OWL + Rules = .. ?

David Carral[1] Matthias Knorr[2] Adila Krisnadhi[1]

[1]Kno.e.sis Center, Wright State University, USA

[2]CENTRIA, Universidade Nova de Lisboa, Portugal

10th Extended Semantic Web Conference (ESWC) 2013

Support

David Carral and Adila Krisnadhi:

At Kno.e.sis Center, supported by NSF grant
III: Small: TROn - Tractable Reasoning with Ontologies

Matthias Knorr:

At Centria, supported by FCT grant

ERRO.
efficient reasoning with rules and ontologies

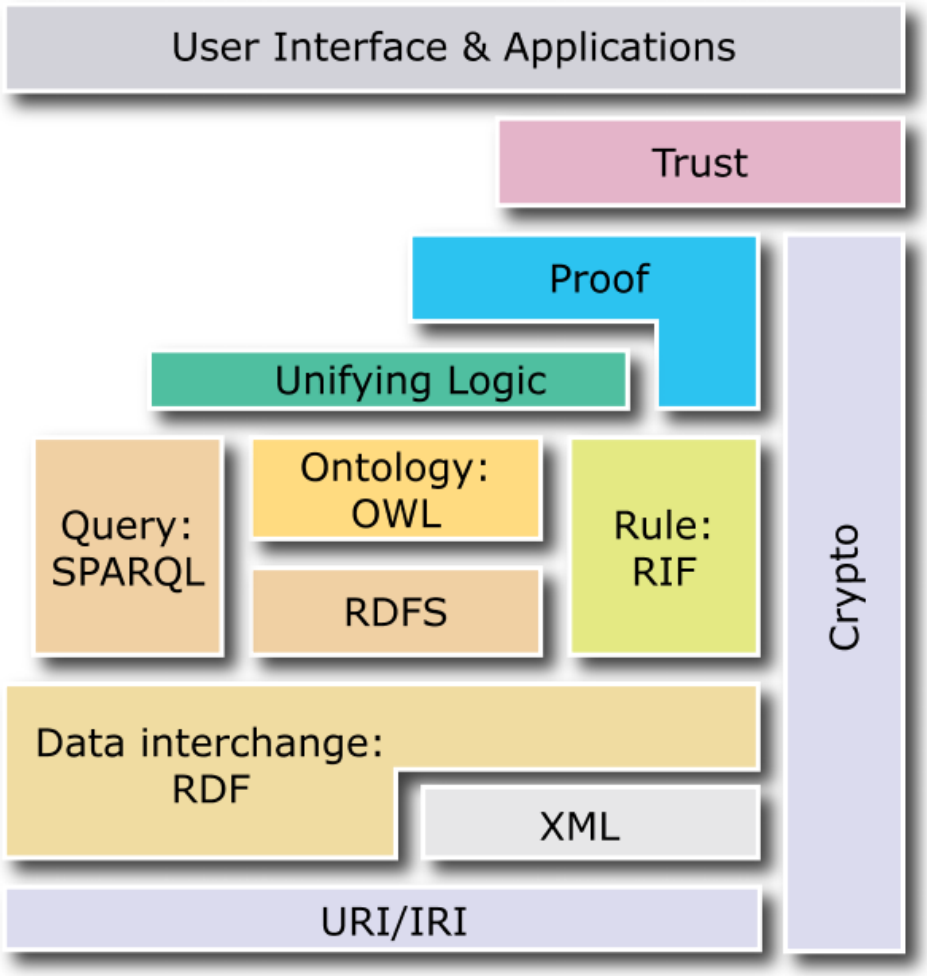
Slides

Latest version available from

`http://centria.di.fct.unl.pt/~mknorr/tutorialESWC2013/`

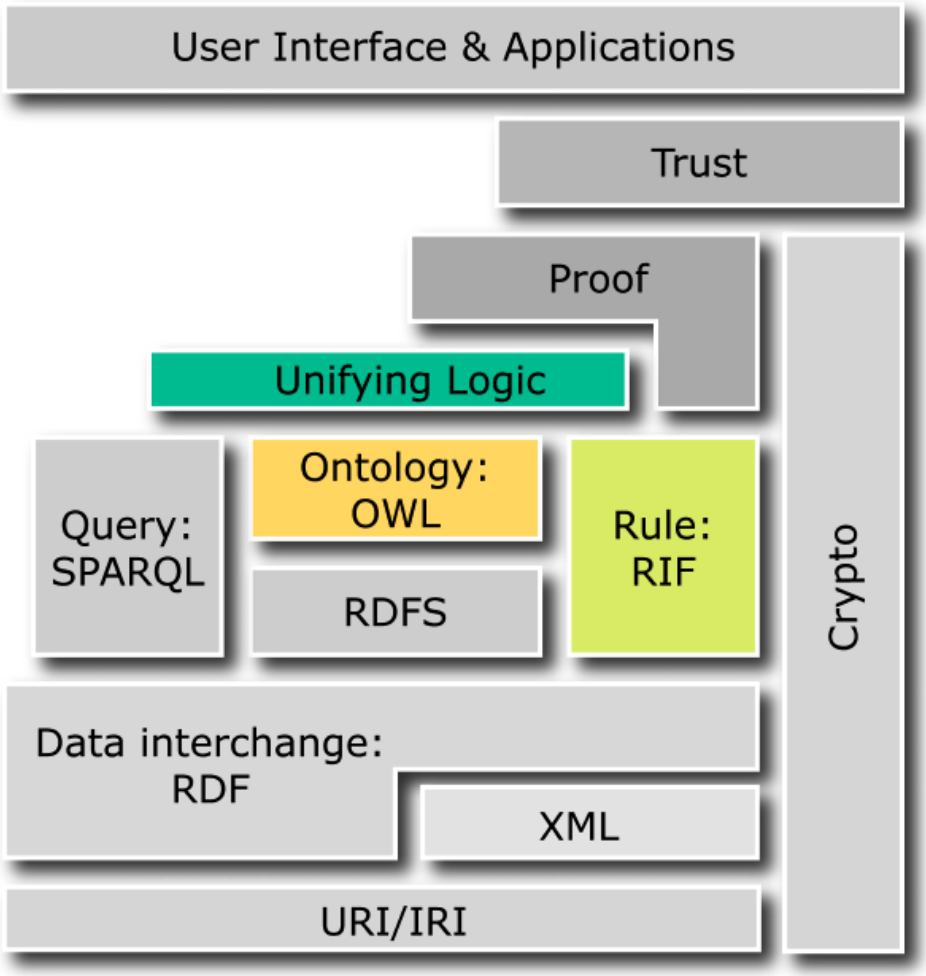
Introduction

Semantic Web Stack / Layer Cake



- Each layer builds on the layers below
- Standardization in progress and driven by W3C
- Hypertext Web technologies and some Semantic Web technologies already standardized

Two Different Paradigms



- Ontologies: OWL
- Rules: RIF, SWRL

Investigation towards a Unifying Logic

OWL 2 DL ($\mathcal{SROIQ}(D)$+...)

Basic Language Constructs

- Classes (concepts) – unary predicates

  Person, Woman, Mother, Uncle

- Properties (roles) – binary predicates

  hasChild, hasParent, hasWife

- Individuals – constants

  Mary, John, Bill

OWL 2 DL ($\mathcal{SROIQ}(D)$+...)

Axioms

- Assertions of named individuals to (complex) classes and properties

$$\text{Woman}(\text{Mary}) \qquad \text{hasMother}(\text{Bill},\text{Mary})$$

- (Sub)class and property hierarchies ($\sqsubseteq$)

$$\text{Woman} \sqsubseteq \text{Person} \qquad \text{hasMother} \sqsubseteq \text{hasParent}$$

- Equivalent classes ($\equiv$ – shortcut for $\sqsubseteq$ and $\sqsupseteq$)

$$\text{Person} \equiv \text{Human}$$

OWL 2 DL ($\mathcal{SROIQ}(D)+\ldots$)

Complex Classes

- Class intersection ($\sqcap$)

$$\text{Mother} \equiv \text{Person} \sqcap \text{Woman}$$

- Class union ($\sqcup$)

$$\text{Parent} \equiv \text{Mother} \sqcup \text{Father}$$

- Class complement ($\neg$)

$$\text{ChildlessPerson} \equiv \text{Person} \sqcap \neg\text{Parent}$$

OWL 2 DL ($\mathcal{SROIQ}(D)+\ldots$)

Complex Classes

- `owl:Thing` ($\top$) and `owl:Nothing` ($\bot$)

$$\text{Man} \sqcap \text{Woman} \sqsubseteq \bot$$

- Existential quantification ($\exists$)

$$\text{Parent} \equiv \exists \text{hasChild}.\text{Person} \qquad \exists \text{hasWife}.\top \sqsubseteq \text{Man}$$

- Universal quantification ($\forall$)

$$\text{NoDaughters} \equiv \forall \text{hasChild}.\text{Male} \qquad \top \sqsubseteq \forall \text{hasWife}.\text{Woman}$$

OWL 2 DL ($\mathcal{SROIQ}(D)$+...)

Complex Classes

- Qualified cardinality constraints ($\leq$ and $\geq$)

$$\leq 2\text{hasChild.Parent}(\text{John}) \qquad \geq 4\text{hasChild.}\top(\text{John})$$

- Nominals/enumerations of individuals (`owl:oneOf`)

$$\{\text{William}\} \equiv \{\text{Bill}\} \qquad \textit{SiblingsOfJohn} \equiv \{\text{Mary, Tom}\}$$

- Self

$$\{\text{NarcisticPerson}\} \equiv \exists \text{loves.Self}$$

OWL 2 DL ($\mathcal{SROIQ}(D)$+...)

Property Characteristics

- Inverse – hasChild$^-$ instead of hasParent
- Symmetric – hasSpouse
- Asymmetric – hasChild
- Disjoint – hasParent and hasChild
- Reflexive – hasRelative
- Irreflexive – hasChild
- Functional – hasSpouse
- Inverse functional – hasSpouse
- Transitive – hasDescendent

OWL 2 DL ($\mathcal{SROIQ}(D)$+...)

And also ...

- Property chains

$$\text{hasParent} \circ \text{hasParent} \sqsubseteq \text{hasGrandparent}$$

- Top property (U universal role)
- Bottom property (not part of $\mathcal{SROIQ}(D)$)
- Datatypes (with facets) (D)
- Keys (not part of $\mathcal{SROIQ}(D)$)

$$\mathit{HasKey}(:\mathit{Person}()(:\mathit{hasSSN}))$$

OWL 2 Profiles

fragments of OWL 2 for better computational properties

- OWL 2 EL:
  - Corresponds to $\mathcal{SROEL}(D)$ / $\mathcal{EL}^{++}$
  - Allows $\sqcap$, $\exists$, $\top$, $\bot$, nominals, property chains and hierarchies, and datatypes
  - Also allows: reflexive and transitive properties, keys
  - Used in large biomedicine ontologies, such as SNOMED CT, or GALEN (containing complex structural descriptions)

OWL 2 Profiles

- OWL 2 QL:
  - Corresponds to DL-Lite
  - Left of $\sqsubseteq$ only allows: $\exists$ limited to $\top$
  - Right of $\sqsubseteq$ only allows: $\sqcap$, $\neg$, $\exists$
  - Allows property inclusions but not property chains
  - Closely related to database technology, query answering can be realized by rewriting queries
- OWL 2 RL:
  - Corresponds to DLP, a rule fragment of OWL 2 DL
  - Well-suited for enriching RDF data
  - Details follow later

What we can(not) do with OWL

- Describe schema level knowledge: class hierarchy, properties about classes, relationships between classes, etc.
- Consistency and class subsumption checking
- Classifying individuals to classes
- Assert the existence of unknown individuals (i.e., those that must exist but cannot be named)
- Cannot specify arbitrary relationships between instances/individuals; due to the inherent tree structure of DLs
- Cannot express n-ary relationships between individuals with $n > 2$; DL extensions with n-ary predicates exist, but not part of OWL

Rules

- Prominent alternative to OWL modeling:
  - Rule-based expert systems
  - Logic Programming/Prolog
  - F-Logic [Kifer et al., 1995]
  - W3C Rule Interchange Format RIF (standard since 2010)
- Often argued to be more intuitive for modeling:

$$\text{worksAt(x,y)},\text{university(y)},\ \text{supervises(x,z)},\text{PhDstudent(z)} \rightarrow \mathit{ProfessorOf}(x, z)$$

Rules

- Rules can be divided into 3 categories:
  - First-order rules: logical implication $F \longrightarrow G$
    - Closely related to RIF-BLD (Basic Logic Dialect)
    - "Open world", declarative (first-order) semantics, monotonic
  - Logic Programming/PROLOG rules:
    - Close to first-order rules but with optional procedural aspects and possible built-ins
    - Covered by RIF-FLD (Framework for Logic Dialects)
    - "Closed world", (semi-)declarative, non-monotonic
  - Production rules:
    - IF condition THEN action
    - Roughly corresponds to RIF-PRD (Production Rule Dialect)
    - Semantics varies, sometimes defined as ad hoc computational mechanisms

First-order rules (Horn clauses)

- $\overbrace{A_1 \wedge A_2 \wedge \cdots \wedge A_k}^{body} \rightarrow \overbrace{H}^{head}$
  - Each A_i and H is a first-order atomic formula $P(t_1, \ldots, t_m)$ with P a predicate symbol with arity m
  - Each t_j is a term: a variable or an expression $f(s_1, \ldots, s_k)$ where f is a function symbol of arity k and $s_1, \ldots, s_k$ are terms
  - No quantifiers; no negation
- Datalog rules: first-order rules without function symbols
  - First used for deductive databases
  - Complexity: PTime data complexity (ExpTime combined)
  - Suitable for large datasets (and relatively small/fixed rule set)

What we can(not) do with Rules

- Specify and infer arbitrary relationships between individuals, including n-ary relationships with $n > 2$
- Many people find rules more natural for modeling
- Non-monotonic extensions are very well-studied, more than that of OWL (ASP solvers, etc.)
- Rules are usually only applied to known constants
- Cannot express the existence of unknown/unnamed individuals (unlike OWL)

Can't we bring them together?

Our Agenda . . .

This tutorial provides a condensed exposition of the recent efforts to answer the previous main question by focusing on the following issues:

- What kind of rules are readily expressible in OWL?
- What is DL-safety notion for rules? Why does it allow one to combine rules and OWL ontologies without losing decidability?
- Can we integrate DL-safe rules seamlessly within OWL framework by some small syntactic extension to OWL?
- Can we add non-monotonic flavor to such integration between DLs and rules?

Some historical bits (not complete...)

