Fast Computation of Explanations for Inconsistency in Large-Scale Knowledge Graphs
(presented at WWW 2020)

Trung Kien Tran[1], Mohamed H. Gad-Elrab[1,2], Daria Stepanova[1], Evgeny Kharlamov[1], Jannik Strötgen[1]

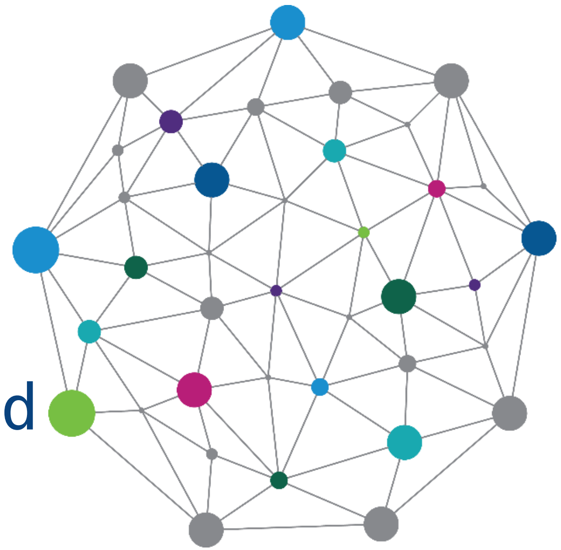
[1]Bosch Center for Artificial Intelligence, Renningen, Germany
[2]Max-Planck Institute for Informatics, Saarbrücken, Germany

BOSCH

Motivation

Knowledge Graphs(KGs)

- Crow-sourcing/hand-crafted
- Automatically constructed
- Semi-automatically constructed



