Technische Universität Dresden
Fakultät Informatik

OWL 2 Profiles

An Introduction to Lightweight Ontology Languages

Markus Krötzsch
TU Dresden

ICCL Summer School 2015

September 18, 2015

Plan

Goal:
Learn about the lightweight ontology languages, and understand why they are lightweight.

1) Review: OWL and DLs
2) Overview: OWL 2 Profiles
3) Reasoning in OWL RL: Instance Retrieval
4) Reasoning in OWL EL: Classification
5) Reasoning in OWL QL: Query Answering
6) Limits of Tractability
7) Advanced Features

The Web Ontology Language OWL

The OWL Language

- W3C standard since 2004, updated in 2009
- An ontology language with two sides:
  - Descriptive: express expert knowledge formally
  - Logical: draw conclusions from this knowledge
    → reasoning
- Compatibility with important technology standards
  - Unicode, IRI, XML Schema, RDF, RDF Schema

The Data Model of OWL

- Ontologies use a vocabulary of **entities**
  - Classes
  - Properties
  - Individuals and data literals
- Entities are combined to form **expressions**
- Relationships of expressions are described by **axioms**

Syntaxes and Semanticses

- 5 official syntactic formats:
  - Functional-Style Syntax
  - Manchester Syntax
  - OWL/XML Syntax
  - RDF-based syntaxes (RDF/XML, Turtle)
- 2 formal semantics:
  - Direct Semantics
  - RDF-Based Semantics


LOGIC LANE

Reasoning Tasks

- Every ontology has infinitely many conclusions
- Which conclusions are we interested in?
  - Instance checking
  - Class subsumption
  - Ontology consistency
  - Class consistency (coherence)

→ Tasks can be reduced to each other with little effort

Hardness of Reasoning

- Main requirements:
  - Soundness: only correct conclusions are computed
  - Completeness: no correct conclusion is missed
- Reasoning is hard:
  - Undecidable for RDF-Based Semantics
  - N2ExpTime-complete for Direct Semantics

→ OWL Profiles: sub-languages with easier reasoning

OWL and Description Logics: Basic Features

	OWL Functional-Style Syntax	DL Syntax
Axioms	SubClassOf(*C D*)	$C \sqsubseteq D$
	ClassAssertion(*C a*)	$C(a)$
	ObjectPropertyAssertion(*P a b*)	$P(a, b)$
Class expressions	ObjectIntersectionOf(*C D*)	$C \sqcap D$
	ObjectUnionOf(*C D*)	$C \sqcup D$
	ObjectComplementOf(*C*)	$\neg C$
	owl:Thing	$\top$
	owl:Nothing	$\bot$
	ObjectSomeValuesFrom(*P C*)	$\exists P.C$
	ObjectAllValuesFrom(*P C*)	$\forall P.C$
Property expressions	ObjectInverseOf(*P*)	P^{-}

Example Axioms in DL Syntax

$\mathit{FelisCatus}(\mathit{silvester})$	Silvester is a cat.
$\mathit{preysOn}(\mathit{silvester}, \mathit{tweety})$	Silvester preys on Tweety.
$\mathit{FelisCatus} \sqsubseteq \mathit{Mammalia}$	Cats are mammals.
$\exists \mathit{preysOn}.\top \sqsubseteq \mathit{Predator}$	What preys on something is a predator.
$\top \sqsubseteq \forall \mathit{preysOn}.\mathit{Animal}$	What is preyed on is an animal.
$\mathit{Animal} \sqcap \mathit{PlaysChess} \sqsubseteq \mathit{HomoSapiens}$	All animals that play chess are humans.
$\mathit{Mammalia} \sqsubseteq \exists \mathit{hasFather}.\mathit{Mammalia}$	Every mammal has a mammal father.

OWL Direct Semantics

- Based on first-order logic interpretations
- Direct correspondences:
  - classes → sets
  - properties → relations
  - individuals → domain elements
- Equivalent to translating OWL to first-order logic

OWL Direct Semantics

Expression ex	Interpretation $ex^{\mathcal{I}}$
$C \sqcap D$	$C^{\mathcal{I}} \cap D^{\mathcal{I}}$
$C \sqcup D$	$C^{\mathcal{I}} \cup D^{\mathcal{I}}$
$\neg C$	$\Delta^{\mathcal{I}} \setminus C^{\mathcal{I}}$
$\top$	$\Delta^{\mathcal{I}}$
$\bot$	$\emptyset$
$\exists P.C$	$\{e \mid \text{there is } f \text{ with } \langle e, f \rangle \in P^{\mathcal{I}} \text{ and } f \in C^{\mathcal{I}}\}$
$\forall P.C$	$\{e \mid \text{for all } f \text{ with } \langle e, f \rangle \in P^{\mathcal{I}} \text{ we have } f \in C^{\mathcal{I}}\}$
P^{-}	$\{\langle f, e \rangle \mid \langle e, f \rangle \in P^{\mathcal{I}}\}$

OWL RDF-Based Semantics

- Based on translating OWL ontologies to RDF graphs
- Interpretations defined on graphs
  - Applicable to all RDF graphs, even if not from OWL
- Sometimes stronger (more entailments), sometimes weaker (fewer entailments) than Direct Semantics
- Direct Semantics and RDF-Based Semantics agree under reasonable conditions

Reasoning in the OWL Profiles

Defining Language Profiles by Grammars

- The OWL sublanguage introduced above is ALCI
- Can be described by a formal grammar:

$\mathcal{ALCI}$

$$
\begin{aligned}
\textbf{Axiom} &::= \textbf{C} \sqsubseteq \textbf{C} \mid \textbf{C}(\textbf{IName}) \mid \textbf{P}(\textbf{IName}, \textbf{IName}) \\
\textbf{C} &::= \textbf{CName} \mid \top \mid \bot \mid \textbf{C} \sqcap \textbf{C} \mid \textbf{C} \sqcup \textbf{C} \mid \neg\textbf{C} \mid \exists\textbf{P}.\textbf{C} \mid \forall\textbf{P}.\textbf{C} \\
\textbf{P} &::= \textbf{PName} \mid \textbf{PName}^{-}
\end{aligned}
$$

A Tiny Version of OWL EL

- OWL EL is based on the description logic EL(++)
- Typical applications in ontology engineering

