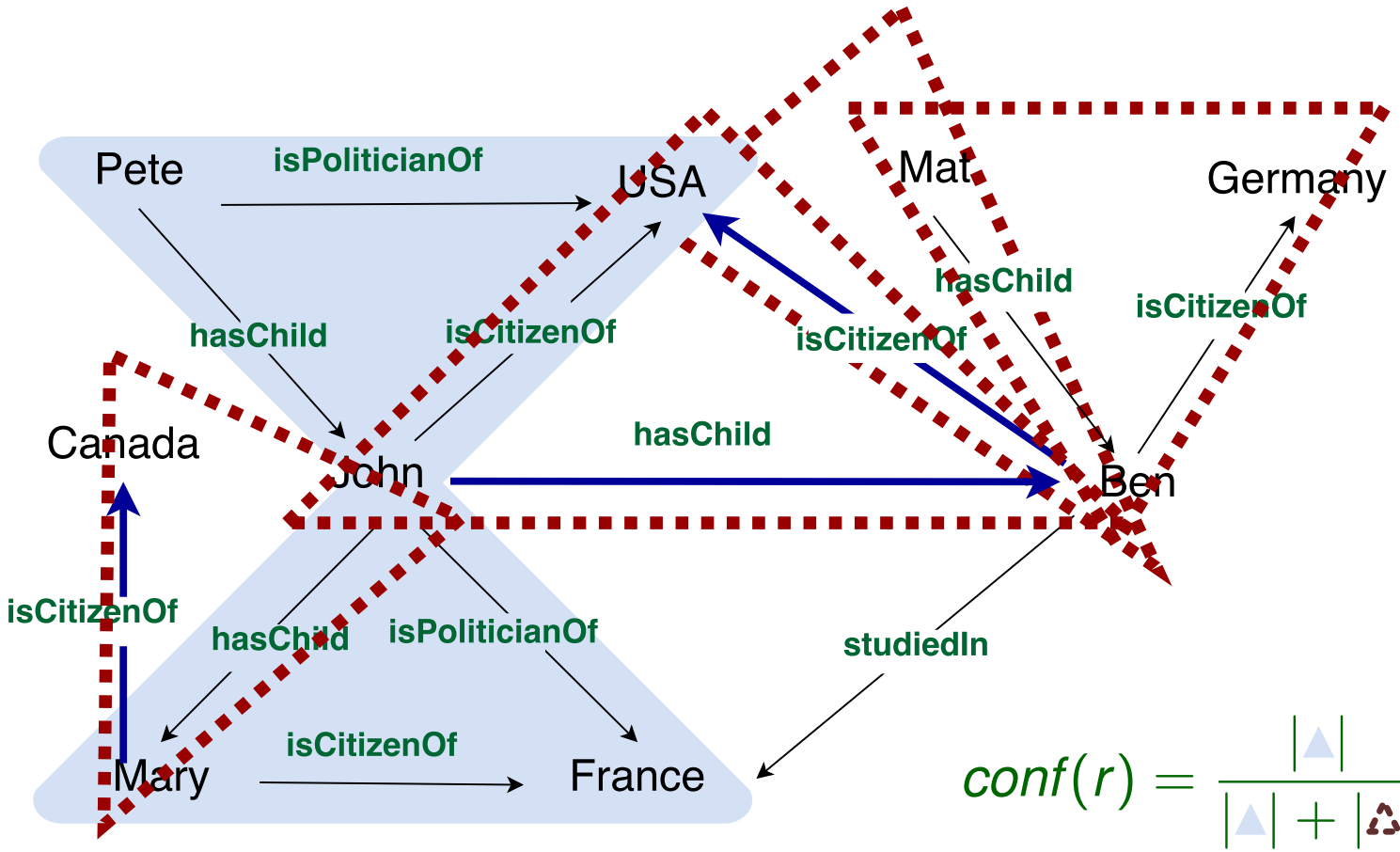


$$conf(r) = \frac{|\blacktriangle|}{|\blacktriangle| + |\blacktriangle\!\!\!\circlearrowright|} = \frac{2}{3}$$

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

Spurious Rules due to Incompleteness

In real world:

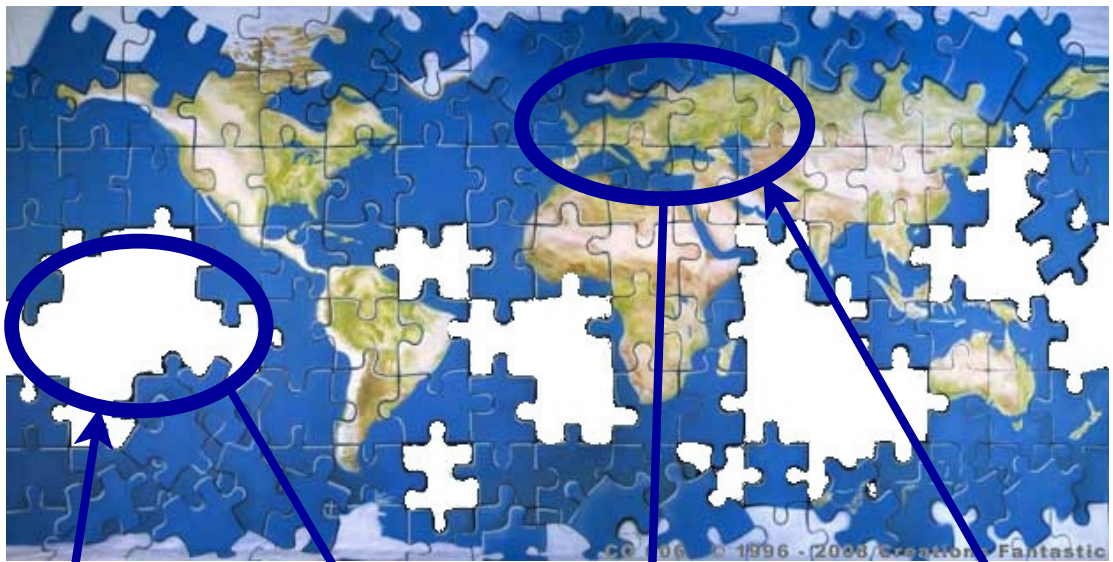


$$conf(r) = \frac{|\blacktriangle|}{|\blacktriangle| + |\blacktriangle|} = \frac{2}{6}$$

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

Completeness-aware Rule Learning

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



T. Pellissier-Tanon, D. Stepanova, S. Razniewski, P. Mirza, G. Weikum, Completeness-aware Rule Learning from Knowledge Graphs, ISWC 2017

Research Topics

Reasoning:

- Theoretical and practical open problems both in ASP and DLs
- Hybrid combinations of ASP with other logics
- Applications, e.g., constraint-based frequent pattern discovery[3]

Learning:

- Learn various types of rules from KGs, e.g.,
  - with cardinalities:
    "If you have X siblings, then your parents have X+1 children"
  - with existentials:
    "If you are a famous actor from Hollywood, there is likely an award that you were nominated for"
  - etc.

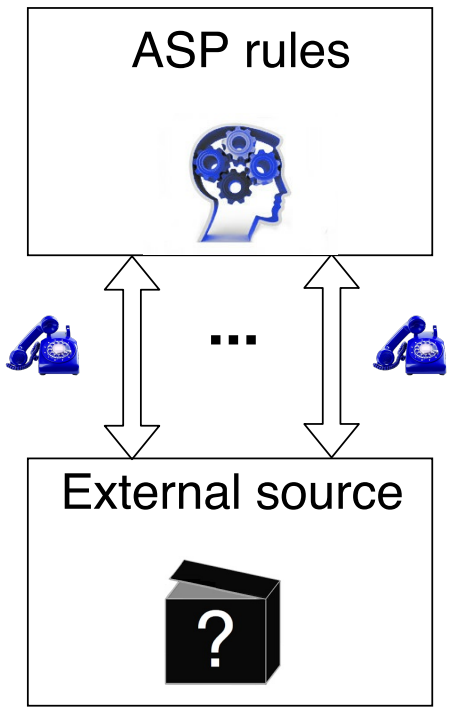
Hybrid topics:

- Use rules for NLP and information extraction tasks
- Angryhex project

[3] S. Paromonov, D. Stepanova, P. Miettinen, Hybrid ASP-based Approach to Pattern Mining, RR 2017

Summary

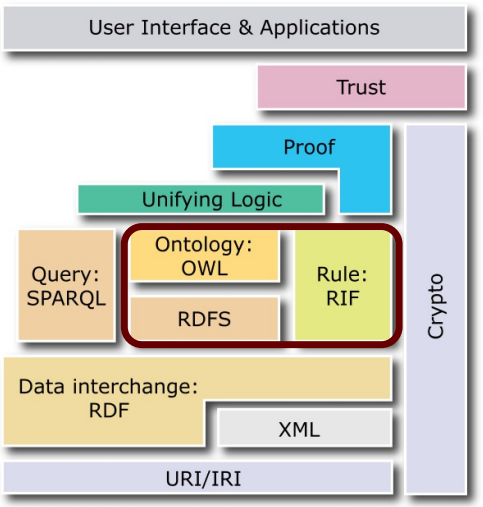
- DIs vs ASP
  - What of DLs can be expressed in ASP
  - What of DIs cannot be expressed in ASP
- DL-programs
  - Example applications
- HEX-programs
  - Example applications
- Rule learning
  - Rules with exceptions
  - Completeness awareness



Conclusion

Main part:

1. **Description Logic ontologies (DL)**
   - Syntax and semantics
   - Reasoning problems
   - Ontology Web Language (OWL)
   - Tools and applications
2. **Answer Set Programming rules (ASP)**
   - Syntax and semantics
   - Declarative programming paradigm
   - Tools and applications



Advanced topics:

3. **ASP extensions and rule learning**
   - DL-programs
   - HEX-programs
   - Applications
   - Approaches to rule learning

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.


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.


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.

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?

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

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

Avoid Data Overfitting

How to distinguish exceptions from noise?

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

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

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

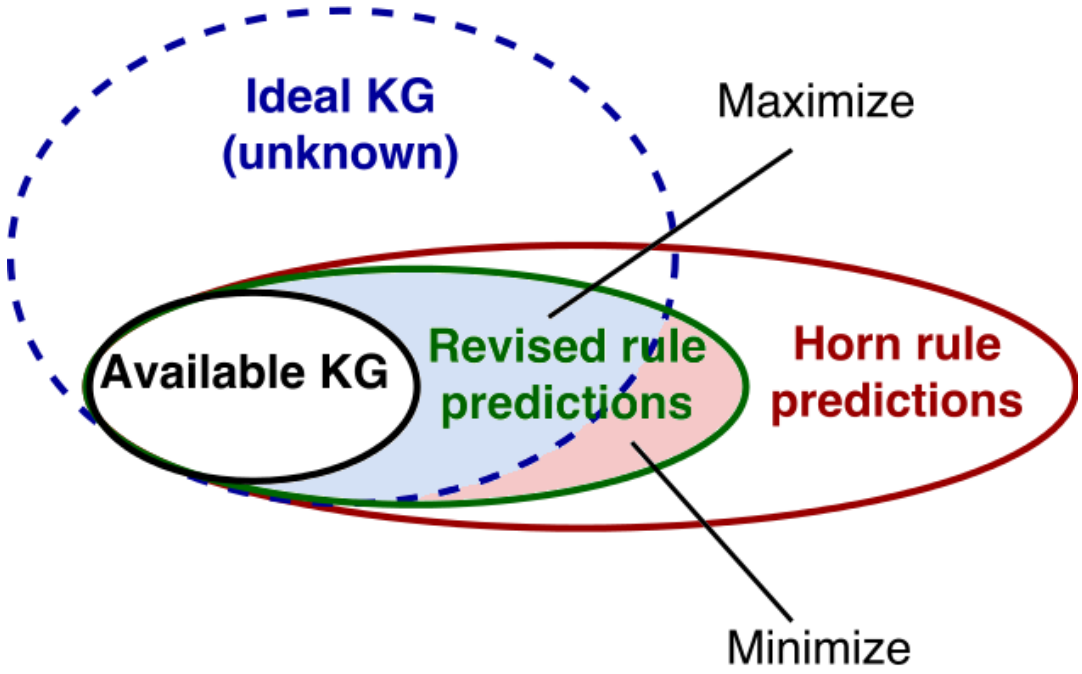
Intuition: Rules with good exceptions should make few conflicting predictions

