Working Modularly with Ontologies

Chiara Del Vescovo Bijan Parsia Uli Sattler
Thomas Schneider

School of Computer Science, University of Manchester, UK

25 March 2010

About the project

Title

Composing and decomposing ontologies: a logic-based approach

People involved/interested

- Chiara Del Vescovo, Rafael Goncalves, Uli Sattler, Bijan Parsia, Thomas Schneider (Manchester)
- Frank Wolter, Boris Konev, Dirk Walther (Liverpool)
- Ian Horrocks, Bernardo Cuenca Grau, Yevgeny Kazakov (Oxford)
- Carsten Lutz (Bremen)
- Michael Zakharyaschev, Roman Kontchakov (London)

And now . . .

Ontology

= collection of statements about a domain *(axioms)*

- Language used: usually logic, often *description logic (DL)*
- *Inferences* can be drawn from axioms

Domains:
biology, medicine, chemistry, business processes, natural language, ...

Example axioms + inferences

- $\underbrace{\text{Duck}}_{\text{concept}} \sqsubseteq \underbrace{\exists \underbrace{\text{feedsOn}}_{\text{role}} . \underbrace{\text{Grass}}_{\text{concept}}}_{\text{concept}}$

$$\forall x \Big(\text{Duck}(x) \rightarrow \exists y \big(\text{feedsOn}(x, y) \wedge \text{Grass}(y) \big) \Big)$$

Example axioms + inferences

- $\underbrace{\text{Duck}}_{\text{concept}} \sqsubseteq \underbrace{\exists \underbrace{\text{feedsOn}}_{\text{role}} . \underbrace{\text{Grass}}_{\text{concept}}}_{\text{concept}}$

$$\forall x \Big(\text{Duck}(x) \rightarrow \exists y \big(\text{feedsOn}(x, y) \wedge \text{Grass}(y) \big) \Big)$$

- $\text{Bird} \equiv \text{Duck} \sqcup \text{Chicken}$

$$\forall x \Big(\text{Bird}(x) \leftrightarrow \big(\text{Duck}(x) \vee \text{Chicken}(x) \big) \Big)$$

Example axioms + inferences

- $\underbrace{\text{Duck}}_{\text{concept}} \sqsubseteq \underbrace{\exists \underbrace{\text{feedsOn}}_{\text{role}} . \underbrace{\text{Grass}}_{\text{concept}}}_{\text{concept}}$

$$\forall x \Big(\text{Duck}(x) \rightarrow \exists y \big(\text{feedsOn}(x, y) \wedge \text{Grass}(y) \big) \Big)$$

- $\text{Bird} \equiv \text{Duck} \sqcup \text{Chicken}$

$$\forall x \Big(\text{Bird}(x) \leftrightarrow \big(\text{Duck}(x) \vee \text{Chicken}(x) \big) \Big)$$

$\models \text{Bird} \sqcap \neg\text{Chicken} \sqsubseteq \exists\, \text{feedsOn}.\text{Grass}$

$$\forall x \Big(\big(\text{Bird}(x) \wedge \neg\text{Chicken}(x) \big) \rightarrow \exists y \big(\text{feedsOn}(x, y) \wedge \text{Grass}(y) \big) \Big)$$

Example axioms + inferences

- $\underbrace{\text{Duck}}_{\text{concept}} \sqsubseteq \underbrace{\exists \underbrace{\text{feedsOn}}_{\text{role}} . \underbrace{\text{Grass}}_{\text{concept}}}_{\text{concept}}$

$$\forall x \Big(\text{Duck}(x) \rightarrow \exists y \big(\text{feedsOn}(x, y) \wedge \text{Grass}(y) \big) \Big)$$

- $\text{Bird} \equiv \text{Duck} \sqcup \text{Chicken}$

$$\forall x \Big(\text{Bird}(x) \leftrightarrow \big(\text{Duck}(x) \vee \text{Chicken}(x) \big) \Big)$$

Example axioms + inferences

- $\underbrace{\text{Duck}}_{\text{concept}} \sqsubseteq \underbrace{\exists \underbrace{\text{feedsOn}}_{\text{role}} . \underbrace{\text{Grass}}_{\text{concept}}}_{\text{concept}}$

$$\forall x \Big(\text{Duck}(x) \rightarrow \exists y \big(\text{feedsOn}(x, y) \wedge \text{Grass}(y) \big) \Big)$$