FreeBase, Wikidata

ConceptNet, OpenCyc

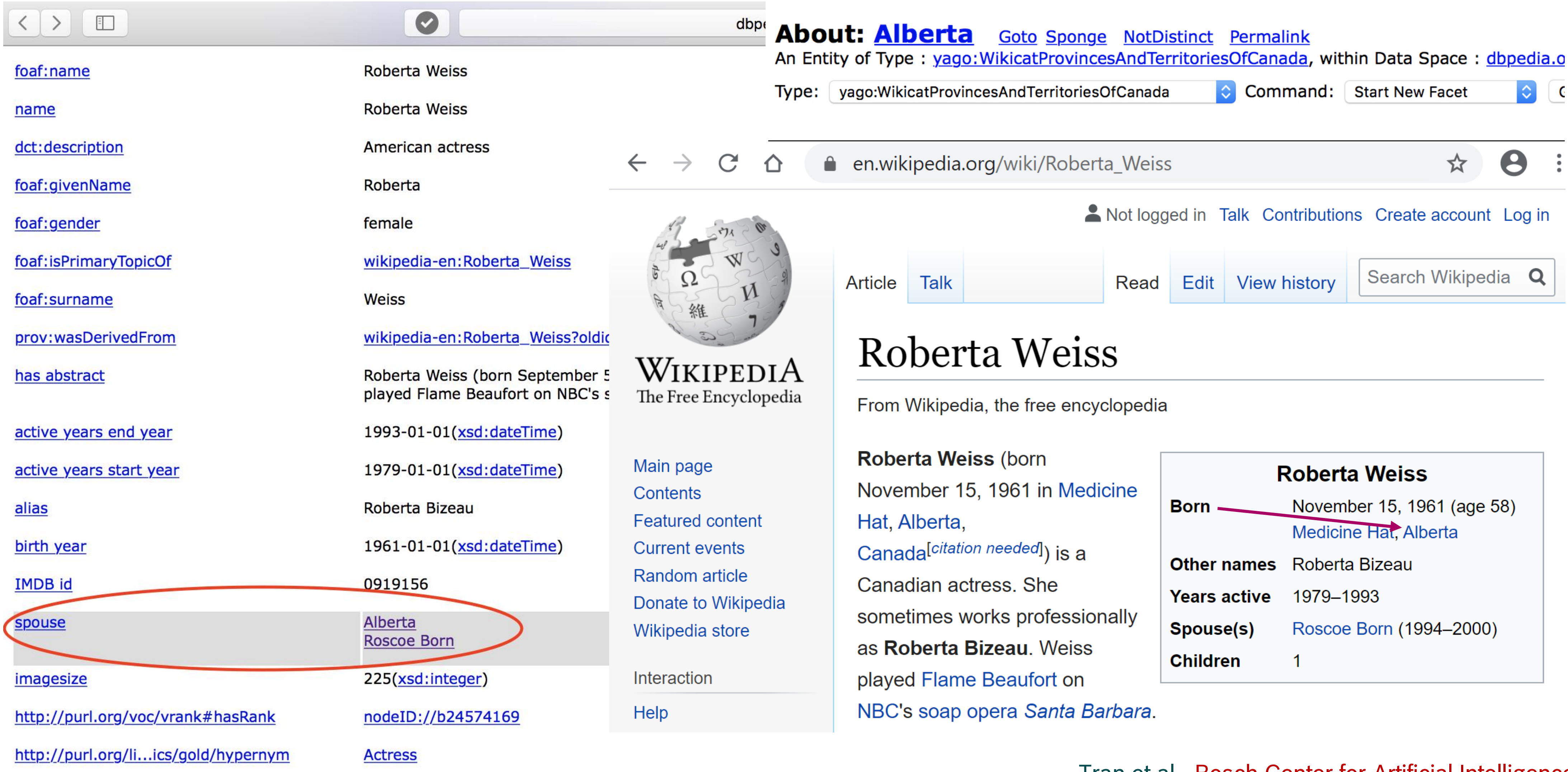
Knowledge Vault

NELL, OpenIE

YAGO, DBpedia, ...

Detecting errors in large KGs is **important** and **challenging** task!

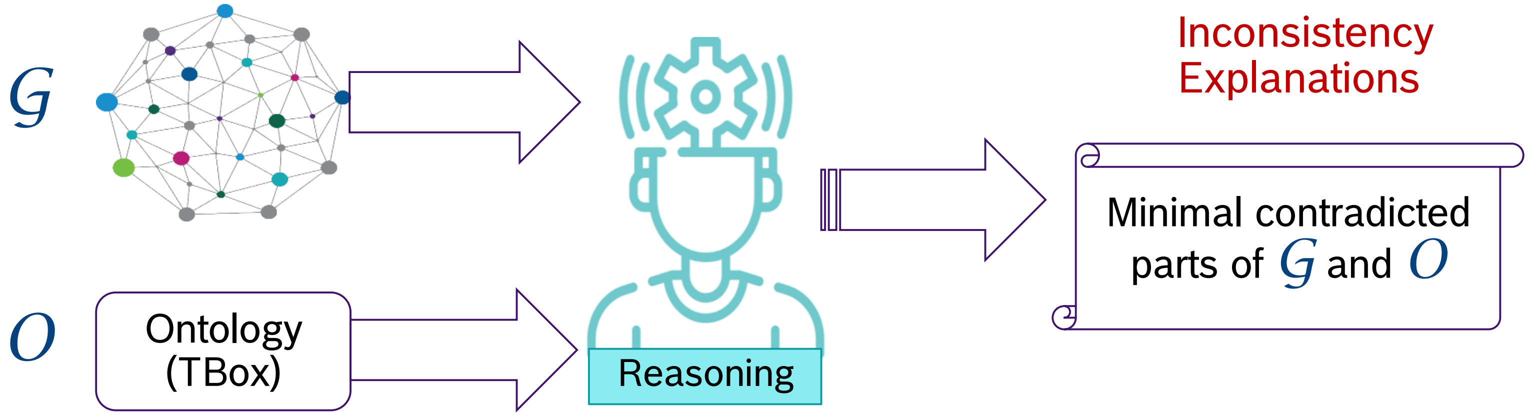
Motivation: Example



Problem Statement

- **Given**: $\mathcal{G}$ and $\mathcal{O}$
- **Task**: check if $\mathcal{G}$ is consistent w.r.t. $\mathcal{O}$

  If $\mathcal{G}$ is inconsistent w.r.t. $\mathcal{O}$, compute all inconsistency explanations



Example:

(1) spouse(roberta_weiss, alberta)

