UNIVERSITY OF
Southampton

UNIVERSITY OF Southampton

A Description Logic Primer

COMP6215 Semantic Web Technologies

Dr Nicholas Gibbins - nmg@ecs.soton.ac.uk

Why do we need Description Logics?

RDF Schema isn't sufficient for all tasks

- There are things you can't express
- There are things you can't infer

Description Logics

A *family* of knowledge representation formalisms

- A subset of first order predicate logic (FOPL)
- Decidable – trade-off of expressivity against algorithmic complexity
- Well understood – derived from work in the mid-80s to early 90s
- Model-theoretic formal semantics
- Simpler syntax than FOPL

This module assumes that you're familiar with FOPL.

If you need a refresher, the following resources are available:

- Lecture notes for COMP1215 Foundations of Computer Science (on ECS intranet)
- Johnsonbaugh, R. (2014) Discrete Mathematics, 7th ed. Chapter 1. (ebook via library)

Description Logics

Description logics restrict the predicate types that can be used

- Unary predicates denote concept membership

$$Person(x)$$

- Binary predicates denote roles between instances

$$hasChild(x, y)$$

Note on terminology: the DL literature uses slightly different terms to those in RDFS

- Class and concept are interchangeable terms
- Role, relation and property are interchangeable terms

Defining ontologies with Description Logics

Describe classes (concepts) in terms of their necessary and sufficient conditions

Consider an attribute A of a class C:

- Attribute A is a necessary condition for membership of C
  - If an object is an instance of C, then it has A
- Attribute A is a sufficient condition for membership of C
  - If an object has A, then it is an instance of C

Description Logic Reasoning Tasks

Satisfaction

- “Can this class have any instances?"

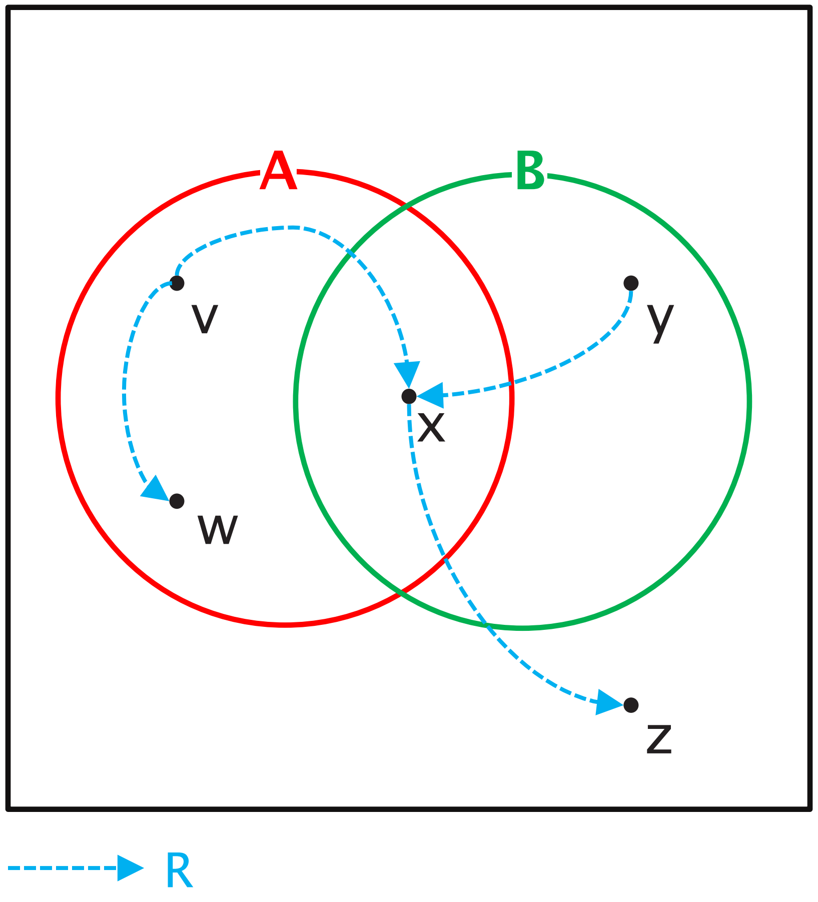
Subsumption

- "Is every instance of class C necessarily an instance of class D?"

Classification

- "What classes is this object an instance of?"

Concepts as sets



Syntax

Expressions

Description logic expressions consist of:

- Concept and role descriptions:
  - Atomic concepts: Person
  - Atomic roles: hasChild
  - Complex concepts: “person with two living parents”
  - Complex roles: “has parent’s brother” (i.e. "has uncle")
- Axioms that make statements about how concepts or roles are related to each other:
  - “Every person with two living parents is thankful”
  - “hasUncle is equivalent to has parent’s brother”

Concept Constructors

Used to construct complex concepts:

- Boolean concept constructors $\neg C \quad C \sqcup D \quad C \sqcap D$
- Restrictions on role successors $\forall R.C \quad \exists R.C$
- Number/cardinality restrictions $\leq n\, R \quad \geq n\, R \quad = nR$
- Nominals (singleton concepts) $\{x\}$
- Universal concept, top $\top$
- Contradiction, bottom $\bot$

Role Constructors

Used to construct complex roles:

- Concrete domains (datatypes)
- Inverse roles R^-
- Role composition $R \circ S$
- Transitive roles R^+

OWL and Description Logics

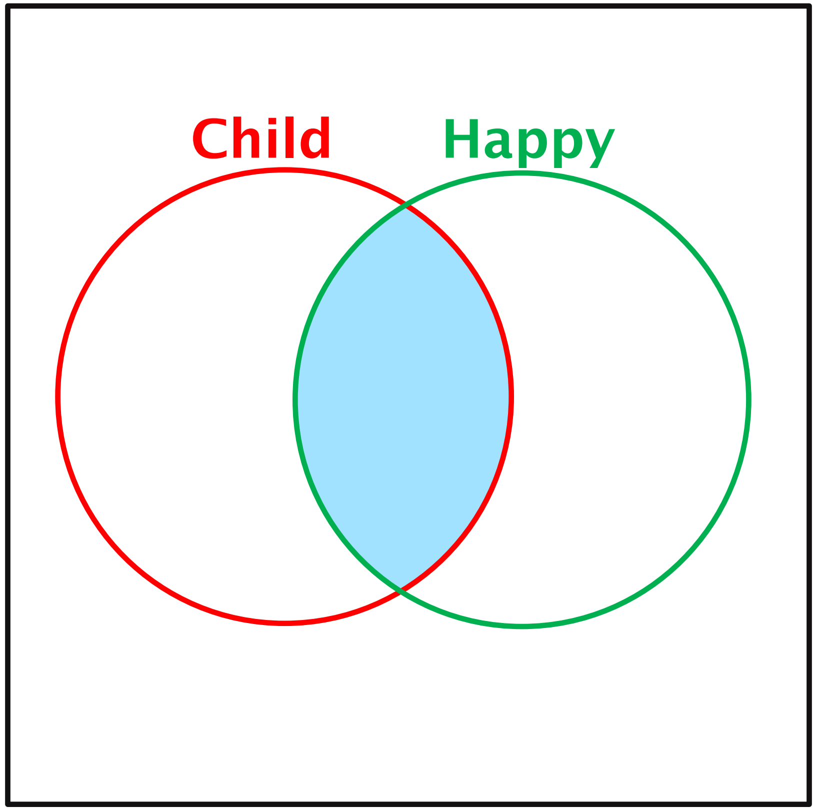
- Not every description logic supports all constructors
- More constructors = more expressive = higher complexity
- For example, OWL DL is equivalent to the logic $\mathcal{SHOIN}(D)$
  - Atomic concepts and roles
  - Boolean operators
  - Universal, existential restrictions, number restrictions
  - Role hierarchies
  - Nominals
  - Inverse and transitive roles (but not role composition)