- $\text{Bird} \equiv \text{Duck} \sqcup \text{Chicken}$

$$\forall x \Big(\text{Bird}(x) \leftrightarrow \big(\text{Duck}(x) \vee \text{Chicken}(x) \big) \Big)$$

- $\underbrace{\text{Tweety}}_{\text{individual}} : \text{Duck}$ $\qquad\qquad$ $\text{Duck}(\text{Tweety})$

Example axioms + inferences

- $\underbrace{\text{Duck}}_{\text{concept}} \sqsubseteq \underbrace{\exists \underbrace{\text{feedsOn}}_{\text{role}} . \underbrace{\text{Grass}}_{\text{concept}}}_{\text{concept}}$

$$\forall x \Big(\text{Duck}(x) \rightarrow \exists y \big(\text{feedsOn}(x, y) \wedge \text{Grass}(y) \big) \Big)$$

- $\text{Bird} \equiv \text{Duck} \sqcup \text{Chicken}$

$$\forall x \Big(\text{Bird}(x) \leftrightarrow \big(\text{Duck}(x) \vee \text{Chicken}(x) \big) \Big)$$

- $\underbrace{\text{Tweety}}_{\text{individual}} : \text{Duck}$

$$\text{Duck}(\text{Tweety})$$

$\models$ $\text{Tweety} : \exists \text{feedsOn}.\text{Grass}$

$$\exists y \big(\text{feedsOn}(\text{Tweety}, y) \wedge \text{Grass}(y) \big)$$

Reasoning tasks

- *Consistency:*
  Does ontology $\mathcal{O}$ have a model?
- *Satisfiability:*
  Is there a model of $\mathcal{O}$ that interprets concept C as nonempty?
- *Subsumption:*
  Does $C \sqsubseteq D$ hold in every model of $\mathcal{O}$?
- *Instance checking:*
  Is individual x an instance of C in every model of $\mathcal{O}$?

Inter-reducible; optimised reasoners available

The Web Ontology Language OWL

- W3C-recommended standard since 2004
- OWL 2 published on 27 Oct. 2009

The Web Ontology Language OWL

- W3C-recommended standard since 2004
- OWL 2 published on 27 Oct. 2009

OWL Full

Consistency?, ~~Reasoning~~

OWL DL

Based on DL $\mathcal{SROIQ}$
$\exists$, $\forall$, counting, role chains and hierarchies, transitivity, inverse roles, nominals

OWL EL, QL, RL

Sub-profiles for efficient reasoning and application orientation

And now . . .

A case for modularity

Common practice in software engineering

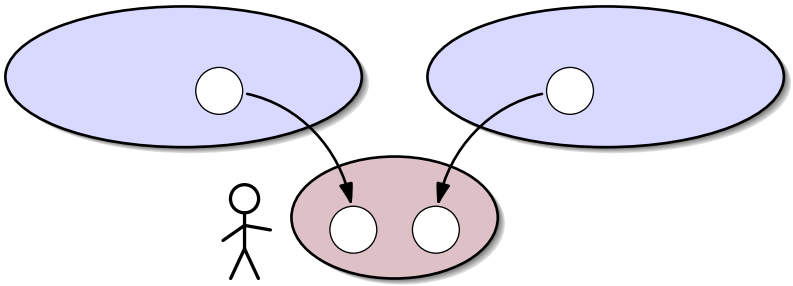
Modular software development allows for:

- Importing/reusing modules
- Collaborative development
- Understanding the code from the interaction between the modules

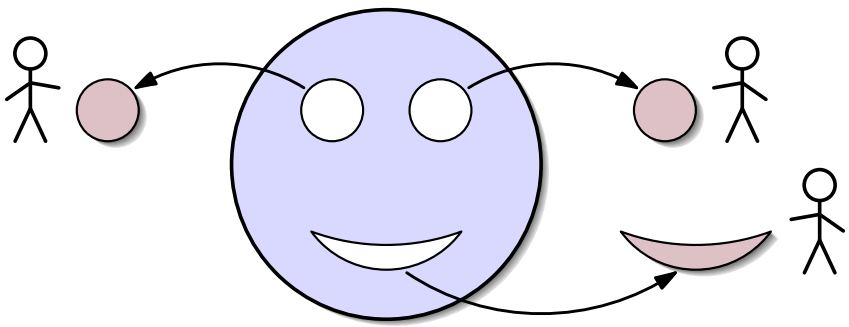
Wouldn't it be nice . . .