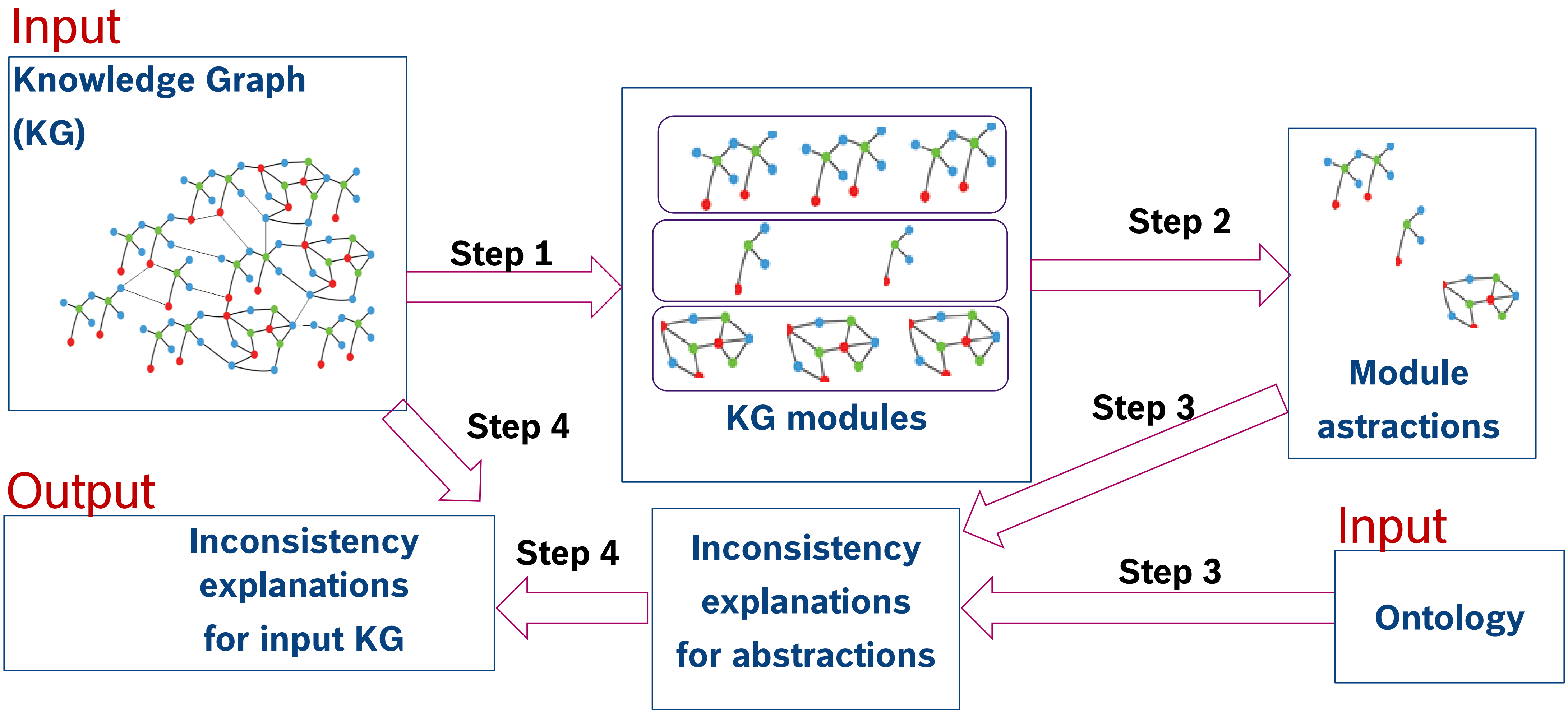
(2) Settlement(alberta)

(3) Actress(roberta_weiss)

(4) alias(roberta_weiss, bizeau)

(5) Person(x), Person(y) ← spouse(x,y)
(6) Place(x) ← Settlement(x)
(7) Inconsistent ← Place(x), Person(x)

