Multi-agent and Semantic Web Systems: Description Logics and OWL

Fiona McNeill

School of Informatics

25th February 2013

Description Logics

- Description Logics allow formal concept definitions that can be reasoned about to be expressed

- Example concept definitions:
  - $Woman \equiv Person \sqcap Female$
  - $Man \equiv Person \sqcap \neg Woman$

- Not a single logic, but a family of KR logics

- Subsets of first-order logic

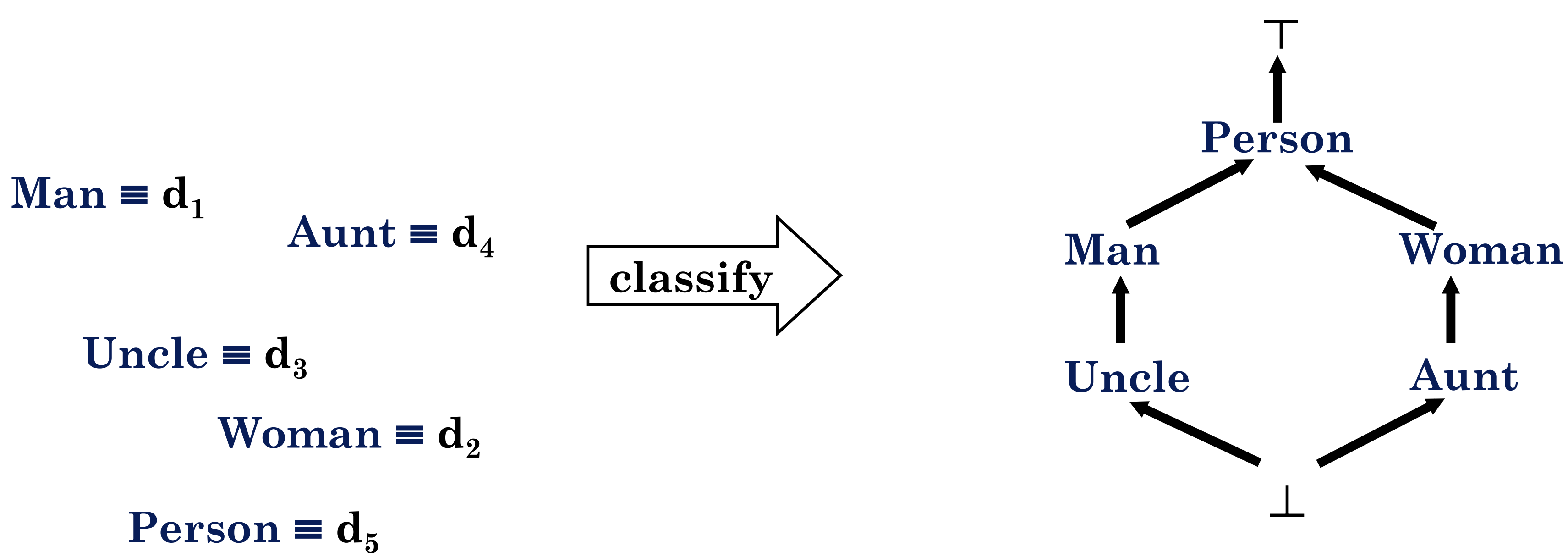
- Well-defined model theory

- Known computational complexity

Description Logics

A classifier (a reasoning engine) can be used to construct the class hierarchy from the definitions of individual concepts in the ontology

Concept definitions are composed from primitive elements and so the ontology is more maintainable



Description Logic Terminology

Description Logics separate assertions and concept definitions:

- A Box: Assertions
  - e.g. *hasChild(john, mary)*
  - This is the knowledge base
- T Box: Terminology
  - The definitions of concepts in the ontology
  - Example axioms for definitions
    - $C \sqsubseteq D$ [C is a subclass of D, D subsumes C]
    - $C \equiv D$ [C is defined by the expression D]

Description Logic Terminology

Concept: class, category or type

Role: binary relation
Attributes are functional roles

Subsumption:
D subsumes C if C is a subclass of D – *i.e.* all Cs are Ds

Unfoldable terminologies:
The defined concept does not occur in the defining expression:
$C \equiv D$ where C does not occur in the expression D

