

Nils Küchenmeister, Alex Ivliev, Dörthe Arndt, Markus Krötzsch
TU Dresden, Knowledge-Based Systems Group

# Stratified Negation for RDF Rules
## A Correct Approach

The 25th International Semantic Web Conference // Bari, 28 October 2026

# Rules for RDF knowledge graphs

- Rules match patterns in an RDF graph and add new triples.

$R_e$ `{?c :teacher ?p.  not ?p rdf:type :Student} => {?c :examiner ?p}.`

$R_b$ `{?c :participant ?p} => {?c :examiner _:}.`

# Rules for RDF knowledge graphs

- Rules match patterns in an RDF graph and add new triples.

test absence

$R_e$ {?c :teacher ?p. not ?p rdf:type :Student} => {?c :examiner ?p}. **default negation**

$R_b$ {?c :participant ?p} => {?c :examiner _:}.

# Rules for RDF knowledge graphs

- Rules match patterns in an RDF graph and add new triples.

$R_e$ `{?c :teacher ?p. not ?p rdf:type :Student} => {?c :examiner ?p}.` **default negation**

$R_b$ `{?c :participant ?p} => {?c :examiner _:}.` **value invention**

blank node requires existence

# Rules for RDF knowledge graphs

- Rules match patterns in an RDF graph and add new triples.

$R_e$ `{?c :teacher ?p. not ?p rdf:type :Student} => {?c :examiner ?p}.` **default negation**

$R_b$ `{?c :participant ?p} => {?c :examiner _:}.` **value invention**

$R_7$ `{?p rdfs:subPropertyOf ?q. ?x ?p ?y} => {?x ?q ?y}.`

$R_5$ `{?p rdfs:subPropertyOf ?q. ?q rdfs:subPropertyOf ?r} => {?p rdfs:subPropertyOf ?r}.`

**RDFS entailment rules** (rdfs7 and rdfs5)

- Rules can also express ontological relationships.

# Rules for RDF knowledge graphs

- Rules match patterns in an RDF graph and add new triples.

$R_e$ `{?c :teacher ?p. not ?p rdf:type :Student} => {?c :examiner ?p}.` **default negation**

$R_b$ `{?c :participant ?p} => {?c :examiner _:}.` **value invention**

$R_7$ `{?p rdfs:subPropertyOf ?q. ?x ?p ?y} => {?x ?q ?y}.`

$R_5$ `{?p rdfs:subPropertyOf ?q. ?q rdfs:subPropertyOf ?r} => {?p rdfs:subPropertyOf ?r}.`

**RDFS entailment rules** (rdfs7 and rdfs5)

- Rules can also express ontological relationships.

Active development: **N3**, **SPARQL RL**, **SHACL Inference Rules**

# Rules for RDF knowledge graphs

- Rules match patterns in an RDF graph and add new triples.

$R_e$ `{?c :teacher ?p. not ?p rdf:type :Student} => {?c :examiner ?p}.` **default negation**

$R_b$ `{?c :participant ?p} => {?c :examiner _:}.` **value invention**

$R_7$ `{?p rdfs:subPropertyOf ?q. ?x ?p ?y} => {?x ?q ?y}.`

$R_5$ `{?p rdfs:subPropertyOf ?q. ?q rdfs:subPropertyOf ?r} => {?p rdfs:subPropertyOf ?r}.`

**RDFS entailment rules** (rdfs7 and rdfs5)

- Rules can also express ontological relationships.

Active development: **N3**, **SPARQL RL**, **SHACL Inference Rules**

⇝ Combining these features is hard. Especially in RDF!

# Example: Teachers, students, and examiners

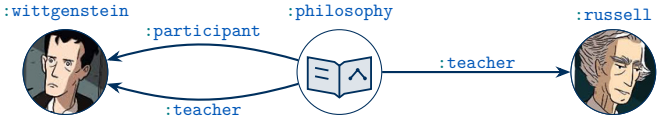
$R_s$ {?c :participant ?p} => {?p rdf:type :Student}.

$R_e$ {?c :teacher ?p. not ?p rdf:type :Student} => {?c :examiner ?p}.

$R_b$ {?c :participant ?p} => {?c :examiner _:}.

$R_7$ {?p rdfs:subPropertyOf ?q. ?x ?p ?y} => {?x ?q ?y}.

$R_5$ {?p rdfs:subPropertyOf ?q. ?q rdfs:subPropertyOf ?r} => {?p rdfs:subPropertyOf ?r}.