- 2001-2004: Description Logics (DLs) turn into the W3C OWL standard (logic programming still used for modeling ontologies);
- 2003: Description Logic Programs (DLP) [Grosof et al., WWW03] – intersection of OWL 1 DL and Datalog;
- 2004: Semantic Web Rules Language (SWRL) – OWL plus unbounded first-order rules, yet undecidable;
- 2004: dl-programs [Eiter et al., KR04] – OWL + non-monotonic rules modularly with limited interaction;
- 2005: DL-safety [Motik et al., JWS05] – DL-safe SWRL is decidable;
- 2006: $\mathcal{DL}$+log [Rosati, KR06] – weakly-DL-safe (non-monotonic) rules plus OWL, still separate semantics for each part;

Some historical bits (not complete...)

- 2007: Hybrid MKNF by [Motik and Rosati, IJCAI07] – seamless integration of OWL and non-monotonic (DL-safe) rules;
- 2006-09: standardization effort of OWL 2 by W3C;
- 2008-10: Description Logic Rules and ELP [Krötzsch et al., ECAI08; ISWC08] and [Rudolph et al., JELIA08] – significantly extended DLP;
- 2011: Well-founded Semantics for Hybrid MKNF [Knorr et al., AI11] – tractable for polynomial DLs;
- 2011: Nominal schemas by [Krötzsch et al., WWW11] – strongly integrate OWL 2 and DL-safe SWRL;
- 2012: Extending DL rules [Carral and Hitzler, ESWC12];
- 2012: Nominal schemas with non-monotonic extensions [Knorr et al., ECAI12].

Rules readily expressible in OWL

Reasoning Needs

z newsFrom rome .
rome locadedIn italy .

Reasoning Needs

z newsFrom rome .
rome locadedIn italy .

We want to conclude

z newsFrom italy .

Reasoning Needs

z newsFrom rome .
rome locadedIn italy .

We want to conclude

z newsFrom italy .

Rule:

$$\text{newsFrom}(x, y) \wedge \text{locatedIn}(y, z) \rightarrow \text{newsFrom}(x, z)$$

Reasoning Needs

z newsFrom rome .
rome locadedIn italy .

We want to conclude

z newsFrom italy .

Rule:

$$\text{newsFrom}(x, y) \wedge \text{locatedIn}(y, z) \rightarrow \text{newsFrom}(x, z)$$

In OWL:

$$\text{newsFrom} \circ \text{locatedIn} \sqsubseteq \text{newsFrom}$$

(using owl:propertyChainAxiom)

Reasoning Needs

e.g. knowledge base of authors and papers

<paper> hasAuthor <Author>

Reasoning Needs

e.g. knowledge base of authors and papers

<paper> hasAuthor <Author>

insufficient because author order is missing

use of RDF-lists not satisfactory due to the lack of formal semantics.

Reasoning Needs

e.g. knowledge base of authors and papers

<paper> hasAuthor <Author>

insufficient because author order is missing

use of RDF-lists not satisfactory due to the lack of formal semantics.

better:

```
<paper>   hasAuthorNumbered   _:x ;
_:x       authorNumber        n^^xsd:positiveInteger ;
          authorName          <author>.
```

Reasoning Needs

<paper>	hasAuthorNumbered	_:x ;
_:x	authorNumber	n^^xsd:positiveInteger ;
	authorName	<author>.

Reasoning Needs

<paper>	hasAuthorNumbered	_:x ;
_:x	authorNumber	n^^xsd:positiveInteger ;
	authorName	<author>.

In OWL:

$$\text{Paper} \sqsubseteq \exists \text{hasAuthorNumbered}.\text{NumberedAuthor}$$

$$\text{NumberedAuthor} \sqsubseteq = 1\ \text{authorNumber}.\ <\ \text{xsd:positiveInteger}\ >$$

$$\text{NumberedAuthor} \sqsubseteq = 1\ \text{authorName}.\text{Name}$$

Reasoning Needs

<paper>	hasAuthorNumbered	_:x ;
_:x	authorNumber	n^^xsd:positiveInteger ;
	authorName	<author>.

In OWL:

$$\text{Paper} \sqsubseteq \exists \text{hasAuthorNumbered}.\text{NumberedAuthor}$$

$$\text{NumberedAuthor} \sqsubseteq = 1\ \text{authorNumber}.\ <\ \text{xsd:positiveInteger}\ >$$

$$\text{NumberedAuthor} \sqsubseteq = 1\ \text{authorName}.\text{Name}$$

$$\text{hasAuthorNumbered} \circ \text{authorName} \sqsubseteq \text{hasAuthor}$$

KNO.E.SIS

CENTRIA

Reasoning Needs

Property hasFirstAuthor:

Reasoning Needs

Property hasFirstAuthor:

$$\text{Paper}(x) \wedge \text{hasAuthorNumbered}(x, y) \wedge \text{authorNumber}(y, 1)$$
$$\wedge \text{authorName}(y, z) \rightarrow \text{hasFirstAuthor}(x, z)$$

Reasoning Needs

Property hasFirstAuthor:

$$\begin{aligned}\text{Paper}(x) \wedge \text{hasAuthorNumbered}(x, y) \wedge \text{authorNumber}(y, 1) \\ \wedge \text{authorName}(y, z) \rightarrow \text{hasFirstAuthor}(x, z)\end{aligned}$$

in OWL:

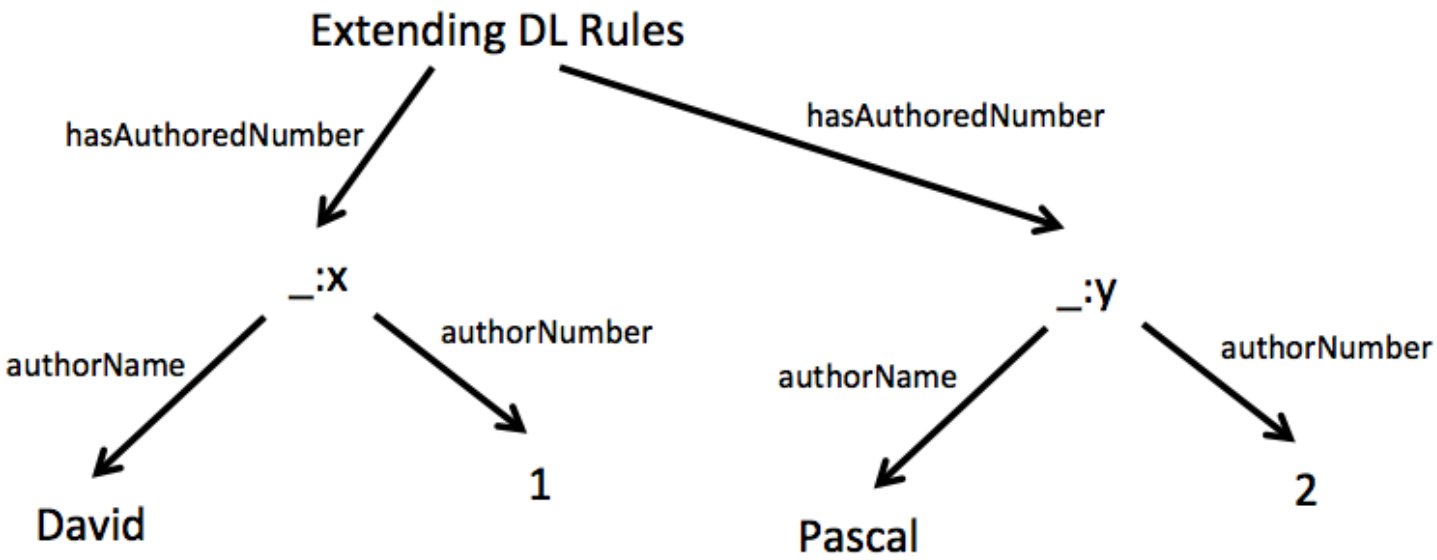
$$\begin{aligned}\text{Paper} &\sqsubseteq \exists \text{reflexivePaper.Self} \\ \exists \text{authorNumber.}\{1\} &\sqsubseteq \text{FirstAuthor} \\ \text{FirstAuthor} &\sqsubseteq \exists \text{reflexiveFirstAuthor.Self}\end{aligned}$$

$$\begin{aligned}\text{reflexivePaper} \circ \text{hasAuthorNumbered} \circ \\ \text{reflexiveFirstAuthor} \circ \text{authorName} \sqsubseteq \text{hasFirstAuthor}\end{aligned}$$