. . . to have this for ontology development as well?

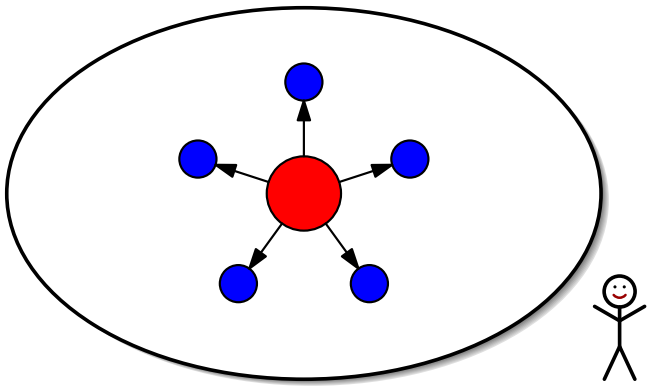
Three scenarios



Import/reuse

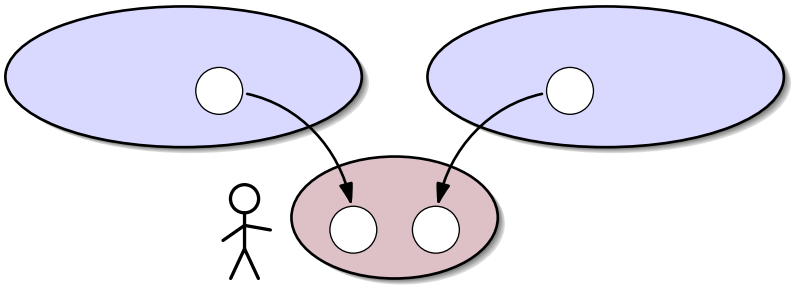


Collaboration

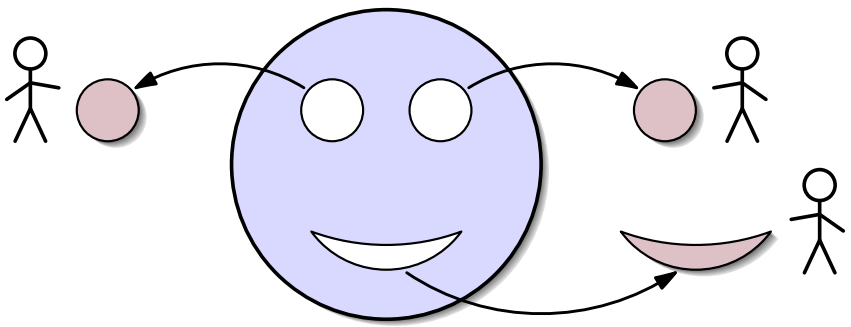


Understanding

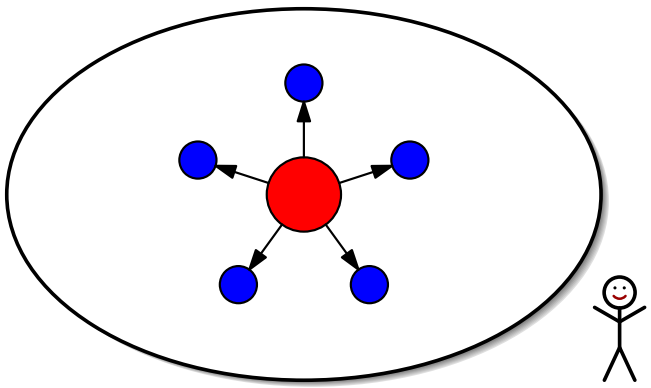
Three scenarios



Import/reuse



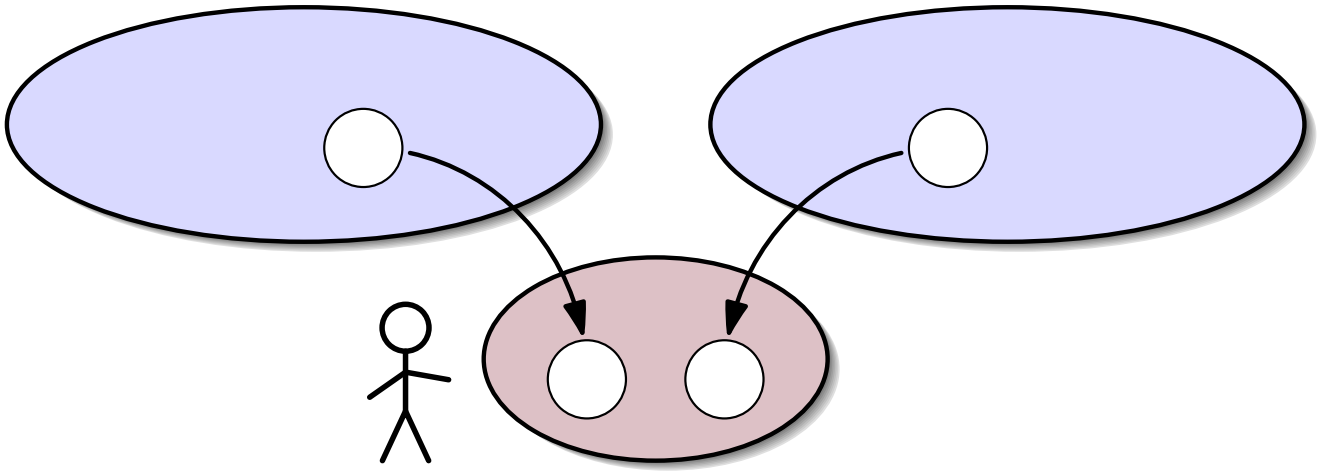