Boolean Concept Constructors: Intersection

Child ⊓ Happy

The class of things which are both children and happy

Read as "Child AND Happy"



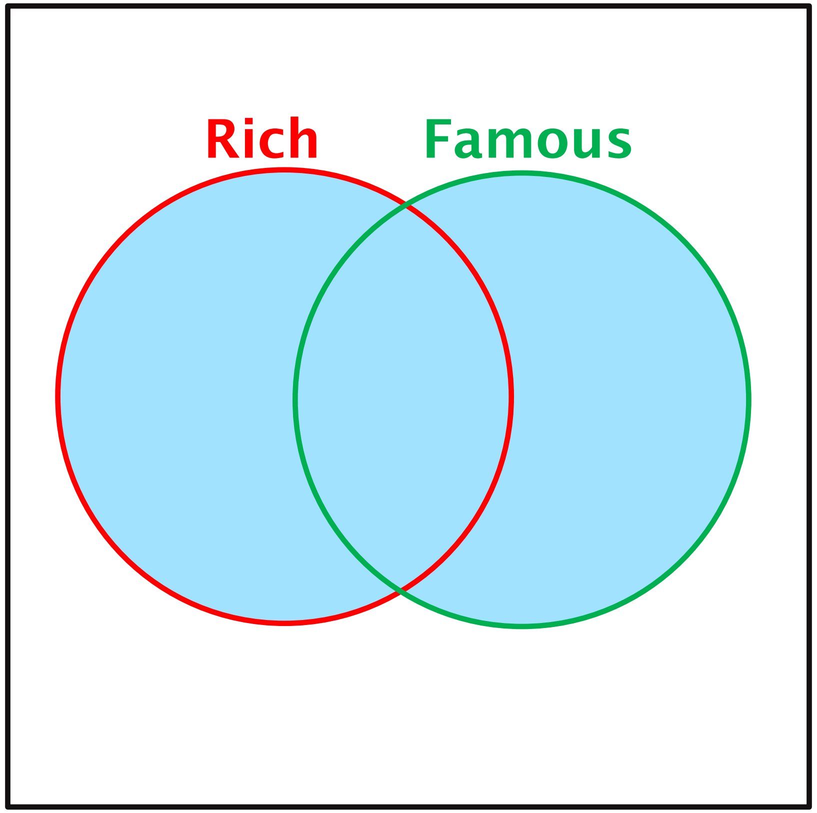
UNIVERSITY OF Southampton

Boolean Concept Constructors: Union

$Rich \sqcup Famous$

The class of things which are rich or famous (or both)

Read as “Rich OR Famous”

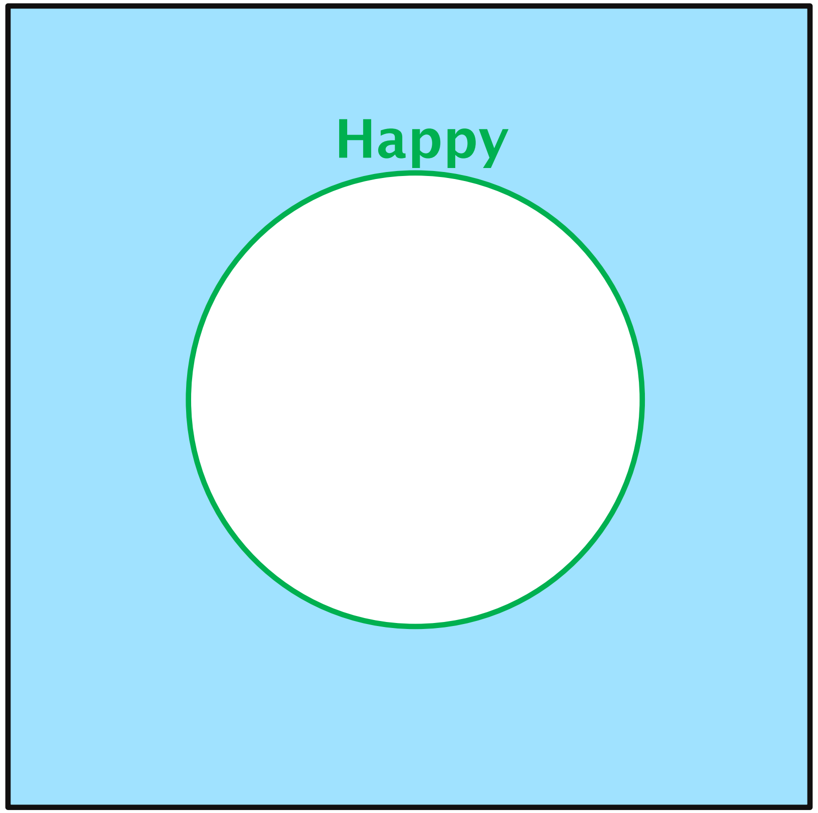


Boolean Concept Constructors: Complement

$\neg$Happy

The class of things which are not happy

Read as “NOT Happy”



Southampton UNIVERSITY OF

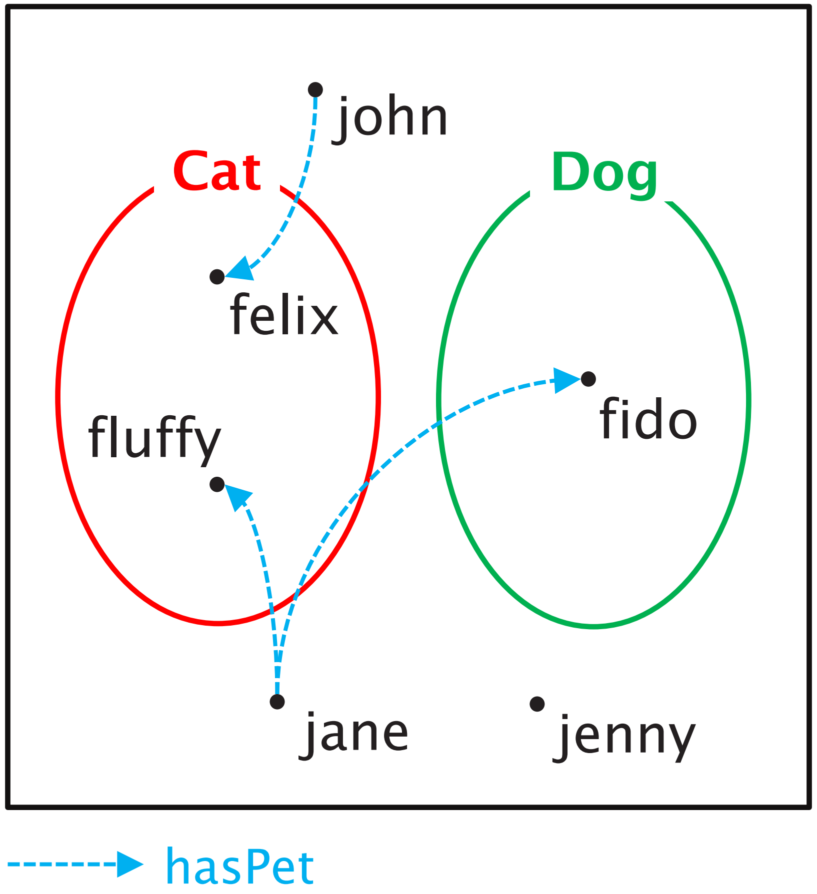
Restrictions: Existential

$\exists hasPet.\, Cat$

The class of things which have some pet that is a cat

- must have at least one pet

Read as "hasPet SOME Cat"