Language families
AL: Attributive Language
ALC adds full negation to AL

Language elements for concept expressions

Symbol	Description	Example	Read
$\top$	all concept names	$\top$	top
$\bot$	empty concept	$\bot$	bottom
$\sqcap$	*intersection* or *conjunction* of concepts	$C \sqcap D$	C and D
$\sqcup$	*union* or *disjunction* of concepts	$C \sqcup D$	C or D
$\neg$	*negation* or *complement* of concepts	$\neg C$	not C
$\forall$	*universal restriction*	$\forall R.C$	all R-successors are in C
$\exists$	*existential restriction*	$\exists R.C$	an R-successor exists in C
$\sqsubseteq$	Concept *inclusion*	$C \sqsubseteq D$	all C are D
$\equiv$	Concept *equivalence*	$C \equiv D$	C is equivalent to D
$\doteq$	Concept *definition*	$C \doteq D$	C is defined to be equal to D
:	Concept *assertion*	$a : C$	a is a C
:	Role *assertion*	$(a, b) : R$	a is R-related to b

Existential and universal restriction

Universal restriction - also called value restriction: $\forall R.C$

The set $\{x | \forall y, R(x, y) \Rightarrow y \in C\}$

The set of things x such that for all y where x and y are related by R, y is in C.

e.g., ∀hasChild.Parent

The set of things x such that for all y where x and y are related by hasChild, y will be in class Parent
So everything in set x is a child, everything in set y is a parent.

That is, anything that is the object of the relation hasChild must be in class Parent, regardless of what the subject is.

This is a local statement: this is true for every statement in your dataset.

Existential and universal restriction

Existential restriction - also called exists restriction: $\exists R.C$

The set $\{x | \exists y, R(x, y) \wedge y \in C\}$

The set of things x such that there exists a y where x and y are related via R and y is in class C.

e.g., ∃hasChild.Doctor

The set of all x's such that x is related to y via hasChild and y is in class Doctor.

the set of all children which have at least one parent who is a doctor

This is a local statement: this is true for at least one statement in your dataset.

Description Logic naming

Three basic logics:

$\mathcal{AL}$ Attributive language - basic language which allows:

- atomic negation
- concept intersection
- universal restrictions
- limited existential quantification

$\mathcal{FL}$ Frame based description language, allows:

- concept intersection
- universal restrictions
- limited existential quantification
- role restriction

$\mathcal{EL}$ Allows:

- concept intersection
- existential restrictions (of full existential quantification)

Description Logic naming

Followed by any of the following extensions:

- $\mathcal{F}$ Functional properties.
- $\mathcal{E}$ Full existential qualification (Existential restrictions that have fillers other than owl:Thing).
- $\mathcal{U}$ Concept union.
- $\mathcal{C}$ Complex concept negation.
- $\mathcal{H}$ Role hierarchy (subproperties - rdfs:subPropertyOf).
- $\mathcal{R}$ Limited complex role inclusion axioms; reflexivity and irreflexivity; role disjointness.
- $\mathcal{O}$ Nominals. (Enumerated classes of object value restrictions - `owl:oneOf, owl:hasValue`).
- $\mathcal{I}$ Inverse properties.
- $\mathcal{N}$ Cardinality restrictions (`owl:cardinality, owl:maxCardinality`).
- $\mathcal{Q}$ Qualified cardinality restrictions (available in OWL 2, cardinality restrictions that have fillers other than owl:Thing).
- $(\mathcal{D})$ Use of datatype properties, data values or data types.

Description Logic naming

Some canonical DLs that do not exactly fit this convention are:

$\mathcal{S}$	An abbreviation for $\mathcal{ALC}$ with transitive roles.
$\mathcal{FL}^-$	A sub-language of $\mathcal{FL}$, which is obtained by disallowing role restriction. This is equivalent to $\mathcal{AL}$ without atomic negation.
$\mathcal{FL}_o$	A sub-language of $\mathcal{FL}^-$ which is obtained by disallowing limited existential quantification.
$\mathcal{EL}^{++}$	Alias for $\mathcal{ELRO}$.

Common DLs

- $\mathcal{ALC}$ is the most common DL. It is $\mathcal{AL}$ with complement of any concept allowed, not just atomic concepts.

- $\mathcal{SHIQ}$ is the logic $\mathcal{ALC}$ plus extended cardinality restrictions, and transverse and inverse roles.