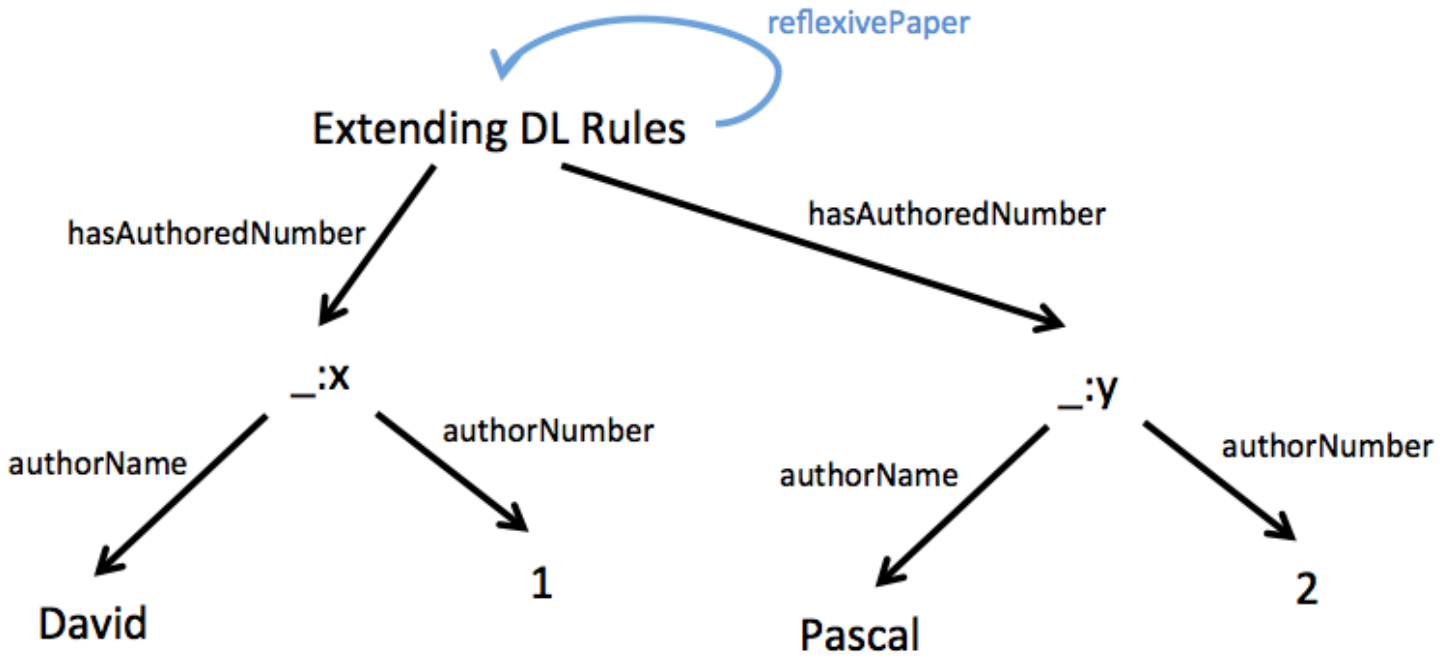
KNO.E.SIS

CENTRIA

Graphical Example

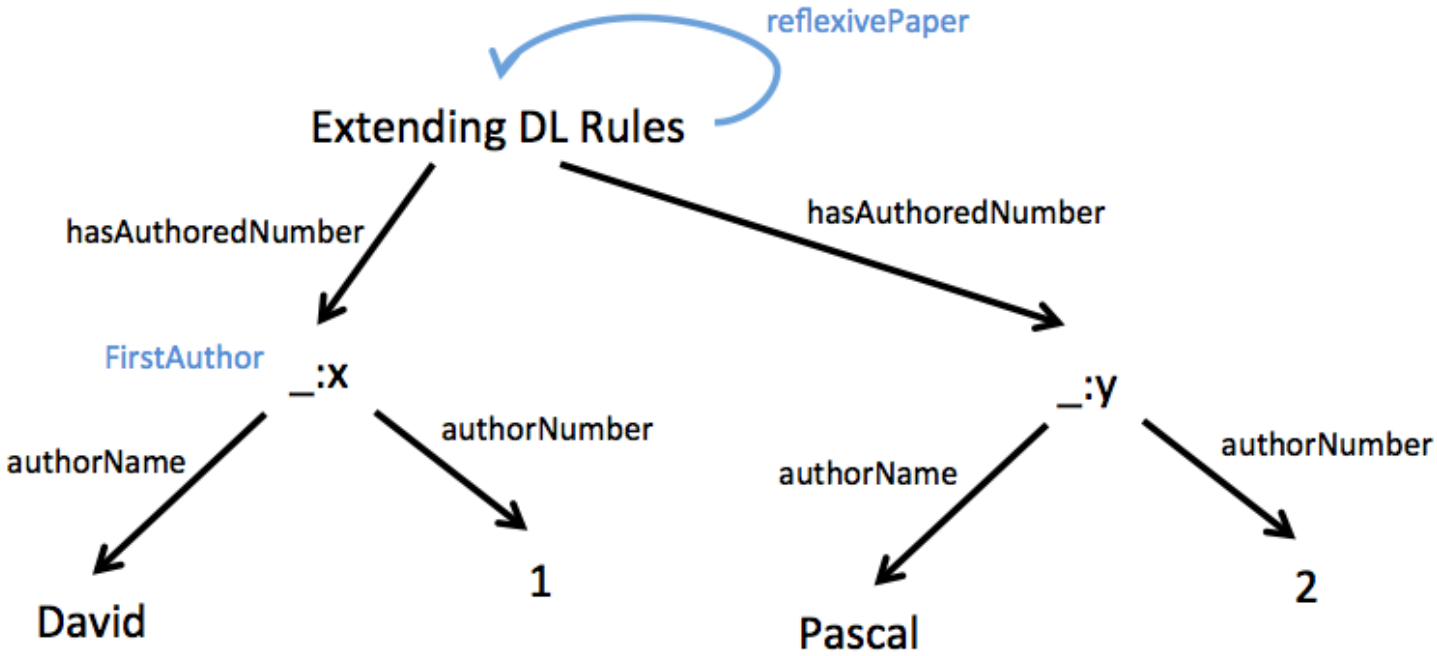


Graphical Example



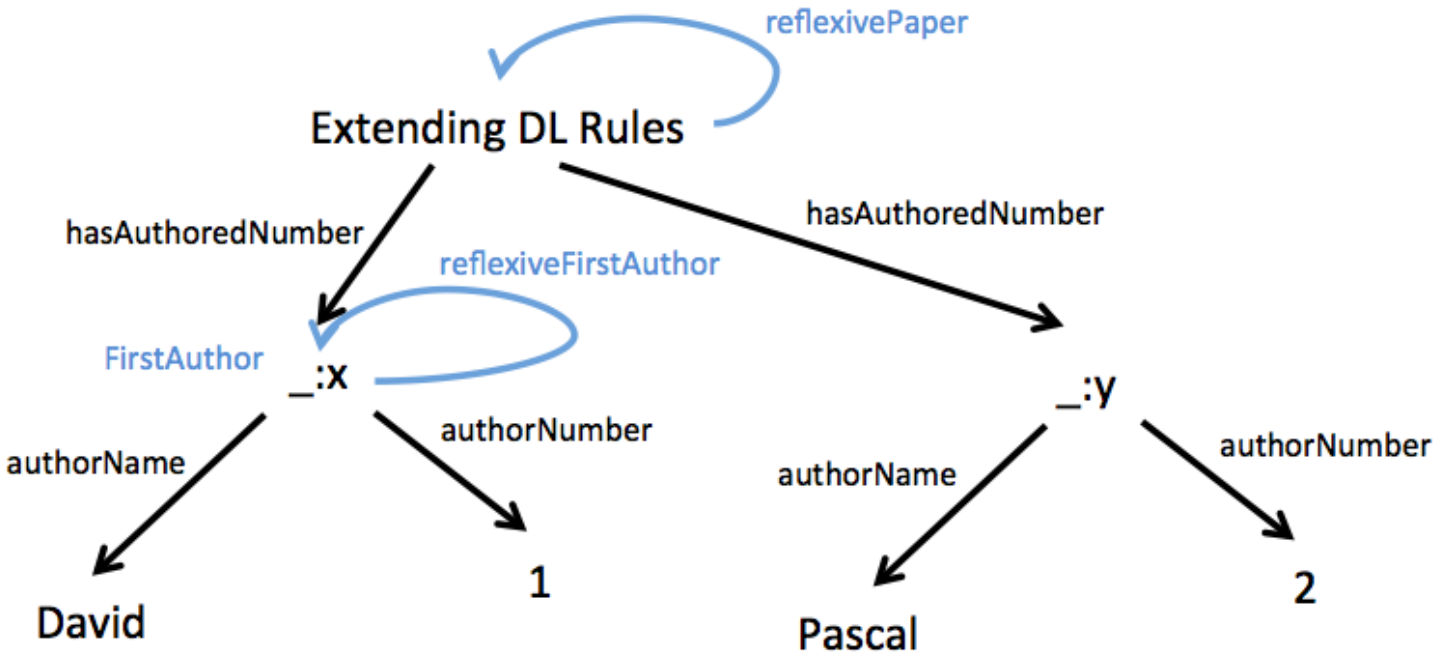
$$\text{Paper} \sqsubseteq \exists \text{reflexivePaper}.\text{Self}$$

Graphical Example



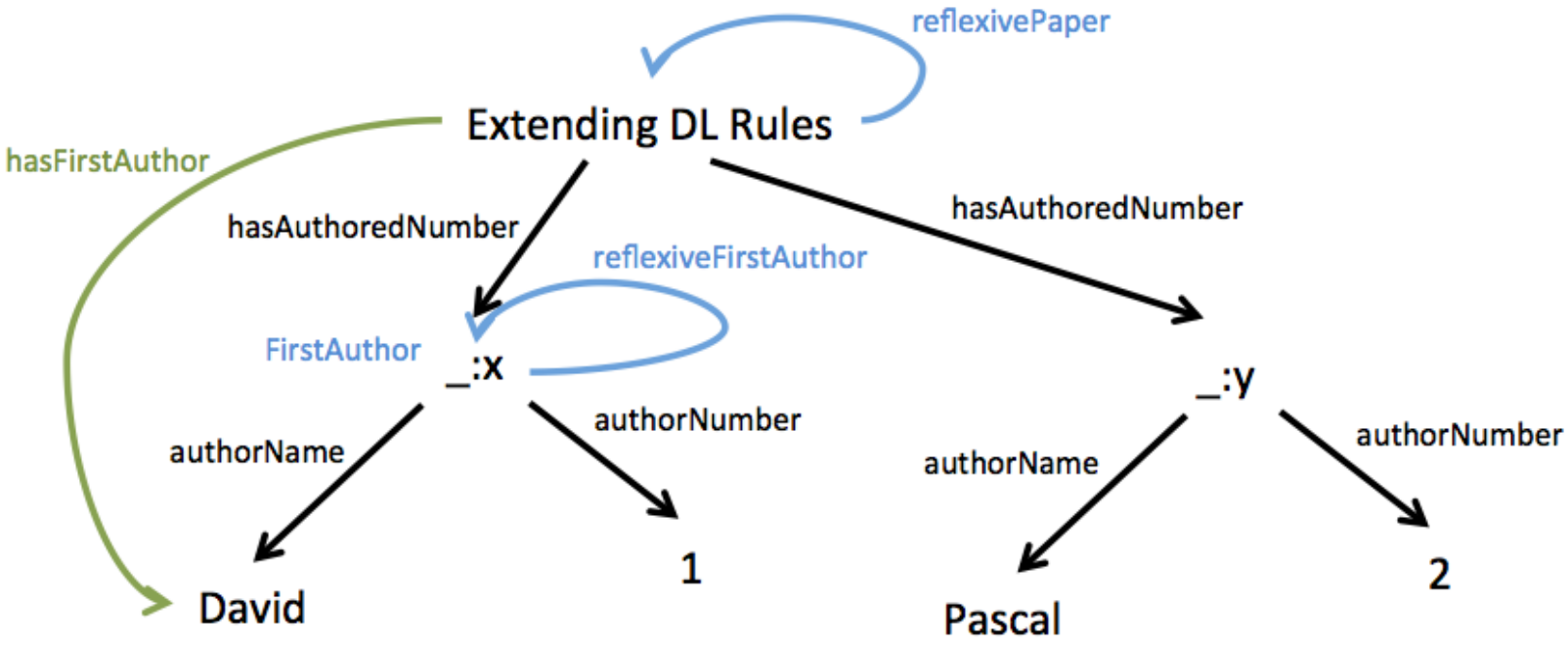
$$\exists \text{authorNumber}.\{1\} \sqsubseteq \text{FirstAuthor}$$

Graphical Example



$$\text{FirstAuthor} \sqsubseteq \exists \text{reflexiveFirstAuthor}.\text{Self}$$

Graphical Example



$$\text{reflexivePaper} \circ \text{hasAuthorNumbered} \circ \text{reflexiveFirstAuthor} \circ \text{authorName} \sqsubseteq \text{hasFirstAuthor}$$

Reasoning as first-class citizen

Why would we want to have knowledge/rules such as

$$\text{newFrom}(x, y) \wedge \text{locatedIn}(y, z) \rightarrow \text{newsFrom}(x, z)$$

if we can also just do this with some software code?

Reasoning as first-class citizen

Why would we want to have knowledge/rules such as

$$\text{newFrom}(x, y) \wedge \text{locatedIn}(y, z) \rightarrow \text{newsFrom}(x, z)$$

if we can also just do this with some software code?

- It declaratively describes what you do.

Reasoning as first-class citizen

Why would we want to have knowledge/rules such as

$$\text{newFrom}(x, y) \wedge \text{locatedIn}(y, z) \rightarrow \text{newsFrom}(x, z)$$

if we can also just do this with some software code?

- It declaratively describes what you do.
- It separates knowledge (as knowledge base) from programming.

Reasoning as first-class citizen

Why would we want to have knowledge/rules such as

$$\text{newFrom}(x, y) \wedge \text{locatedIn}(y, z) \rightarrow \text{newsFrom}(x, z)$$

if we can also just do this with some software code?

- It declaratively describes what you do.
- It separates knowledge (as knowledge base) from programming.
- It makes knowledge shareable.

Reasoning as first-class citizen

Why would we want to have knowledge/rules such as

$$\text{newFrom}(x, y) \wedge \text{locatedIn}(y, z) \rightarrow \text{newsFrom}(x, z)$$

if we can also just do this with some software code?

- It declaratively describes what you do.
- It separates knowledge (as knowledge base) from programming.
- It makes knowledge shareable.
- It makes knowledge easier to maintain.

Translating OWL Axioms

Which OWL axioms can be encoded as rules?

Let's see some examples

Translating OWL Axioms

Which OWL axioms can be encoded as rules?

Translating OWL Axioms

Which OWL axioms can be encoded as rules?

$$A \sqsubseteq B \text{ becomes } A(x) \rightarrow B(x)$$
$$R \sqsubseteq S \text{ becomes } R(x, y) \rightarrow S(x, y)$$

Translating OWL Axioms

Which OWL axioms can be encoded as rules?

$$A \sqsubseteq B \text{ becomes } A(x) \rightarrow B(x)$$
$$R \sqsubseteq S \text{ becomes } R(x, y) \rightarrow S(x, y)$$

$$A \sqcap \exists R.\exists S.B \sqsubseteq C \text{ becomes } A(x) \wedge R(x, y) \wedge S(y, z) \wedge B(z) \rightarrow C(x)$$

Translating OWL Axioms

Which OWL axioms can be encoded as rules?

$$A \sqsubseteq B \text{ becomes } A(x) \rightarrow B(x)$$
$$R \sqsubseteq S \text{ becomes } R(x, y) \rightarrow S(x, y)$$

$$A \sqcap \exists R.\exists S.B \sqsubseteq C \text{ becomes } A(x) \wedge R(x, y) \wedge S(y, z) \wedge B(z) \rightarrow C(x)$$

$$A \sqsubseteq \forall R.B \text{ becomes } A(x) \wedge R(x, y) \rightarrow B(y)$$

Rules in OWL

Which OWL axioms can be encoded as rules?

Rules in OWL

Which OWL axioms can be encoded as rules?

$$A \sqsubseteq \neg B \sqcup C \text{ becomes } A(x) \wedge B(x) \rightarrow C(x)$$

Rules in OWL

Which OWL axioms can be encoded as rules?

$$A \sqsubseteq \neg B \sqcup C \text{ becomes } A(x) \wedge B(x) \rightarrow C(x)$$

$$A \sqsubseteq \neg B \text{ becomes } A(x) \wedge B(x) \rightarrow f$$

Rules in OWL

Which OWL axioms can be encoded as rules?

$$A \sqsubseteq \neg B \sqcup C \text{ becomes } A(x) \wedge B(x) \rightarrow C(x)$$

$$A \sqsubseteq \neg B \text{ becomes } A(x) \wedge B(x) \rightarrow f$$

$$\top \sqsubseteq \leq 1R.\top \text{ becomes } R(x,y) \wedge R(x,z) \rightarrow y = z$$

Rules in OWL

Which OWL axioms can be encoded as rules?

$$A \sqsubseteq \neg B \sqcup C \text{ becomes } A(x) \wedge B(x) \rightarrow C(x)$$

$$A \sqsubseteq \neg B \text{ becomes } A(x) \wedge B(x) \rightarrow f$$

$$\top \sqsubseteq \leq 1R.\top \text{ becomes } R(x,y) \wedge R(x,z) \rightarrow y = z$$

$$A \sqcap \exists R.\{b\} \sqsubseteq C \text{ becomes } A(x) \wedge R(x,b) \rightarrow C(x)$$

Rules in OWL

Which OWL axioms can be encoded as rules?

$$\{a\} \equiv \{b\} \text{ becomes } \rightarrow a = b$$

Rules in OWL

Which OWL axioms can be encoded as rules?

$$\{a\} \equiv \{b\} \text{ becomes } \rightarrow a = b$$

$$A \sqcap B \sqsubseteq \bot \text{ becomes } A(x) \wedge B(x) \rightarrow f$$

Rules in OWL

Which OWL axioms can be encoded as rules?

$$\{a\} \equiv \{b\} \text{ becomes } \rightarrow a = b$$

$$A \sqcap B \sqsubseteq \bot \text{ becomes } A(x) \wedge B(x) \rightarrow f$$

$$A \sqsubseteq B \sqcap C \text{ becomes } A(x) \rightarrow B(x) \text{ and } A(x) \rightarrow C(x)$$
$$A \sqcup B \sqsubseteq C \text{ becomes } A(x) \rightarrow C(x) \text{ and } B(x) \rightarrow C(x)$$

Rules in OWL

A DL axiom α can be translated into rules if, after translating α into a first-order predicate logic expression α', and after normalizing this expression into a set of clauses M, each formula in M is a Horn clause (i.e., a rule).

Issue: How complicated a translation is allowed?

Rules in OWL

A DL axiom α can be translated into rules if, after translating α into a first-order predicate logic expression α', and after normalizing this expression into a set of clauses M, each formula in M is a Horn clause (i.e., a rule).

Issue: How complicated a translation is allowed?

Naive translation: DLP plus some more (OWL 2)
e.g.,

$$R \circ S \sqsubseteq T \text{ becomes } R(x, y) \wedge S(y, z) \rightarrow T(x, z)$$

This essentially results in OWL RL

Which rules can be translated into OWL axioms?

