It's all Connected:

Knowledge Representation for Graph Data

Magdalena Ortiz

KR 2026 @ FLoC, Lisboa, Portugal, 20 July 2026

Supported by FWF projects 10.55776/P30873 and 10.55776/COE12

Thanks to a fantastic team!

Shqiponja Ahmetaj

Janos Arpasi

Bartosz Bednarczyk

Bente Gortworst

Bianca Löhnert

Sanja Lukumbuzya

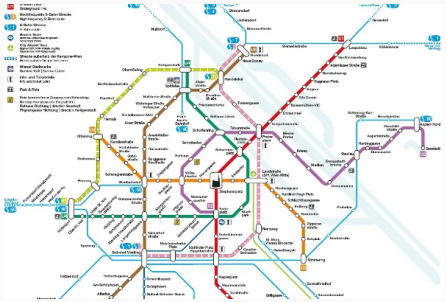
Cem Okulmus

Anouk Oudshoorn

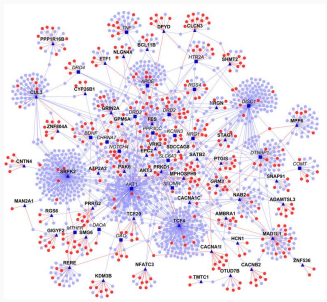
Mantas Šimkus

Graph Data

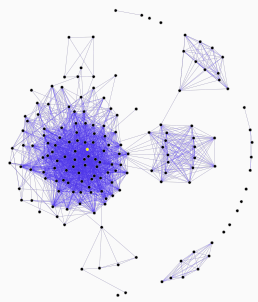
Graphs are everywhere



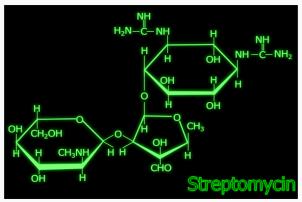
Transport networks



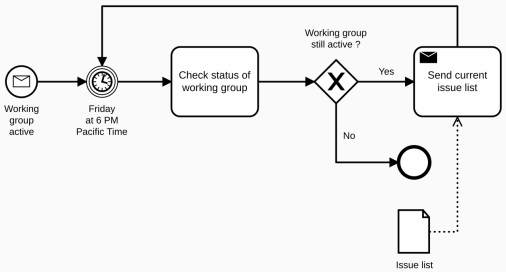
Interactome



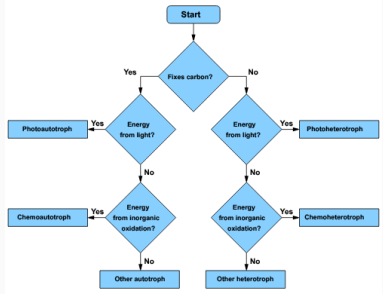
Social networks



Molecules



Business processes



Flow charts

Fundamental abstractions for modelling and computation.

Graphs as a Data Model

Graph data is increasingly adopted as a **data model**, particularly when connections between nodes convey important information.

Papers affected by the retraction of AB24 to degree ≤ 3

Graph query: AB24 $\xrightarrow{\text{cites}^- \cdot (\varepsilon|\text{cites}^-) \cdot (\varepsilon|\text{cites}^-)}$ x **vs. SQL:**

```
SELECT src FROM Cites WHERE dst='AB24'
     UNION
SELECT c2.src FROM Cites c1 JOIN Cites c2
          ON c1.src=c2.dst WHERE c1.dst='AB24'
     UNION
SELECT c3.src FROM Cites c1
     JOIN Cites c2 ON c1.src=c2.dst
     JOIN Cites c3 ON c2.src=c3.dst
          WHERE c1.dst='AB24'
```

An

Biko

Cruz

AB2

st

23

324

C25

Two **graphs-as-data** paradigms: RDF and Property Graphs

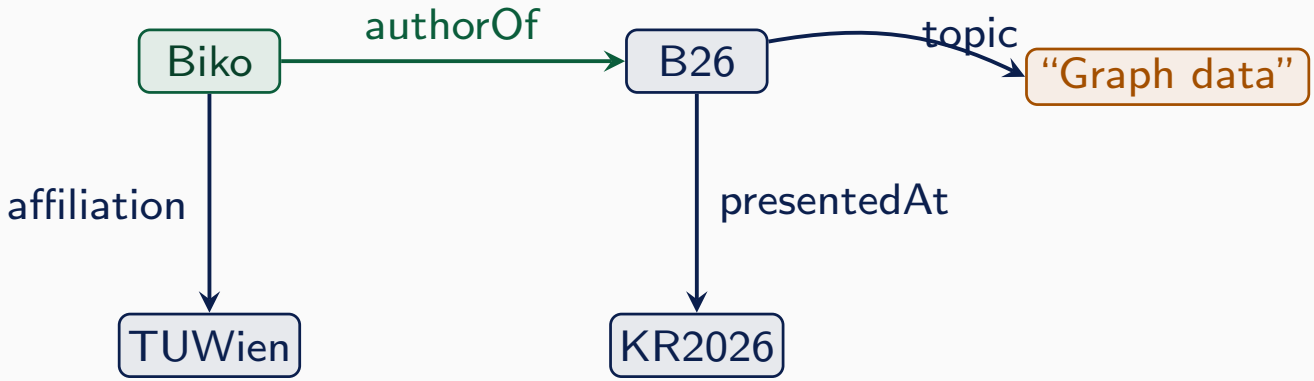
Resource Description Framework (RDF) triple stores