$\mathcal{EL}$ *tiny*

$$
\begin{aligned}
\mathbf{Axiom} &::= \mathbf{C} \sqsubseteq \mathbf{C} \mid \mathbf{C(IName)} \mid \mathbf{P(IName, IName)} \\
\mathbf{C} &::= \mathbf{CName} \mid \top \mid \bot \mid \mathbf{C} \sqcap \mathbf{C} \mid \exists \mathbf{P.C} \\
\mathbf{P} &::= \mathbf{PName}
\end{aligned}
$$

A Tiny Version of OWL RL

- OWL RL is a “rule language”
- Typical applications in data management

$\mathcal{RL}\,tiny$

$$
\begin{aligned}
\mathbf{Axiom} &::= \mathbf{CL} \sqsubseteq \mathbf{CR} \mid \mathbf{CR(IName)} \mid \mathbf{P(IName, IName)} \\
\mathbf{CL} &::= \mathbf{CName} \mid \bot \mid \mathbf{CL} \sqcap \mathbf{CL} \mid \mathbf{CL} \sqcup \mathbf{CL} \mid \exists \mathbf{P.CL} \\
\mathbf{CR} &::= \mathbf{CName} \mid \bot \mid \mathbf{CR} \sqcap \mathbf{CR} \mid \neg \mathbf{CL} \mid \forall \mathbf{P.CR} \\
\mathbf{P} &::= \mathbf{PName} \mid \mathbf{PName}^{-}
\end{aligned}
$$

A Tiny Version of OWL QL

- OWL QL is a "query language"
- Typical applications in data access

$\mathcal{QL}$ *tiny*

$$
\begin{aligned}
\textbf{Axiom} &::= \textbf{CL} \sqsubseteq \textbf{CR} \mid \textbf{CR}(\textbf{IName}) \mid \textbf{P}(\textbf{IName}, \textbf{IName}) \\
\textbf{CL} &::= \textbf{CName} \mid \top \mid \bot \mid \exists \textbf{P}.\top \\
\textbf{CR} &::= \textbf{CName} \mid \top \mid \bot \mid \textbf{CR} \sqcap \textbf{CR} \mid \neg \textbf{CL} \mid \exists \textbf{P}.\textbf{CR} \\
\textbf{P} &::= \textbf{PName} \mid \textbf{PName}^{-}
\end{aligned}
$$

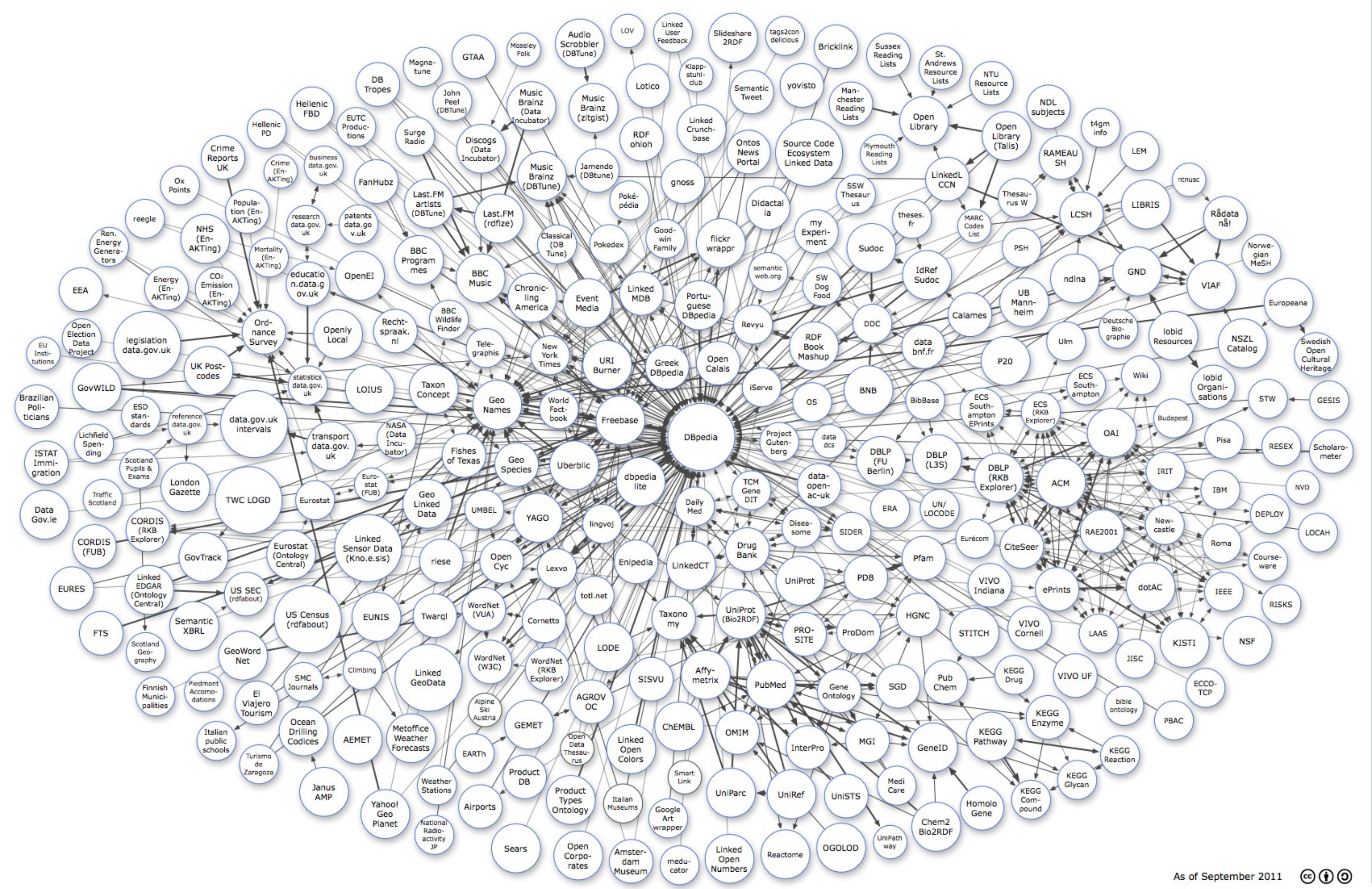
Tiny OWL Profiles: Feature Overview

- RL and QL allow for inverse properties, EL doesn't.
- Features for sub- and superclasses:

Sub	$\top$	$\bot$	$\sqcap$	$\sqcup$	$\neg$	$\exists$	$\exists\top$	$\forall$	$\forall\bot$
RL		×	×	×		×	×		
EL	×	×	×			×	×		
QL	×	×					×		