Overview of the approach

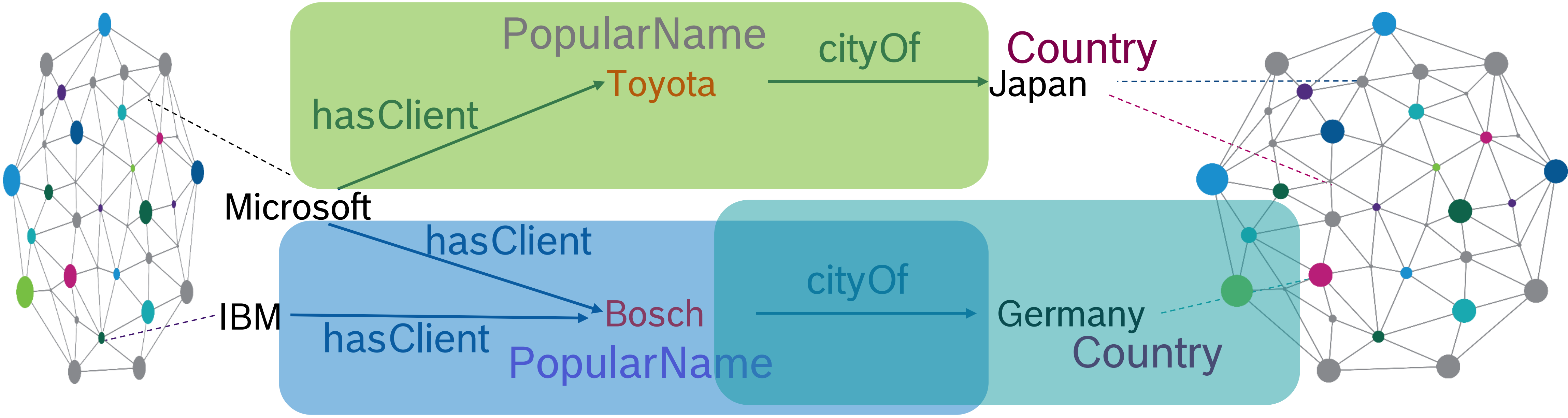


The Web Ontology Language (OWL 2) fragment

Syntax	Semantics	
R	$R^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$	Relations
`ObjectInverseOf`(R)	$\{\langle e, d\rangle \mid \langle d, e\rangle \in R^{\mathcal{I}}\}$	
A	$A^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$	Classes/complex types
`owl:Thing`	$\Delta^{\mathcal{I}}$	
`owl:NoThing`	$\emptyset$	
`ObjectComplementOf`(C)	$\Delta^{\mathcal{I}} \setminus C^{\mathcal{I}}$	
`ObjectIntersectionOf`(C)	$C^{\mathcal{I}} \cap D^{\mathcal{I}}$	
`ObjectUnionOf`(C, D)	$C^{\mathcal{I}} \cup D^{\mathcal{I}}$	
`ObjectSomeValuesFrom`(P, `owl:Thing`)	$\{d \mid \exists e \in \Delta^{\mathcal{I}} : \langle d, e\rangle \in P^{\mathcal{I}}\}$	
`SubClassOf`(C, D)	$C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$	Ontology axioms
`SubObjectPropertyOf`(P, S)	$P^{\mathcal{I}} \subseteq S^{\mathcal{I}}$	
`TransitiveObjectProperty`(P)	$P^{\mathcal{I}} \circ P^{\mathcal{I}} \subseteq P^{\mathcal{I}}$	
`ClassAssertion`(C, a)	$a^{\mathcal{I}} \in C^{\mathcal{I}}$	Triples/assertions
`ObjectPropertyAssertion`(R, a, b)	$\langle a^{\mathcal{I}}, b^{\mathcal{I}}\rangle \in R^{\mathcal{I}}$	