RDF: open standard for sharing data on the web

- format of knowledge graphs and open data repositories
- labelled graphs as **triples** (s, p, o)

  (Biko, authorOf, B26) (Biko, affiliation, TUWien)

  (B26, presentedAt, KR2026) (B26, topic, "Graph data")
- SPARQL query language, access via (web) endpoints

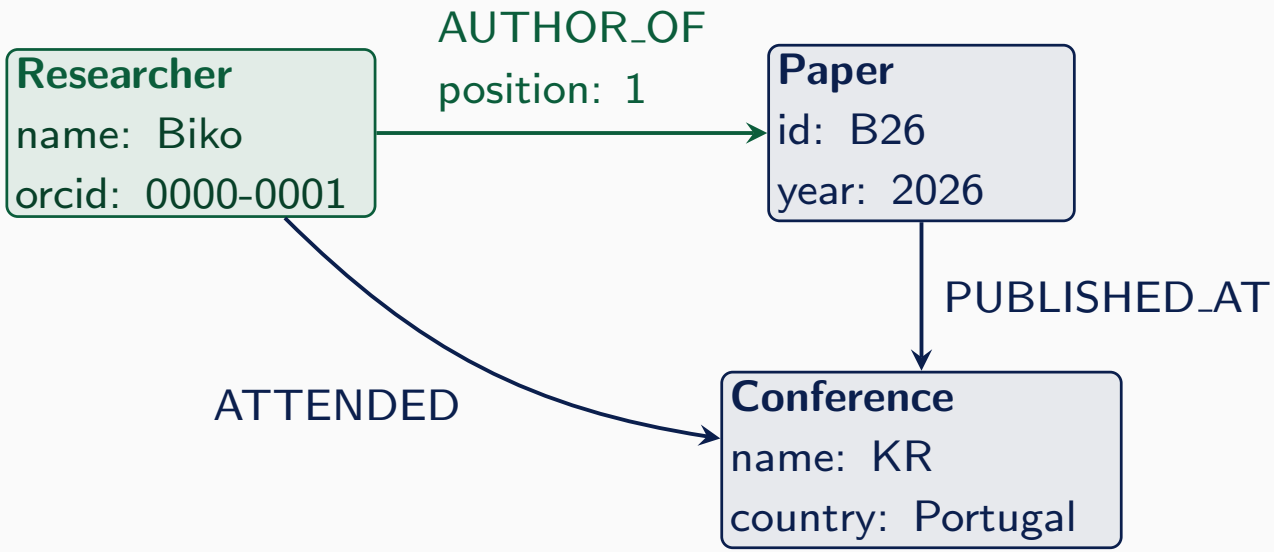


W3C, 2014, *RDF 1.1 Concepts and Abstract Syntax*, W3C.

Property Graphs

Property graphs (PGs): data model of graph databases

- Labelled graph with property values in nodes and edges
- Broad industrial adoption
- A zoo of proprietary engines and query languages

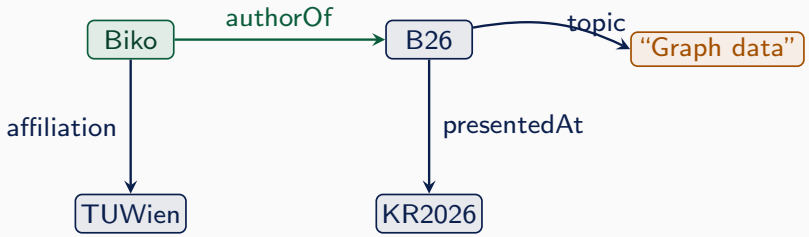


Gutierrez, Hidders & Wood, 2018, *Graph Data Models*, ACM CSUR.

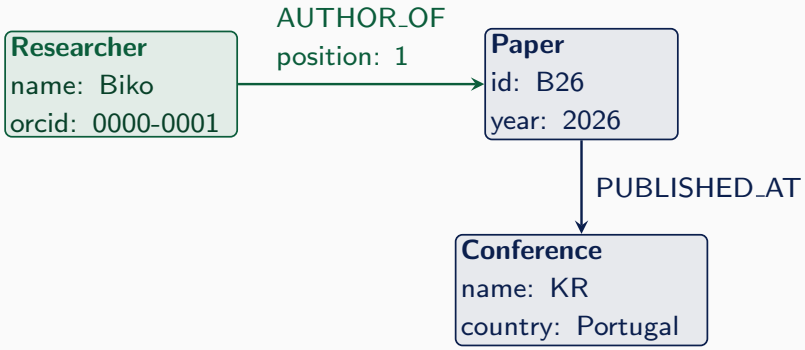
New standards for Graph Data

Different models with **a large common core**

RDF graph



Property graph



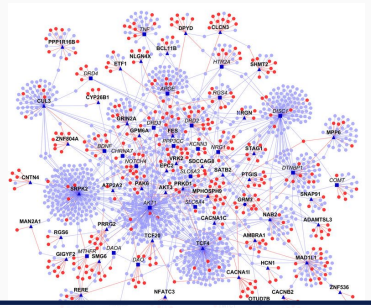
Recent standardisation efforts

aim at narrowing the gap.