Sup	$\top$	$\bot$	$\sqcap$	$\sqcup$	$\neg$	$\exists$	$\exists\top$	$\forall$	$\forall\bot$
RL		×	×		×			×	×
EL	×	×	×			×	×		
QL	×	×	×		×	×	×		

Rule-Based Instance Retrieval in RL



Rule-Based Instance Retrieval in RL

- **Goal:** for an ontology O, compute all entailments of the form C(a) and P(a,b) for a class name C in O and a property name P in O
- **Approach:** apply inference rules until no new conclusions are found
- Known under many names: saturation, deductive closure, materialisation, bottom-up reasoning, forward chaining, consequence-based reasoning

Derivation Rules for RL

$$\mathbf{A}_{\sqsubseteq}\ \frac{D(c)}{E(c)} : D \sqsubseteq E \in \mathcal{O}$$

$$\mathbf{A}^{-}_{\sqcap}\ \frac{D_1 \sqcap D_2(c)}{D_1(c) \quad D_2(c)} \qquad \mathbf{A}^{+}_{\sqcap}\ \frac{D_1(c) \quad D_2(c)}{D_1 \sqcap D_2(c)} : D_1 \sqcap D_2 \text{ occurs in } \mathcal{O}$$

$$\mathbf{A}^{-}_{\forall}\ \frac{\forall P.E(c) \quad P(c,d)}{E(d)} \qquad \mathbf{A}^{+}_{\exists}\ \frac{P(c,d) \quad E(d)}{\exists P.E(c)} : \exists P.E \text{ occurs in } \mathcal{O}$$

$$\mathbf{A}^{-}_{\neg}\ \frac{\neg D(c) \quad D(c)}{\bot(c)} \qquad \mathbf{A}^{+}_{\sqcup}\ \frac{D(c)}{D_1 \sqcup D_2(c)} : \begin{array}{l} D = D_1 \text{ or } D = D_2 \\ D_1 \sqcup D_2 \text{ occurs in } \mathcal{O} \end{array}$$

$$\mathbf{A}^{-}_{\mathbf{inv}}\ \frac{P^{-}(c,d)}{P(d,c)} \qquad \mathbf{A}^{+}_{\mathbf{inv}}\ \frac{P(c,d)}{P^{-}(d,c)} : P^{-} \text{ occurs in } \mathcal{O}$$

Derivation Calculus

- Saturate under the derivation rules
  (this is uniquely defined: Section 3.3 lecture notes)

- An axiom is inferred if
  - the axiom was derived by the rules, or
  - $\perp(c)$ was derived for some constant c.

→ Second case takes inconsistent ontologies into account

Example Derivation for RL

FelisCatus $\sqsubseteq$ ∀preysOn.(Animal $\sqcap$ Small)
Animal $\sqcap$ ∃preysOn.Animal $\sqsubseteq$ Predator
FelisCatus $\sqsubseteq$ Animal
FelisCatus(silvester)
preysOn(silvester, tweety)

$$\mathbf{A}_{\sqsubseteq}\ \frac{D(c)}{E(c)} : D \sqsubseteq E \in O$$

$$\mathbf{A}_{\sqcap}^{-}\ \frac{D_1 \sqcap D_2(c)}{D_1(c) \quad D_2(c)}$$

$$\mathbf{A}_{\forall}^{-}\ \frac{\forall P.E(c) \quad P(c, d)}{E(d)}$$

$$\mathbf{A}_{\sqcap}^{+}\ \frac{D_1(c) \quad D_2(c)}{D_1 \sqcap D_2(c)} : D_1 \sqcap D_2 \text{ occu}$$

$$\mathbf{A}_{\exists}^{+}\ \frac{P(c, d) \quad E(d)}{\exists P.E(c)} : \exists P.E \text{ occurs}$$

Example Derivation for RL

FelisCatus $\sqsubseteq \forall$preysOn.(Animal $\sqcap$ Small)
Animal $\sqcap$ $\exists$preysOn.Animal $\sqsubseteq$ Predator
FelisCatus $\sqsubseteq$ Animal
FelisCatus(silvester)
preysOn(silvester, tweety)

Animal(silvester)
$\forall$preysOn.(Animal $\sqcap$ Small)(silvester)
Animal $\sqcap$ Small(tweety)
Animal(tweety)
Small(tweety)
$\exists$preysOn.Animal(silvester)
Animal $\sqcap$ $\exists$preysOn.Animal(silvester)
Predator(silvester)

$$\mathbf{A}_{\sqsubseteq}\ \frac{D(c)}{E(c)} : D \sqsubseteq E \in \mathcal{O}$$

$$\mathbf{A}^{-}_{\sqcap}\ \frac{D_1 \sqcap D_2(c)}{D_1(c) \quad D_2(c)}$$

$$\mathbf{A}^{-}_{\forall}\ \frac{\forall P.E(c) \quad P(c,d)}{E(d)}$$

$$\mathbf{A}^{+}_{\sqcap}\ \frac{D_1(c) \quad D_2(c)}{D_1 \sqcap D_2(c)} : D_1 \sqcap D_2 \text{ occu}$$

$$\mathbf{A}^{+}_{\exists}\ \frac{P(c,d) \quad E(d)}{\exists P.E(c)} : \exists P.E \text{ occurs}$$

Correctness of the RL Rule Calculus

Correctness of the RL Rule Calculus

Soundness

Completeness

Termination

Soundness of the Calculus

Soundness of the Calculus

- Proof strategy:
  - Show that every single rule is sound
  - If we start with true statements, only true statements can be derived

    (that's an induction argument)
  - Easy to see

Termination of the Calculus

Termination of the Calculus

- Proof strategy:
  - Show that only a limited number of inferences can be derived
  - Main observation: every derived axiom only uses expressions from the ontology (or $\perp$)
  - Only finite number of axioms possible (at most size^3 many)

Completeness of the Calculus (for instance retrieval!)

Completeness of the Calculus (for instance retrieval!)

- Proof strategy:
  - Show that, if a axiom is not inferred, then there is a model of O here the axiom does not hold
  - There even is a single **universal model** that refutes every axiom that is not inferred
  - Proof steps:
    - Define this model
    - Show that it is a model
    - Show that it refutes non-inferred axioms

Defining a Universal Model

- Let O' be the saturation of O.
- We define an interpretation I:
  - The domain Δ^I of I is the set of all individual symbols (w.l.o.g., we can assume that there is one).
  - For every individual symbol c, define $c^I := c$.
  - For every class name A, define $c \in A^I$ iff $A(c) \in O'$.
  - For every property name P, define $\langle c, d \rangle \in P^I$ iff $P(c, d) \in O'$.