- Rolification
- Examples
- Formal definition: Rule Graphs

Rolification

$$\text{Elephant}(x) \wedge \text{Mouse}(y) \rightarrow \text{biggerThan}(x, y)$$

Rolification

$$\text{Elephant(x)} \wedge \text{Mouse(y)} \rightarrow \text{biggerThan(x, y)}$$

Rolification of a concept A: $A \sqsubseteq \exists R_A.\text{Self}$

Rolification

$$\text{Elephant(x)} \wedge \text{Mouse(y)} \rightarrow \text{biggerThan(x, y)}$$

Rolification of a concept A: $A \sqsubseteq \exists R_A.\text{Self}$

$$\text{Elephant} \sqsubseteq \exists R_{\text{Elephant}}.\text{Self}$$
$$\text{Mouse} \sqsubseteq \exists R_{\text{Mouse}}.\text{Self}$$
$$R_{\text{Elephant}} \circ U \circ R_{\text{Mouse}} \sqsubseteq \mathit{biggerThan}$$

Rolification

$$\mathrm{A}(x) \wedge \mathrm{R}(x, y) \rightarrow \mathrm{S}(x, y) \text{ becomes } R_A \circ R \sqsubseteq S$$

$$\mathrm{A}(y) \wedge \mathrm{R}(x, y) \rightarrow \mathrm{S}(x, y) \text{ becomes } R \circ R_A \sqsubseteq S$$

$$\mathrm{A}(x) \wedge \mathrm{B}(y) \wedge \mathrm{R}(x, y) \rightarrow \mathrm{S}(x, y) \text{ becomes } R_A \circ R \circ R_B \sqsubseteq S$$

Rolification

$$\mathsf{A}(x) \wedge \mathsf{R}(x, y) \rightarrow \mathsf{S}(x, y) \text{ becomes } R_A \circ R \sqsubseteq S$$
$$\mathsf{A}(y) \wedge \mathsf{R}(x, y) \rightarrow \mathsf{S}(x, y) \text{ becomes } R \circ R_A \sqsubseteq S$$
$$\mathsf{A}(x) \wedge \mathsf{B}(y) \wedge \mathsf{R}(x, y) \rightarrow \mathsf{S}(x, y) \text{ becomes } R_A \circ R \circ R_B \sqsubseteq S$$

$$\mathsf{Woman}(x) \wedge \mathsf{marriedTo}(x, y) \wedge \mathsf{Man}(y) \rightarrow \mathsf{hasHusband}(x, y)$$
$$R_{\mathsf{Woman}} \circ \mathsf{marriedTo} \circ R_{\mathsf{Man}} \sqsubseteq \mathsf{hasHusband}$$

careful - role regularity needs to be preserved

$$\mathsf{hasHusband} \sqsubseteq \mathsf{marriedTo}$$

Rolification

$$\text{worksAt}(x, y) \wedge \text{University}(y) \wedge \text{supervises}(x, z)$$
$$\wedge \text{PhDStudent}(z) \rightarrow \text{professorOf}(x, z)$$

$$R_{\exists \text{worksAt.University}} \circ \text{supervises} \circ R_{\text{PhDStudent}} \sqsubseteq \text{professorOf}$$

Rules in OWL 2

$$\text{Man}(x) \wedge \text{hasBrother}(x, y) \wedge \text{hasChild}(y, z) \rightarrow \text{Uncle}(x)$$

$$\text{Man} \sqcap \exists \text{hasBrother}.\exists \text{hasChild}.\top \sqsubseteq \text{Uncle}$$

$$\text{NutAllergic}(x) \wedge \text{NutProdcut}(y) \rightarrow \text{dislikes}(x, y)$$

$$\text{NutAllergic} \sqsubseteq \exists \text{nutAllergic}.\text{Self}$$

$$\text{NutProduct} \sqsubseteq \exists \text{nutProduct}.\text{Self}$$

$$\text{nutAllergic} \circ U \circ \text{nutProduct} \sqsubseteq \text{dislikes}$$

$$\text{dislikes}(x, z) \wedge \text{Dish}(y) \wedge \text{contains}(y, z) \rightarrow \text{dislikes}(x, y)$$

$$\text{Dish} \sqsubseteq \exists \text{dish}.\text{Self}$$

$$\text{dislikes} \circ \text{contains}^{-} \circ \text{dish} \sqsubseteq \text{dislikes}$$

KNO.E.SIS

CENTRIA

So how can we pinpoint this?

- Tree-shaped bodies (variables)
- First argument of the conclusion is the root

$$C(x) \wedge R(x, a) \wedge S(x, \mathbf{y}) \wedge D(\mathbf{y}) \wedge T(\mathbf{y}, a) \rightarrow E(x)$$

$$C \sqcap \exists R.\{a\} \sqcap \exists S.(D \sqcap \exists T.\{a\}) \sqsubseteq E$$

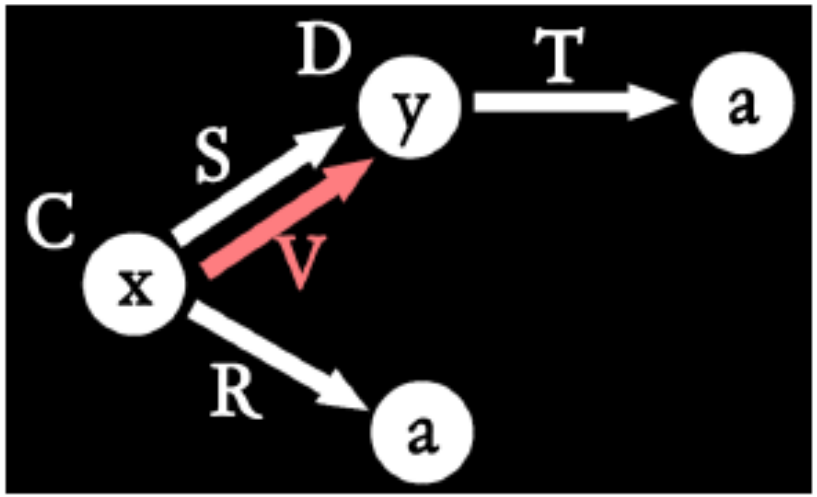
So how can we pinpoint this?

$$C(x) \wedge R(x, a) \wedge S(x, \mathbf{y}) \wedge D(\mathbf{y}) \wedge T(\mathbf{y}, a) \rightarrow V(x, y)$$

So how can we pinpoint this?

$$C(x) \wedge R(x, a) \wedge S(x, \mathbf{y}) \wedge D(\mathbf{y}) \wedge T(\mathbf{y}, a) \rightarrow V(x, y)$$

$$C \sqcap \exists R.\{a\} \sqsubseteq \exists R_1.\text{Self}$$
$$D \sqcap \exists T.\{a\} \sqsubseteq \exists R_2.\text{Self}$$
$$R_1 \circ S \circ R_2 \sqsubseteq V$$



KNO.E.SIS

CENTRIA

So how can we pinpoint this?

Rule graph: $C(x) \wedge R(x, a) \wedge S(x, y) \wedge D(y) \wedge T(y, a) \rightarrow P(x, y)$

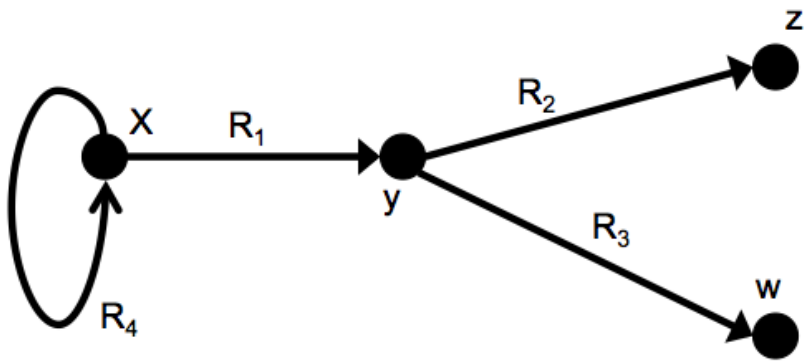
Graph analysis: determine whether a rule is expressible within a given profile
Automatic Transformation

DLs Rules: $\mathcal{EL}^{++}$

$$R_1(x, y) \wedge C_1(y) \wedge R_2(y, w) \wedge R_3(y, z) \wedge C_2(z) \wedge R_4(x, x) \rightarrow C_3(x)$$

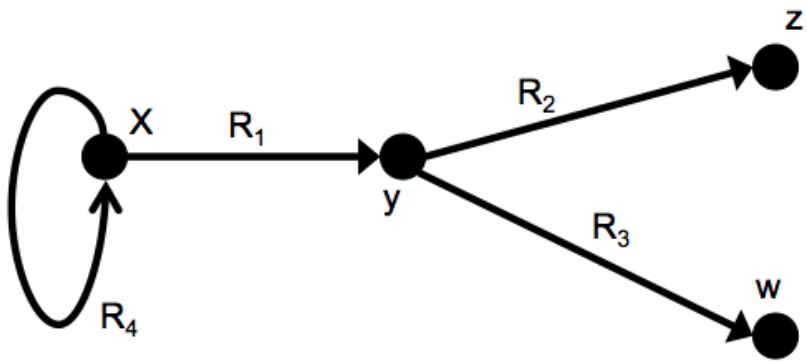
DLs Rules: $\mathcal{EL}^{++}$

$$R_1(x, y) \wedge C_1(y) \wedge R_2(y, w) \wedge R_3(y, z) \wedge C_2(z) \wedge R_4(x, x) \rightarrow C_3(x)$$



DLs Rules: $\mathcal{EL}^{++}$

$$R_1(x,y) \wedge C_1(y) \wedge R_2(y,w) \wedge R_3(y,z) \wedge C_2(z) \wedge R_4(x,x) \rightarrow C_3(x)$$



$$\exists R_1.(C_1 \sqcap \exists R_2.\top \sqcap \exists R_3.C_2) \sqcap \exists R_4.Self \sqsubseteq C_3$$

KNO.E.SIS

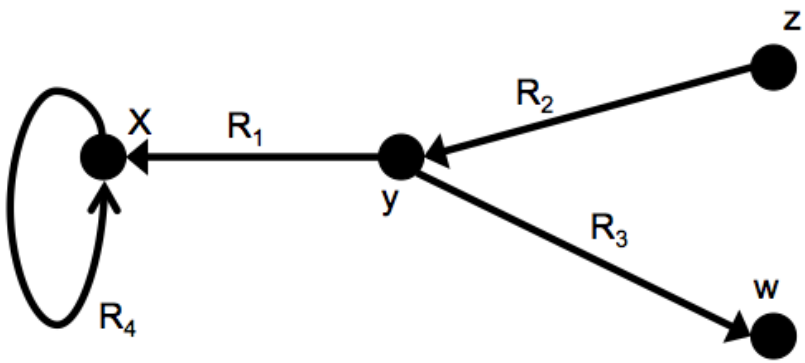
CENTRIA

DLs Rules: $\mathcal{SROIQ}$

$$R_1(y,x) \wedge C_1(y) \wedge R_2(w,y) \wedge R_3(y,z) \wedge C_2(z) \wedge R_4(x,x) \rightarrow C_3(x)$$

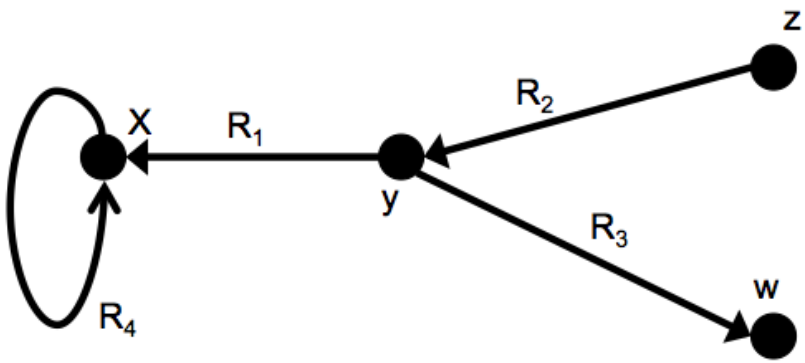
DLs Rules: $\mathcal{SROIQ}$

$$R_1(y,x) \wedge C_1(y) \wedge R_2(w,y) \wedge R_3(y,z) \wedge C_2(z) \wedge R_4(x,x) \rightarrow C_3(x)$$



DLs Rules: $\mathcal{SROIQ}$

$$R_1(y,x) \wedge C_1(y) \wedge R_2(w,y) \wedge R_3(y,z) \wedge C_2(z) \wedge R_4(x,x) \rightarrow C_3(x)$$



$$\exists R_1^-.(C_1 \sqcap \exists R_2^-.\top \sqcap \exists R_3.C_2) \sqcap \exists R_4.Self \sqsubseteq C_3$$

Extending DL Rules

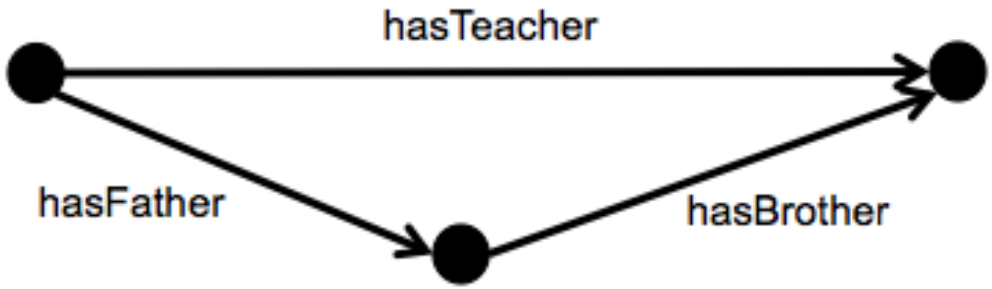
Extended Description Logic Rules
[Carral and Hitzler, ESWC12]

Conjunction over complex roles?

- Some cyclic rules

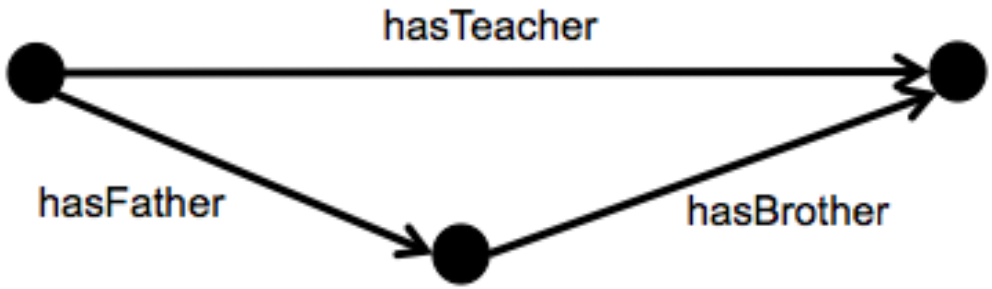
Extending DL: $\mathcal{SROIQ}(\sqcap)$

$\text{hasFather}(x, y) \wedge \text{hasBrother}(y, z) \wedge \text{hasTeacher}(x, z) \rightarrow \text{TaughtByUncle}(x)$



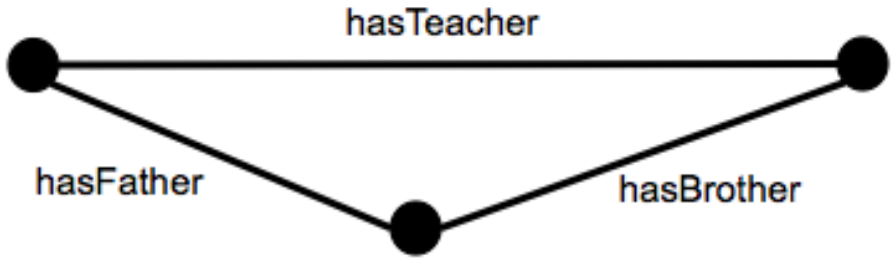
Extending DLs: $\mathcal{SROIQ}(\sqcap)$

$\text{hasFather}(x, y) \wedge \text{hasBrother}(y, z) \wedge \text{hasTeacher}(x, z) \rightarrow \text{TaughtByUncle}(x)$



Extending DLs: $\mathcal{SROIQ}(\sqcap)$

$\text{hasFather}(x, y) \wedge \text{hasBrother}(y, z) \wedge \text{hasTeacher}(x, z) \rightarrow$
$\text{TaughtByUncle}(x)$