- The Protégé editor supports $\mathcal{SHOIN}^{(\mathcal{D})}$

- OWL-2 provides the expressivness of $\mathcal{SROIQ}^{(\mathcal{D})}$

- OWL-DL is based on $\mathcal{SHOIN}^{(\mathcal{D})}$

- OWL-Lite is based on $\mathcal{SHIF}^{(\mathcal{D})}$

Example concept expressions

Parent ≡ *"Persons who have (amongst other things) some children"*

$\text{Person} \sqcap \exists \text{hasChild}.\top$

ParentOfBoys ≡ *"Persons who have some children, and only have children that are male"*

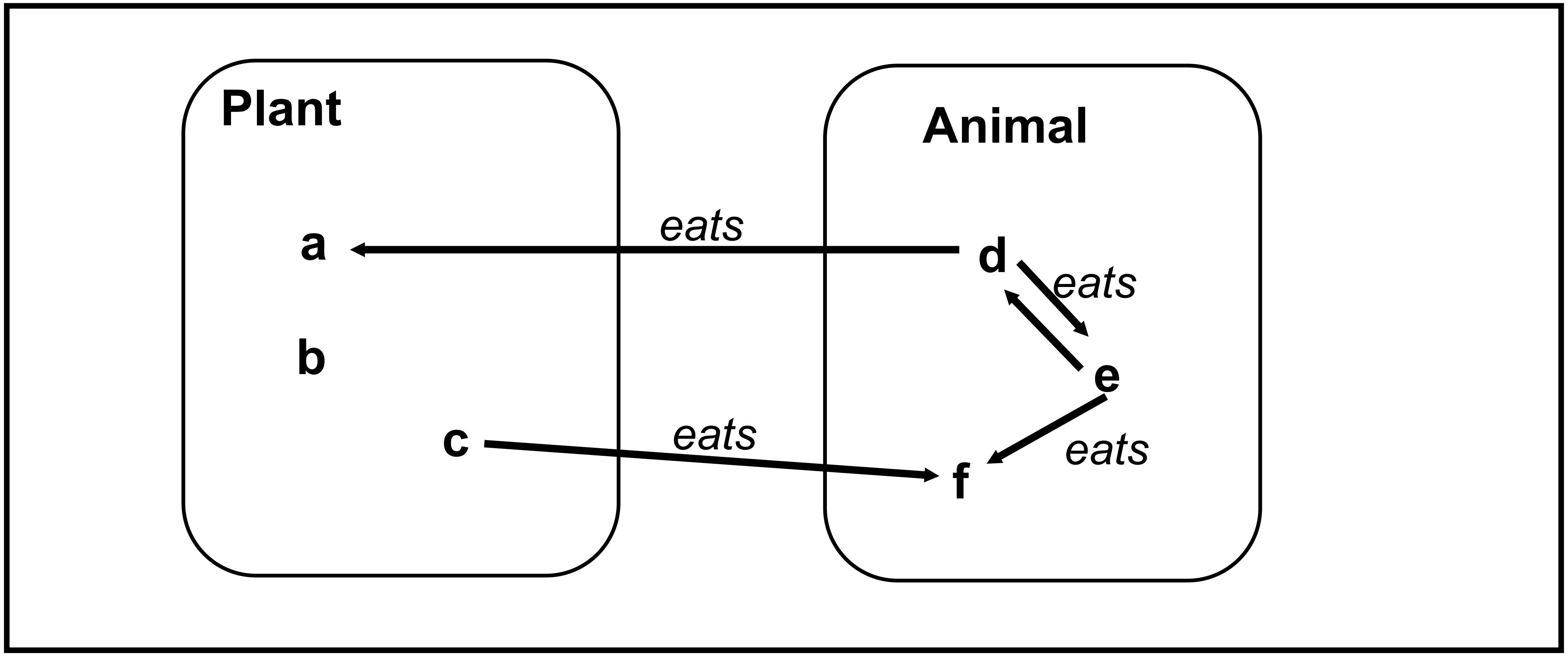
$\text{Person} \sqcap (\exists \text{hasChild}.\ \top)\ (\forall \text{hasChild}.\text{Male})$

ScottishParent ≡ *"Persons who only have children that drink (amongst other things) some IrnBru"*

$\text{Person} \sqcap (\forall \text{hasChild}.\ (\exists \text{drink}.\text{IrnBru}))$