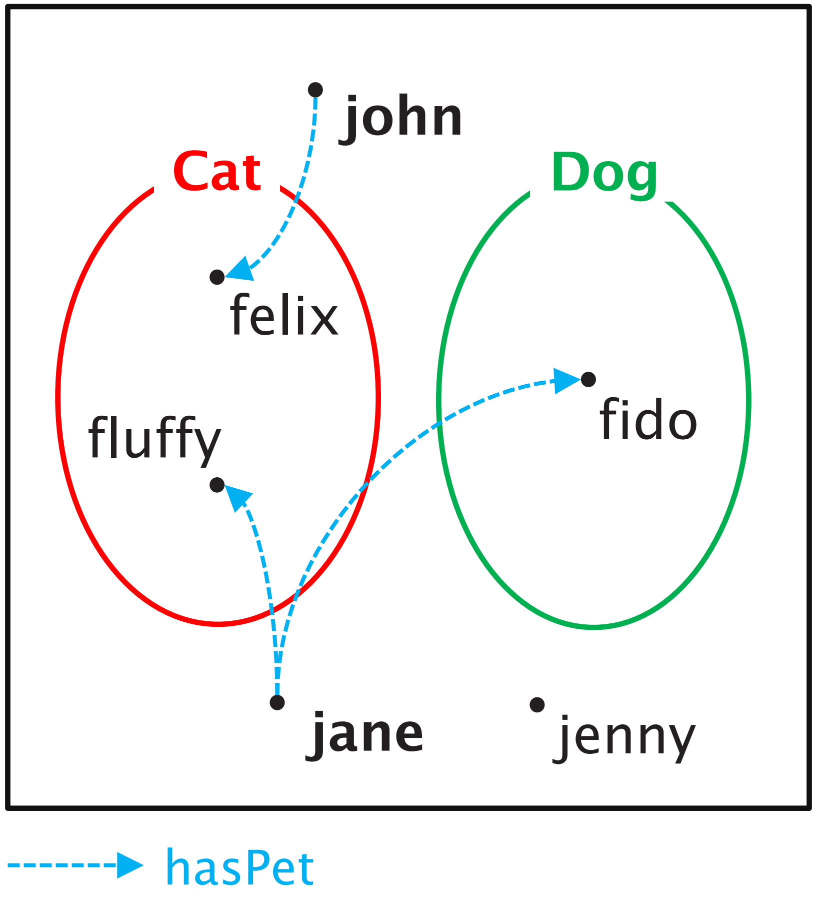
Restrictions: Existential

$\exists hasPet.\, Cat$

The class of things which have some pet that is a cat

- must have at least one pet

Read as "hasPet SOME Cat"



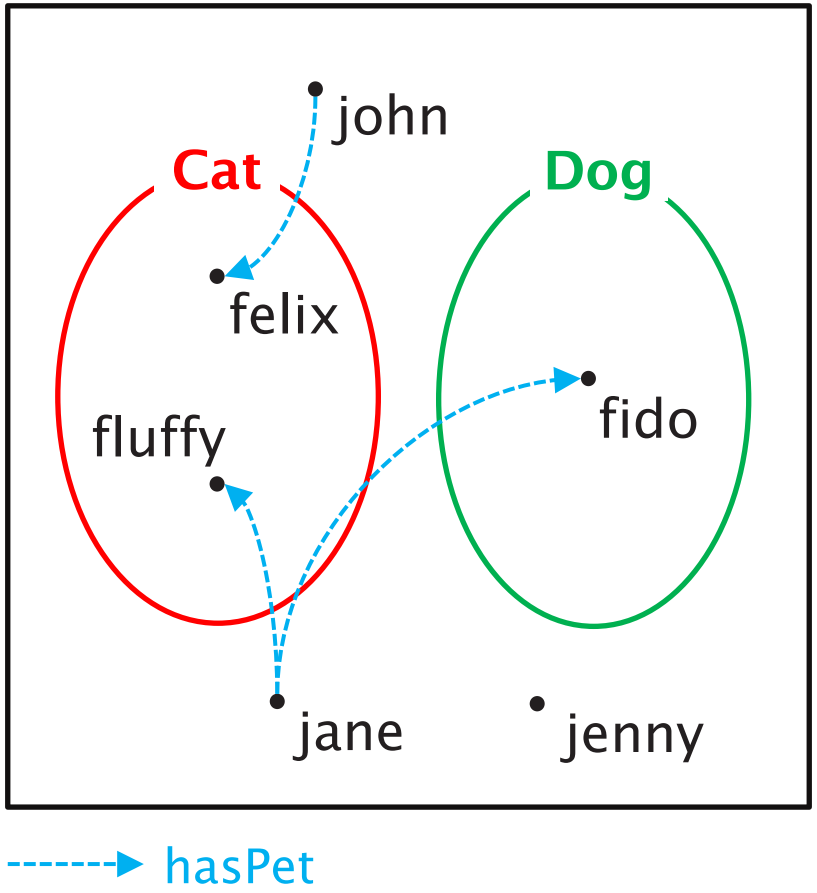
Restrictions: Universal

$\forall \text{hasPet.Cat}$

The class of things all of whose pets are cats

- Or, which only have pets that are cats
- includes those things which have no pets

Read as "hasPet ONLY Cat"



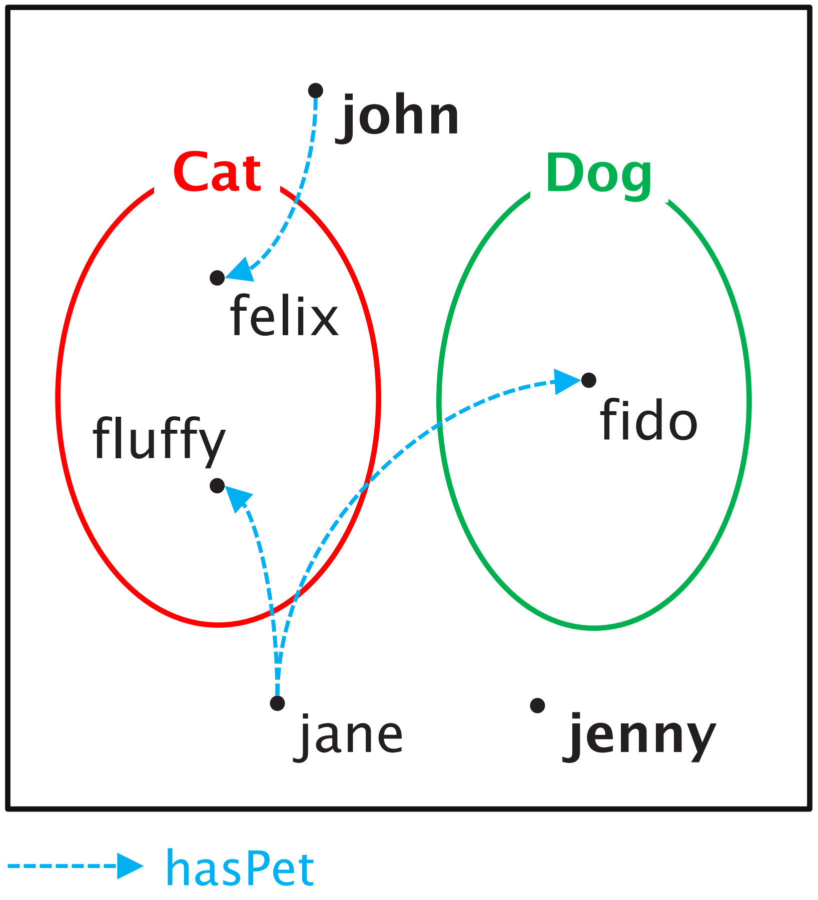
Restrictions: Universal

$\forall \text{hasPet}.\text{Cat}$

The class of things all of whose pets are cats

- Or, which only have pets that are cats
- includes those things which have no pets