# Example: Teachers, students, and examiners

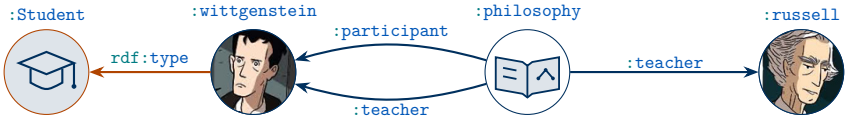
$R_s$ {?c :participant ?p} => {?p rdf:type :Student}.

$R_e$ {?c :teacher ?p. not ?p rdf:type :Student} => {?c :examiner ?p}.

$R_b$ {?c :participant ?p} => {?c :examiner _:}.

$R_7$ {?p rdfs:subPropertyOf ?q. ?x ?p ?y} => {?x ?q ?y}.

$R_5$ {?p rdfs:subPropertyOf ?q. ?q rdfs:subPropertyOf ?r} => {?p rdfs:subPropertyOf ?r}.




$R_s \prec^{-} R_e$ **Negative reliance:** Student inference blocks the examiner test.

# Example: Teachers, students, and examiners

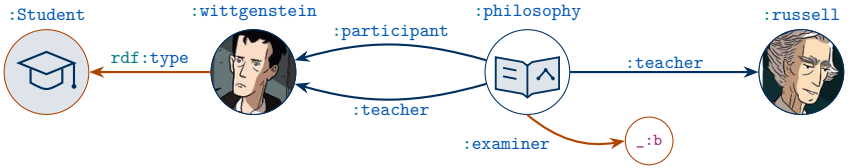
$R_s$ {?c :participant ?p} => {?p rdf:type :Student}.

$R_e$ {?c :teacher ?p. not ?p rdf:type :Student} => {?c :examiner ?p}.

$R_b$ {?c :participant ?p} => {?c :examiner _:}.

$R_7$ {?p rdfs:subPropertyOf ?q. ?x ?p ?y} => {?x ?q ?y}.

$R_5$ {?p rdfs:subPropertyOf ?q. ?q rdfs:subPropertyOf ?r} => {?p rdfs:subPropertyOf ?r}.




$R_b$ Introduce a fresh examiner blank node.

# Example: Teachers, students, and examiners

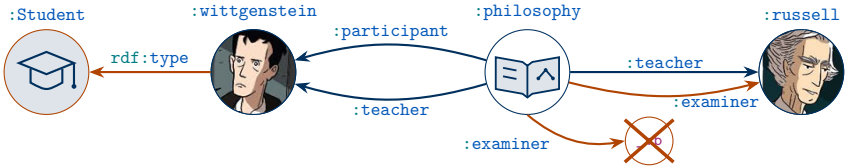
$R_s$ {?c :participant ?p} => {?p rdf:type :Student}.

$R_e$ {?c :teacher ?p. not ?p rdf:type :Student} => {?c :examiner ?p}.

$R_b$ {?c :participant ?p} => {?c :examiner _:}.

$R_7$ {?p rdfs:subPropertyOf ?q. ?x ?p ?y} => {?x ?q ?y}.

$R_5$ {?p rdfs:subPropertyOf ?q. ?q rdfs:subPropertyOf ?r} => {?p rdfs:subPropertyOf ?r}.

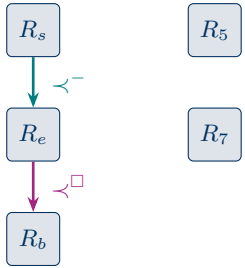



$R_e \prec^{\square} R_b$ **Restraint:** the named examiner makes the blank node redundant.

# Stratification: Order rule applications

$\prec^{-}$ can invalidate a negative test.

$\prec^{\square}$ can make an invented value redundant.




Apply rules in an order that keeps tests valid and avoids redundant values.

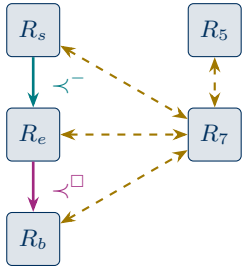
[Krötzsch, KR, 2020]

# Stratification: Order rule applications

$\prec^-$ can invalidate a negative test.

$\prec^\square$ can make an invented value redundant.

$\prec^+$ can enable a new match.



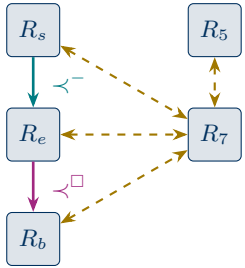

Apply rules in an order that keeps tests valid and avoids redundant values.

[Krötzsch, KR, 2020]

# Stratification: Order rule applications

$\prec^-$ can invalidate a negative test.

$\prec^\square$ can make an invented value redundant.




$\prec^+$ can enable a new match.

Apply rules in an order that keeps tests valid and avoids redundant values.

$(\prec^+)^\star$ positive paths can carry an effect across several rules.