Equivalent Translation:

$$\text{hasFather}(x, y) \wedge \text{hasBrother}(y, z) \rightarrow \textit{hasUncle}(x, z)$$
$$\textit{hasUncle}(x, z) \wedge \text{hasTeacher}(x, z) \rightarrow \text{TaughtbyUncle}(x)$$

$$\text{hasFather} \circ \text{hasBrother} \sqsubseteq \text{hasUncle}$$

Extending DLs: $\mathcal{SROIQ}(\sqcap)$

Middle rule:

$$\mathit{hasUncle}(x, z) \wedge \mathrm{hasTeacher}(x, z) \rightarrow \mathrm{TaughtbyUncle}(x)$$

Equivalent Translation:

$$\mathrm{hasUncle}(x, z) \wedge \mathrm{hasTeacher}(x, z) \rightarrow \mathit{hasUncleAndTeacher}(x, z)$$
$$\mathit{hasUncleAndTeacher}(x, z) \rightarrow \mathrm{TaughtbyUncle}(x)$$

$$\mathbf{hasUncle} \sqcap \mathbf{hasTeacher} \sqsubseteq \mathbf{hasUncle}$$
$$\exists \mathit{hasUncleAndTeacher}.\top \sqsubseteq \mathit{TaughtByUncle}$$

Extending DLs: $\mathcal{SROIQ}(\sqcap)$

$$\text{hasFather}(x, y) \wedge \text{hasBrother}(y, z) \wedge \text{hasTeacher}(x, z) \rightarrow \text{TaughtByUncle}(x)$$

$$\text{hasFather} \circ \text{hasBrother} \sqsubseteq \text{hasUncle}$$

$$\textbf{hasUncle} \sqcap \textbf{hasTeacher} \sqsubseteq \textbf{hasUncle}$$

$$\exists \text{hasTeacherAndUncle}.\top \sqsubseteq \text{TaughtByUncle}$$

Conclusions and Future Work

- Definition of DL Rules
- Automatic Transformation: implementation
- Extending DL Rules: $\sqcap$

Nominal Schemas

What we learned about rules expressible in OWL

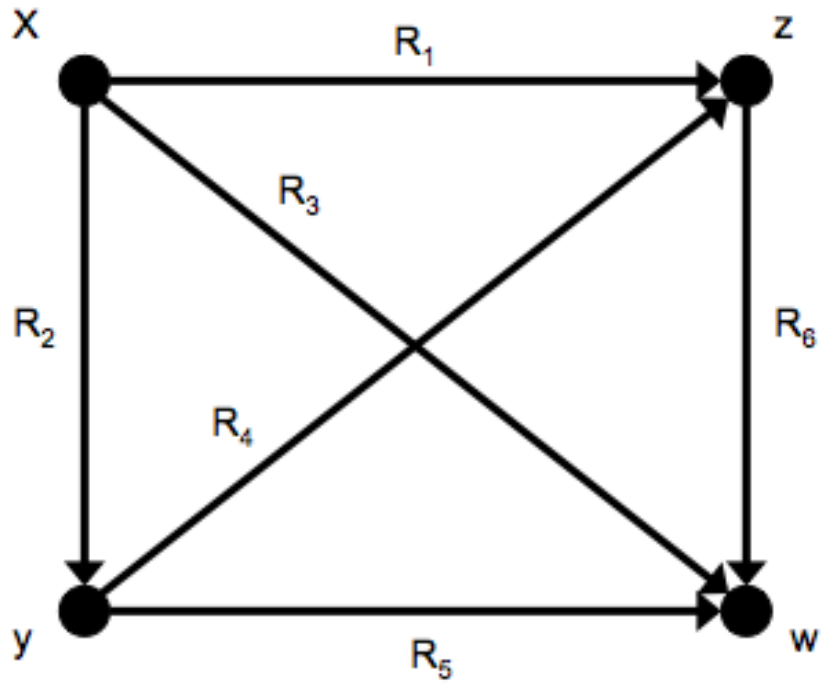
- Rules with tree-shaped body can be expressed in DL,
- role conjunction allows DLs to express some rules with non-tree-shaped body, but
- many rules are not covered.

Clique of 4

$$R_1(x,y) \wedge R_2(x,z) \wedge R_3(x,w) \wedge R_4(y,z) \wedge R_5(y,w) \wedge R_6(w,z) \rightarrow C(x)$$

Clique of 4

$$R_1(x,y) \wedge R_2(x,z) \wedge R_3(x,w) \wedge R_4(y,z) \wedge R_5(y,w) \wedge R_6(w,z) \rightarrow C(x)$$



Nominal Schemas

A Better Uncle for OWL
[Krötzsch et al; WWW 2011]

$$\mathsf{hasParent}(x, y) \wedge \mathsf{married}(y, z) \wedge \mathsf{hasParent}(x, z) \rightarrow C(x)$$

$$\exists \mathsf{hasParent}.\exists \mathsf{married}.\{z\} \sqcap \exists \mathsf{hasParent}.\{z\} \sqsubseteq C$$

Nominal Schemas

A Better Uncle for OWL
[Krötzsch et al; WWW 2011]

$\text{hasParent}(x, y) \wedge \text{married}(y, z) \wedge \text{hasParent}(x, z) \rightarrow C(x)$

$\exists \text{hasParent}.\exists \text{married}.\{z\} \sqcap \exists \text{hasParent}.\{z\} \sqsubseteq C$

$\{z\}$ only binds to known/named individuals!

Covers DL-safe datalog (arbitrary arity of predicates)

Complex Rules to OWL

Theorem

Any rule R containing m different free variables, where $m > 3$, *can be directly expressed in DL using n nominal schemas s.t.* $n \leq m - 2$.

DL-safe Rules

- How about simply adding rules "as-is" to the ontology?

DL-safe Rules

- How about simply adding rules "as-is" to the ontology?
- Problem: although the DL-part and rule-part are both decidable, the combination is undecidable!

DL-safe Rules

- How about simply adding rules "as-is" to the ontology?
- Problem: although the DL-part and rule-part are both decidable, the combination is undecidable!
- Work around: weaken the rule semantics?

DL-safety

- Decidability guaranteed if rules only operate on named individuals
  - Named individuals are finite.

DL-safety

- Decidability guaranteed if rules only operate on named individuals
  - Named individuals are finite.
- DL-safety: variables in the rules refer to only named individuals in the OWL ontology.

DL-safety

- Decidability guaranteed if rules only operate on named individuals
  - Named individuals are finite.
- DL-safety: variables in the rules refer to only named individuals in the OWL ontology.
- In the rule:

$$R(u,x) \wedge A(x) \wedge S(u,y) \wedge T(x,z) \wedge T(y,z) \wedge R(u,z) \wedge S(x,y) \rightarrow B(u)$$

the variables u, x, y, z refer only to named individuals.

DL-safety

- Decidability guaranteed if rules only operate on named individuals
  - Named individuals are finite.
- DL-safety: variables in the rules refer to only named individuals in the OWL ontology.
- In the rule:

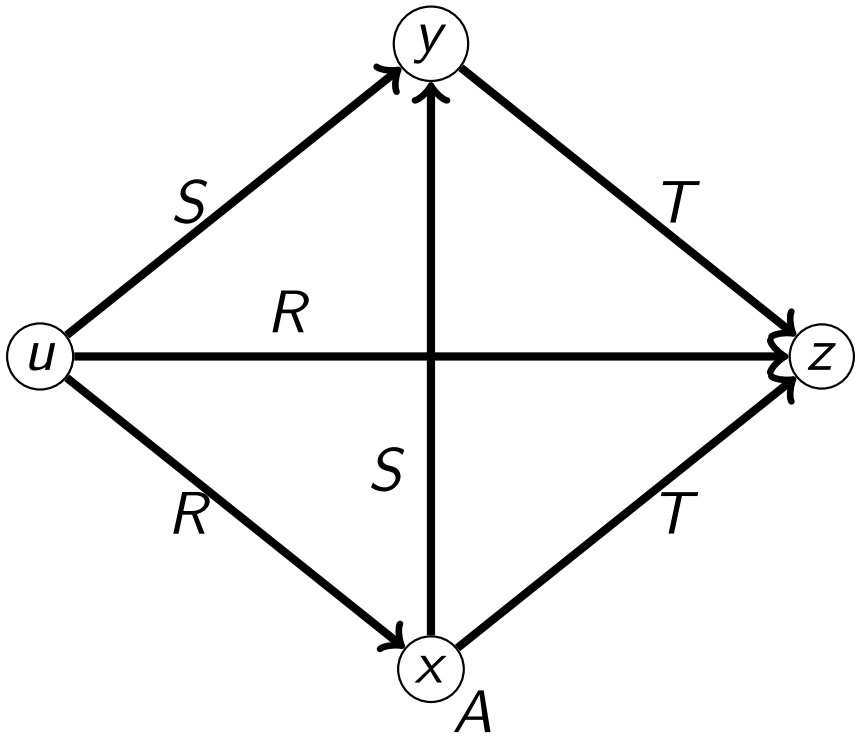
$$R(u,x) \wedge A(x) \wedge S(u,y) \wedge T(x,z) \wedge T(y,z) \wedge R(u,z) \wedge S(x,y) \rightarrow B(u)$$

  the variables u, x, y, z refer only to named individuals.
- Approach taken by DL-safe SWRL.

Revisiting DL-safety: can it be relaxed?

$$R(u,x) \wedge A(x) \wedge S(u,y) \wedge T(x,z) \wedge T(y,z) \wedge R(u,z) \wedge S(x,y) \rightarrow B(u)$$

Rule body forms complicated graph:



Revisiting DL-safety: can it be relaxed?

$$R(u,x) \wedge A(x) \wedge S(u,y) \wedge T(x,z) \wedge T(y,z) \wedge R(u,z) \wedge S(x,y) \rightarrow B(u)$$

If y and z refer to named individuals, say a, b, it represents:

$$R(u,x) \wedge A(x) \wedge S(u,a) \wedge T(x,a) \wedge T(a,a) \wedge R(u,a) \wedge S(x,a) \rightarrow B(u)$$
$$R(u,x) \wedge A(x) \wedge S(u,a) \wedge T(x,b) \wedge T(a,b) \wedge R(u,b) \wedge S(x,a) \rightarrow B(u)$$
$$R(u,x) \wedge A(x) \wedge S(u,b) \wedge T(x,a) \wedge T(b,a) \wedge R(u,a) \wedge S(x,b) \rightarrow B(u)$$
$$R(u,x) \wedge A(x) \wedge S(u,b) \wedge T(x,b) \wedge T(b,b) \wedge R(u,b) \wedge S(x,b) \rightarrow B(u)$$

Revisiting DL-safety: can it be relaxed?

$$R(u,x) \wedge A(x) \wedge S(u,y) \wedge T(x,z) \wedge T(y,z) \wedge R(u,z) \wedge S(x,y) \rightarrow B(u)$$

$$R(u,x) \wedge A(x) \wedge S(u,a) \wedge T(x,a) \wedge T(a,a) \wedge R(u,a) \wedge S(x,a) \rightarrow B(u)$$
$$R(u,x) \wedge A(x) \wedge S(u,a) \wedge T(x,b) \wedge T(a,b) \wedge R(u,b) \wedge S(x,a) \rightarrow B(u)$$
$$R(u,x) \wedge A(x) \wedge S(u,b) \wedge T(x,a) \wedge T(b,a) \wedge R(u,a) \wedge S(x,b) \rightarrow B(u)$$
$$R(u,x) \wedge A(x) \wedge S(u,b) \wedge T(x,b) \wedge T(b,b) \wedge R(u,b) \wedge S(x,b) \rightarrow B(u)$$

- Only y and z need to be DL-safe.
- Expressible in OWL EL.

Nominal schemas: DL-safety only to some variables

- In the rule

$$R(u,x) \wedge A(x) \wedge S(u,y) \wedge T(x,z) \wedge T(y,z) \wedge R(u,z) \wedge S(x,y) \rightarrow B(u)$$

only y and z need to be DL-safe to be expressible in OWL EL.

- Expressing it in OWL EL need multiple axioms:

$$\exists R.\{a\} \sqcap \exists R.(A \sqcap \exists S.\{a\} \sqcap \exists T.\{a\}) \sqcap \exists S.(\{a\} \sqcap \exists T.\{a\}) \sqsubseteq B$$
$$\exists R.\{b\} \sqcap \exists R.(A \sqcap \exists S.\{a\} \sqcap \exists T.\{b\}) \sqcap \exists S.(\{a\} \sqcap \exists T.\{b\}) \sqsubseteq B$$
$$\exists R.\{a\} \sqcap \exists R.(A \sqcap \exists S.\{b\} \sqcap \exists T.\{a\}) \sqcap \exists S.(\{b\} \sqcap \exists T.\{a\}) \sqsubseteq B$$
$$\exists R.\{b\} \sqcap \exists R.(A \sqcap \exists S.\{b\} \sqcap \exists T.\{b\}) \sqcap \exists S.(\{b\} \sqcap \exists T.\{b\}) \sqsubseteq B$$

- With nominal schemas, the above 4 axioms can be condensed:
$\exists R.\{z\} \sqcap \exists R.(A \sqcap \exists S.\{y\} \sqcap \exists T.\{z\}) \sqcap \exists S.(\{y\} \sqcap \exists T.\{z\}) \sqsubseteq B$

Nominal schemas: syntax and semantics

- Nominal schemas: a "variable nominal" construct in the form of $\{x\}$ where x is a variable.

Nominal schemas: syntax and semantics

- Nominal schemas: a "variable nominal" construct in the form of $\{x\}$ where x is a variable.
- Semantically, each occurrence of a nominal schema represents all possible nominals in the ontology.

Nominal schemas: syntax and semantics

- Nominal schemas: a "variable nominal" construct in the form of $\{x\}$ where x is a variable.
- Semantically, each occurrence of a nominal schema represents all possible nominals in the ontology.
- If an axiom α contains n different nominal schemas (each may occur more than once) while the ontology has m different named individuals, then α represents m^n different axioms, each is obtained by substituting nominal schemas with named individuals.

Nominal schemas: syntax and semantics

- Nominal schemas: a "variable nominal" construct in the form of $\{x\}$ where x is a variable.
- Semantically, each occurrence of a nominal schema represents all possible nominals in the ontology.
- If an axiom α contains n different nominal schemas (each may occur more than once) while the ontology has m different named individuals, then α represents m^n different axioms, each is obtained by substituting nominal schemas with named individuals.
- All those m^n axioms are called *groundings* of α.

What can we express with nominal schemas?

- Let $\mathcal{SROELV}(\times, \sqcap) = \mathcal{SROEL}(\times, \sqcap)$ + nominal schemas.

What can we express with nominal schemas?