Read as “hasPet ONLY Cat”



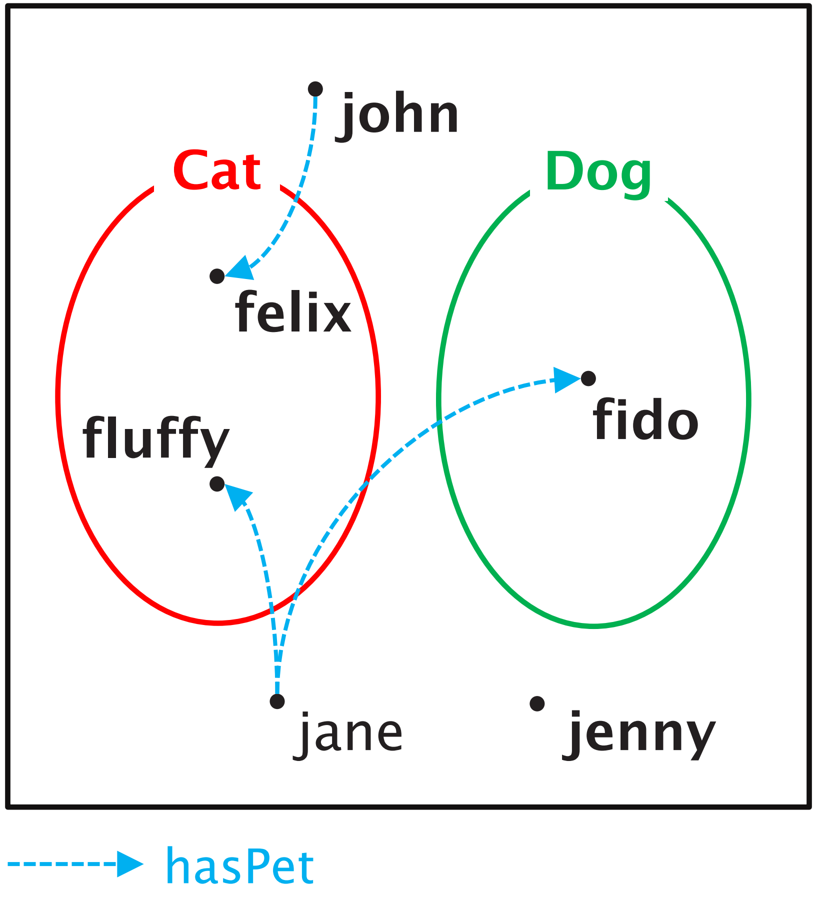
Restrictions: Universal

$\forall \text{hasPet}.\text{Cat}$

The class of things all of whose pets are cats

- Or, which only have pets that are cats
- includes those things which have no pets

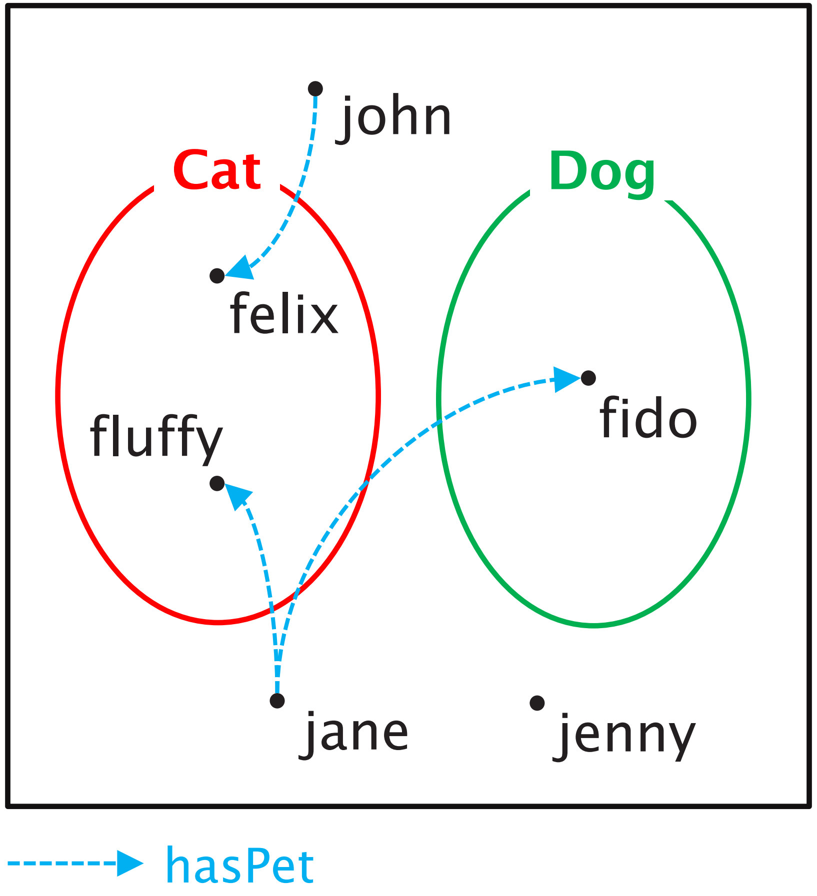
Read as "hasPet ONLY Cat"