→ I refutes atomic assertions that are not in O'

Example Derivation for RL

$\mathrm{FelisCatus} \sqsubseteq \forall \mathrm{preysOn}.(\mathrm{Animal} \sqcap \mathrm{Small})$
$\mathrm{Animal} \sqcap \exists \mathrm{preysOn}.\mathrm{Animal} \sqsubseteq \mathrm{Predator}$
$\mathrm{FelisCatus} \sqsubseteq \mathrm{Animal}$
$\mathrm{FelisCatus}(\mathrm{silvester})$
$\mathrm{preysOn}(\mathrm{silvester}, \mathrm{tweety})$

$\mathrm{Animal}(\mathrm{silvester})$
$\forall \mathrm{preysOn}.(\mathrm{Animal} \sqcap \mathrm{Small})(\mathrm{silvester})$
$\mathrm{Animal} \sqcap \mathrm{Small}(\mathrm{tweety})$
$\mathrm{Animal}(\mathrm{tweety})$
$\mathrm{Small}(\mathrm{tweety})$
$\exists \mathrm{preysOn}.\mathrm{Animal}(\mathrm{silvester})$
$\mathrm{Animal} \sqcap \exists \mathrm{preysOn}.\mathrm{Animal}(\mathrm{silvester})$
$\mathrm{Predator}(\mathrm{silvester})$

Constructing a Universal Model

FelisCatus(silvester)
preysOn(silvester, tweety)

Animal(silvester)

Animal(tweety)
Small(tweety)

Predator(silvester)

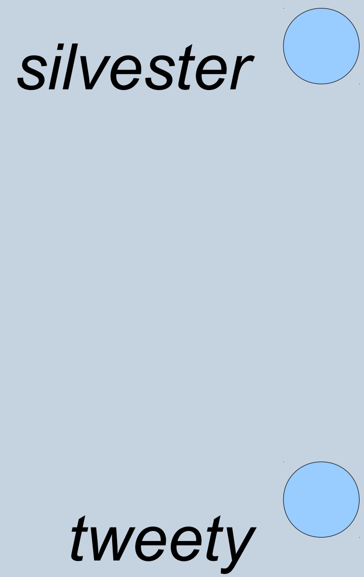
Constructing a Universal Model

FelisCatus(silvester)
preysOn(silvester, tweety)

Animal(silvester)

Animal(tweety)
Small(tweety)

Predator(silvester)



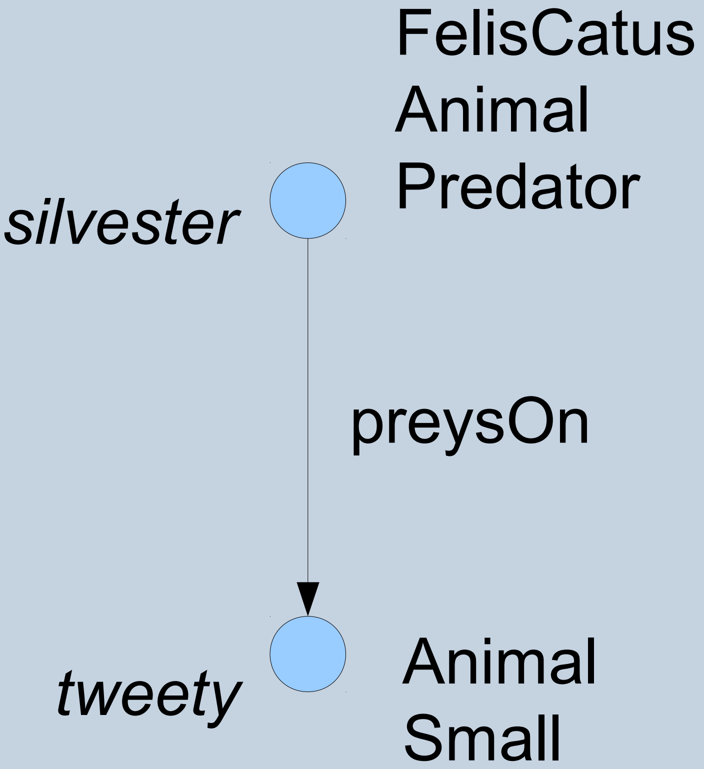
Constructing a Universal Model

FelisCatus(silvester)
preysOn(silvester, tweety)

Animal(silvester)

Animal(tweety)
Small(tweety)

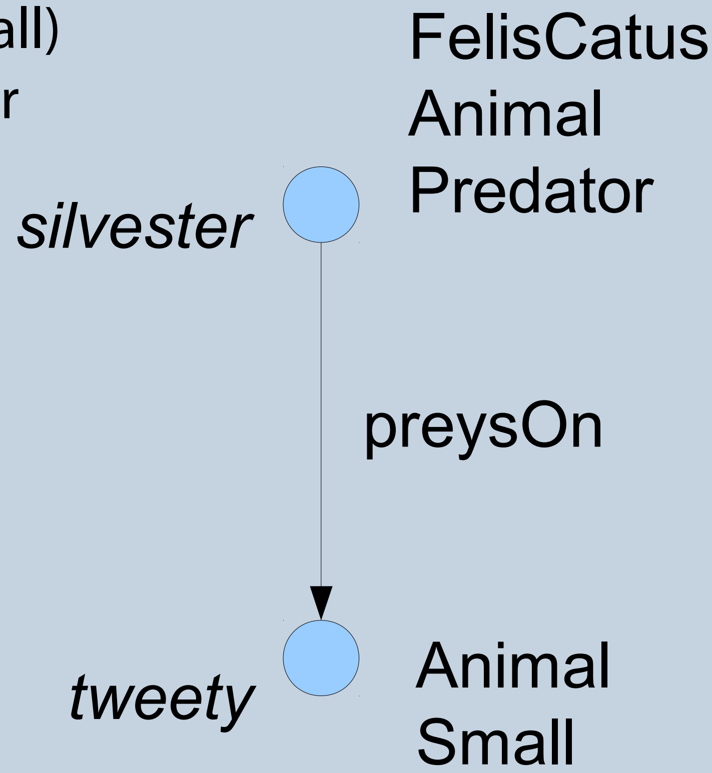
Predator(silvester)