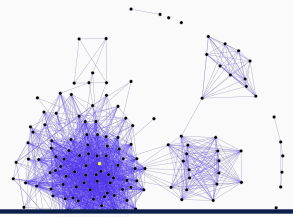
Description Logics: KR for Graphs

Description Logics (DLs) are logics for describing graphs

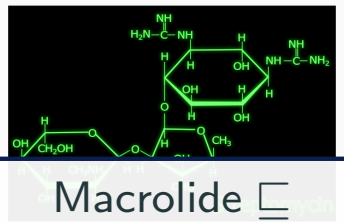
$\text{Tram} \sqsubseteq \forall \text{stopsAt.Station}$



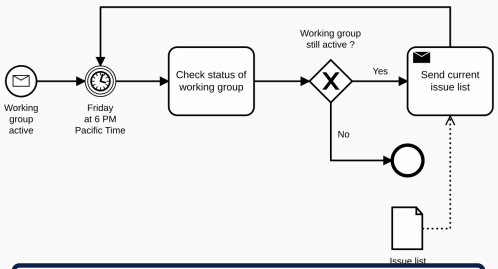
$\text{Complex} \sqsubseteq \exists \text{hasPart.Protein}$



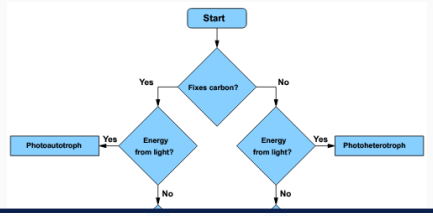
$\text{Researcher} \sqsubseteq \exists \text{collaborates.Researcher}$



$\text{Macrolide} \sqsubseteq \exists \text{hasPart.LactoneRing}$



$\text{Approved} \sqsubseteq \exists \text{hasStep.Review}$



$\text{Decision} \sqsubseteq \forall \text{next.}(\text{Decision} \sqcup \text{Action})$

DLs: a flexible toolbox of decidable logics

	NL	P	ExpTime	
	DL-Lite	$\mathcal{EL}$	$\mathcal{ALCI}$	$\mathcal{SHIQ}$
$A \sqsubseteq B$	✓	✓	✓	✓
$A \sqsubseteq B \sqcup C$			✓	✓
$A \sqsubseteq \exists r.B$	✓	✓	✓	✓
$A \sqcap \exists r.B \sqsubseteq C$		✓	✓	✓

- well-charted complexity vs. expressiveness trade-off
- highly optimised off-the-shelf reasoners for many DLs
- plethora of services: explanations, repairs, updates, refinements, forgetting, module extraction, ...

Baader, Horrocks, Lutz & Sattler, 2017, *An Introduction to Description Logic*, CUP.

KR and DLs for Graph Data

DLs have been a powerful KR tool in data management:

- To describe background knowledge and infer implicit facts
- As constraint languages describing requirements on datasets
- As query languages
- As foundations for ontology-mediated data access

Today's Talk: DLs are ready to face new challenges!

Part 1: revisiting an old problem

Can we finally have true ontology-mediated querying?

Part 2: addressing new ones

KR as foundations for new graph constraint languages

Schneider & Šimkus, 2020, *Ontologies and Data Management: A Brief Survey*, KI.

Part 1: revisiting an old problem

Ontology-Mediated Query Answering

An ontology-mediated query (OMQ)

is a pair $(\mathsf{q}, \mathcal{O})$: a database query q together with a DL ontology $\mathcal{O}$ providing background knowledge.

OMQ $(\mathsf{q}, \mathcal{O})$

$$\mathsf{q}(x) = \mathsf{Paper}(x)$$

$$\begin{array}{rcl} \exists\, \mathsf{authored}^{-}.\top & \sqsubseteq & \mathsf{Paper} \\ \exists\, \mathsf{publishedIn}.\top & \sqsubseteq & \mathsf{Paper} \end{array}$$

Data $\mathcal{D}$

$$\begin{array}{l} \mathsf{authored}(\mathsf{An}, \mathsf{ANV26}) \\ \mathsf{authored}(\mathsf{Biko}, \mathsf{ABK23}) \\ \mathsf{publishedIn}(\mathsf{KR24}, \mathsf{AIJ}) \\ \vdots \end{array}$$

$$\mathsf{Ans}_{\mathcal{O}}(\mathsf{q}, \mathcal{D}) \;=\; \{\mathsf{ANV26}, \mathsf{ABK23}, \mathsf{KR24}\}$$

- infer missing facts from incomplete data
- define high-level vocabulary for better query formulation
- unified access to heterogenous data . . .

Bienvenu, 2016, *Ontology-Mediated Query Answering: Harnessing Knowledge to Get More from Data*, IJCAI.

Query Rewriting Makes OBDA Possible

OMQs are the pillar of virtual knowledge graphs (VKGs), one of the biggest DL success stories of the last decade.