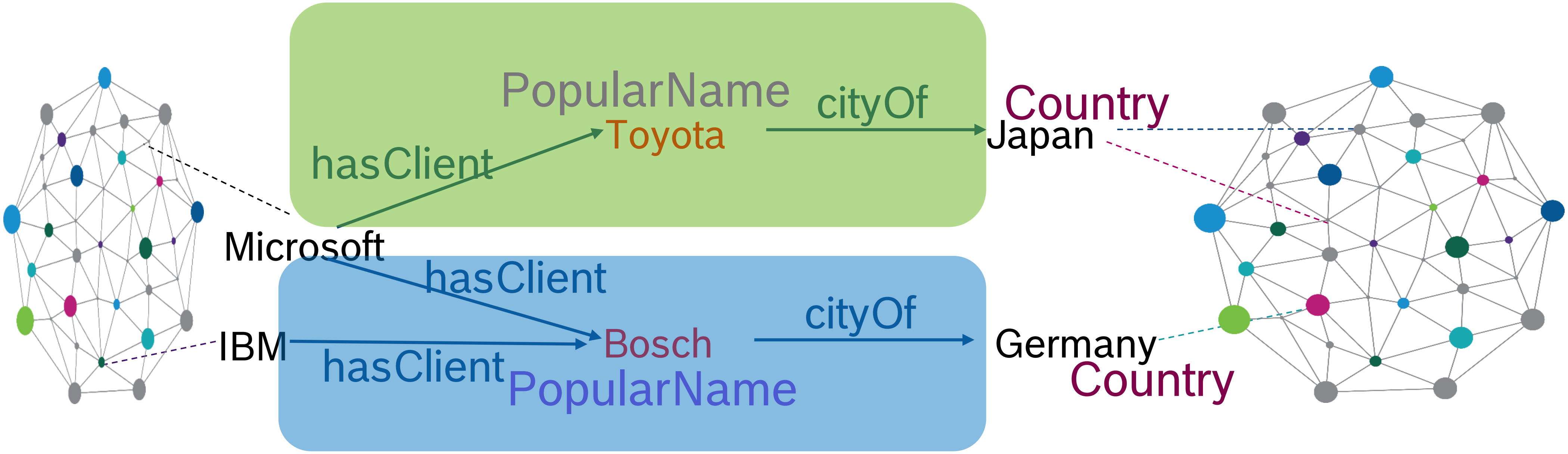
The language can express other axioms: ObjectPropertyDomain, ObjectPropertyRange, EquivalentClasses, DisjointClasses, EquivalentProperties

Modularization



Locality property: For any ontology $\mathcal{O}$ (in the specified language), consistency checking of $\mathcal{G}$ can be reduced to consistency checking of all modules of $\mathcal{G}$

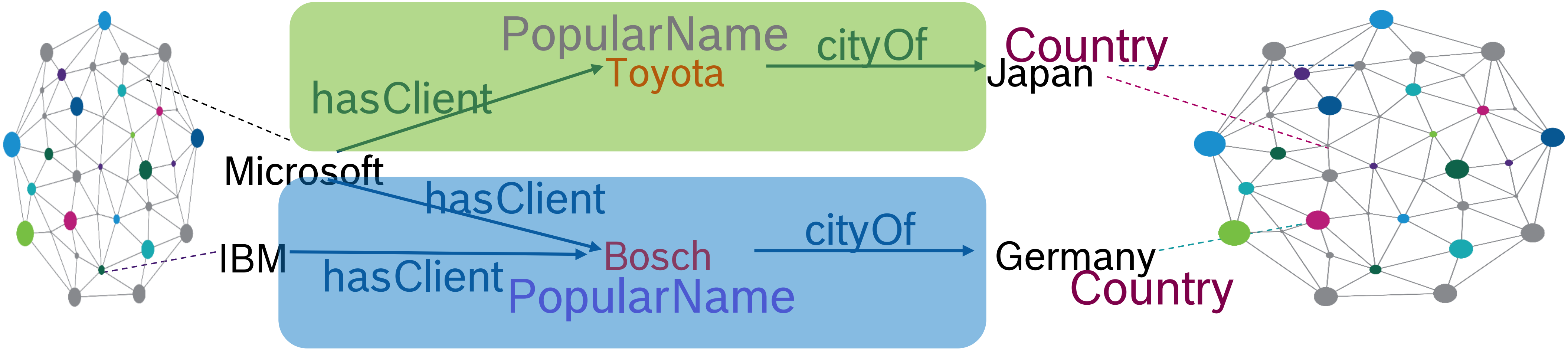
Module Abstraction



- **Locality property** enables parallel and incremental processing
- Still there are too many modules