Constructing a Universal Model

$\text{FelisCatus} \sqsubseteq \forall \text{preysOn}.(\text{Animal} \sqcap \text{Small})$
$\text{Animal} \sqcap \exists \text{preysOn}.\text{Animal} \sqsubseteq \text{Predator}$
$\text{FelisCatus} \sqsubseteq \text{Animal}$
$\text{FelisCatus}(\text{silvester})$
$\text{preysOn}(\text{silvester}, \text{tweety})$

$\text{Animal}(\text{silvester})$
$\forall \text{preysOn}.(\text{Animal} \sqcap \text{Small})(\text{silvester})$
$\text{Animal} \sqcap \text{Small}(\text{tweety})$
$\text{Animal}(\text{tweety})$
$\text{Small}(\text{tweety})$
$\exists \text{preysOn}.\text{Animal}(\text{silvester})$
$\text{Animal} \sqcap \exists \text{preysOn}.\text{Animal}(\text{silvester})$
$\text{Predator}(\text{silvester})$



Completing the Completeness Proof (1)

- I and O' agree on class and property names.
- Extend this to complex expressions:
  1) P^{-} occurs in O and $<c, d> \in P^{-I}$ iff $P^{-}(c, d) \in O'$
  2) If $E \in CL$ occurs in O, then $c \in E^{I}$ implies $E(c) \in O'$
  3) If $E \in CR$ and $E(c) \in O'$, then E occurs in O and $c \in E^{I}$

- Easy consequence: I satisfies O

Completing the Completeness Proof (2)

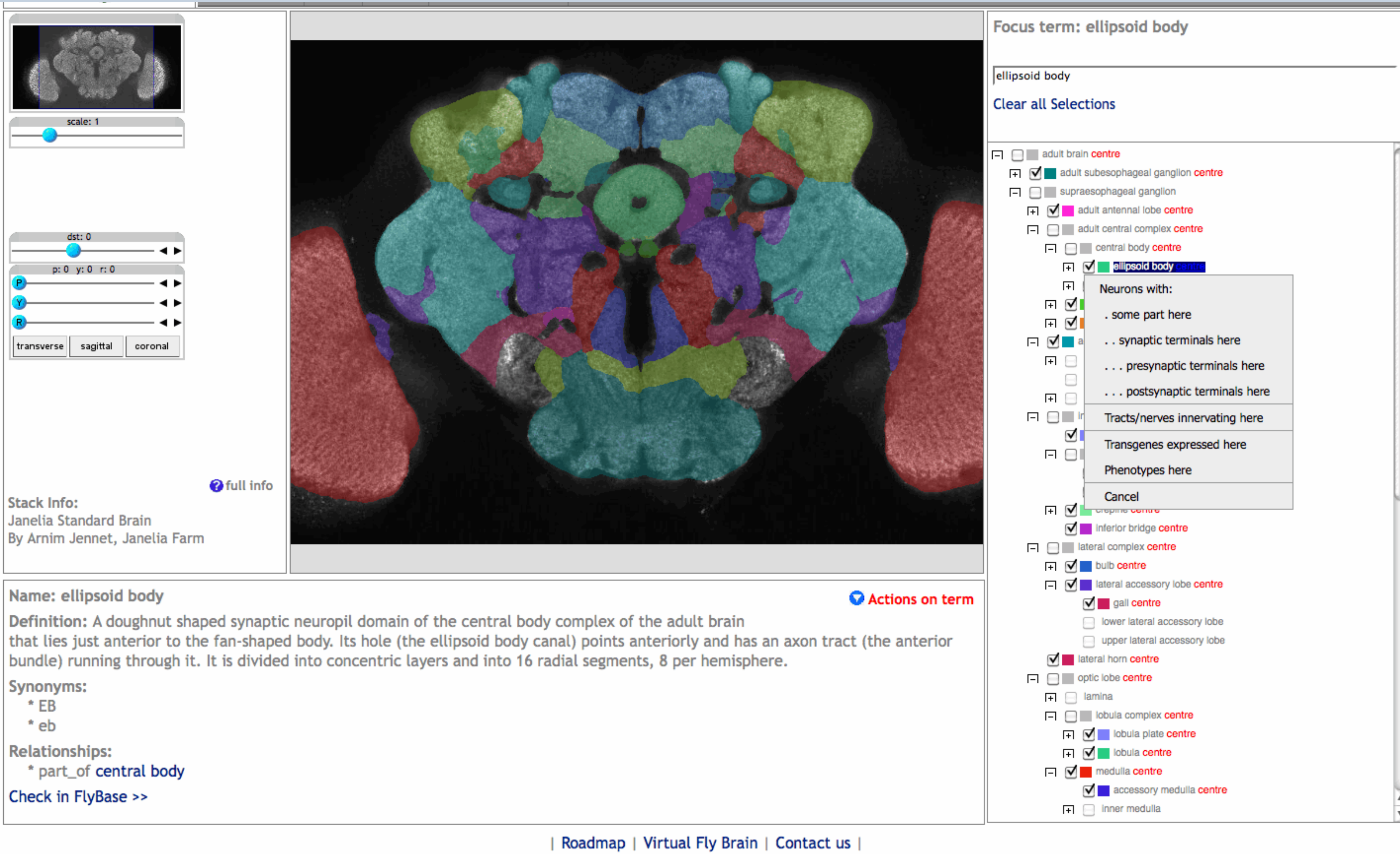
- **Example Claim 2:**
  If E ∈ CL occurs in O, then c ∈ E^I implies E(c) ∈ O'
- **Proof technique:**
  structural induction on the grammatical definition of CL

$$\mathbf{CL ::= CName \mid \bot \mid CL \sqcap CL \mid CL \sqcup CL \mid \exists P.CL}$$

- Not hard in each case; for example:

Case $E = D_1 \sqcap D_2$. By the semantics of $\sqcap$, we find $c \in D_1^{\mathcal{I}}$ and $c \in D_2^{\mathcal{I}}$. Clearly, D_1 and D_2 occur in $\mathcal{O}$ since E does. Thus, the induction hypothesis implies $D_1(c) \in \mathcal{O}'$ and $D_2(c) \in \mathcal{O}'$. Since E occurs in $\mathcal{O}$, rule $\mathbf{A}^+_\sqcap$ applies and $E(c) \in \mathcal{O}'$ as required.