OMQ rewriting

Compile $(\mathsf{q}, \mathcal{O})$ into a *pure* database query $\mathsf{q}^{\mathcal{O}}$ that gives the same answers over every database.

$$\mathsf{q}(x) = \mathsf{Paper}(x),$$
$$\{\exists \mathsf{authored}^{-}.\top \sqsubseteq \mathsf{Paper},$$
$$\exists \mathsf{publishedIn}.\top \sqsubseteq \mathsf{Paper}\}$$

$$\mathsf{q}^{\mathcal{O}}(x) = \mathsf{Paper}(x)$$
$$\vee\, \exists y\, \mathsf{authored}(y, x)$$
$$\vee\, \exists y\, \mathsf{publishedIn}(x, y)$$

Key: $\mathsf{q}^{\mathcal{O}}$ can be evaluated with an ordinary database engine

Xiao et al., 2018, *Ontology-Based Data Access: A Survey*, IJCAI.
Calvanese et al., 2007, *The DL-Lite Family*, JAR.

For two decades, we have been stuck with un-graphy OMQs

Relational-like queries

joins, but no graph navigation:

$$r(x, y) \wedge r(x, z) \wedge s(z, y).$$

No reachability $r^*(x, y)$.

DL-Lite ontologies

Deliberately weak ontology language:

$$B \sqsubseteq C \qquad B \sqsubseteq \exists r \qquad \exists r \sqsubseteq B$$

Local, blind to graph structure.

Every graph query language has some form of **graph pattern matching** using **path expressions**

Papers affected by a retraction $\quad q(x) = \langle \text{cites}^+ \cdot \text{Retrd}? \rangle ?(x, y)$

- A **2-way regular path query (2RPQ)** is an atom $E(x, y)$ with E a regular expression over (possibly inverted) edge labels.
- Conjunctive 2RPQs join 2RPQs through shared variables

$$q(\vec{x}) = \exists \vec{y} \bigwedge E_i(u_i, v_i)$$

Angles, Arenas, Barceló, Hogan, Reutter & Vrgoč, 2017, *Foundations of Modern Query Languages for Graph Databases*, ACM CSUR.

The big barrier

Navigational OMQs are not rewritable into C2RPQs

$$\mathrm{q}(x, y) = (r \cdot A?)^{+}(x, y), \qquad \mathcal{O} = \{\exists s.\top \sqsubseteq A\}.$$

A regular path cannot leave the path to check s-edge and return.

Nested paths let us branch off tests without leaving the main path.

$$\mathrm{q}^{\mathcal{O}}(x, y) = \big(r \cdot (A? + \langle s \rangle?)\big)^{+}(x, y).$$

- OMQs with *nested regular expressions* studied a decade ago
- but graph database engines did not support them!

Löhnert, Augstein, Okulmus & O., 2025, *Query Rewriting for Nested Navigational Queries over Property Graphs*, ESWC.
Bienvenu, Calvanese, O. & Šimkus, 2014, *Nested Regular Path Queries in Description Logics*, KR.

The barrier is finally overcome!

The New Standards Supply the Missing Expressiveness

GQL: new standard for querying PGs (ISO/IEC 39075:2024)

- commercial engines are quickly moving towards GQL support
- SQL/PGQ (SQL:2023) defines graph-pattern queries over property-graph views of relational data

Their graph-pattern core supports nested path navigation

- the target language needed for DL-Lite rewritings now exists!
- Ten-year-old techniques are finally implementable.
- Ongoing work on making the techniques more goal-oriented and the rewritings more succinct.
- First prototype compiling OWL QL + CN2RPQ into Cypher.

Francis et al., 2023, *A Researcher's Digest of GQL*, ICDT.
Löhnert, Augstein, Okulmus & O., 2025, *Query Rewriting for Nested Navigational Queries over Property Graphs*, ESWC.

What About Going Beyond DL-Lite?

DL-Lite is too poor for describing graphs, e.g., it cannot propagate labels to/from neighbouring nodes.

$$\exists r.\text{Red} \sqsubseteq \text{Blue} \qquad \text{Blue} \sqsubseteq \forall r.\text{Red}$$

- DL-Lite reasoning is in AC_0.
- Graph query evaluation is NL-complete in data complexity.

Can we spend the additional complexity available in graph query languages on a more expressive ontology language?

Unfortunately, the answer is not in the DL toolbox

$\mathcal{EL}$: conjunction simulates monotone circuits

$$\begin{aligned} \mathsf{And} \sqcap \exists \mathsf{left}.\mathsf{True} \sqcap \exists \mathsf{right}.\mathsf{True} &\sqsubseteq \mathsf{True}, \\ \mathsf{Or} \sqcap \exists \mathsf{left}.\mathsf{True} &\sqsubseteq \mathsf{True}, \\ \mathsf{Or} \sqcap \exists \mathsf{right}.\mathsf{True} &\sqsubseteq \mathsf{True}. \end{aligned}$$

$\mathcal{ELI}^{lin}$: inverses recover conjunction

$$A \sqcap B \sqsubseteq C \quad \rightsquigarrow \quad \begin{cases} A \sqsubseteq \exists r.\top, \\ \exists r^{-}.B \sqsubseteq D, \\ \exists r.D \sqsubseteq C. \end{cases}$$

Entailment is P-hard in data complexity.