Collaboration



Understanding

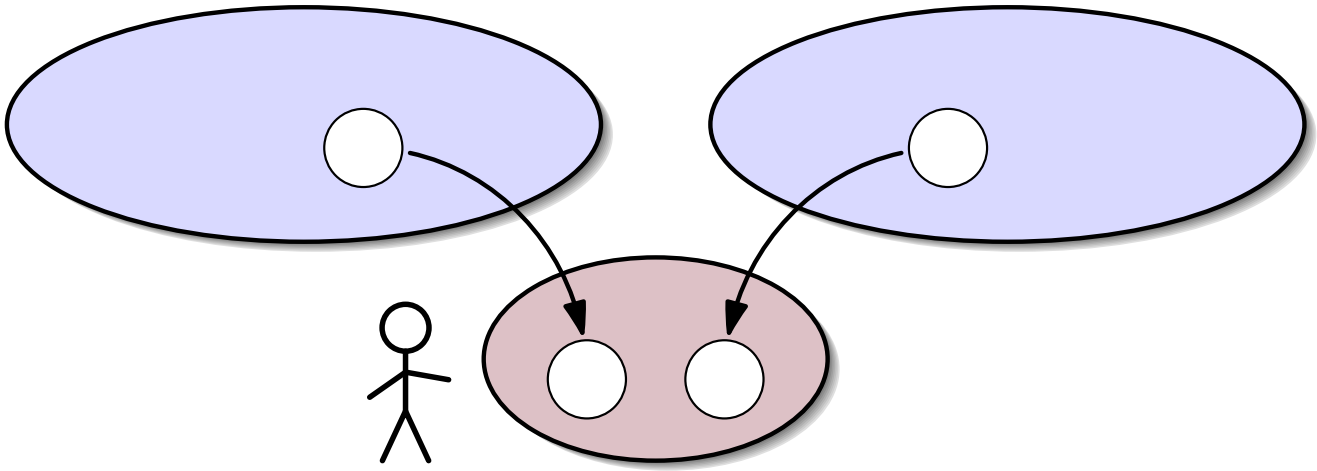
Scenario 1: Import/reuse

"Borrow" knowledge about certain terms from external ontologies



Scenario 1: Import/reuse

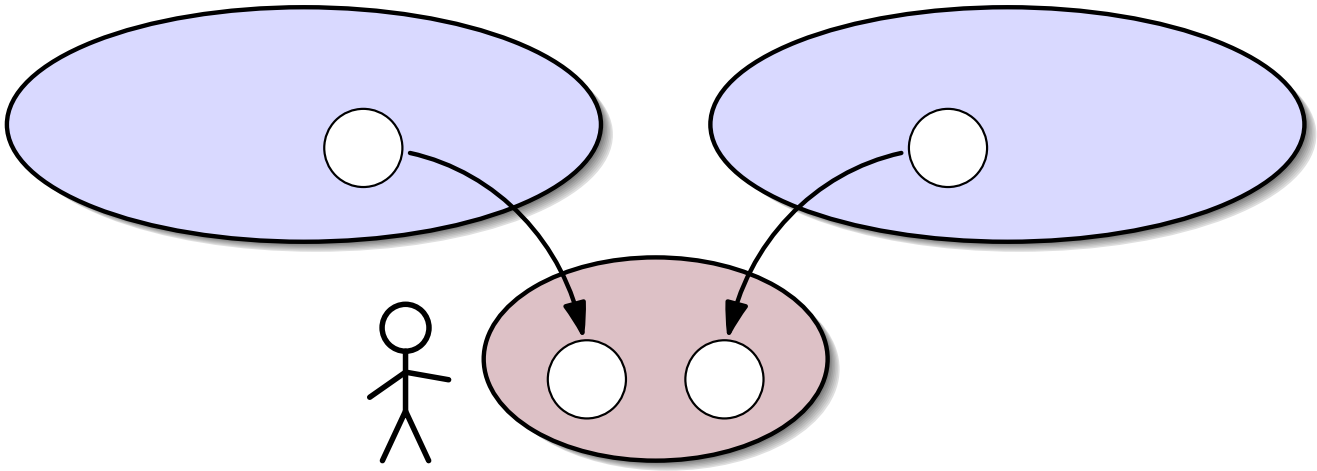
"Borrow" knowledge about certain terms from external ontologies



- Provides access to well-established knowledge