Rule-Based Classification in EL



DRESDEN

Rule-Based Classification in EL

- **Goal:** for an ontology O, compute all entailments of the form $A \sqsubseteq B$ for class names A and B in O.

  → Possible with similar inference rules as for RL

Derivation Rules for EL

- Ignore assertions: assume we only have class inclusions here

$$\mathbf{T}_{\sqsubseteq}\ \frac{C \sqsubseteq D}{C \sqsubseteq E} : D \sqsubseteq E \in O \qquad \mathbf{T}_{\mathsf{i}}^{+}\ \frac{}{C \sqsubseteq C \quad C \sqsubseteq \top} : C \text{ occurs in } O$$

$$\mathbf{T}_{\sqcap}^{-}\ \frac{C \sqsubseteq D_1 \sqcap D_2}{C \sqsubseteq D_1 \quad C \sqsubseteq D_2} \qquad \mathbf{T}_{\sqcap}^{+}\ \frac{C \sqsubseteq D_1 \quad C \sqsubseteq D_2}{C \sqsubseteq D_1 \sqcap D_2} : D_1 \sqcap D_2 \text{ occurs in } O$$

$$\mathbf{T}_{\exists}^{-}\ \frac{C \sqsubseteq \exists P.\bot}{C \sqsubseteq \bot} \qquad \mathbf{T}_{\exists}^{+}\ \frac{C \sqsubseteq \exists P.D \quad D \sqsubseteq E}{C \sqsubseteq \exists P.E} : \exists P.E \text{ occurs in } O$$

Derivation Calculus

- Saturate under the derivation rules

- An axiom $A \sqsubseteq B$ is inferred if
  - $A \sqsubseteq B$ was derived by the rules, or
  - $A \sqsubseteq \bot$ was derived, or
  - $\top \sqsubseteq \bot$ was derived

→ Second case takes inconsistent class into account
→ Third case takes inconsistent ontology into account

Correctness of the EL Rule Calculus

Soundness

Completeness

Termination

Correctness of the EL Rule Calculus

- Soundness and termination as for RL
- Completeness similar to RL, but with different model I:
  - Introduce one representative domain element e_C for every class name C that is not inconsistent
  - For class name A, define $e_C \in A^I$ iff $C \sqsubseteq A \in O'$.
  - For property name P, define

    $\langle e_C, e_D \rangle \in P^I$ iff $C \sqsubseteq \exists P.D \in O'$.

→ Not a universal model, but a “canonical” one

Towards a Practical Implementation

- Optimisation is essential
- Main aspects of optimisation:
  - Derivation rules: avoid redundant inferences
  - Control flow: avoid unnecessary rule applications
  - Engineering: fast rule matching
  - Concurrency: parallelise computation

- Important insight: main cost is in checking applicability of rules, not in actually applying them

ELK Reasoner

- Fastest OWL EL reasoner today
  (according to OWL Reasoner Evaluation 2013, 2014, 2015)

  http://elk.semanticweb.org/

- For further details see

  Y. Kazakov, MK, F. Simancik: "The Incredible ELK"
  J Automated Reasoning 53:1, Springer, 2013. Available online.

Rewriting-Based Query Answering in QL

Querying ontologies

- DL is often not enough for data access:
  - "Who lives together with their parents?"
  - "Who has married parents?"
  - "Which properties connect Alice and Bob?"

  → Not expressible in DL

- More advanced query languages can be used, e.g., SPARQL

  → Semantics not always clear

  → Standard reasoning algorithms often not applicable

Ontology-Based Data Access (OBDA)

- The OBDA paradigm:
  - Query data from a database, taking ontological axioms into account
  - Models of ontology+database can be viewed as (possibly very large) *completed* database
  - Query answers in OBDA: **certain answers** (answers that hold in every model of the ontology)

→ Query answering under constraints (background knowledge), e.g., for data integration tasks

Conjunctive Queries (CQs)

- Most basic query language
- A query is a formula of the form

$$\exists x1\ldots\ \exists xn.\ C1 \wedge \ldots \wedge Cm$$

where C1, …, Cm are atomic first-order formulae
(in our case: predicates are only unary [class] or binary [property])

- A query can have free variables, called **distinguished**
- The bound variables x1, …, xn are **non-distinguished**

Conjunctive Queries: Examples

- Who lives together with their parents?

$\exists y.\ \mathrm{livesWith}(x,y) \wedge \mathrm{hasParent}(x,y)$

- Who has married parents?

$\exists y.\exists z.\ \mathrm{hasMother}(x,y) \wedge \mathrm{hasFather}(x,z) \wedge \mathrm{married}(y,z)$

Conjunctive Queries: Relationships

- CQs are essentially equivalent to:
  - Basic Graph Patterns of SPARQL
  - SELECT-PROJECT-JOIN fragment of SQL

→ Definition of distinguished/non-distinguished variables must be clarified in either case

- Unions of CQs (UCQs): allow disjunctions of CQs
- Path queries: allow regular expressions over properties

CQs and OBDA

- Challenge:
  Existence of unnamed individuals may be entailed
  → Can we query for these? How?

- Solution:
  Distinguished variables can only bind to named ind's
  Non-distinguished variables can bind to anything

Example

- Query: FelisCatus(x) ∧ preysOn(x,y) ∧ Animal(y)

$$FelisCatus(silvester)$$
$$FelisCatus(tom)$$
$$SerinusCanaria(tweety)$$
$$preysOn(silvester, tweety)$$
$$SerinusCanaria \sqsubseteq Animal$$
$$FelisCatus \sqsubseteq \exists preysOn.Animal$$

Example

- Query: FelisCatus(x) $\wedge$ preysOn(x,y) $\wedge$ Animal(y)

$$\mathit{FelisCatus}(\mathit{silvester})$$
$$\mathit{FelisCatus}(\mathit{tom})$$
$$\mathit{SerinusCanaria}(\mathit{tweety})$$
$$\mathit{preysOn}(\mathit{silvester}, \mathit{tweety})$$
$$\mathit{SerinusCanaria} \sqsubseteq \mathit{Animal}$$
$$\mathit{FelisCatus} \sqsubseteq \exists \mathit{preysOn}.\mathit{Animal}$$

→ Only solution is x=silvester, y=tweety
→ No solution with x=tom (no named value for y)