- Let $\mathcal{SROELV}(\times, \sqcap) = \mathcal{SROEL}(\times, \sqcap)$ + nominal schemas.
- $\mathcal{SROELV}(\sqcap, \times)$ covers:
  - DL-safe Datalog rules with predicates of arbitrary arity is also in $\mathcal{SROELV}$.
  - OWL 2 EL without datatypes
  - DL-safe OWL 2 RL without datatypes — but, preserves only ABox entailments (the main inference task for OWL 2 RL)
  - most of OWL 2 QL

Complexity bounds

- Reasoning for $\mathcal{SROIQV} = \mathcal{SROIQ}$ + nominal schemas, is theoretically no worse than $\mathcal{SROIQ}$ [Krötzsch et al., WWW11]

Complexity bounds

- Reasoning for $\mathcal{SROIQV} = \mathcal{SROIQ}$ + nominal schemas, is theoretically no worse than $\mathcal{SROIQ}$ [Krötzsch et al., WWW11]
- Reasoning for $\mathcal{SROELV}$ is:
  - still polynomial (like $\mathcal{SROEL}$) if the number of occurrences of different nominal schemas in an axiom is bounded by a fixed constant;
  - in general, it is exponential (c.f., combined complexity of Datalog is ExpTime)

An Efficient Implementation for Nominal Schemas

Naive grounding

- Naive reasoning directly from the semantics:

Naive grounding

- Naive reasoning directly from the semantics:
  - Ground each axiom containing nominal schemas into exponentially many axioms without nominal schemas.
  - The resulting ontology is in standard DL or OWL and equivalent to the original one.
  - Use any existing reasoning algorithm for the corresponding standard DL on the resulting ontology.

Naive grounding

- Naive reasoning directly from the semantics:
  - Ground each axiom containing nominal schemas into exponentially many axioms without nominal schemas.
  - The resulting ontology is in standard DL or OWL and equivalent to the original one.
  - Use any existing reasoning algorithm for the corresponding standard DL on the resulting ontology.
- Practically inefficient.

Naive grounding example

$$\exists \mathsf{hParent}.\exists \mathsf{married}.\{z\} \sqcap \exists \mathsf{hParent}.\{z\} \sqsubseteq C$$

Naive grounding example

$$\exists \text{hParent}.\exists \text{married}.\{z\} \sqcap \exists \text{hParent}.\{z\} \sqsubseteq C$$

Full grounding:

$$\exists \text{hParent}.\exists \text{married}.\{a\} \sqcap \exists \text{hParent}.\{a\} \sqsubseteq C$$
$$\exists \text{hParent}.\exists \text{married}.\{b\} \sqcap \exists \text{hParent}.\{b\} \sqsubseteq C$$
$$\exists \text{hParent}.\exists \text{married}.\{c\} \sqcap \exists \text{hParent}.\{c\} \sqsubseteq C$$

where a, b, and c are the only individuals in the knowledge base

Naive grounding example

$$\exists R_1.(\exists R_4.\{z\} \sqcap \exists R_5.(\{w\} \sqcap \exists R_6.\{z\})) \sqcap \exists R_2.\{z\} \sqcap \exists R_3.\{w\} \sqsubseteq C$$

Naive grounding example

$$\exists R_1.(\exists R_4.\{z\} \sqcap \exists R_5.(\{w\} \sqcap \exists R_6.\{z\})) \sqcap \exists R_2.\{z\} \sqcap \exists R_3.\{w\} \sqsubseteq C$$

Full grounding:

$$\exists R_1.(\exists R_4.\{a\} \sqcap \exists R_5.(\{a\} \sqcap \exists R_6.\{a\})) \sqcap \exists R_2.\{a\} \sqcap \exists R_3.\{a\} \sqsubseteq C$$
$$\exists R_1.(\exists R_4.\{a\} \sqcap \exists R_5.(\{b\} \sqcap \exists R_6.\{a\})) \sqcap \exists R_2.\{a\} \sqcap \exists R_3.\{b\} \sqsubseteq C$$
$$\exists R_1.(\exists R_4.\{a\} \sqcap \exists R_5.(\{c\} \sqcap \exists R_6.\{a\})) \sqcap \exists R_2.\{a\} \sqcap \exists R_3.\{c\} \sqsubseteq C$$
$$\exists R_1.(\exists R_4.\{a\} \sqcap \exists R_5.(\{b\} \sqcap \exists R_6.\{a\})) \sqcap \exists R_2.\{a\} \sqcap \exists R_3.\{b\} \sqsubseteq C$$
$$\exists R_1.(\exists R_4.\{b\} \sqcap \exists R_5.(\{c\} \sqcap \exists R_6.\{b\})) \sqcap \exists R_2.\{a\} \sqcap \exists R_3.\{c\} \sqsubseteq C$$
$$\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots$$

where a, b, and c are the only individuals in the knowledge base

Defining Optimizations

- Delayed grounding: [Cong et al. at JIST 2012]
- Ordered Resolution: [Adila et al. at RR 2012]

Towards an Efficient Algorithm for DL Extended with Nominal Schemas

Algorithm extension presented at [Markus Krotzsch at Jelia 2010]

- Define a mapping from a normalized OWL EL knowledge base to a Datalog program.
- Make use of an existing Datalog engine to derive all inferences.

OWL EL Normal Form Transformation

$$C(a) \quad R(a,b) \quad A \sqsubseteq \bot \quad \top \sqsubseteq C \quad A \sqsubseteq \{c\} \quad \{a\} \sqsubseteq \{c\}$$

$$A \sqsubseteq C \quad A \sqcap B \sqsubseteq C \quad \exists R.A \sqsubseteq C \quad A \sqsubseteq \exists R.B \quad \exists R.\mathrm{Self} \sqsubseteq C \quad A \sqsubseteq \exists R.\mathrm{Self}$$

$$R \sqsubseteq T \quad R \circ S \sqsubseteq T \quad R \sqcap S \sqsubseteq T \quad A \times B \sqsubseteq R \quad R \sqsubseteq C \times D$$

where $A, B, C, D \in \mathbf{N_C}$, $R, S, T \in \mathbf{N_R}$, *and* $a, b, c \in \mathbf{N_I}$.

KNO.E.SIS

CENTRIA

Input Transformation

$C(a) \mapsto \{\mathtt{subClass}(a, C)\}$	$R(a, b) \mapsto \{\mathtt{subEx}(a, R, b, b)\}$	$a \in \mathbf{N_I} \mapsto \{\mathtt{nom}(a)\}$
$\top \sqsubseteq C \mapsto \{\mathtt{top}(C)\}$	$A \sqsubseteq \bot \mapsto \{\mathtt{bot}(A)\}$	$A \in \mathbf{N_C} \mapsto \{\mathtt{cls}(A)\}$
$\{a\} \sqsubseteq C \mapsto \{\mathtt{subClass}(a, C)\}$	$A \sqsubseteq \{c\} \mapsto \{\mathtt{subClass}(A, c)\}$	$R \in \mathbf{N_R} \mapsto \{\mathtt{rol}(R)\}$
$A \sqsubseteq C \mapsto \{\mathtt{subClass}(A, C)\}$	$A \sqcap B \sqsubseteq C \mapsto \{\mathtt{subConj}(A, B, C)\}$	
$\exists R.\mathsf{Self} \sqsubseteq C \mapsto \{\mathtt{subSelf}(R, C)\}$	$A \sqsubseteq \exists R.\mathsf{Self} \mapsto \{\mathtt{supSelf}(A, R)\}$	
$\exists R.A \sqsubseteq C \mapsto \{\mathtt{subEx}(R, A, C)\}$	$A \sqsubseteq \exists R.B \mapsto \{\mathtt{supEx}(A, R, B, aux_1)\}$	
$R \sqsubseteq T \mapsto \{\mathtt{subRole}(R, T)\}$	$R \circ S \sqsubseteq T \mapsto \{\mathtt{subRChain}(R, S, T)\}$	
$R \sqsubseteq C \times D \mapsto \{\mathtt{supProd}(R, C, D)\}$	$A \times B \sqsubseteq R \mapsto \{\mathtt{subProd}(A, B, R)\}$	
$R \sqcap S \sqsubseteq T \mapsto \{\mathtt{subRConj}(R, S, T)\}$		

Fig. 2. Input translation for K_{inst}

KNO.E.SIS

CENTRIA

Input Transformation: some mappings

$$A \sqsubseteq C \mapsto \text{subClass}(a, C)$$

Input Transformation: some mappings

$$A \sqsubseteq C \mapsto \text{subClass}(a, C)$$

$$R \sqsubseteq S \mapsto \text{subRole}(R, T)$$

Input Transformation: some mappings

$$A \sqsubseteq C \mapsto \text{subClass}(a, C)$$

$$R \sqsubseteq S \mapsto \text{subRole}(R, T)$$

$$A \times B \sqsubseteq R \mapsto \text{subProd}(A, B, R)$$

Set of Rules

(1)	$\texttt{nom}(x) \rightarrow \texttt{inst}(x, x)$
(2)	$\texttt{nom}(x) \wedge \texttt{triple}(x, v, x) \rightarrow \texttt{self}(x, v)$
(3)	$\texttt{top}(z) \wedge \texttt{inst}(x, z') \rightarrow \texttt{inst}(x, z)$
(4)	$\texttt{bot}(z) \wedge \texttt{inst}(u, z) \wedge \texttt{inst}(x, z') \wedge \texttt{cls}(y) \rightarrow \texttt{inst}(x, y)$
(5)	$\texttt{subClass}(y, z) \wedge \texttt{inst}(x, y) \rightarrow \texttt{inst}(x, z)$
(6)	$\texttt{subConj}(y_1, y_2, z) \wedge \texttt{inst}(x, y_1) \wedge \texttt{inst}(x, y_2) \rightarrow \texttt{inst}(x, z)$
(7)	$\texttt{subEx}(v, y, z) \wedge \texttt{triple}(x, v, x') \wedge \texttt{inst}(x', y) \rightarrow \texttt{inst}(x, z)$
(8)	$\texttt{subEx}(v, y, z) \wedge \texttt{self}(x, v) \wedge \texttt{inst}(x, y) \rightarrow \texttt{inst}(x, z)$
(9)	$\texttt{supEx}(y, v, z, x') \wedge \texttt{inst}(x, y) \rightarrow \texttt{triple}(x, v, x')$
(10)	$\texttt{supEx}(y, v, z, x') \wedge \texttt{inst}(x, y) \rightarrow \texttt{inst}(x', z)$
(11)	$\texttt{subSelf}(v, z) \wedge \texttt{self}(x, v) \rightarrow \texttt{inst}(x, z)$
(12)	$\texttt{supSelf}(y, v) \wedge \texttt{inst}(x, y) \rightarrow \texttt{self}(x, v)$
(13)	$\texttt{subRole}(v, w) \wedge \texttt{triple}(x, v, x') \rightarrow \texttt{triple}(x, w, x')$
(14)	$\texttt{subRole}(v, w) \wedge \texttt{self}(x, v) \rightarrow \texttt{self}(x, w)$
(15)	$\texttt{subRChain}(u, v, w) \wedge \texttt{triple}(x, u, x') \wedge \texttt{triple}(x', v, x'') \rightarrow \texttt{triple}(x, w, x'')$
(16)	$\texttt{subRChain}(u, v, w) \wedge \texttt{self}(x, u) \wedge \texttt{triple}(x, v, x') \rightarrow \texttt{triple}(x, w, x')$
(17)	$\texttt{subRChain}(u, v, w) \wedge \texttt{triple}(x, u, x') \wedge \texttt{self}(x', v) \rightarrow \texttt{triple}(x, w, x')$
(18)	$\texttt{subRChain}(u, v, w) \wedge \texttt{self}(x, u) \wedge \texttt{self}(x, v) \rightarrow \texttt{triple}(x, w, x)$
(19)	$\texttt{subRConj}(v_1, v_2, w) \wedge \texttt{triple}(x, v_1, x') \wedge \texttt{triple}(x, v_2, x') \rightarrow \texttt{triple}(x, w, x')$
(20)	$\texttt{subRConj}(v_1, v_2, w) \wedge \texttt{self}(x, v_1) \wedge \texttt{self}(x, v_2) \rightarrow \texttt{self}(x, w)$
(21)	$\texttt{subProd}(y_1, y_2, w) \wedge \texttt{inst}(x, y_1) \wedge \texttt{inst}(x', y_2) \rightarrow \texttt{triple}(x, w, x')$
(22)	$\texttt{subProd}(y_1, y_2, w) \wedge \texttt{inst}(x, y_1) \wedge \texttt{inst}(x, y_2) \rightarrow \texttt{self}(x, w)$
(23)	$\texttt{supProd}(v, z_1, z_2) \wedge \texttt{triple}(x, v, x') \rightarrow \texttt{inst}(x, z_1)$
(24)	$\texttt{supProd}(v, z_1, z_2) \wedge \texttt{self}(x, v) \rightarrow \texttt{inst}(x, z_1)$
(25)	$\texttt{supProd}(v, z_1, z_2) \wedge \texttt{triple}(x, v, x') \rightarrow \texttt{inst}(x', z_2)$
(26)	$\texttt{supProd}(v, z_1, z_2) \wedge \texttt{self}(x, v) \rightarrow \texttt{inst}(x, z_2)$
(27)	$\texttt{inst}(x, y) \wedge \texttt{nom}(y) \wedge \texttt{inst}(x, z) \rightarrow \texttt{inst}(y, z)$
(28)	$\texttt{inst}(x, y) \wedge \texttt{nom}(y) \wedge \texttt{inst}(y, z) \rightarrow \texttt{inst}(x, z)$
(29)	$\texttt{inst}(x, y) \wedge \texttt{nom}(y) \wedge \texttt{triple}(z, u, x) \rightarrow \texttt{triple}(z, u, y)$

Fig. 3. Deduction rules P_{inst}

KNO.E.SIS

CENTRIA

Set of Rules: some mappings

$$\text{subClass}(y, z) \wedge \text{inst}(x, y) \rightarrow \text{inst}(x, z)$$

Set of Rules: some mappings

$$\text{subClass}(y, z) \wedge \text{inst}(x, y) \rightarrow \text{inst}(x, z)$$

$$\text{subRole}(v, w) \wedge \text{triple}(x, v, x') \rightarrow \text{inst}(v, w)$$

Set of Rules: some mappings

$$\text{subClass}(y, z) \wedge \text{inst}(x, y) \rightarrow \text{inst}(x, z)$$

$$\text{subRole}(v, w) \wedge \text{triple}(x, v, x') \rightarrow \text{inst}(v, w)$$

$$\text{subProd}(y_1, y_2, w) \wedge \text{inst}(x, y_1) \wedge \text{inst}(z, y_2) \rightarrow \text{triple}(x, w, x')$$

Extending the Algorithm: Nominal Schemas

- Normalization and Input Transformation.
- Set of Rules.
- Set of facts + Set of Rules +
  Set of Rules derived from axioms with nominal schemas.