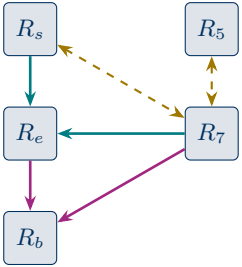
[Krötzsch, KR, 2020]

# Stratification: Order rule applications

$\prec^-$ can invalidate a negative test.

$\prec^\square$ can make an invented value redundant.

$\prec^+$ can enable a new match.




Apply rules in an order that keeps tests valid and avoids redundant values.

$(\prec^+)^\star$ positive paths can carry an effect across several rules.

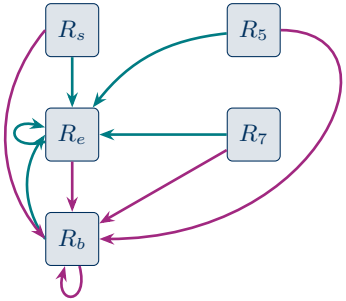
[Krötzsch, KR, 2020]

# Stratification: Order rule applications

$\prec^-$ can invalidate a negative test.

$\prec^\square$ can make an invented value redundant.

$\prec^+$ can enable a new match.




Apply rules in an order that keeps tests valid and avoids redundant values.

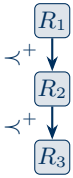
$(\prec^+)^\star$ positive paths can carry an effect across several rules.

**The required application order is cyclic.** This does not work.

[Krötzsch, KR, 2020]

# Idea 1: Trails track multi-step rule interactions

Can we do better than $(\prec^+)^\star$?



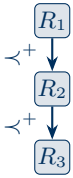

```
{?c :participant ?p.  ?p rdf:type :Student} => {?c rdf:type :Course}.

{?c rdf:type :Course} => {?c rdf:type :Module}.

{?c rdf:type :Module} => {?c :participant _:p.  _:p rdf:type :Student}.
```

# Idea 1: Trails track multi-step rule interactions

Can we do better than $(\prec^+)^*$?



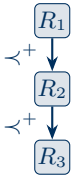

```
{?c :participant ?p.  ?p rdf:type :Student} => {?c rdf:type :Course}.
{?c rdf:type :Course} => {?c rdf:type :Module}.
{?c rdf:type :Module} => {?c :participant _:p.  _:p rdf:type :Student}.
```

Existing participant already fills the role: $\texttt{_:p} \mapsto \texttt{?p}$

**No new blank node is needed.** The final application is blocked.

# Idea 1: Trails track multi-step rule interactions

Can we do better than $(\prec^+)^\star$?




$R_1$: {?c :participant ?p. ?p rdf:type :Student} => {?c rdf:type :Course}.

$R_2$: {?c rdf:type :Course} => {?c rdf:type :Module}.

$R_3$: {?c rdf:type :Module} => {?c :participant _:p. _:p rdf:type :Student}.

Existing participant already fills the role: $\texttt{_:p} \mapsto \texttt{?p}$

**No new blank node is needed.** The final application is blocked.

A positive path need not describe an actual sequence of applications.

$\rightsquigarrow$ **Trails** track actual multi-step interactions.

# The problem & our solution

**Trails:** deciding whether two rules are connected by such applications is **undecidable**.

# The problem & our solution

**Trails:** deciding whether two rules are connected by such applications is **undecidable**.

1. **Chains** approximate trails with a decidable criterion.

# The problem & our solution

**Trails:** deciding whether two rules are connected by such applications is **undecidable**.

1. **Chains** approximate trails with a decidable criterion.

2. Chains can still be **arbitrarily long**.
   A **regular language** characterises them and enables a decision procedure.

# The problem & our solution

**Trails:** deciding whether two rules are connected by such applications is **undecidable**.

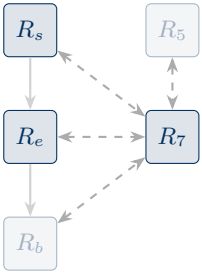
1. **Chains** approximate trails with a decidable criterion.

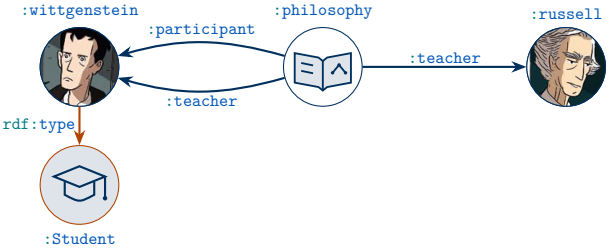
2. Chains can still be **arbitrarily long**.
   A **regular language** characterises them and enables a decision procedure.

3. **Chain stratification:** prototype implementation available.