- Doesn't require expertise in external disciplines

Scenario 1: Import/reuse

"Borrow" knowledge about certain terms from external ontologies

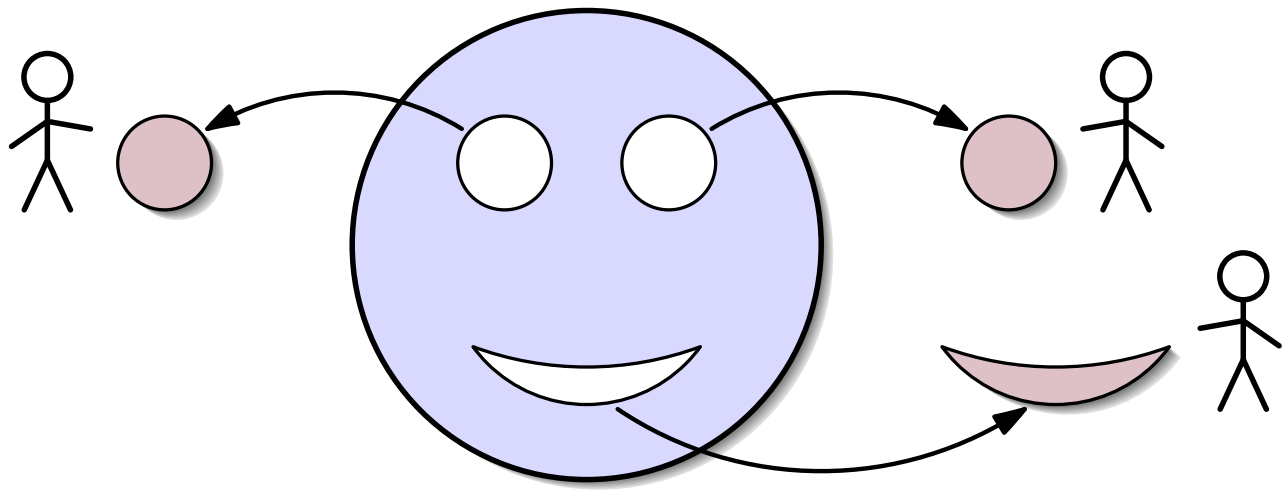


- Provides access to well-established knowledge
- Doesn't require expertise in external disciplines

This scenario is well-understood and implemented.