Calvanese et al., 2007, *Tractable Reasoning and Efficient Query Answering in Description Logics: The DL-Lite Family*, JAR.

Can We Recover a Rewritable Fragment?

$\mathcal{ELHI}$ provides the graph features that DL-Lite lacks:

- conjunction of node labels,
- existential restrictions on either side of inclusions,
- inverse roles and role inclusions.

Keep these constructors, but stratify their recursive interaction so that ontology reasoning remains navigational.

$$\mathcal{ELHI}_{\preceq}$$

Stratifying Conjunctions

The fragment $\mathcal{ELHI}_{\preceq}$

An ontology $\mathcal{O}$ belongs to $\mathcal{ELHI}_{\preceq}$ if its axioms have the form

$$A \sqsubseteq B, \quad A \sqcap B \sqsubseteq C, \quad A \sqsubseteq \exists s.B, \quad \exists s.A \sqsubseteq B, \quad r \sqsubseteq s.$$

Stratification is witnessed by a preorder $\preceq$:

- predicates on the left are no larger than the predicate on the right;
- the comparison is strict for all but at most one left-hand predicate;
- inverse roles occupy the same stratum.

Recursion is allowed; recursive conjunction is controlled.

Stratification Brings Reasoning Back into NL

Query rewriting

For every $\mathcal{ELHI}_{\preceq}$-ontology $\mathcal{O}$ and N2RPQ q, there is an UCN2RPQ q' such that

$$(\mathcal{D}, \mathcal{O}) \models q(a) \quad \Longleftrightarrow \quad \mathcal{D} \models q'(a).$$

- A nested automaton simulates derivations with bounded alternation.
- Each atomic query compiles into a nested path query.

For $\mathcal{ELHI}_{\preceq}$, CN2RPQs rewrite into equivalent CN2RPQs and can be evaluated directly over the graph data.

Arpasi, Bednarczyk & O., submitted, *Towards Practicable Algorithms for Rewriting Graph Queries Beyond DL-Lite*.

The Fragment Covers Most Tested $\mathcal{ELHI}$ Ontologies

Preliminary check on the ORE 2015 corpus:

$455/509 = 89.4\%$ of the $\mathcal{ELHI}$ ontologies admit an $\mathcal{ELHI}_{\preceq}$ stratification.

Natural recursive propagation remains available

$$\exists \text{partOf}.\text{Contaminated} \sqsubseteq \text{Contaminated}.$$

Mutually recursive conjunction is excluded

$$\exists \text{contacted}.\text{Infected} \sqcap \exists \text{coResident}.\text{Infected} \sqsubseteq \text{Infected}.$$

We can finally have ontology-mediated graph querying.

Part 2: addressing new ones

SHACL: A Short History

- Web data stores on the web are enormous
- They are interconnected and usually open
- RDF is schema-less: data can be heterogeneous and messy

$(\text{P1}, \text{author}, \text{An})$ $(\text{An}, \text{authorOf}, \text{P1})$ $(\text{P1}, \text{dc:creator}, \text{An})$

We need a way to **describe and validate** datasets!

- W3C RDF Data Shapes Working (2014): let's standardize a language for structural constraints on RDF graphs!
- The Shapes Constraint language **SHACL** becomes a **W3C Recommendation in 2017**.

SHACL Shapes

Shape expressions (Logic syntax)

$$\varphi ::= c \mid A \mid \varphi \mid \neg\varphi \mid \varphi \wedge \varphi \mid \exists_{\geq n} E.\varphi \mid \mathsf{eq}(E, r) \mid \mathsf{disj}(E, r)$$

where c is a node name (constant),
A is a class name (unary predicate),
r is a relation name (binary predicate), and
E is a **regular expression** over roles and their inverses.

$\varphi \vee \varphi$, $\exists E.\varphi$, and $\forall E.\varphi$ expressible as usual

Shapes are **concepts** in a very **expressive description logic**

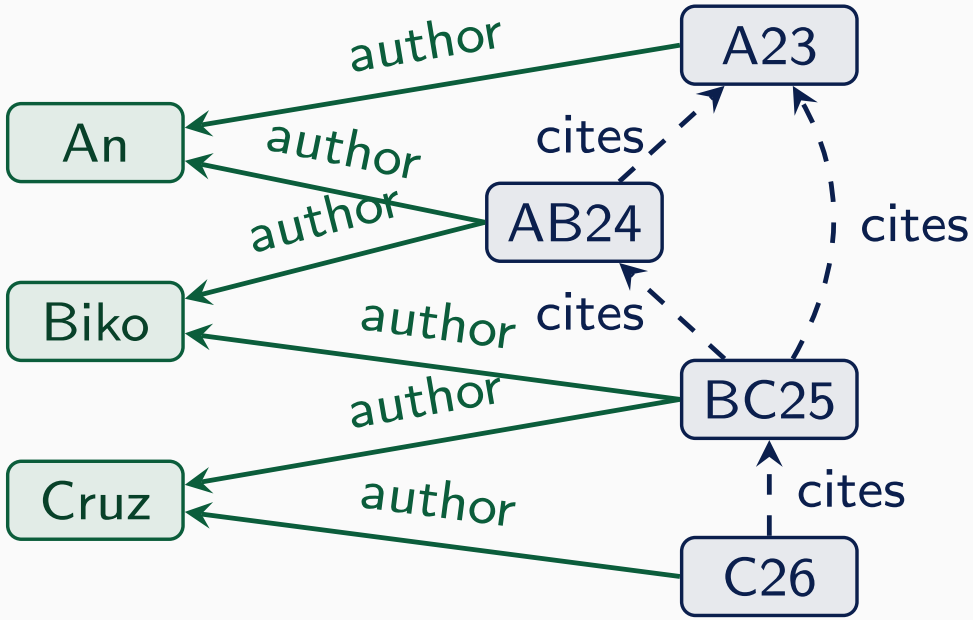
O., 2023, *A Short Introduction to SHACL for Logicians*, WoLLIC.
Bogaerts, Jakubowski & Van den Bussche, 2022, *SHACL: A Description Logic in Disguise*, LPNMR.