Restrictions: Number

$= 1$ hasPet

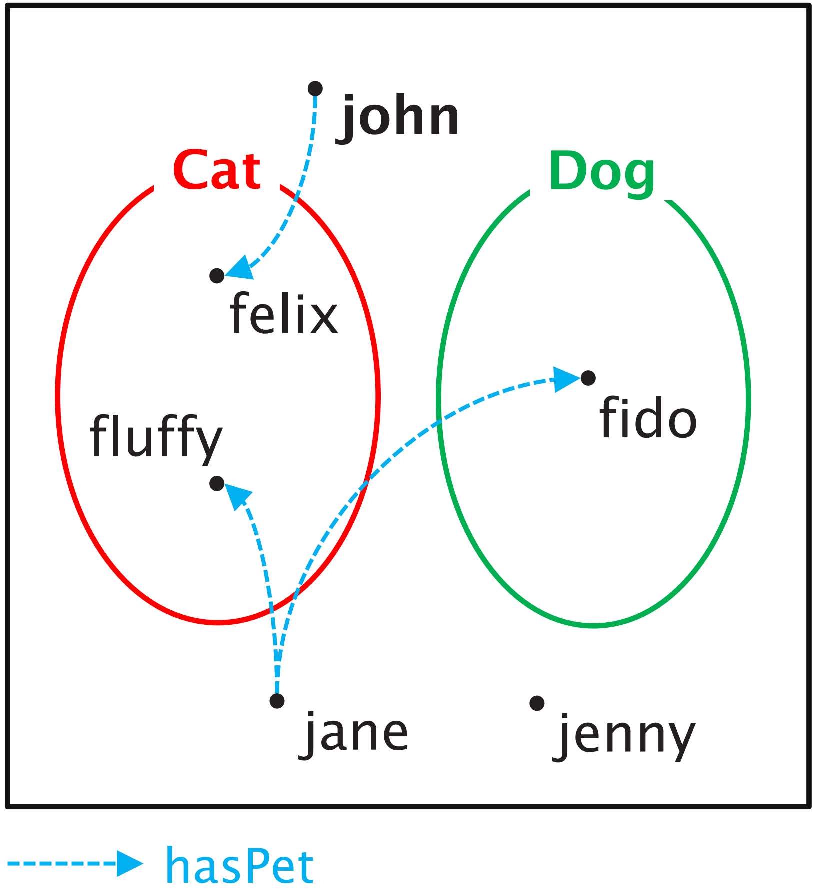
The class of things which have exactly one pet



Restrictions: Number

= 1 hasPet

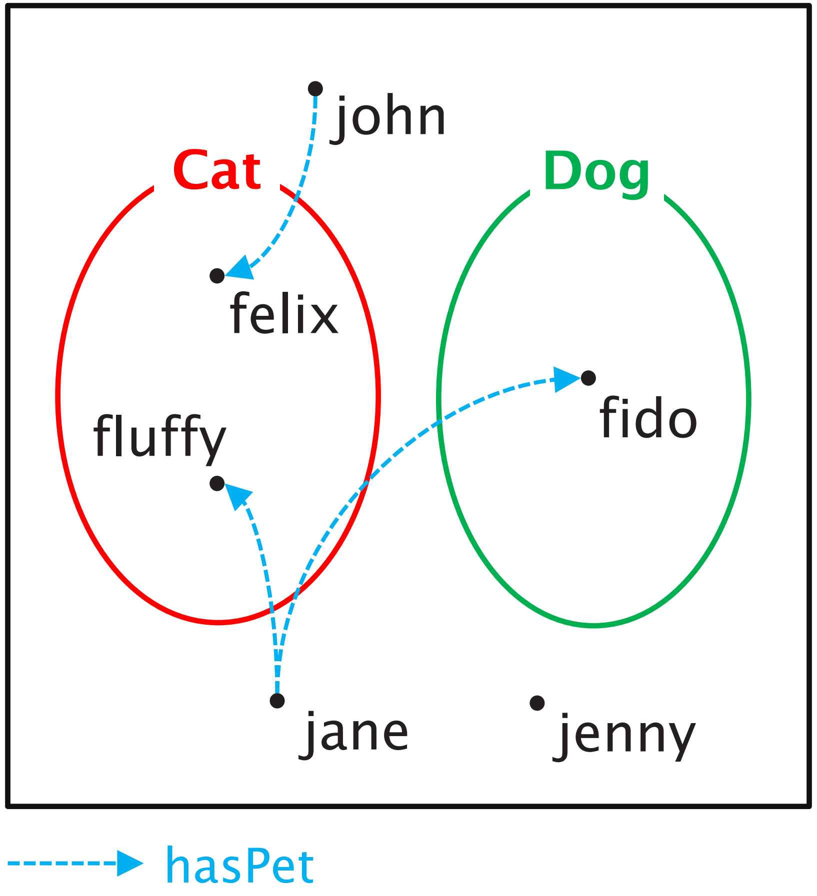
The class of things which have exactly one pet



Restrictions: Number

≥ 2 hasPet

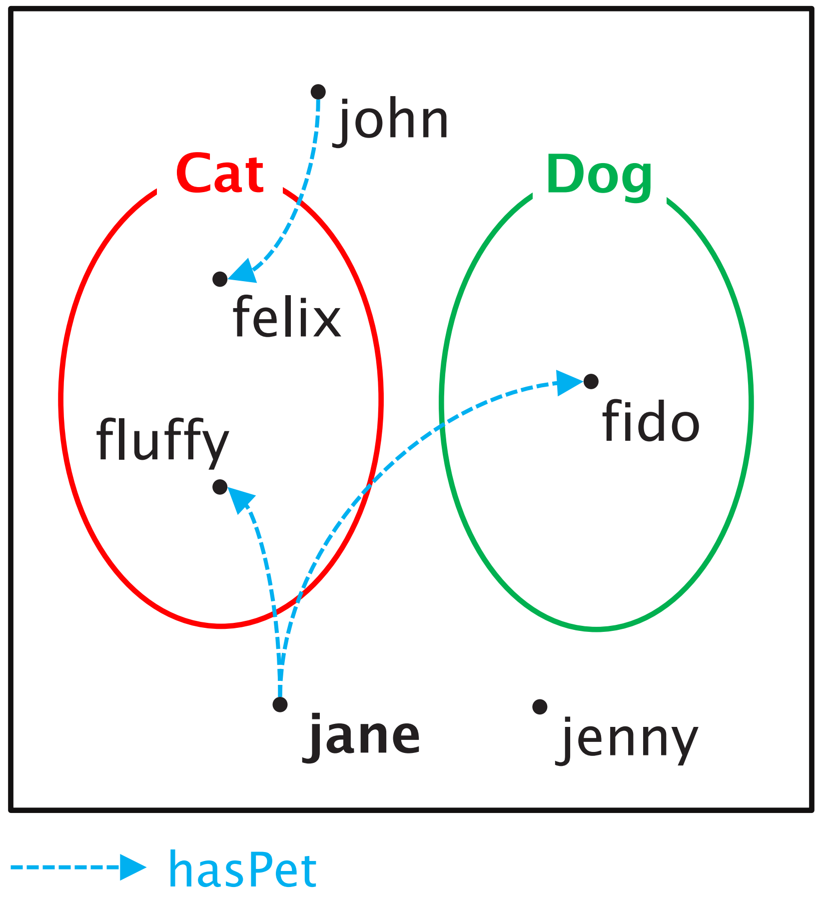
The class of things which have at least two pets



Restrictions: Number

≥ 2 hasPet

The class of things which have at least two pets



Knowledge Bases

A description logic knowledge base (KB) has two parts:

- TBox: terminology
  - A set of axioms describing the structure of the domain (i.e., a conceptual schema)
  - Concepts, roles
- ABox: assertions
  - A set of axioms describing a concrete situation (data)
  - Instances

TBox Axioms

Concept inclusion (C is a subclass of D)	$C \sqsubseteq D$
Concept equivalence (C is equivalent to D)	$C \equiv D$
Role inclusion (R is a subproperty of S)	$R \sqsubseteq S$
Role equivalence (R is equivalent to S)	$R \equiv S$
Role transitivity (R composed with itself is a subproperty of R)	$R^+ \sqsubseteq R$

UNIVERSITY OF Southampton

Revisiting Necessary and Sufficient Conditions

“Attribute A is a necessary/sufficient condition for membership of C”

Instead of talking directly about A, we can make a class expression (using the concept constructors) that represents the class of things with attribute A – call it D