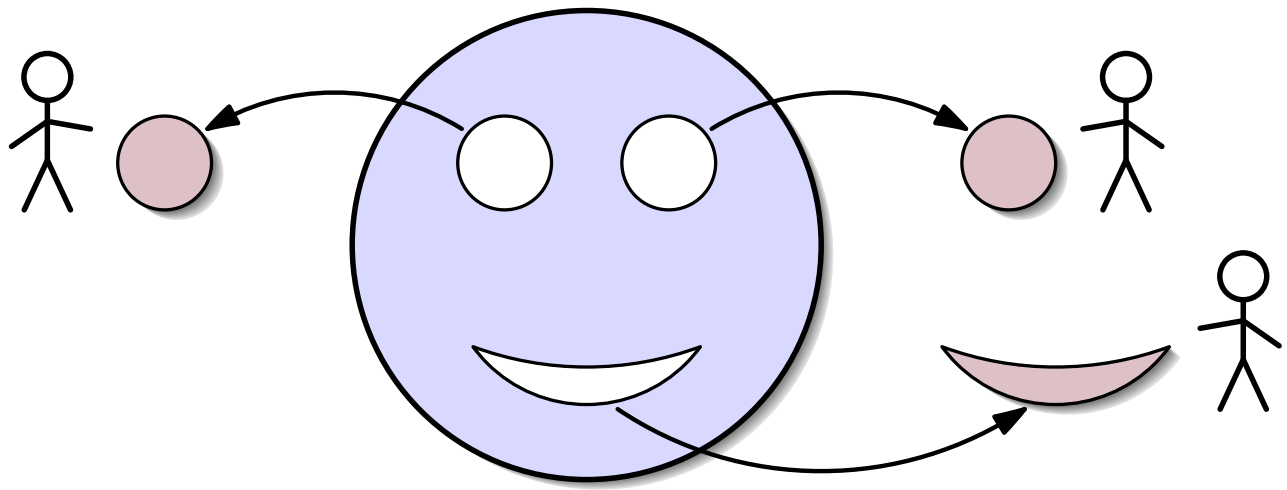
Scenario 2: Collaboration

Collaborative ontology development



Scenario 2: Collaboration

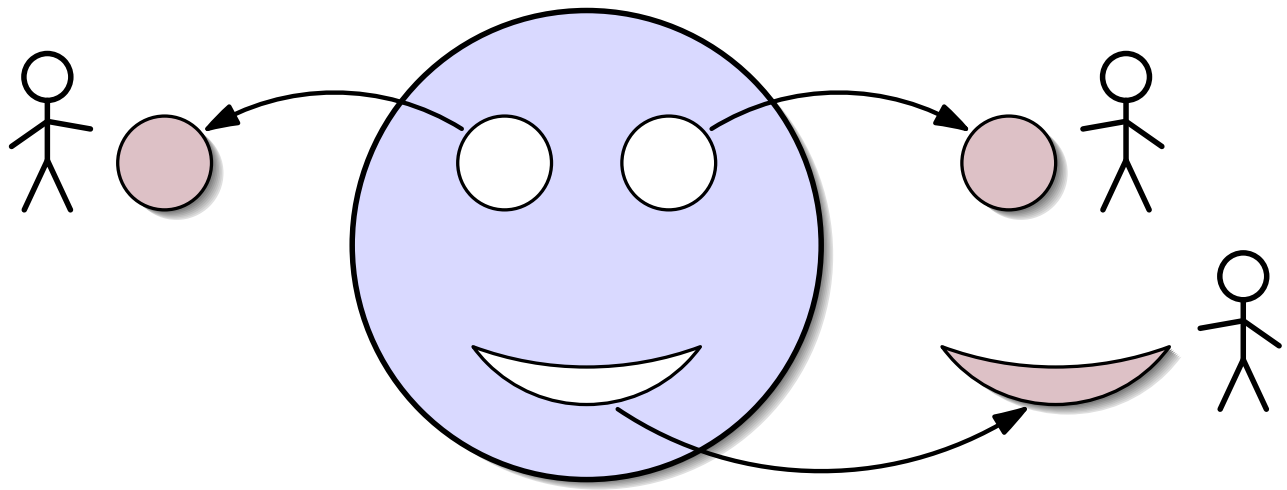
Collaborative ontology development



- Developers work (edit, classify) locally