Mapping for Axioms Containing Nominal Schemas

$$\mathsf{hasParent}(x, y) \wedge \mathsf{married}(y, z) \wedge \mathsf{hasParent}(x, z) \rightarrow C(x)$$

Mapping for Axioms Containing Nominal Schemas

$$\mathsf{hasParent}(x, y) \wedge \mathsf{married}(y, z) \wedge \mathsf{hasParent}(x, z) \rightarrow C(x)$$

$$\exists \mathsf{hasParent}.\exists \mathsf{married}.\{z\} \sqcap \exists \mathsf{hasParent}.\{z\} \sqsubseteq C$$

Mapping for Axioms Containing Nominal Schemas

$$\text{hasParent}(x, y) \wedge \text{married}(y, z) \wedge \text{hasParent}(x, z) \rightarrow C(x)$$

$$\exists \text{hasParent}.\exists \text{married}.\{z\} \sqcap \exists \text{hasParent}.\{z\} \sqsubseteq C$$

$$\text{triple}(x, \text{hasParent}, y) \wedge \text{triple}(y, \text{married}, z) \wedge \text{triple}(x, \text{hasParent}, z) \wedge \text{nom}(z) \rightarrow \text{inst}(x, C)$$

Execution Example

$$\text{hasFather}(\text{Mike},\ \text{Joe})$$

$$\text{hasParent}(\text{Mike},\ \text{Mary})$$

$$\text{married}(\text{Joe},\ \text{Mary})$$

$$\text{hasFather} \sqsubseteq \text{hasParent}$$

$$\text{hasParent}(x, y) \wedge \text{married}(y, z) \wedge \text{hasParent}(x, z) \rightarrow C(x)$$

Execution Example

$$\text{triple}(\text{Mike}, \text{hasFather}, \text{Joe})$$
$$\text{triple}(\text{Mike}, \text{hasParent}, \text{Mary})$$
$$\text{triple}(\text{Joe}, \text{married}, \text{Mary})$$
$$\text{subRole}(\text{hasFather}, \text{hasParent})$$

$$\text{subRole}(v, w) \wedge \text{triple}(x, v, x') \rightarrow \text{triple}(x, w, x')$$

$$\text{triple}(x, \text{hasParent}, y) \wedge \text{triple}(y, \text{married}, z) \wedge \text{triple}(x, \text{hasParent}, z) \wedge$$
$$\text{nom}(z) \rightarrow \text{inst}(x, C)$$

Experimental Results

Ontology	Individuals	no ns	1 ns	2 ns	3 ns	4 ns	5 ns
Rex (full ground.)	100	263	263 (321)	267 (972)	273	275	259
	1000	480	518 (1753)	537 (OOM)	538	545	552
	10000	2904	2901 (133179)	3120 (OOM)	3165	3192	3296
Spatial (full ground.)	100	22	191 (222)	201 (1163)	198	202	207
	1000	134	417 (1392)	415 (OOM)	421	431	432
	10000	1322	1792 (96437)	1817 (OOM)	1915	1888	1997
Xenopus (full ground.)	100	62	332 (383)	284 (1629)	311	288	280
	1000	193	538 (4751)	440 (OOM)	430	456	475
	10000	1771	2119 (319013)	1843 (OOM)	1886	2038	2102

Conclusions

- DL Logic Rules
  - Allows to express simple rules
  - Push the DL fragments
- Nominal Schemas
  - Convoluted Rules
  - Efficient implementations

Non-monotonic rule extension to OWL

Why Non-monotonic Extensions?

- Open World Assumption (OWA) in general preferable on the Web
  - Without clinical test, no assumptions can be made on outcome
- But with complete knowledge, Closed World Assumption (CWA) is better
  - Patient's medication is fully known
- Requirement for local closure of certain information

Why Non-monotonic Extensions?

$$\begin{aligned} \mathit{Person} &\sqsubseteq \mathit{HeartLeft} \sqcup \mathit{HeartRight} \\ \mathit{HeartLeft} \sqcap \mathit{HeartRight} &\sqsubseteq \bot \\ \mathit{Person} &\sqsubseteq \exists \mathit{has}.\mathit{SpinalColumn} \\ \exists \mathit{has}.\mathit{SpinalColumn} &\sqsubseteq \mathit{Vertebrate} \\ \mathit{Person}(\mathit{Bob}) & \end{aligned}$$

$\Rightarrow \mathit{Vertebrate}(\mathit{Bob})$, $\mathit{Person} \sqsubseteq \mathit{Vertebrate}$, and $\exists x.\mathit{SpinalColumn}(x)$ derivable

Why Non-monotonic Extensions?

$$\begin{aligned}
Person &\sqsubseteq HeartLeft \sqcup HeartRight \\
HeartLeft \sqcap HeartRight &\sqsubseteq \bot \\
Person &\sqsubseteq \exists has.SpinalColumn \\
\exists has.SpinalColumn &\sqsubseteq Vertebrate \\
Person(Bob) & \\
SSN_OK(x) &\leftarrow hasSSN(x, y) \\
\mathbf{f} &\leftarrow Person(x), \mathbf{not} SSN_OK(x) \\
HeartLeft(x) &\leftarrow Vertebrate(x), \mathbf{not} HeartRight(x)
\end{aligned}$$

$\Rightarrow$ $HeartLeft(Bob)$; $hasSSN(Bob, y)$ for ground y required

Model defaults and exceptions, and integrity constraints

KNO.E.SIS

CENTRIA

Why focus on Rules?

- Non-monotonic extensions have been studied directly for DLs
- Extensions for, e.g., default logic, epistemic logic, and circumscription
- But non-trivial and few results in terms of implementations (apart from ad-hoc solutions in, e.g., recommender systems)
- Rules easily extended to non-monotonic features
- Well-studied field Logic Programming including fast reasoners
- Leverage the Knowledge and Reasoners available

Ways to Combine Ontologies and Rules

Non-trivial matter because of Non-monotonicity

- Rules "on top" of ontologies
  - First define concepts then define rules "on top"
  - Rules deal with ontologies as external code
  - Separate rule and ontology predicates
- Tight (full) integration
  - Allow for "defining" predicates both in the ontology and the rule layer
    - Rules may use concepts defined in the ontology
    - Ontology can use predicates of the rules
- Modular combination
  - Trades some expressiveness for an easier to implement interface integration

Difficulties of Tight Integration

- Rule languages (LP) use CWA
- Ontology languages (DL) use OWA
- What if a predicate is "defined" using both DL and LP?
  - Should its negation be assumed by default?
  - Or should it be kept open?
  - How exactly can one define what is CWA or OWA in this context?

Hybrid MKNF Knowledge Bases

- Seamless integration, expressive, yet competitive w.r.t. computational complexity
- Introduced in [Motik and Rosati, IJCAI07] (extended in [Motik and Rosati, JACM10])
- Based on Logics of Minimal Knowledge and Negation as Failure: first-order logics with equality and modal operators **K** and **not** [Lifschitz, IJCAI91]
- Consist of a (decidable) DL knowledge base $\mathcal{O}$ and a finite set of rules, $\mathcal{P}$, of the form

$$\mathbf{K}H_1 \vee \ldots \vee \mathbf{K}H_l \leftarrow \mathbf{K}A_1, \ldots, \mathbf{K}A_n, \mathbf{not}\,B_1, \ldots, \mathbf{not}\,B_m$$

- Combined decidability ensured by DL-safety (restriction of application of rules to known individuals)

MKNF KB Example

$$PortCity(Barcelona) \quad OnSea(Barcelona, Mediterranean)$$
$$PortCity(Hamburg) \quad NonSeaSideCity(Hamburg)$$
$$RainyCity(Manchester) \quad Has(Manchester, AquaticsCenter)$$
$$Recreational(AquaticsCenter)$$
$$SeaSideCity \sqsubseteq \exists Has.Beach$$
$$Beach \sqsubseteq Recreational$$
$$\exists Has.Recreational \sqsubseteq RecreationalCity$$
$$\mathbf{K}SeaSideCity(x) \leftarrow \mathbf{K}PortCity(x), \mathbf{not}\, NonSeaSideCity(x)$$
$$\mathbf{K}interestingCity(x) \leftarrow \mathbf{K}RecreationalCity(x), \mathbf{not}\, RainyCity(x)$$
$$\mathbf{K}hasOnSea(x) \leftarrow \mathbf{K}OnSea(x, y)$$
$$\mathbf{K}false \leftarrow \mathbf{K}SeaSideCity(x), \mathbf{not}\, hasOnSea(x)$$
$$\mathbf{K}summerDestination(x) \leftarrow \mathbf{K}interestingCity(x), \mathbf{K}OnSea(x, y)$$

KNO.E.SIS

CENTRIA

Properties of Hybrid MKNF

- Generalizes/captures (sometimes not entirely) quite a number of different approaches
- Faithful w.r.t. Stable Models for empty $\mathcal{O}$ and w.r.t. OWL for empty $\mathcal{P}$
- Data complexity of instance checking in MKNF:

rules	$\mathcal{DL} = \emptyset$	$\mathcal{DL} \in$ P	$\mathcal{DL} \in$ coNP
definite	P	P	coNP
stratified	P	P	Δ_2^p
normal	coNP	coNP	Π_2^p
disjunctive	Π_2^p	Π_2^p	Π_2^p

Problems of two-valued Hybrid MKNF

- Models have to be guessed and checked
- Unrestricted rules increase computational complexity
- Queries for particular information require computation of the entire model
- Limited robustness, e.g., w.r.t. merging of KBs ($\mathbf{K}u \leftarrow \mathbf{not}u$)

[Knorr et al., AI11] provides alternative based on well-founded semantics for non-disjunctive logic programs

Stable Models vs. Well-Founded Model in LP

$$p \leftarrow \mathbf{not} q \qquad q \leftarrow \mathbf{not} p \qquad a \leftarrow \mathbf{not} b \qquad b \leftarrow$$

has two stable models $\{p, b\}$ and $\{q, b\}$, while the unique well-founded model assigns **t** to b, **f** to a, and **u** to both p and q. For

$$p \leftarrow \mathbf{not} p \qquad q \leftarrow \mathbf{not} q \qquad a \leftarrow \mathbf{not} b \qquad b \leftarrow$$

the well-founded model is the same, but there are no stable models!

Stable Models vs. Well-Founded Model in LP

Stable Models/Answer Sets

- More expressive language
- More derivable information
- Fast ASP solvers available

Well-founded Model

- Lower computational complexity
- always exists
- top-down derivations possible

Similar for combinations of rules and ontologies

Properties of Well-Founded MKNF

- Sound w.r.t. two-valued MKNF semantics
- Faithful w.r.t. first-order semantics for empty $\mathcal{P}$ and w.r.t. the Well-Founded Semantics for empty $\mathcal{O}$
- given complexity $\mathcal{C}$ for instance checking in $\mathcal{O}$ we obtain a data complexity $\mathrm{P}^{\mathcal{C}}$; for $\mathcal{C} = \mathrm{P}$, polynomial data complexity
- Top-down procedure **SLG($\mathcal{O}$)** [Alferes et al., ACM TOCL13] combining a DL reasoner and XSB Prolog, special procedures defined for OWL 2 QL and a large fragment of OWL 2 EL

Non-monotonic DL Extension with MKNF

$\mathcal{ALCK}_{\mathcal{NF}}$ [Donini et al., ACM TOCL02]

$\mathcal{ALC}$ with MKNF logic-style modal operators **K** – minimal knowledge – and **A** – autoepistemic assumption (corresponds to $\neg$**not**)

K can be used to derive new information, **A** to verify if information is already known

Different expressiveness compared to Hybrid MKNF

Non-monotonic Features of $\mathcal{ALCK}_{\mathcal{NF}}$

from [Donini et al., ACM TOCL02]

- Defaults:

$$\mathbf{K}I \sqcap \mathbf{K}(employee \sqcap \exists belongsTo.programmingDept) \sqcap$$
$$\neg \mathbf{A}manager \sqsubseteq \mathbf{K}(engineer \sqcup mathematican)$$

- Integrity Constraints:

$$\mathbf{K}employee \sqsubseteq (\mathbf{A}male \sqcup \mathbf{A}female)$$

$$\mathbf{K}employee \sqsubseteq \exists \mathbf{A}SSN.\mathbf{A}valid$$

Non-monotonic Features of $\mathcal{ALCK}_{\mathcal{NF}}$

- Concept and Role Closure

$$\neg \mathit{UScitizen}(\mathit{Paula}) \qquad \mathit{Manages}(\mathit{Ann}, \mathit{Marc})$$
$$\neg \mathit{UScitizen}(\mathit{Carl}) \qquad \mathit{UScitizen}(\mathit{Marc})$$

adding $(\forall \mathbf{K}\mathit{Manages}.\mathbf{K}\mathit{UScitizen})(\mathit{Ann})$ closes the role

adding $\exists \mathbf{K}\mathit{Manages}.\mathbf{A}\neg \mathit{UScitizen}(\mathit{Ann})$ closes the concept

Can We find a joint formalism for both MKNF extensions?