- Membership of D is necessary/sufficient for membership of C

Revisiting Necessary and Sufficient Conditions

Membership of D is a necessary condition for membership of C

$$C \sqsubseteq D$$

Membership of D is a sufficient condition for membership of C

$$C \sqsupseteq D$$

Membership of D is both a necessary and a sufficient condition for membership of C

$$C \equiv D$$

Revisiting Necessary and Sufficient Conditions

Some common terminology:

$C \sqsubseteq D$

- C is a *primitive* or *partial class*

$C \equiv D$

- C is a *defined class*

(you'll see these terms used in the Protégé OWL Tutorial)

ABox Axioms

Concept instantiation

$C(x)$

- x is of type C

Role instantiation

$R(x, y)$

- x has R of y

Axiom Examples

Every person is either living or dead

Every happy child has a loving parent

Every child who eats only cake is unhealthy

No elephants can fly

A mole is a sauce from Mexico that contains chili

All Englishmen are mad

Axiom Examples

Every person is either living or dead	$\text{Person} \sqsubseteq \text{Living} \sqcup \text{Dead}$
Every happy child has a loving parent	$\text{Child} \sqcap \text{Happy} \sqsubseteq \exists \text{hasParent}.\text{Loving}$
Every child who eats only cake is unhealthy	$\text{Child} \sqcap \forall \text{eats}.\text{Cake} \sqcap \exists \text{eats}.\text{Cake} \sqsubseteq \neg \text{Healthy}$
No elephants can fly	$\text{Elephant} \sqcap \text{FlyingThing} \equiv \bot$
A mole is a sauce from Mexico that contains chili	$\text{Mole} \equiv \text{Sauce} \sqcap \exists \text{hasOrigin}.\{\text{Mexico}\} \sqcap \exists \text{hasIngredient}.\text{Chili}$
All Englishmen are mad	$\exists \text{bornIn}.\{\text{England}\} \sqcap \text{Male} \sqsubseteq \text{Mad}$

Tips for Description Logic Axioms

- No single ‘correct’ answer - different modelling choices
- Break sentence down into pieces
  - e.g. “successful man”, “spicy ingredient” etc
  - Look for nouns and adjectives (concepts)
  - Look for verb phrases (roles)
- Look for indicators of axiom type:
  - “Every X is Y” - inclusion axiom
  - “X is Y” - equivalence axiom
- Remember that $\forall R.C$ is satisfied by instances which have no value for R

DON'T PANIC!

Semantics

Description Logics and Predicate Logic

Description Logics are a subset of first order Predicate Logic with a simplified syntax

Every DL expression can be converted into an equivalent FOPL expression

Description Logics and Predicate logic

Every concept C is translated to a formula $\phi_C(x)$

Every role R is translated to a formula $\phi_R(x, y)$

Boolean concept constructors:

$$\phi_{\neg C}(x) = \neg\phi_C(x)$$
$$\phi_{C \sqcup D}(x) = \phi_C(x) \vee \phi_D(x)$$
$$\phi_{C \sqcap D}(x) = \phi_C(x) \wedge \phi_D(x)$$

Restrictions:

$$\phi_{\exists R.C}(x) = \exists y.\, \phi_R(x, y) \wedge \phi_C(y)$$
$$\phi_{\forall R.C}(x) = \forall y.\, \phi_R(x, y) \Rightarrow \phi_C(y)$$

Description Logics and Predicate logic

Axioms are translated as follows:

Concept inclusion $C \sqsubseteq D$

$$\forall x. \phi_C(x) \Rightarrow \phi_D(x)$$

Concept equivalence $C \equiv D$

$$\forall x. \phi_C(x) \Leftrightarrow \phi_D(x)$$

Example

"Every child who eats cake is happy"

Example

"Every child who eats cake is happy"

$$\mathrm{Child} \sqcap \exists \mathrm{eats}.\mathrm{Cake} \sqsubseteq \mathrm{Happy}$$

Example

"Every child who eats cake is happy"

$$\text{Child} \sqcap \exists \text{eats.Cake} \sqsubseteq \text{Happy}$$

$$\forall x\ \phi_{Child \sqcap \exists eats.Cake}(x) \Rightarrow \phi_{Happy}(x)$$

Example

"Every child who eats cake is happy"

$$\text{Child} \sqcap \exists \text{eats.Cake} \sqsubseteq \text{Happy}$$

$$\forall x \; \phi_{Child \sqcap \exists eats.Cake}(x) \Rightarrow \phi_{Happy}(x)$$

Example

"Every child who eats cake is happy"

$$\text{Child} \sqcap \exists \text{eats.Cake} \sqsubseteq \text{Happy}$$

$$\forall x \; \phi_{Child \sqcap \exists eats.Cake}(x) \Rightarrow \phi_{Happy}(x)$$

Example

"Every child who eats cake is happy"

$$\text{Child} \sqcap \exists \text{eats}.\text{Cake} \sqsubseteq \text{Happy}$$

$$\forall x\ \phi_{Child \sqcap \exists eats.Cake}(x) \Rightarrow \phi_{Happy}(x)$$

Example

"Every child who eats cake is happy"

$$\text{Child} \sqcap \exists \text{eats}.\text{Cake} \sqsubseteq \text{Happy}$$

$$\forall x\ \phi_{Child \sqcap \exists eats.Cake}(x) \Rightarrow \phi_{Happy}(x)$$

$$\forall x\ \phi_{Child}(x) \wedge \phi_{\exists eats.Cake}(x) \Rightarrow \phi_{Happy}(x)$$

Example

"Every child who eats cake is happy"

$$\mathrm{Child} \sqcap \exists \mathrm{eats}.\mathrm{Cake} \sqsubseteq \mathrm{Happy}$$

$$\forall x\ \phi_{Child \sqcap \exists eats.Cake}(x) \Rightarrow \phi_{Happy}(x)$$

$$\forall x\ \phi_{Child}(x) \land \phi_{\exists eats.Cake}(x) \Rightarrow \phi_{Happy}(x)$$

Example

"Every child who eats cake is happy"

$$\text{Child} \sqcap \exists \text{eats.Cake} \sqsubseteq \text{Happy}$$

$$\forall x\ \phi_{Child \sqcap \exists eats.Cake}(x) \Rightarrow \phi_{Happy}(x)$$

$$\forall x\ \phi_{Child}(x) \wedge \phi_{\exists eats.Cake}(x) \Rightarrow \phi_{Happy}(x)$$

Example

"Every child who eats cake is happy"

$$\text{Child} \sqcap \exists \text{eats}.\text{Cake} \sqsubseteq \text{Happy}$$

$$\forall x\ \phi_{Child \sqcap \exists eats.Cake}(x) \Rightarrow \phi_{Happy}(x)$$

$$\forall x\ \phi_{Child}(x) \land \phi_{\exists eats.Cake}(x) \Rightarrow \phi_{Happy}(x)$$

$$\forall x\ \phi_{Child}(x) \land \exists y\ \phi_{eats}(x, y) \land \phi_{Cake}(y) \Rightarrow \phi_{Happy}(x)$$

Example

"Every child who eats cake is happy"

$$\text{Child} \sqcap \exists \text{eats}.\text{Cake} \sqsubseteq \text{Happy}$$

$$\forall x\ \phi_{Child \sqcap \exists eats.Cake}(x) \Rightarrow \phi_{Happy}(x)$$

$$\forall x\ \phi_{Child}(x) \wedge \phi_{\exists eats.Cake}(x) \Rightarrow \phi_{Happy}(x)$$

$$\forall x\ \phi_{Child}(x) \wedge \exists y\ \phi_{eats}(x, y) \wedge \phi_{Cake}(y) \Rightarrow \phi_{Happy}(x)$$