Value and exists restrictions

{a,b,c,d,e,f} are instances; Plant and Animal are classes

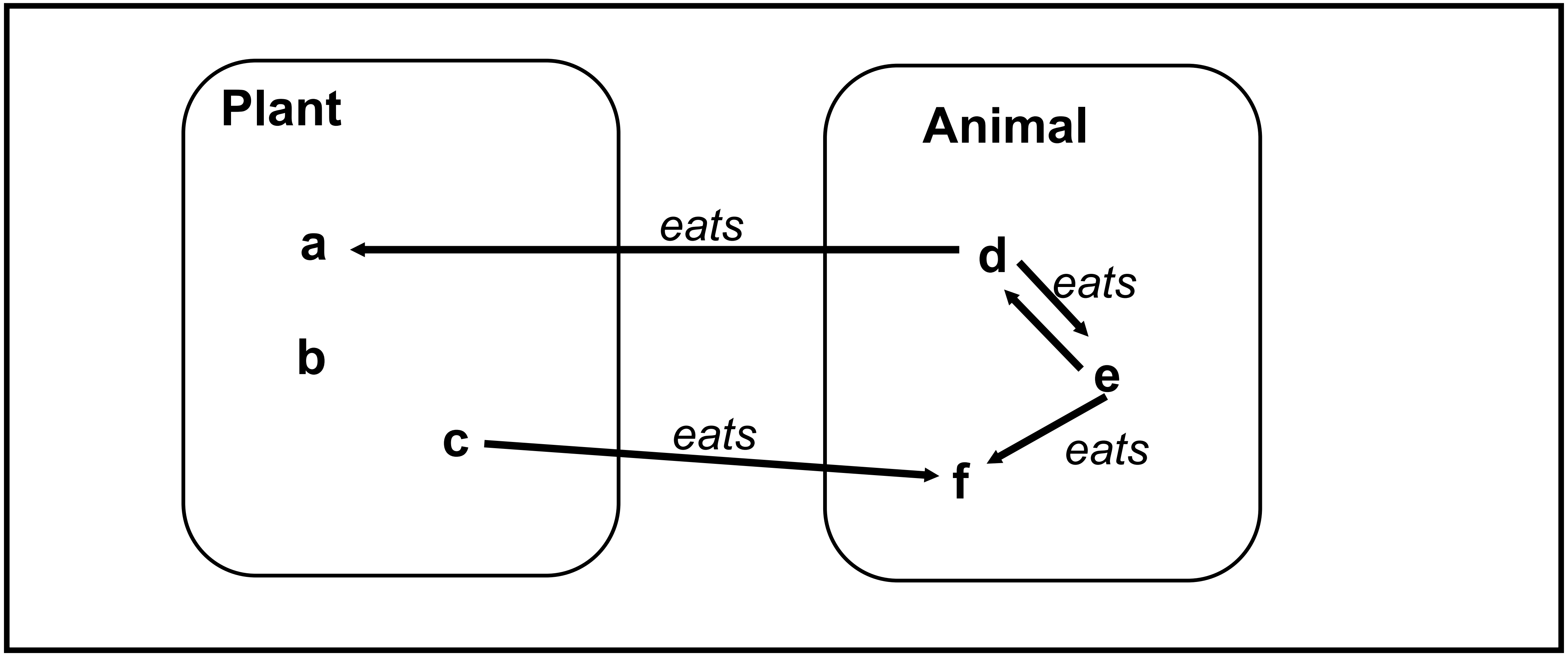


$\text{Plant} \sqcap \text{Animal} \sqsubseteq \bot$
(disjointness)

$\top \sqsubseteq \text{Plant} \sqcap \text{Animal}$
(partition)

Value and exists restrictions

{a,b,c,d,e,f} are instances; Plant and Animal are classes



$\exists$eats.Animal = {c,d,e}

$\forall$eats.Animal = {a,b,c,e,f}

$\exists$eats.Animal $\sqcap$ $\forall$eats.Animal = {c,e}

Model Theory

$\triangle^I$ universal domain of individuals, let

$\triangle^I = \{a,b,c,d,e,f\}$

$eats^I$ set of pairs for the relation eats, let

$eats^I = \{\langle d,a\rangle, \langle d,e\rangle, \langle e,d\rangle, \langle e,f\rangle, \langle c,f\rangle\}$

For all concepts C:

i) $C^I \subseteq \triangle^I$

ii) $C^I \neq \emptyset$

Let $Animal^I = \{d,e,f\}$

$\therefore (\neg Animal)^I = \{a,b,c\}$

$\therefore (\forall eats.Animal)^I = \{a,b,c,e,f\}$

$\therefore (\exists eats.Animal)^I = \{c,d,e\}$

Inference

MeatEater $\equiv$ $\forall$eats.Animal = {a,b,c,e,f}

Vegetarian $\equiv$ $\forall$eats. $\neg$Animal = {a,b,f}

Omnivore $\equiv$ $\exists$eats.Animal = {c,d,e}

Inference:

From the above classes we can see that:

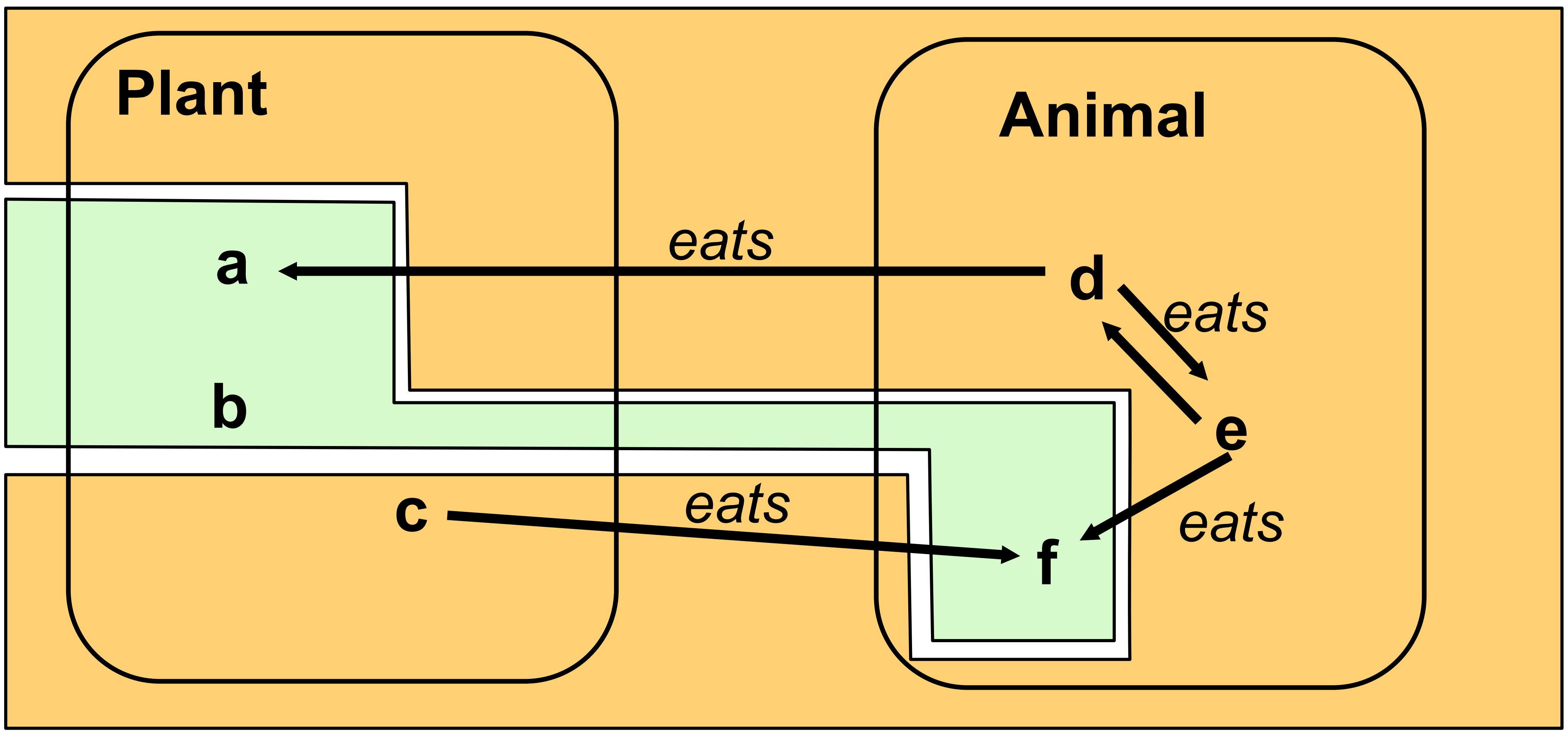
- MeatEater subsumes Vegetarian
- Vegetarian is disjoint from Omnivore

in this model, with these definitions.

The problem is to prove this for ALL models.

Value and exists restrictions

{a,b,c,d,e,f} are instances; Plant and Animal are classes



Vegetarian = {a,b,f}
Omnivore = {c,d,e}

disjoint?