Semantics of SHACL Shapes: an example

Semantics as expected

$[\![\varphi]\!]^G$ denotes the **set of nodes** in G that are in the extension of φ

$\varphi = \exists_{\geq 2}\text{author}$



$$[\![\varphi]\!]^G = \{\text{AB24}, \text{BC25}\}$$

SHACL Validation

Shape Validation

given a graph G, a **target node** n, and a shape φ,
decide whether n validates φ in G $\quad n \in [\![\varphi]\!]^G$

$\varphi =$ $\exists\, \text{authorOf}.\text{Paper}$

Target: Biko

$\text{Biko} \in [\![\varphi]\!]^G$ ✓

Validation of a shape is **model checking** over a **given graph**

- No open-world, no completion
- An **interpretation**, not an ABox

SHACL Documents

1. **Catalogue** of **shape definitions**

$$\mathtt{S}_1 \equiv \varphi_1 \quad \cdots \quad \mathtt{S}_n \equiv \varphi_n$$

   - defined shape names $\mathtt{S}_1, \ldots \mathtt{S}_n$
   - each $\mathtt{S}_i$ has a definition φ_i
   - shape expressions φ can mention **shape names** $\mathtt{S}_1, \ldots, \mathtt{S}_n$

$$\mathtt{PAINTER} \equiv \exists\, \mathsf{creatorOf}.\mathtt{PAINTING}$$

$$\mathtt{PAINTING} \equiv \exists\, \mathsf{creatorOf}^{-}.\mathtt{PAINTER}$$

2. **Targets** pairs $(n, \mathtt{S})$ for which we want to validate $n \in [\![\mathtt{S}]\!]^G$
   - more complex targets allowed
     e.g. *class targets*: validate $\mathtt{S}$ for all objects in class C,

From Model Checking to Model Finding

$$\mathsf{S}_1 \equiv \varphi_1 \quad \cdots \quad \mathsf{S}_n \equiv \varphi_n,$$

Solution (aka shape assignment, model)

node labellings S that satisfy all shape definitions: $[\![\mathsf{S}_i]\!]^G_S = [\![\varphi_i]\!]^G_S$

$$\texttt{PAINTER} \equiv \exists\, \mathsf{creatorOf}.\texttt{PAINTING}$$
$$\texttt{PAINTING} \equiv \exists\, \mathsf{creatorOf}^-.\texttt{PAINTER}$$

Solutions:

$$[\![\texttt{PAINTER}]\!]^G_{S_1} = \{\}$$

$$[\![\texttt{PAINTING}]\!]^G_{S_1} = \{\}$$

$$[\![\texttt{PAINTER}]\!]^G_{S_2} = \{\mathsf{Biko}\}$$

$$[\![\texttt{PAINTING}]\!]^G_{S_2} = \{\mathsf{KR25}\}$$

Semantics of Recursive SHACL

Not all models are equally intuitive. . .

moreover, **validation becomes intractable!**

Maybe we need some form of minimality?

Positive SHACL

- If the defined shapes occur positively, there's a unique minimal model derivable by a fixed point computation

Close to monadic regular Datalog and modal substitution calculus.

But in general, no unique obvious semantics

$$\texttt{PAINTER} \equiv \text{Artist} \wedge \neg\texttt{WRITER} \qquad \texttt{WRITER} \equiv \text{Artist} \wedge \neg\texttt{PAINTER}$$

Corman, Reutter & Savković, 2018, *Semantics and Validation of Recursive SHACL*, ISWC.

Recursion in the Specification

According to `https://www.w3.org/TR/shacl/`:

W3C Recommendation

A shape `s1` in an RDF graph `G` ***refers*** to shape `s2` in `G` if it has `s2` as value for some non-list-taking, shape-expecting parameter of some constraint component or `s2` as a member of the value for some list-taking, shape-expecting parameter of some constraint component. A shape in an RDF graph `G` is a ***recursive shape*** in `G` if it is related to itself by the transitive closure of the refers relationship in `G`.

The validation with recursive shapes is not defined in SHACL and is left to SHACL processor implementations. For example, SHACL processors may support recursion scenarios or produce a failure when they detect recursion.