Description Logic Semantics

Δ is the domain (non-empty set of individuals)

Interpretation function $\cdot^{\mathcal{I}}$ (or $ext()$) maps:

- Concept expressions to their extensions
  (set of instances of that concept, subsets of Δ)
- Roles to subsets of $\Delta \times \Delta$
- Individuals to elements of Δ

Examples:

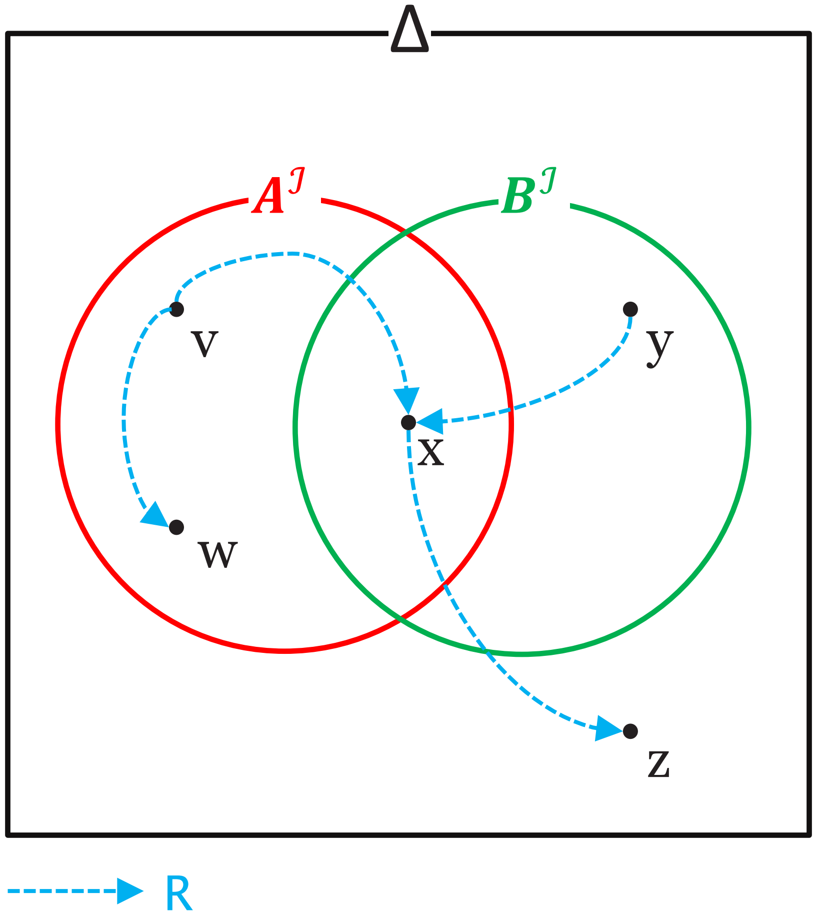
- $C^{\mathcal{I}}$ is the set of members of C
- $(C \sqcup D)^{\mathcal{I}}$ is the set of members of either C or D

Description Logic Semantics

Syntax	Semantics	Notes
$(C \sqcap D)^{\mathcal{I}}$	$C^{\mathcal{I}} \cap D^{\mathcal{I}}$	Conjunction
$(C \sqcup D)^{\mathcal{I}}$	$C^{\mathcal{I}} \cup D^{\mathcal{I}}$	Disjunction
$(\neg C)^{\mathcal{I}}$	$\Delta \backslash \mathrm{C}^{\mathcal{I}}$	Complement
$(\exists R.C)^{\mathcal{I}}$	$\{x \vert \exists y \,.\, \langle x, y \rangle \in R^{\mathcal{I}} \wedge y \in C^{\mathcal{I}}\}$	Existential
$(\forall R.C)^{\mathcal{I}}$	$\{x \vert \forall y \, \langle x, y \rangle \in R^{\mathcal{I}} \Rightarrow y \in C^{\mathcal{I}}\}$	Universal
$(\geq n\ R)^{\mathcal{I}}$	$\{x \vert \#\{y \vert \langle x, y \rangle \in R^{\mathcal{I}}\} \geq n\}$	Min cardinality
$(\leq n\ R)^{\mathcal{I}}$	$\{x \vert \#\{y \vert \langle x, y \rangle \in R^{\mathcal{I}}\} \leq n\}$	Max cardinality
$(= n\ R)^{\mathcal{I}}$	$\{x \vert \#\{y \vert \langle x, y \rangle \in R^{\mathcal{I}}\} = n\}$	Exact cardinality
$(\bot)^{\mathcal{I}}$	$\emptyset$	Bottom
$(\top)^{\mathcal{I}}$	Δ	Top

Interpretation Example

$$\Delta = \{v, w, x, y, z\}$$
$$A^{\mathcal{I}} = \{v, w, x\}$$
$$B^{\mathcal{I}} = \{x, y\}$$
$$R^{\mathcal{I}} = \{\langle v, w\rangle, \langle v, x\rangle, \langle y, x\rangle, \langle x, z\rangle\}$$