# Taking stock: This feedback is real

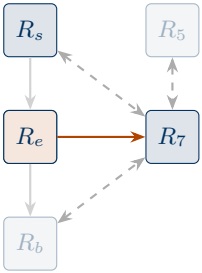
`:examiner rdfs:subPropertyOf :participant`

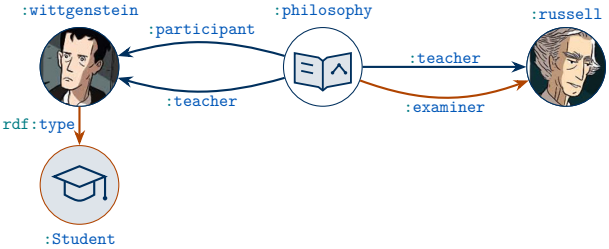


# Taking stock: This feedback is real

```
:examiner rdfs:subPropertyOf :participant
```

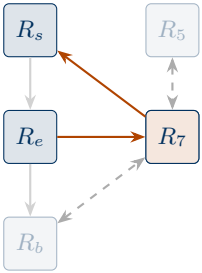


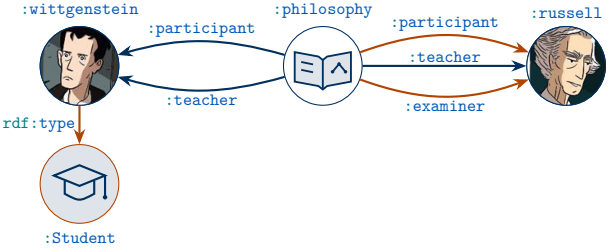




$R_e$ The teacher becomes an examiner.

# Taking stock: This feedback is real

`:examiner rdfs:subPropertyOf :participant`



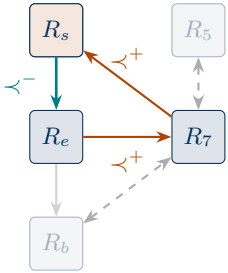



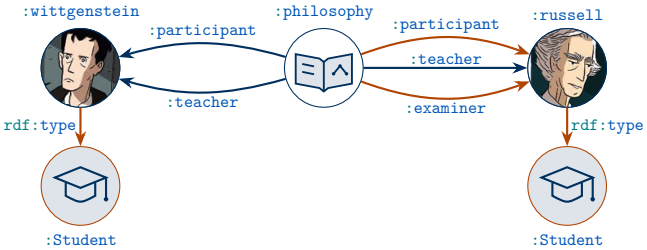

$R_e$ The teacher becomes an examiner.

$R_7$ RDFS also makes the examiner a participant.

# Taking stock: This feedback is real

`:examiner rdfs:subPropertyOf :participant`




$R_e$ The teacher becomes an examiner.

$R_7$ RDFS also makes the examiner a participant.

$R_s$ Student inference invalidates the earlier examiner test.

⇝ **Better analysis cannot remove a real interaction.**

# Idea 2: Constraints

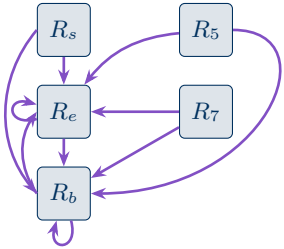
These schema facts must neither be given nor inferred.

{:examiner rdfs:subPropertyOf :participant} => ⊥

{:examiner rdfs:subPropertyOf rdfs:subPropertyOf} => ⊥

{:examiner rdfs:subPropertyOf rdf:type} => ⊥

Exclude harmful multi-step interactions, not every direct dependency.



# Idea 2: Constraints

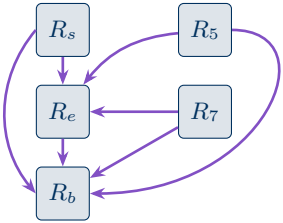
These schema facts must neither be given nor inferred.

{:examiner rdfs:subPropertyOf :participant} => ⊥

{:examiner rdfs:subPropertyOf rdfs:subPropertyOf} => ⊥

{:examiner rdfs:subPropertyOf rdf:type} => ⊥

Exclude harmful multi-step interactions, not every direct dependency.




**Constraint-aware precedence**

**acyclic ⇒ chain stratified under constraints.**

⇝ Follow this precedence when selecting a rule to apply to get a unique **perfect core model.**

## Main insight

Value invention and default negation can be combined, even for RDF rules.

## Contributions

- Analyse **multi-step** rule interactions.
- Use **constraints** to sharpen the analysis.
- **Chains** as a decidable approximation.




## Future work

- Integrate in RDF rule engines. **Nemo** integration in progress
- Constraint mining for rule repair.
- Simplify the criterion and contribute to **standardization**.

Nemo
Graph Rule Engine