- Extra care at re-combination

Scenario 2: Collaboration

Collaborative ontology development

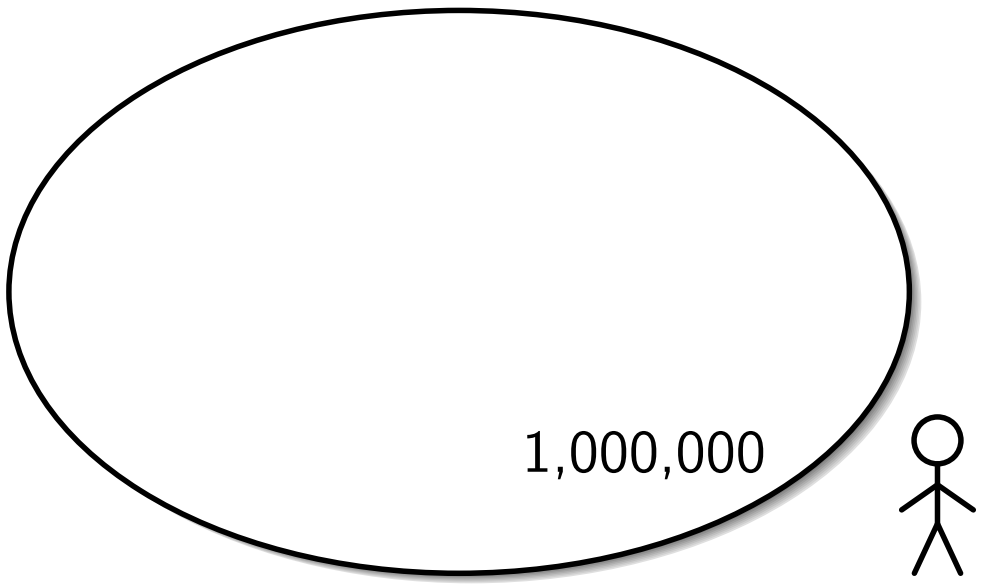


- Developers work (edit, classify) locally
- Extra care at re-combination

This approach is understood, but not implemented yet.