→Solution: abstract and process multiple modules **at the same time**

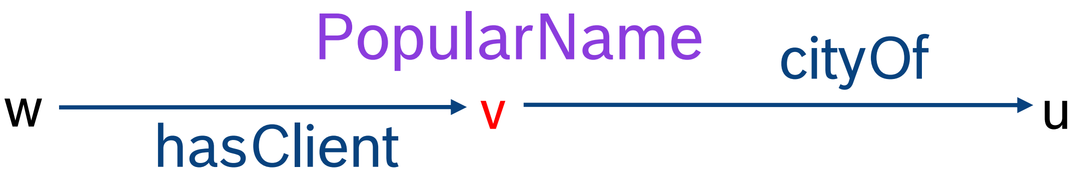
Module Abstraction



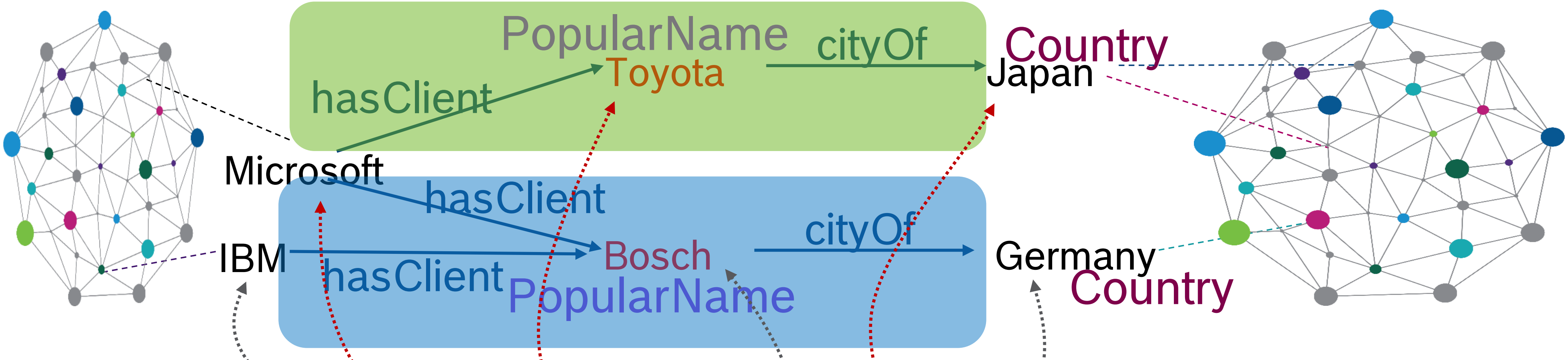
- **Local Type (lt):** ({types}, {incoming relations}, {outgoing relations})

lt(Toyota) = lt(Bosch) = ({PopularName}, {hasClient}, {cityOf})

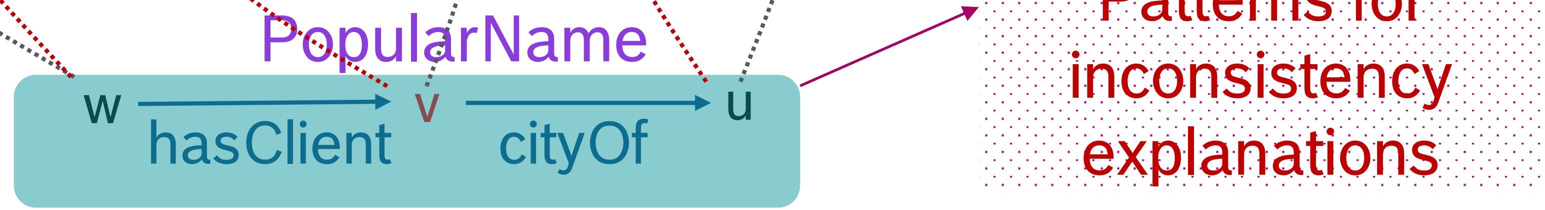
- **Abstraction of module(s):**



Compute inconsistency explanations



- **Compute inconsistency explanations over abstractions and map back to the input KG**



Ontololgy O	PropertyRange(hasClient, Company) PropertyDomain(cityOf, City) DisjointClasses(Company,City, Country)

Experiments

- Datasets: YAGO, DBpedia, and NPD
- Baselines: Pellet Reasoner, Modularization
- Statistics of the datasets