Rewriting-Based Query Answering in QL

- **Goal:** for an ontology O, compute all certain answers for a conjunctive query over O
- **Approach:** rewrite input query into a union of many conjunctive queries that yield the result
  - Rewriting only depends on terminological axioms, not on assertions
  - Rewritten queries can be answered by relational DB systems (SQL)

QL Syntax Revisited

$\mathcal{QL}$ *tiny*

$$
\begin{aligned}
\textbf{Axiom} &::= \textbf{CL} \sqsubseteq \textbf{CR} \mid \textbf{CR}(\textbf{IName}) \mid \textbf{P}(\textbf{IName}, \textbf{IName}) \\
\textbf{CL} &::= \textbf{CName} \mid \top \mid \bot \mid \exists \textbf{P}.\top \\
\textbf{CR} &::= \textbf{CName} \mid \top \mid \bot \mid \textbf{CR} \sqcap \textbf{CR} \mid \neg \textbf{CL} \mid \exists \textbf{P}.\textbf{CR} \\
\textbf{P} &::= \textbf{PName} \mid \textbf{PName}^{-}
\end{aligned}
$$

A Simpler Normal Form for Axioms

- Axioms of QLtiny can be rewritten to a simpler form:

QL normal form:

$$A \sqsubseteq B \qquad A \sqsubseteq \bot \qquad A \sqsubseteq \exists P.B$$

$$A \sqcap A' \sqsubseteq B \qquad \top \sqsubseteq B \qquad \exists P.\top \sqsubseteq B$$

$$A(c) \qquad P(c, d)$$

where A, A', and B are class names, and P is a property or an inverse property

Rules for Rewriting Queries

- Derive new queries by **replacing** atoms:

$$\mathbf{Q}_{\sqsubseteq}\ \frac{E(x)}{D(x)} : D \sqsubseteq E \in \mathcal{O} \qquad\qquad \mathbf{Q}_{\mathsf{inv}}\ \frac{P(x,y)}{P^{-}(y,x)}$$

$$\mathbf{Q}^{-}_{\sqcap}\ \frac{D_1 \sqcap D_2(x)}{D_1(x) \quad D_2(x)} \qquad\qquad \mathbf{Q}^{-}_{\top}\ \frac{\top(x)}{}$$

$$\mathbf{Q}^{+}_{\exists}\ \frac{\exists P.\top(x) \quad \exists P^{-}.\top(y) \quad P(x,y) \quad P^{-}(y,x) \quad B(y)}{\exists P.B(x)} :$$

y a non-distinguished variable that occurs only in the query atoms in the premise; $\exists P.B$ occurs in $\mathcal{O}$

plus any rule obtained from $\mathbf{Q}^{+}_{\exists}$ by leaving away some (but not all) of the premises

Example Rewriting for QL

FelisCatus $\sqsubseteq$ $\exists$preysOn.Animal
SerinusCanaria $\sqsubseteq$ Animal
FelisCatus(silvester)
FelisCatus(tom)
SerinusCanaria(tweety)
preysOn(silvester, tweety)

Example Rewriting for QL

FelisCatus $\sqsubseteq$ $\exists$preysOn.Animal
SerinusCanaria $\sqsubseteq$ Animal
FelisCatus(silvester)
FelisCatus(tom)
SerinusCanaria(tweety)
preysOn(silvester, tweety)

$\exists$y.FelisCatus(x) $\wedge$ preysOn(x, y) $\wedge$ Animal(y)
$\exists$y.FelisCatus(x) $\wedge$ preysOn(x, y) $\wedge$ SerinusCanaria(y)
$\exists$y.FelisCatus(x) $\wedge$ preysOn^{-}(y, x) $\wedge$ Animal(y)
$\exists$y.FelisCatus(x) $\wedge$ preysOn^{-}(y, x) $\wedge$ SerinusCanaria(y)
FelisCatus(x) $\wedge$ $\exists$preysOn.Animal(x)
FelisCatus(x)

Missing Bits

- Check if ontology is consistent

  → Check if the "query" $\exists y.\bot(y)$ is entailed

- Allow query simplification by unification

  → Factorise query atoms

Correctness of the QL Rewriting Calculus

- **Soundness:** easy
- **Termination:** not hard (query building blocks finite)
- **Correctness:** not entirely trivial
  - Construct a universal model, step by step (may be infinite now!)
  - Every query match can be found in this model → can also be found in the partially constructed model after some number n of construction steps
  - Show that there is a rewritten query that has a match after only n-1 construction steps
  - Induction: some rewriting matches at n=0 (assertions in O)

The Limits of Lightweight Ontologies

Tiny OWL Profiles: Possible Extensions

- OWL RL and QL allow inverse properties, EL doesn't.
- Features for sub- and superclasses:

Sub	$\top$	$\bot$	$\sqcap$	$\sqcup$	$\neg$	$\exists$	$\exists\top$	$\forall$	$\forall\bot$
RL		×	×	×		×	×		
EL	×	×	×			×	×		
QL	×	×					×		

Sup	$\top$	$\bot$	$\sqcap$	$\sqcup$	$\neg$	$\exists$	$\exists\top$	$\forall$	$\forall\bot$
RL		×	×		×			×	×
EL	×	×	×			×	×		
QL	×	×	×		×	×	×		

Tiny OWL Profiles: Possible Extensions

- OWL RL and QL allow inverse properties, EL doesn't.
- Features for sub- and superclasses: **more is possible**

Sub	$\top$	$\bot$	$\sqcap$	$\sqcup$	$\neg$	$\exists$	$\exists\top$	$\forall$	$\forall\bot$
RL	**×**	×	×	×		×	×		
EL	×	×	×	**×**		×	×		
QL	×	×	**×**	**×**			×		

Sup	$\top$	$\bot$	$\sqcap$	$\sqcup$	$\neg$	$\exists$	$\exists\top$	$\forall$	$\forall\bot$
RL	**×**	×	×		×			×	×
EL	×	×	×		**×**	×	×		**×**
QL	×	×	×		×	×	×		**×**

Unions are Hard

- No tractable language can have $\sqcap$ in subclasses and $\sqcup$ in superclasses
- Proof idea:
  Show NP-hardness by expressing 3SAT in OWL
- Try it at home (solution in Section 4.3 lecture notes)

Universal + Existential = Exponential

- Reasoning in any ontology language with
  - $\exists$ and $\sqcap$ in subclasses, and
  - $\forall$ and $\exists$ in superclasses

  is ExpTime-hard.

- This covers the union of RL and EL and the union of RL and QL

- How can we show this?