Interpretation Example

$(\neg B)^{\mathcal{I}} =$

$(A \sqcup B)^{\mathcal{I}} =$

$(\neg A \sqcap B)^{\mathcal{I}} =$

$(\exists R.B)^{\mathcal{I}} =$

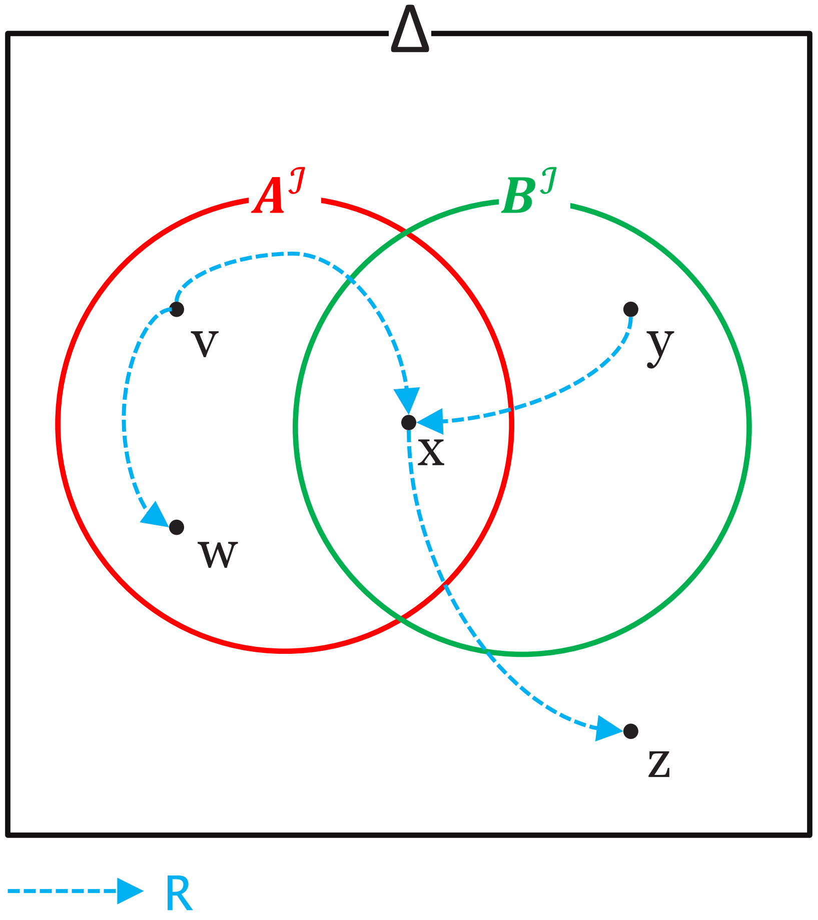
$(\forall R.B)^{\mathcal{I}} =$

$\left(\exists R.(\exists R.A)\right)^{\mathcal{I}} =$

$\left(\exists R.\neg(A \sqcap B)\right)^{\mathcal{I}}$

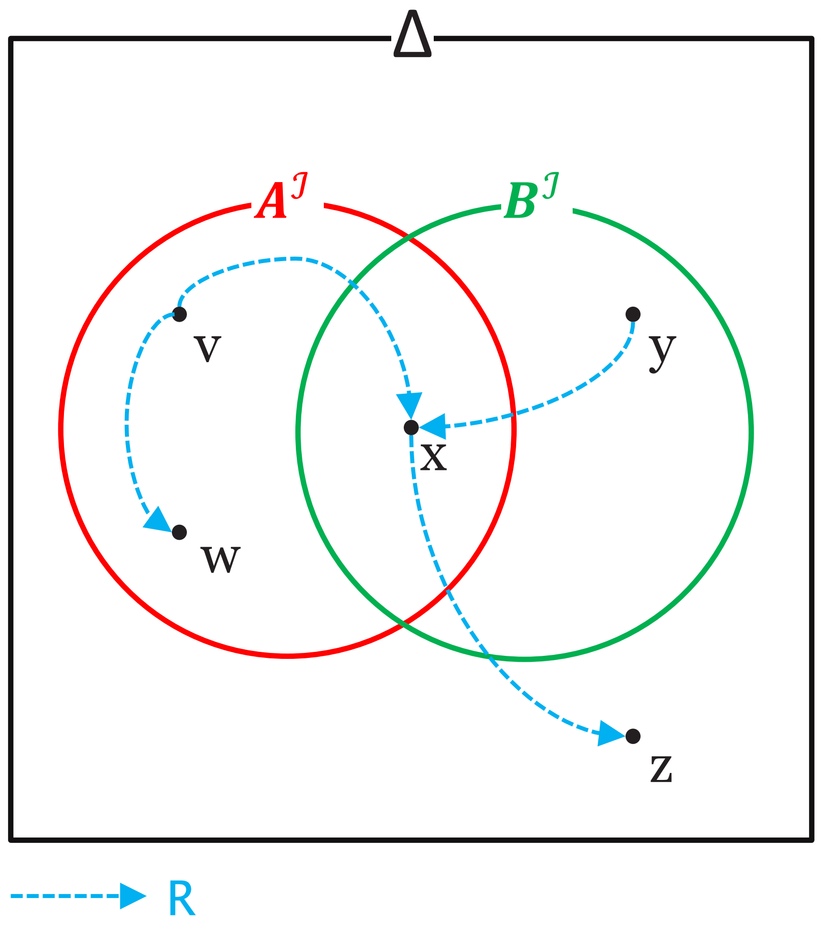
$(\exists R^{-}.A)^{\mathcal{I}} =$

$(R^{+})^{\mathcal{I}} =$



Answers

$(\neg B)^{\mathcal{I}} = \{\mathrm{v}, \mathrm{w}, \mathrm{z}\}$
$(A \sqcup B)^{\mathcal{I}} = \{\mathrm{v}, \mathrm{w}, \mathrm{x}, \mathrm{y}\}$
$(\neg A \sqcap B)^{\mathcal{I}} = \{\mathrm{y}\}$
$(\exists R.B)^{\mathcal{I}} = \{\mathrm{v}, \mathrm{y}\}$
$(\forall R.B)^{\mathcal{I}} = \{\mathrm{y}, \mathrm{w}, \mathrm{z}\}$
$\left(\exists R.(\exists R.A)\right)^{\mathcal{I}} = \{\}$
$\left(\exists R.\neg(A \sqcap B)\right)^{\mathcal{I}} = \{\mathrm{v}, \mathrm{x}\}$
$(\exists R^{-}.A)^{\mathcal{I}} = \{w, x, z\}$
$(R^{+})^{\mathcal{I}} = \{\langle v, w\rangle, \langle v, x\rangle, \langle v, z\rangle, \langle y, x\rangle, \langle y, z\rangle, \langle x, z\rangle\}$



DL Reasoning Revisited

DL Reasoning Revisited

A description logic knowledge base comprises:

- A TBox defining concepts and roles
- An ABox containing assertions about instances

$$K = \langle TBox, ABox \rangle$$

We can construct an interpretation $\mathcal{I} = \langle \Delta, \cdot^{\mathcal{I}} \rangle$ which maps the instances, concepts and roles in K onto a domain Δ via an interpretation function $\cdot^{\mathcal{I}}$

We can redefine the reasoning tasks in terms of $\mathcal{I}$

Satisfaction

"Can this class have any instances?"

A class C is satisfiable with respect to a KB K iff there exists an interpretation $\mathcal{I}$ of K with $C^{\mathcal{I}} \neq \emptyset$

Subsumption

"Is every instance of this class necessarily an instance of this other class?"

A class C is subsumed by a class D with respect to a KB K iff
for every interpretation $\mathcal{I}$ of K, $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$

Equivalence

"Is every instance of this class necessarily an instance of this other class, and vice versa?"

A class C is equivalent to a class D with respect to a KB K iff
for every interpretation $\mathcal{I}$ of K, $C^{\mathcal{I}} = D^{\mathcal{I}}$

Classification

"Is this individual necessarily an instance of this class?"

An individual x is an instance of class C wrt a KB K iff for every interpretation $\mathcal{I}$ of K, $x^{\mathcal{I}} \in C^{\mathcal{I}}$

Reduction to Satisfaction

Tableau-based reasoners for description logics (the predominant modern approach) reduce all reasoning tasks to satisfaction:

Subsumption

- C is subsumed by $D \Leftrightarrow (C \sqcap \neg D)$ is unsatisfiable

Equivalence

- C is equivalent to $D \Leftrightarrow$ both $(C \sqcap \neg D)$ and$(\neg C \sqcap D)$ are unsatisfiable

Classification

- x is an instance of $C \Leftrightarrow (\neg C \sqcap \{x\})$ is unsatisfiable

Further Reading

Daniele Nardi and Ronald J. Brachman (2003) An Introduction to Description Logics, in Franz Baader, Diego Calvanese, Deborah L. McGuinness, Daniele Nardi and Peter F. Patel-Schneider (eds) The Description Logic Handbook: Theory, implementation and applications, Cambridge University Press, 2003, pp.1-40.

F. Baader and W. Nutt (2003) Basic Description Logics, in Franz Baader, Diego Calvanese, Deborah L. McGuinness, Daniele Nardi and Peter F. Patel-Schneider (eds) The Description Logic Handbook: Theory, implementation and applications, Cambridge University Press, 2003, pp.47-100.

UNIVERSITY OF Southampton

Next Lecture: OWL