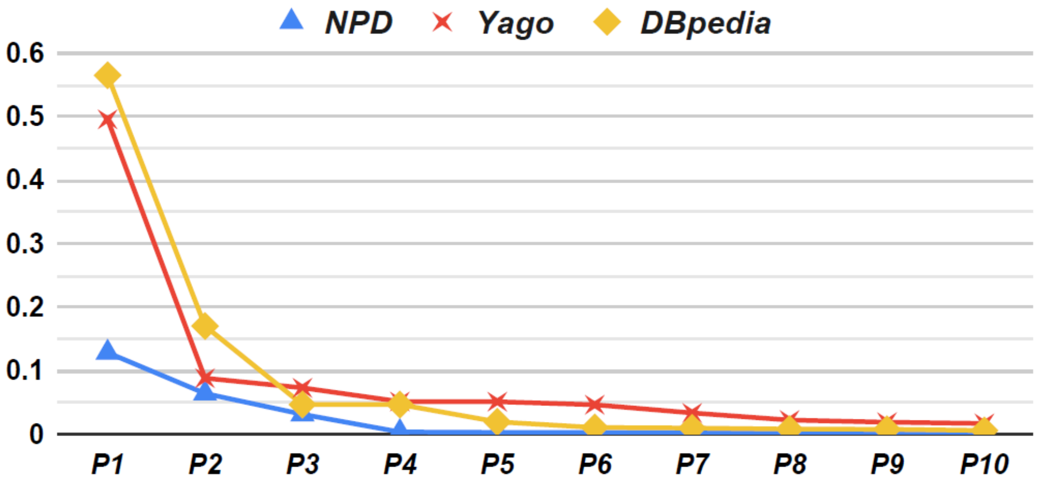
KG	Axioms	Triples	Entities	Classes	Relations
NPD	678	929,710	264,081	343	142
Yago	1,045	7,321,308	3,275,593	960	38
DBpedia	4,287	22,955,173	5,867,913	685	663

Experimental Results

	Method	Total Modules	Processed Modules	Patterns	Explanations	Time (hours)
NPD	*Pellet*	1	0	-	-	72
	Modular	264,081	243,467	-	41,128	72
	Abstraction	11,970	11,866	20,251	60,554	4.5
Yago	*Pellet*	1	0	-	-	72
	Modular	3,275,593	1,624,196	-	1,547	72
	Abstraction	2,821	2,821	69	3,565	0.1
DBpedia	*Pellet*	1	0	-	-	72
	Modular	5,867,913	647,443	-	73,152	72
	Abstraction	91,084	90446	1,797	20,093,617	21

Note: Each abstraction is regarded as a module

Distribution of inconsistency explanations



	Pattern	Ontology axioms	Example
YAGO	wn:building(v) isLocatedIn(u,v)	DisjointClasses(wn:building, GeoEntity) ObjectPropertyRange(isLocatedIn,GeoEntity)	wn:building(<J._Forrestal_Building>) isLocatedIn(<NNSA>,<J._Forrestal_Building>)
DBpedia	bandMember(u_1,v) instrument(u_2,v)	SubClassOf(Ship,MeanOfTransportation) SubClassOf(Instrument,Product), SubClassOf(Product,Ship) DisjointClasses(MeanOfTransportation,Person) ObjectPropertyRange(bandMember,Person) ObjectPropertyRange(instrument,Instrument)	instrument(<African_blues>,<Djembe>) bandMember(<Baka_Beyond>,<Djembe>)

Conclusions

- A scalable symbolic-reasoning approach to compute inconsistency explanations in large KGs
- Explanations are grouped into patterns, which could reveal systematic errors of the KG construction process
- Works for a practical fragment of Web Ontology Language (OWL 2)

THANK YOU!

Putting all together

Input : A knowledge graph $\mathcal{G}$ and an ontology $\mathcal{O}$

Output: The set `allExpls` of all explanations for inconsistency of $\mathcal{G} \cup \mathcal{O}$

`allExpls` $\leftarrow \emptyset$

`/* Step 1 & 2: compute local types of all individuals in` $\mathcal{G}$ `*/`

`types` $\leftarrow \{\tau(a, \mathcal{G}) \mid a \in \mathsf{ind}(\mathcal{G})\}$

foreach *maximal* $\tau \in$ `types` **do**

    `/* Step 3: compute explanations for the star shape abstraction of` τ `using a reasoner */`

    $X \leftarrow$ all explanations for inconsistency of $\mathsf{abs}(\tau) \cup \mathcal{O}$

    `/* Step 4: obtain the explanations for` $\mathcal{G}$ `*/`

    foreach $\mathcal{E} = \mathcal{E}_{\mathcal{G}} \cup \mathcal{E}_{\mathcal{O}} \in X$ **do**

        `/* compute the local type of` v_τ `in` $\mathcal{E}_{\mathcal{G}}$ `*/`

        $\tau' = \tau(v_\tau, \mathcal{E}_{\mathcal{G}})$

        `newExpls` $\leftarrow$ all realizations of τ' in $\mathcal{G}$

        `allExpls` $\leftarrow$ `allExpls` $\cup$ `newExpls`

return `allExpls`