The expectation is that future work . . . will lead to the definition of specific dialects of SHACL where recursion is well-defined.

KR to the Rescue!

Stratified SHACL

- Stratified negation only over atoms defined in lower strata.
- One natural perfect model computable in polynomial time.

Full recursive SHACL: three choices

- **Supported**: accept all solutions. Intractable!
- **Stable**: well-known from Logic Programming.
  Intuitively, only choices where everything is justified.
  Hard to compute, but tractable for stratified definitions.
- **Well-founded**: unique three-valued model.
  Polynomial approximation of the stable semantics.
  Well-suited for SHACL validation.

Andreșel et al., 2020, *Stable Model Semantics for Recursive SHACL*, WWW.
Okulmus & Šimkus, 2024, *SHACL Validation under the Well-Founded Semantics*, KR.

KR & Foundations of SHACL Reasoning: selected contributions

With the KR toolbox, we have managed to address a few SHACL challenges:

- Validation of incomplete graphs: combining SHACL+ontologies
- Static analysis: SHACL satisfiability and implication
- Validation preservation in evolving graphs
- Learning SHACL from data

Validating SHACL in the Presence of Ontologies

- **SHACL**: closed-world validation
- **Ontologies**: implicit facts, open-world view.

Goal: validate incomplete graphs with ontology consequences.

$$\text{Biko} \in [\![\texttt{AUTHOR}]\!]^{\text{cl}_{\mathcal{O}}(G)}$$

Ontology axiom

$$\text{firstAuthorOf} \sqsubseteq \text{authorOf}$$

Shape and target

$$\texttt{AUTHOR} \equiv \exists\, \text{authorOf}.\top$$

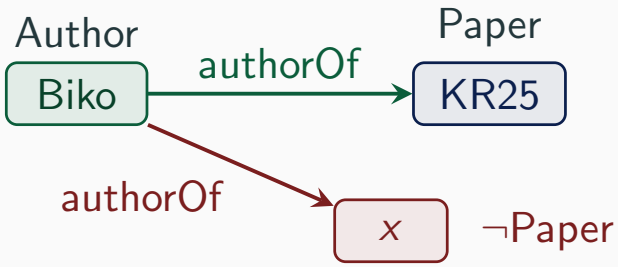
target: Biko

Ahmetaj, O., Oudshoorn & Šimkus, 2023, *Reconciling SHACL and Ontologies: Semantics and Validation via Rewriting*, ECAI.

From Universal to Core Models

Data graph

Skolem chase



Austere model

Ontology axiom

$$\text{Author} \sqsubseteq \exists\, \text{authorOf}.\top$$

Shape

$$\texttt{ONLYPAPERS} \equiv \forall\, \text{authorOf}.\text{Paper}$$

$$\text{Biko} \notin [\![\texttt{ONLYPAPERS}]\!]^{G_{\text{sk}}} \quad \times$$

$$\text{Biko} \in [\![\texttt{ONLYPAPERS}]\!]^{G} \quad \checkmark$$

We need a core that adds only genuinely needed successors.

Ontology-Aware Validation: Contributions

- Semantics based on validation over **austere canonical model**
  - prove that it is a **core**
  - construction does not require repeated core tests
- Rewriting of SHACL plus an $\mathcal{ELHI}$ TBox into plain SHACL
  - ontology consequences compiled into SHACL constraints
  - output checkable by ordinary validators
- Validation is $\mathrm{ExpTime}$-complete combined complexity.
- Validation is PTime-complete data complexity.
- Preliminary ideas towards a practical rewriting: data-aware rewriting builds a compact core representation

Oudshoorn, O. & Šimkus, 2026, *SHACL Validation in the Presence of Ontologies: Semantics and Rewriting Techniques*, AIJ.

SHACL Satisfiability and Implication

Static analysis questions

- Is a SHACL shape satisfied in some graph?
- Are two SHACL shapes equivalent?
  That is, they have the same extension on every graph
- Does one shape imply another?
- Over arbitrary and over finite graphs