- Contribute towards a unifying logic
- Reconcile OWL and Datalog together with CWA extensions (on both sides)
- Usage of one (DL-style) syntax in opposite to common hybrid languages
- Coverage of many different previous approaches

$\mathcal{SROIQV}(\mathcal{B}^s, \times)\mathcal{K}_{\mathcal{NF}}$

- OWL 2 DL ($\mathcal{SROIQ}$) with concept products ($\times$ - [Krötzsch, SSW10]) and Boolean constructors over simple roles ($\mathcal{B}^s$ - [Rudolph et al., JELIA08])
- Nominal schemas ($\mathcal{V}$ - [Krötzsch et al., WWW11]) - variable nominals that can only bind to known individuals
- MKNF logic-style modal operators **K** – minimal knowledge – and **A** – autoepistemic assumption – ($\mathcal{K}_{\mathcal{NF}}$ - from $\mathcal{ALCK}_{\mathcal{NF}}$ [Donini et al., ACM TOCL02])

Syntax

signature $\Sigma = \langle N_I, N_C, N_R, N_V \rangle$

Definition

The set of $\mathcal{SROIQV}(\mathcal{B}^s, \times)\mathcal{K}_{\mathcal{NF}}$ concepts C and (*simple*/*non-simple*) $\mathcal{SROIQV}(\mathcal{B}^s, \times)\mathcal{K}_{\mathcal{NF}}$ roles R ($\mathsf{R}^s/\mathsf{R}^n$) are defined by the following grammar.

$$
\begin{aligned}
\mathsf{R}^s ::= & N_R^s \mid (N_R^s)^- \mid U \mid N_C \times N_C \mid \neg \mathsf{R}^s \mid \mathsf{R}^s \sqcap \mathsf{R}^s \mid \mathsf{R}^s \sqcup \mathsf{R}^s \mid \\
& \mathbf{K}\mathsf{R}^s \mid \mathbf{A}\mathsf{R}^s \\
\mathsf{R}^n ::= & N_R^n \mid (N_R^n)^- \mid U \mid N_C \times N_C \mid \mathbf{K}\mathsf{R}^n \mid \mathbf{A}\mathsf{R}^n \\
\mathsf{R} ::= & \mathsf{R}^s \mid \mathsf{R}^n \\
\mathsf{C} ::= & \top \mid \bot \mid N_C \mid \{N_I\} \mid \{N_V\} \mid \neg \mathsf{C} \mid \mathsf{C} \sqcap \mathsf{C} \mid \mathsf{C} \sqcup \mathsf{C} \mid \\
& \exists \mathsf{R}.\mathsf{C} \mid \forall \mathsf{R}.\mathsf{C} \mid \exists \mathsf{R}^s.\mathsf{Self} \mid \leqslant k\, \mathsf{R}^s.\mathsf{C} \mid \geqslant k\, \mathsf{R}^s.\mathsf{C} \mid \mathbf{K}\mathsf{C} \mid \mathbf{A}\mathsf{C}
\end{aligned}
$$

Semantics – Principal Notions

- Based on interpretations $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ plus variable assignments (for nominal variables) mapping each variable to the interpretation of one element in N_I
- Variant of *Standard Name Assumption* applied: essentially $\mathcal{I}$ is a bijective function on N_I while still allowing that elements of N_I may be identified ($\rightarrow$ only one Δ)

An *MKNF structure* is a triple $(\mathcal{I}, \mathcal{M}, \mathcal{N})$ where $\mathcal{I}$ is an interpretation, $\mathcal{M}$ and $\mathcal{N}$ are sets of interpretations, and $\mathcal{I}$ and all interpretations in $\mathcal{M}$ and $\mathcal{N}$ are defined over Δ. For any such $(\mathcal{I}, \mathcal{M}, \mathcal{N})$ and assignment $\mathcal{Z}$, the function $\cdot^{(\mathcal{I},\mathcal{M},\mathcal{N}),\mathcal{Z}}$ is defined.

Function $\cdot^{(\mathcal{I},\mathcal{M},\mathcal{N}),\mathcal{Z}}$ (parts of it)

Syntax	Semantics
a	$a^{\mathcal{I}} \in \Delta$
x	$\mathcal{Z}(x) \in \Delta$
$\neg C$	$\Delta \setminus C^{(\mathcal{I},\mathcal{M},\mathcal{N}),\mathcal{Z}}$
$\{t\}$	$\{a \mid a \approx t^{(\mathcal{I},\mathcal{M},\mathcal{N}),\mathcal{Z}}\}$
$\mathbf{K}C$	$\bigcap_{\mathcal{J}\in\mathcal{M}} C^{(\mathcal{J},\mathcal{M},\mathcal{N}),\mathcal{Z}}$
$\mathbf{A}C$	$\bigcap_{\mathcal{J}\in\mathcal{N}} C^{(\mathcal{J},\mathcal{M},\mathcal{N}),\mathcal{Z}}$
$\mathbf{K}R$	$\bigcap_{\mathcal{J}\in\mathcal{M}} R^{(\mathcal{J},\mathcal{M},\mathcal{N}),\mathcal{Z}}$
$\mathbf{A}R$	$\bigcap_{\mathcal{J}\in\mathcal{N}} R^{(\mathcal{J},\mathcal{M},\mathcal{N}),\mathcal{Z}}$
$C \sqsubseteq D$	$C^{(\mathcal{I},\mathcal{M},\mathcal{N}),\mathcal{Z}} \subseteq D^{(\mathcal{I},\mathcal{M},\mathcal{N}),\mathcal{Z}}$

(Monotonic) Semantics

Definition

$(\mathcal{I}, \mathcal{M}, \mathcal{N})$ *satisfies* axiom α, written $(\mathcal{I}, \mathcal{M}, \mathcal{N}) \models \alpha$, if $(\mathcal{I}, \mathcal{M}, \mathcal{N}), \mathcal{Z} \models \alpha$ for all variable assignments $\mathcal{Z}$.

A (non-empty) set of interpretations $\mathcal{M}$ *satisfies* α, written $\mathcal{M} \models \alpha$, if $(\mathcal{I}, \mathcal{M}, \mathcal{M}) \models \alpha$ holds for all $\mathcal{I} \in \mathcal{M}$.

$\mathcal{M}$ *satisfies* a $\mathcal{SROIQV}(\mathcal{B}^s, \times)\mathcal{K}_{\mathcal{NF}}$ knowledge base *KB*, written $\mathcal{M} \models KB$, if $\mathcal{M} \models \alpha$ for all axioms $\alpha \in KB$.

(Non-monotonic) Semantics

Definition

Given a $\mathcal{SROIQV}(\mathcal{B}^s, \times)\mathcal{K}_{\mathcal{NF}}$ knowledge base KB, a (non-empty) set of interpretations $\mathcal{M}$ is an *MKNF model* of KB if

(1) $\mathcal{M} \models KB$, and

(2) for each $\mathcal{M}'$ with $\mathcal{M} \subset \mathcal{M}'$, $(\mathcal{I}', \mathcal{M}', \mathcal{M}) \not\models KB$ for some $\mathcal{I}' \in \mathcal{M}'$.

Example

C Persons whose parents are married

$$\mathsf{HasParent}(\mathsf{mary}, \mathsf{john}) \quad (1)$$

$$(\exists \mathsf{HasParent}.\exists \mathsf{Married}.\{\mathsf{john}\})(\mathsf{mary}) \quad (2)$$

$$\exists \mathsf{HasParent}.\{z\} \sqcap \exists \mathsf{HasParent}.\exists \mathsf{Married}.\{z\} \sqsubseteq C \quad (3)$$

We can substitute (3) by

$$\mathbf{K}\exists \mathsf{HasParent}.\{z\} \sqcap \mathbf{K}\exists \mathsf{HasParent}.\exists \mathsf{Married}.\{z\} \sqsubseteq \mathbf{K}C$$

Example

C Persons whose parents are married

$$\mathsf{HasParent}(\mathsf{mary}, \mathsf{john}) \quad (1)$$

$$(\exists \mathsf{HasParent}.\exists \mathsf{Married}.\{\mathsf{john}\})(\mathsf{mary}) \quad (2)$$

$$\exists \mathsf{HasParent}.\{z\} \sqcap \exists \mathsf{HasParent}.\exists \mathsf{Married}.\{z\} \sqsubseteq C \quad (3)$$

We can also substitute (3) by

$$\mathbf{K}\exists \mathsf{HasParent}.\{z\} \sqcap \mathbf{K}\exists \mathsf{HasParent}.\exists \mathsf{Married}.\{z\} \sqsubseteq \mathbf{A}C$$

Example

C Persons whose parents are married

$$\text{HasParent}(\text{mary}, \text{john}) \quad (1)$$

$$(\exists \text{HasParent}.\exists \text{Married}.\{\text{john}\})(\text{mary}) \quad (2)$$

$$\exists \text{HasParent}.\{z\} \sqcap \exists \text{HasParent}.\exists \text{Married}.\{z\} \sqsubseteq C \quad (3)$$

We can also substitute (3) by

$$\exists \text{HasParent}.\{z\} \sqcap \exists \text{HasParent}.\exists \neg \mathbf{A}\text{Married}.\{z\} \sqsubseteq C$$

Now C are Persons that are known to be not married

Decidability

- First, reduce reasoning in $\mathcal{SROIQV}(\mathcal{B}^s, \times)\mathcal{K}_{\mathcal{NF}}$ to reasoning in $\mathcal{SROIQ}(\mathcal{B}^s)\mathcal{K}_{\mathcal{NF}}$ by grounding and by simulating concept products
- Then, follow approach for $\mathcal{ALCK}_{\mathcal{NK}}$:
  - each model of a knowledge base in $\mathcal{SROIQ}(\mathcal{B}^s)\mathcal{K}_{\mathcal{NF}}$ is cast into a $\mathcal{SROIQ}(\mathcal{B}^s)$ KB. Consequently, reasoning in $\mathcal{SROIQ}(\mathcal{B}^s)\mathcal{K}_{\mathcal{NF}}$ is reduced to a number of reasoning tasks in the non-modal $\mathcal{SROIQ}(\mathcal{B}^s)$
  - For simplicity, appearance of modal operators restricted to simple KBs as in $\mathcal{ALCK}_{\mathcal{NF}}$ (finitely many, finite representations of models)

(Monotonic) Coverage

- $\mathcal{SROIQ}$ (a.k.a. OWL 2 DL);
- The tractable profiles OWL 2 EL, OWL 2 RL, OWL 2 QL;
- RIF-Core, i.e., *n*-ary Datalog, interpreted as DL-safe Rules (general case new result in [Knorr et al., ECAI12]);
- DL-safe SWRL [Motik et al., JWS05], $\mathcal{AL}$-log [Donini et al., JIIS98], and CARIN [Levy and Rousset, AI98].

(Non-monotonic) Coverage

- $\mathcal{ALCK}_{\mathcal{NF}}$ [Donini et al., ACM TOCL02]; includes notions of concept and role closure present in this formalism;
- Closed Reiter defaults covered through the coverage of $\mathcal{ALCK}_{\mathcal{NF}}$; includes coverage of DLs extended with default rules [Baader and Hollunder, JAR95];
- Hybrid MKNF [Motik and Rosati, JACM10];
- Answer Set Programming, i.e., disjunctive Datalog with classical negation and non-monotonic negation under the answer set semantics; follows from the coverage of Hybrid MKNF.

N-nary Datalog

$N_R = N_{P,2} \cup \{U\} \cup S$, where S is a special set of roles: If $P \in N_{P,>2}$ has arity k, then $P_1, \ldots, P_k \in S$ are unique binary predicates associated with P;

Translation: $\mathrm{dl}(P(t_1, \ldots, t_k)) := \exists U.(\exists P_1.\{t_1\} \sqcap \ldots \sqcap \exists P_k.\{t_k\})$;

Family of interpretations of $\mathcal{J}$ for interpretation $\mathcal{I}$ of Datalog RB:

(a) To each $(d_1, \ldots, d_k) \in P^{\mathcal{I}}$, assign a unique element e in Δ (i.e., we define a total, injective function from the set of tuples to Δ).

(d) For each $P \in N_{P,>2}$, if $(d_1, \ldots, d_k) \in P^{\mathcal{I}}$, then $(e, d_i) \in P_i^{\mathcal{J}}$, where e is the element assigned to $(d_1, \ldots, d_k)$ in point (a).

Hybrid MKNF

Seamless integration of DL ontology $\mathcal{O}$ and rules of the form

$$\mathbf{K}H_1 \vee \mathbf{K}H_l \leftarrow \mathbf{K}A_1, \ldots, \mathbf{K}A_n, \mathbf{not}B_1, \ldots, \mathbf{not}B_m$$

Based on the n-nary Datalog embedding, additionally:

$$\begin{aligned}&\mathsf{dl}(\mathbf{K}H_1 \vee \mathbf{K}H_l \leftarrow \mathbf{K}A_1, \ldots, \mathbf{K}A_n, \mathbf{not}B_1, \ldots, \mathbf{not}B_m) := \\ &\mathbf{K}\mathsf{dl}(A_1) \sqcap \ldots \sqcap \mathbf{K}\mathsf{dl}(A_n) \sqcap \neg\mathbf{A}\mathsf{dl}(B_1) \sqcap \ldots \sqcap \neg\mathbf{A}\mathsf{dl}(B_m) \\ &\sqsubseteq \mathbf{K}\mathsf{dl}(H_1) \sqcup \ldots \sqcup \mathbf{K}\mathsf{dl}(H_l)\end{aligned}$$

KNO.E.SIS

CENTRIA

Example

$$\mathbf{K}C(x) \leftarrow \mathbf{K}\mathsf{HasParent}(x, y), \mathbf{K}\mathsf{HasParent}(x, z), \mathbf{K}(y \not\approx z), \\ \mathbf{not}\mathsf{Married}(y, z).$$

can be translated into

$$\mathbf{K}\exists U.(\{x\} \sqcap \exists \mathsf{HasParent}.\{y\}) \sqcap \mathbf{K}\exists U.(\{x\} \sqcap \\ \exists \mathsf{HasParent}.\{z\}) \sqcap \mathbf{K}\exists U.(\{y\} \sqcap \exists \not\approx .\{z\}) \sqcap \neg\mathbf{A}\exists U.(\{y\} \sqcap \\ \exists \mathsf{Married}.\{z\}) \sqsubseteq \mathbf{K}\exists U.(\{x\} \sqcap C)$$

Thank you!

References

Rules expressible in OWL:

- Description logic programs: combining logic programs with description logic. Benjamin Grosof, Ian Horrocks, Raphael Volz, and Stefan Decker, WWW 2003, 48–57.
- Description logic rules. Markus Krötzsch, Sebastian Rudolph, and Pascal Hitzler, ECAI 2008, 80–84.
- ELP: tractable rules for OWL 2. Markus Krötzsch, Sebastian Rudolph, and Pascal Hitzler, ISWC 2008, 649–664.
- Cheap boolean role constructors for description logics. Sebastian Rudolph, Markus Krötzsch, and Pascal Hitzler, JELIA 2008, 362–374.
- Description logic rules. Markus Krötzsch, Studies on the Semantic Web, Vol. 08, 2010.

References

Rules expressible in OWL and with nominal schemas:

- A better uncle for OWL: nominal schemas for integrating rules and ontologies. Markus Krötzsch, Frederick Maier, Adila A. Krisnadhi, and Pascal Hitzler, WWW 2011, 645–654.
- Extending description logic rules. David Carral Martínez and Pascal Hitzler, ESWC 2012, 345–359.
- A tableau algorithm for description logics with nominal schemas. Adila Krisnadhi and Pascal Hitzler, RR2012, 234-237.
- A resolution procedure for description logics with nominal schemas. Cong Wang and Pascal Hitzler, JIST 2012, 1-16.

References

Nonmonotonic Extensions:

- Description logics of minimal knowledge and negation as failure. Francesco M. Donini, Daniele Nardi, and Riccardo Rosati, ACM Transactions on Computational Logic, 3(2), 2002, 227–252.
- Reconciling Description Logics and Rules. Boris Motik and Riccardo Rosati, Journal of the ACM, 57(5), 2010, 1–62.
- Local closed world reasoning with description logics under the well-founded semantics. Matthias Knorr, José J. Alferes, and Pascal Hitzler, Artificial Intelligence, 175(9-10), 2011, 1528–1544.
- Reconciling OWL and non-monotonic rules for the Semantic Web. Matthias Knorr, Pascal Hitzler, and Frederick Maier, ECAI 2012, 474–479.