Alternation (1)

- An **Alternating Turing Machine (ATM)** is a non-deterministic TM whose states are partitioned into two sets of **existential states** and **universal states**.
- **Intuition:**
  - Existential state: the ATM nondeterministically picks one possible transition to move on
  - Universal state: the ATM branches into many ATMs that explore each possible transition

Alternation (2)

- A configuration of an ATM is **accepting** if:
  - it is in an existential state and one of the possible transitions leads to an accepting state, or
  - it is in a universal state and all of the possible transitions lead to an accepting state.

  (note: inductive definition; universal states with no transitions are accepting)

- An ATM accepts an input if the initial state is accepting on this input.

Alternation (3)

- Time and space complexity for ATMs defined as usual (considering the time/space used by a single sequence of choices, whether existential or universal)
- What makes ATMs so interesting for us:
  - ALogSpace = PTime
  - APTime = PSpace
  - APSpace = ExpTime

Simulating a PSpace ATM with OWL

- **Input:** an ATM and an input word
- **Goal:**
  - Construct an OWL ontology that derives a certain entailment iff the ATM can accept the input in polynomial space.
  - The construction should only take polynomial time

Simulating a PSpace ATM with OWL

- **Idea:** individuals represent ATM configurations, classes describe configurations, properties model transitions
- **Encoding:**
  - Aq: the ATM is in state q
  - Hi: the ATM head is at position i
  - Cσ,i: the tape position i contains symbol σ
  - Acc: the configuration is accepting
  - Iw: the initial configuration for input word w
  - Sδ: property linking to configuration obtained by applying transition δ

Simulating a PSpace ATM with OWL

(1) Left and right transition rules

$A_q \sqcap H_i \sqcap C_{\sigma,i} \sqsubseteq \exists S_\delta.(A_{q'} \sqcap H_{i+1} \sqcap C_{\sigma',i})$ if $\delta = \langle q, \sigma, q', \sigma', r\rangle$ and $i < p(|w|) - 1$

$A_q \sqcap H_i \sqcap C_{\sigma,i} \sqsubseteq \exists S_\delta.(A_{q'} \sqcap H_{i-1} \sqcap C_{\sigma',i})$ if $\delta = \langle q, \sigma, q', \sigma', l\rangle$ and $i > 0$

(2) Memory

$H_j \sqcap C_{\sigma,i} \sqsubseteq \forall S_\delta.C_{\sigma,i}$ if $i \neq j$

(3) Final configurations

$A_q \sqcap H_i \sqcap C_{\sigma,i} \sqsubseteq Acc$ if there is no transition from q and σ

(4) Existential acceptance

$A_q \sqcap \exists S_\delta.Acc \sqsubseteq Acc$ if $q \in E$

(5) Universal acceptance

$A_q \sqcap H_i \sqcap C_{\sigma,i} \sqcap \bigsqcap_{\delta \in \Delta(q,\sigma)} \exists S_\delta.Acc \sqsubseteq Acc$ if $q \in U$ and where $\Delta(q,\sigma)$ is the set of all transitions from q and σ

Simulating a PSpace ATM with OWL

- Finally, we define the initial configuration for input w: (p defines the polynomial space bound for the ATM)

$$I_w := A_{q_0} \sqcap H_0 \sqcap C_{\sigma_0,0} \sqcap \ldots \sqcap C_{\sigma_{|w|-1},|w|-1} \sqcap C_{\Box,|w|} \sqcap \ldots \sqcap C_{\Box,p(|w|)-1},$$

- One can show (Section 4.5 of lecture notes): The ontology implies Iw ⊑ Acc iff the ATM accepts w in polynomial space.

EL + QL = ExpTime

- Only a small change is needed in the ATM simulation.

  Replace:

(2) Memory

$$H_j \sqcap C_{\sigma,i} \sqsubseteq \forall S_\delta . C_{\sigma,i} \qquad \text{if } i \neq j$$

By:

$$\exists S_\delta^- . (H_j \sqcap C_{\sigma,i}) \sqsubseteq C_{\sigma,i} \qquad \text{if } i \neq j$$

Advanced Features

Further Features of all OWL Profiles

- **Datatypes**
  - Many types (numbers, strings, dates, …)
  - Used with DataProperties
  - Datatype expressions usable like class expressions
  - Restrictions to avoid non-determinism
- **Property Hierarchies**

Further Features of OWL EL and OWL RL

- **Property Chains**
  - Generalisation of transitivity
  - Example: hasParent o hasBrother ⊑ hasUncle
  - Subject to global restrictions in OWL DL
- **Equality**
  - State that two individuals are the same or different
- **Nominals**
  - Classes with exactly one instance, given by an individual
  - Example: ∃ livesIn.{europe} ⊑ European

Further Features of OWL EL and OWL RL

- **Functional Properties**
  - Properties that have at most one value
  - Limited to DataProperties in OWL EL
  - Missing in lecture notes
- **Keys**
  - “Rules” that imply the equality of individuals
  - Semantics restricted to named individuals
  - No description logic syntax
  - Example: HasKey(Person hasName birthday)

Further Features of OWL EL

- **Local Reflexivity (Self)**
  - Example: CEO $\sqsubseteq$ $\exists$ supervisedBy.Self
  - Can be used to refer to classes in property chains:

$$\text{Man} \sqsubseteq \exists\, \text{manProperty.Self}$$
$$\text{manProperty} \circ \text{hasChild} \sqsubseteq \text{fatherOf}$$

  - Not in OWL RL (no technical reason) or OWL QL (technical status not known, but should work)

Sugar

- Many OWL features can also be expressed by using other features → Syntactic sugar
- Can be more efficient for encoding something
- What is sugar depends on the available features (for example: $\sqcap$ is sugar if $\sqcup$ and $\neg$ are available)

Summary & Conclusions

Summary: Reasoning in the Profiles

- Reasoning with the OWL 2 Profiles
  - Saturation (bottom-up): EL and RL
  - Rewriting (top-down): QL

  → other approaches possible in each case!

- Completeness of inference methods:
  - Relate computation to (canonical/universal) models
  - Main tool: (structural) induction

- Optimisation on several levels is important

Summary: Extending the Profiles

- Various features can be added
- Some features are generally problematic
  - Unions (in superclasses)
  - Combination of universals and existentials
  - Combination of inverses and existentials
- Hardness by simulating hard problems in OWL
  - ATMs as a powerful tool

Conclusions