MeatEater = {a,b,c,e,f}

DL Inference

- Inference can expressed in terms of the model
  - Satisfiability of C: C^I is non-empty
  - Subsumption $C \sqsubseteq D$ iff $C^I \subseteq D^I$ ("C is subsumed by D")
  - Equivalence $C \equiv D$ iff $C^I = D^I$
  - Disjointness$(C \sqcap D) \sqsubseteq$ iff $C^I \cap D^I \equiv \emptyset$

- Tractable/terminating inference algorithms exist

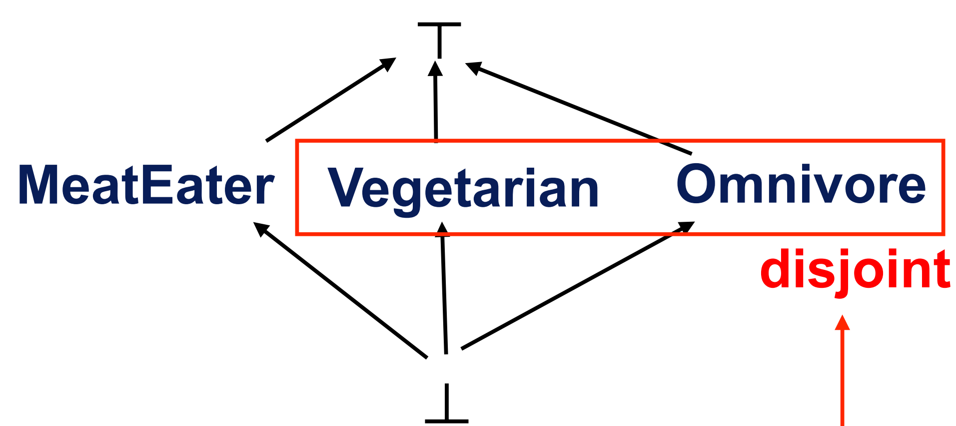
DL Inference

$\text{MeatEater} \equiv \forall \text{eats.Animal}$

$\text{Vegetarian} \equiv \forall \text{eats.} \neg \text{Animal}$

$\text{Omnivore} \equiv \exists \text{eats.Animal}$

Query	Answer
a) $\text{Vegetarian} \sqsubseteq \text{MeatEater}$	No
b) $(\text{MeatEater} \sqcap \text{Vegetarian}) \sqsubseteq \bot$	No
c) $(\text{Omnivore} \sqcap \text{Vegetarian}) \sqsubseteq \bot$	Yes



DL Inference

Inference has 2 equivalent notions - so implementing one lets us prove all 4 properties

- Reduction to subsumption $\sqsubseteq$:
  - Unsatisfiability of C: $C \sqsubseteq \bot$
  - Equivalence $C \equiv D$ iff $C \sqsubseteq D$ and $D \sqsubseteq C$
  - Disjointness $(C \sqcap D) \sqsubseteq \bot$
- Reduction to unsatisfiability $C^I = \emptyset$:
  - Subsumption $C \sqsubseteq D$ iff $(C \sqcap \neg D)$ is unsatisfiable
  - Equivalence $C \equiv D$ iff $(C \sqcap \neg D)$ and $(D \sqcap \neg C)$ are unsatisfiable
  - Disjointness $(C \sqcap D)$ is unsatisfiable

DL Summary

- DLs are a family of languages based on subsets of first-order logic.
  - The level of expressivity depends on the attributes of the language.
  - Attributes are indicated by letters; DL language names consist of a series of these letters. The expressivity of any DL language can therefore be inferred from its name.

- DLs allow complex expressions of how *concepts* related to one another.

- There are many algorithms (e.g., Tableaux Algorithms) that allow efficient reasoning over DLs.