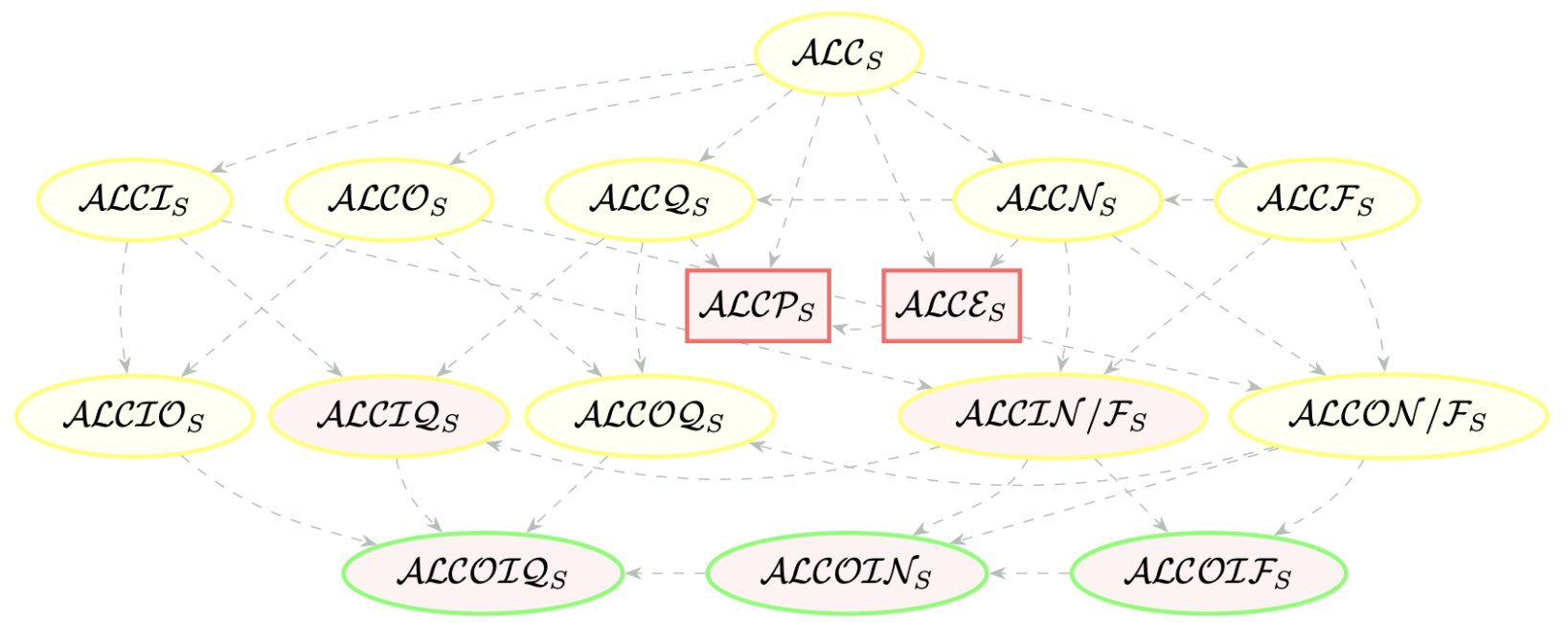
Transfer results from the DL literature! (Oudshoorn 2025)

- Many core fragments inherit $\mathrm{ExpTime}$ bounds from DLs.
- Richer counting fragments reach $\mathrm{NExpTime}$ or undecidability.
- Finite satisfiability is more fragile: several fragments lack the finite model property.

Oudshoorn, 2025, *SHACL Satisfiability: What Can We Learn from DLs?*, DL.

SHACL Satisfiability: Complexity Landscape

Using DL syntax: (Oudshoorn 2025)



Ellipses: decidable; squares: undecidable.

Border color distinguishes $\mathrm{ExpTime}$ from $\mathrm{NExpTime}$.

Yellow fill indicates means finite model property.

Static Analysis of SHACL Documents

Document satisfiability and implication (definitions+targets)

Given documents (C_1, T_1) and (C_2, T_2), decide whether every graph validating (C_1, T_1) also validates (C_2, T_2).

Main results

- For supported and stable semantics: **undecidable**
  - stratification regains decidability in the stable case
  - but not in the supported case
- For the well-founded semantics: $\mathrm{ExpTime}$-**complete**.

Upper bound

Translation of well-founded SHACL into full hybrid μ-calculus

Don't miss Anouk's talk tomorrow 11:25! (B1.03)

Oudshoorn, O. & Šimkus, 2026, *Static Analysis of Recursive SHACL*, KR.

SHACL Validation Under Graph Updates

Static validation of RDF graph updates

Given a (sequential) update $\mathcal{U}$ and a SHACL specification $\mathcal{C}$, does **every graph** that **satisfying** $\mathcal{C}$ still satisfy it **after applying** $\mathcal{U}$?

- RDF graphs evolve; avoid failure after updates.
- The paper defines a SHACL-based update language.
- Adds or removes labels based on shapes.

Main technique: regression

- update effects are compiled into SHACL constraints $tr^{\mathcal{U}}(\mathcal{C})$.
- Static validation reduces to satisfiability/containment.
- Undecidable in general, decidable fragments identified.

Ahmetaj et al., 2025, *SHACL Validation under Graph Updates*, ISWC.

Learning Shapes from Examples

Considered setting

Fitting SHACL shapes from positive and negative example nodes in a graph, possibly using shape names from a recursive catalogue.

- Find a shape in a core SHACL fragment (essentially $\mathcal{ELI}^*$).
- All semantics considered: well-founded, stable, and supported.
- Fitting existence is $\mathrm{ExpTime}$-complete for recursive SHACL under all three semantics.
- Most-specific fitting existence is in $\mathrm{ExpTime}$.
- When validation is tractable, polynomial for bounded positive examples.

Gortworst, Okulmus, Turhan & O., 2025, *Learning Shapes from Examples*, ISWC.

That's it for today

- Many open problems remain in the SHACL world
- Bridging the gap with other forms of graph reasoning and learning offers exciting perspectives!

Reliable reasoning from knowledge and data
still poses many challenges to KR.

- As new formalisms emerge, we face new challenges
- and reencounter old challenges in new forms,
- but we also find new opportunities to overcome previous limitations.