Horn Theory Revision

Quality-based Horn Theory Revision

Given:

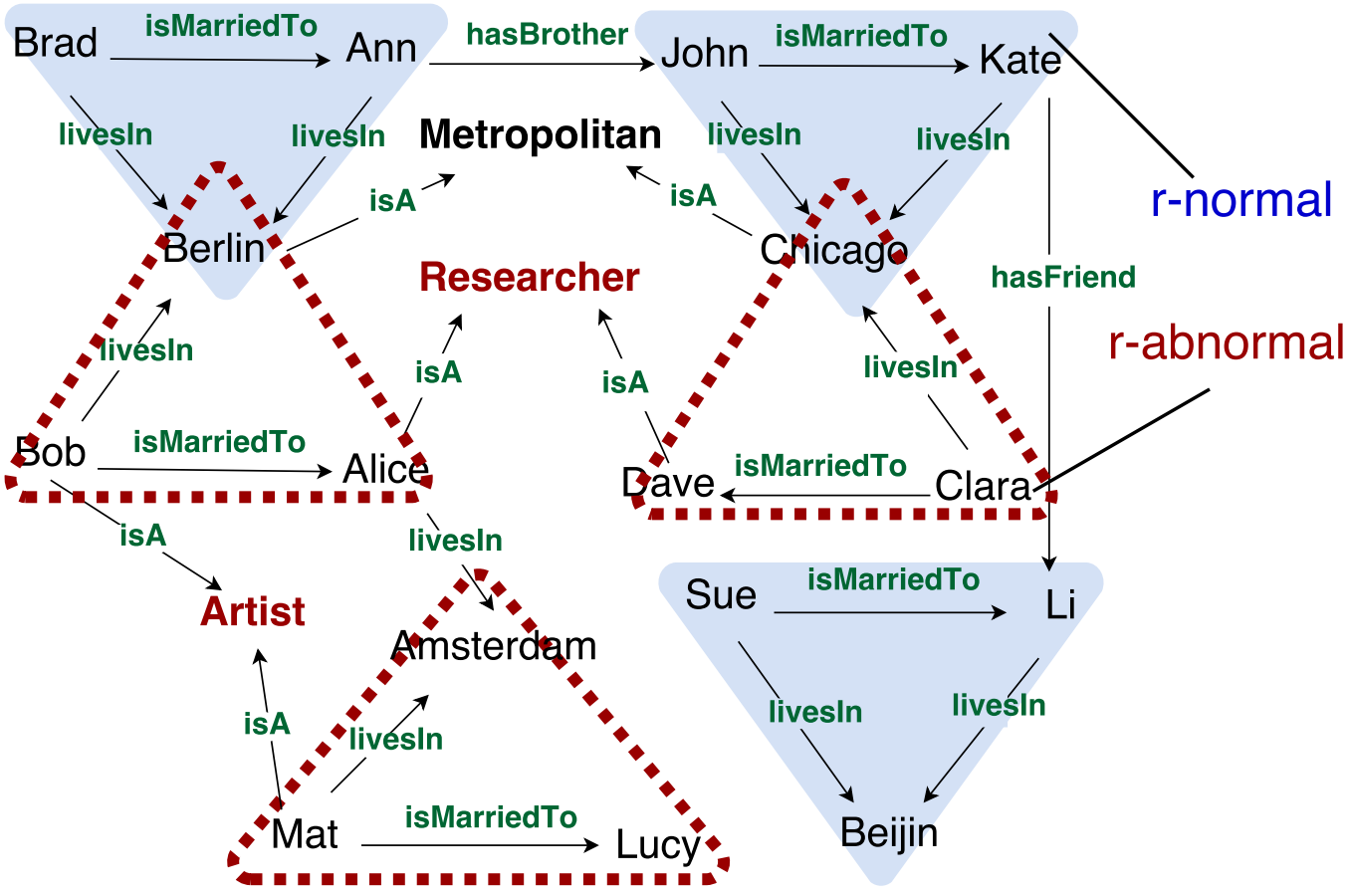
- Available KG
- Horn rule set



Find:

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

Exception Candidates



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

$$\left\{ \begin{array}{l} not\ researcher(X) \\ not\ artist(Y) \end{array} \right\}$$

Exception Ranking

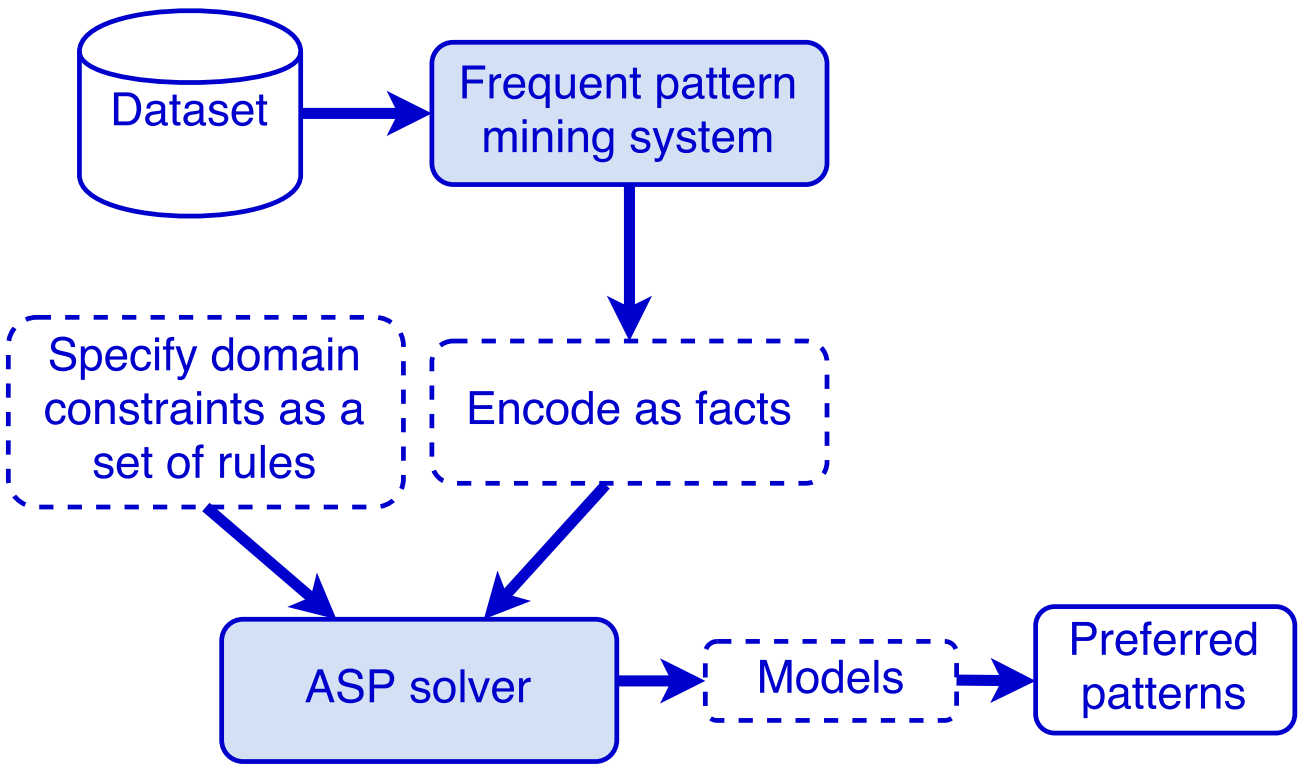
$$\begin{array}{ll} \textit{rule1} & \{\underline{\mathbf{e_1}}, e_2, e_3, \ldots\} \\ \textit{rule2} & \{e_1, \underline{\mathbf{e_2}}, e_3, \ldots\} \\ \textit{rule3} & \{\underline{\mathbf{e_1}}, e_2, e_3, \ldots\} \end{array}$$

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

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