Web Ontology Language: OWL

- Web Ontology Language (OWL) is W3C Recommendation for an ontology language for the web
  - Has an XML syntax
- OWL is layered on RDF and RDFS (other W3C standards)
  - Conforms to the RDF/RDFS semantics
  - OWL has 3 versions:
    - OWL-Lite - the simpler OWL DL
    - OWL-DL - more expressive DL
    - OWL-Full - not confined to DL, closer to FOL
  - OWL DLs extend ALC
    - Allow instances to be represented (A Box)
    - Provides datatypes
    - Provides number restrictions
- OWL 1.1 and 2 extend OWL DL

OWL Object Properties

OWL makes a distinction between Object types and Datatypes Object types and Object properties are the same as in ALC

CN, DN	Atomic concepts	Non-empty sets CN^I, $DN^I \subseteq \Delta^I$
$\bot^I$	owl:Nothing	ϕ
$\top^I$	owl:Thing	Δ^I
$(\neg C)^I$	Full Negation	$\Delta^I \setminus C^I$
$(C \sqcup D)^I$	Union	$C^I \cup D^I$
$(C \sqcap D)^I$	Intersection	$C^I \cap D^I$
$(\forall R.C)^I$	Value restriction	$\{x \in \Delta^I \mid \forall y <x,y> \in R^I \Rightarrow y \in C^I\}$
$(\exists R.C)^I$	Full existential quantification	$\{x \in \Delta^I \mid \exists y <x,y> \in R^I \wedge y \in C^I\}$

Terminological axioms: Inclusions and equalities

Concepts: $C \sqsubseteq D$ iff $C^I \subseteq D^I$

$C \equiv D$ iff $C^I = D^I$

OWL Datatypes

Datatypes $\triangle^{I}_{D}$ are distinct from Object types $\triangle^{I}$.

- A datatype relation U, e.g. age, relates an object type, e.g. Person to an integer
  - $\exists$age.Integer (the set of things that have some Integer as age)

- Data types correspond to XML Schema types

- OWL also provides hasValue: U:v to represent specific datatype values
  - age:29 (the set of things age 29)

D	Data Range	$D^I \subseteq \Delta_D^I$
$(\forall U.D)^I$	Value restriction	$\{x \in \Delta^I \mid \forall y <x,y> \in U^I \Rightarrow y \in D^I\}$
$(\exists U.D)^I$	Full existential quantification	$\{x \in \Delta^I \mid \exists y <x,y> \in U^I \wedge y \in D^I\}$

OWL Number Restrictions

OWL adds (unqualifying) number restrictions to ALC

$\geq$ n R

- Defines the set of instances, x, for which there n or more instances, y, such that R(x, y)
- BusyParent ≡ $\geq$ 3 hasChild

$\leq$ n R

- Defines the set of instances, x, for which there n or less instances, y, such that R(x, y)

$\geq$ n R	**Minimum cardinality**	**$\{x \in \Delta^I \mid \#(\langle x,y\rangle \in R^I) \geq n\}$**
$\leq$ n R	**Maximum cardinality**	**$\{x \in \Delta^I \mid \#(\langle x,y\rangle \in R^I) \leq n\}$**

Datatypes $\triangle^I{}_D$ and Object types $\triangle^I$

BN, CN	Non-empty sets $BN^I, CN^I \subseteq \Delta^I$
D	$D^I \subseteq \Delta_D{}^I$
$(B \sqcup C)^I$	$\{x \in \Delta^I \mid x \in B^I \vee x \in C^I\}$
$(B \sqcap C)^I$	$\{x \in \Delta^I \mid x \in B^I \wedge x \in C^I\}$
$(\forall R.C)^I$	$\{x \in \Delta^I \mid \forall y\ (\langle x,y\rangle \in R^I \Rightarrow y \in C^I)\}$
$(\exists R.C)^I$	$\{x \in \Delta^I \mid \exists y\ \langle x,y\rangle \in R^I \wedge y \in C^I\}$
$(\forall U.D)^I$	$\{x \in \Delta^I \mid \forall y\ (\langle x,y\rangle \in U^I \Rightarrow y \in D^I)\}$
$(\exists U.D)^I$	$\{x \in \Delta^I \mid \exists y\ \langle x,y\rangle \in U^I \wedge y \in D^I\}$

OWL-DL Cardinality

BN, CN	Non-empty sets $BN^I, CN^I \subseteq \Delta^I$
$(\forall R.C)^I$	$\{x \in \Delta^I \mid \forall y\ (\langle x,y\rangle \in R^I \Rightarrow y \in C^I)\}$
$(\exists R.C)^I$	$\{x \in \Delta^I \mid \exists y\ \langle x,y\rangle \in R^I \wedge y \in C^I\}$
$(\geq n\ R)^I$	$\{x \in \Delta^I \mid \#(\langle x,y\rangle \in R^I) \geq n\}$
$(\leq n\ R)^I$	$\{x \in \Delta^I \mid \#(\langle x,y\rangle \in R^I) \leq n\}$

OWL-DL Cardinality

$\text{Bicycle} \equiv \geq 2\ \text{hasWheel} \sqcap \leq 2\ \text{hasWheel} \sqcap \forall \text{hasPart}. \neg \text{Engine}$

- Unicyles would have 1 wheel, tricycles 3 wheels, motorcycles would have 2 wheels and an Engine......
- hasWheel is needed, rather than hasPart, as OWL-DL cannot specify the type of the range to be Wheel
    - Define hasWheel a subProperty of hasPart
    - Range of hasWheel: Wheel

OWL domain and range axioms

Domain and range specifications

domain(R, C) :: $\geq 1\ R \sqsubseteq C$

Consider:

1) $\exists$hasChild.Male :anything with a male child

2) $\boxed{\text{Person}} \sqcap \exists$hasChild.Male :person with a male child:

The Person intersection in 2) is implicit in 1) if the domain of hasChild is defined as Person

range(R, C) :: $\top \sqsubseteq$ R.C

OWL abstract syntax

- The ALC-style syntax is not suitable for the WWW
- OWL needs to conform to the RDF/XML syntax

OWL/ALC DL Syntax		OWL Abstract Syntax
$(\neg C)$	Full Negation	`< complementOf C >`
$(C \sqcup D)$	Union	`< unionOf C D >`
$(C \sqcap D)$	Intersection	`< intersectionOf C D >`
$(\forall R.C)$	Value restriction	`< Restriction < onProperty R > < allValuesFrom C >>`
$(\exists R.C)$	Full existential quantification	`< Restriction < onProperty R > < someValuesFrom C >>`
$(C \sqcap D) = \perp$	Disjoint concepts	`< disjoint C D >`
$C \sqsubseteq D$	Subclass of /subsumes	`< C <subClassOf D>>`
$C \equiv D$	Equivalent	`<C <equivalentClass D>>`

OWL in RDF/XML format

Class definitions $C \sqsubseteq D$ and Property restrictions $\forall R.C$ in RDF/XML syntax: DieselEngine is a subclass of Engine: DieselEngine $\sqsubseteq$ Engine

```
<owl:Class rdf:ID =”DieselEngine">
    <rdfs:subClassOf rdf:resource=”&base;Engine"/>
</owl:Class>
```

CarPart is a subclass of the parts of the Car:
CarPart $\sqsubseteq$ $\forall$partOf.Car

```
<owl:Class rdf:ID=”CarPart">
    <rdfs:subClassOf>
        <owl:Restriction>
            <owl:onProperty rdf:resource=“&base;partOf”/>
            <owl:allValuesFrom rdf:resource=“#Car”/>
        </owl:Restriction>
    </rdfs:subClassOf>
</owl:Class>
```

Defined locally

Imported

<owl:Class> is used to specify the rdf:type
rdf:ID introduces new terms (compare with rdf:about to refer to terms)
&base; is a namespace (assumed to be defined)

OWL in RDF/XML format

CarEngine is equivalent to the intersection of Engine and $\forall$partOf.Car :

$\text{CarEngine} \equiv \text{Engine} \sqcap \forall \text{partOf.Car}$

```
<owl:Class rdf:ID=”CarEngine">
    <owl:equivalentClass>!
        <owl:Class>
        <owl:intersectionOf rdf:parseType=“Collection”>
                <owl:Class rdf:about=“#Engine”/>
            <owl:Restriction>
                <owl:onProperty rdf:resource=“&base;partOf”/>
                <owl:allValuesFrom rdf:resource=“#Car”/>
            </owl:Restriction>
        </owl:intersectionOf>
        </owl:Class>
    </owl:equivalentClass>
</owl:Class>
```

Protégé reads and writes this syntax

Use HP’s Jena toolkit in Java applications that need to read/write/manipulate RDF/S or OWL.

OWL summary

OWL:

- Is a web-compatible ontology language
- Syntax based on RDF/XML
- Semantics compatible with RDF and RDFS
- OWL-Lite and OWL-DL have a formal interpretation based on DLs
- Extensive documentation at http://www.w3c.org
- Editing Tools
  - Protégé 4

Reading

Ian Horrocks, Peter F. Patel-Schneider, and Frank van Harmelen. From SHIQ and RDF to OWL: The making of a web ontology language. J. of Web Semantics, 1(1):7-26, 2003.

Task

Write down a few universal and existential restriction statements in DL.

Add some OWL cardinality restriction statements.