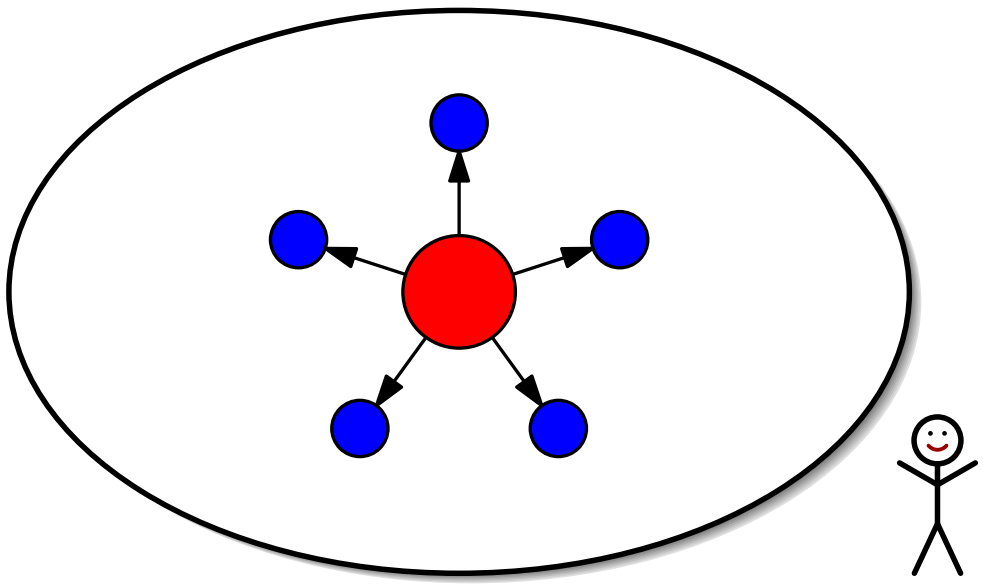
Scenario 3: Understanding

Visualise the modular structure of an ontology



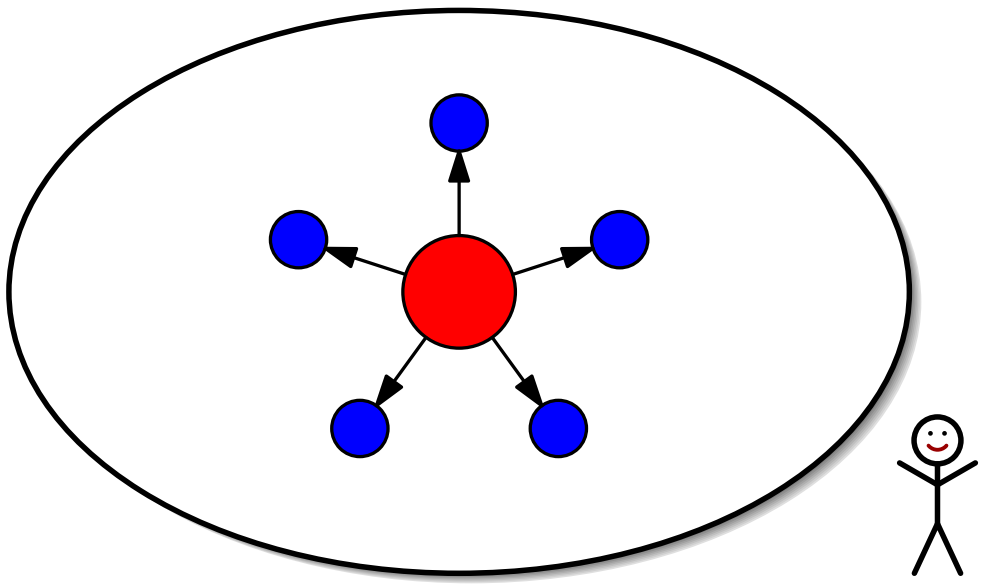
Scenario 3: Understanding

Visualise the modular structure of an ontology



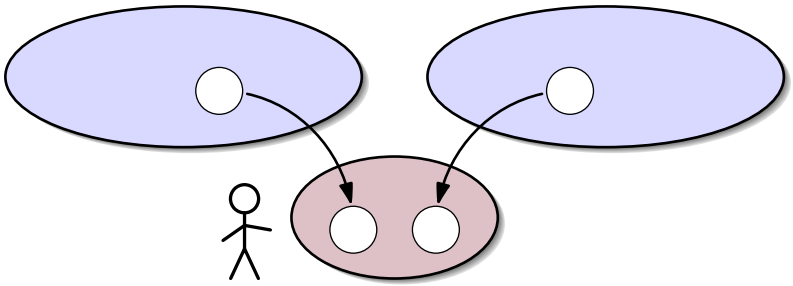
Scenario 3: Understanding

Visualise the modular structure of an ontology

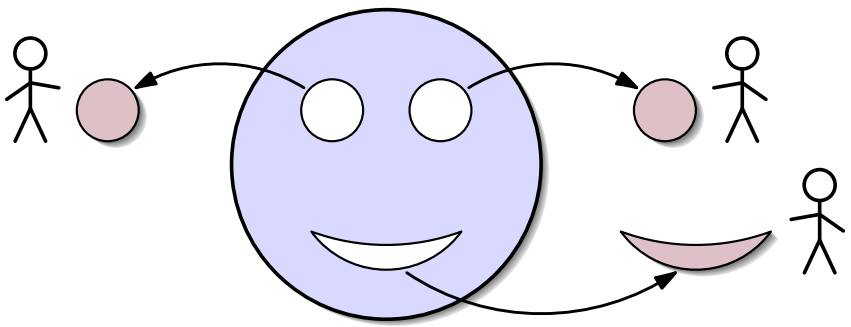


We're still playing with this.

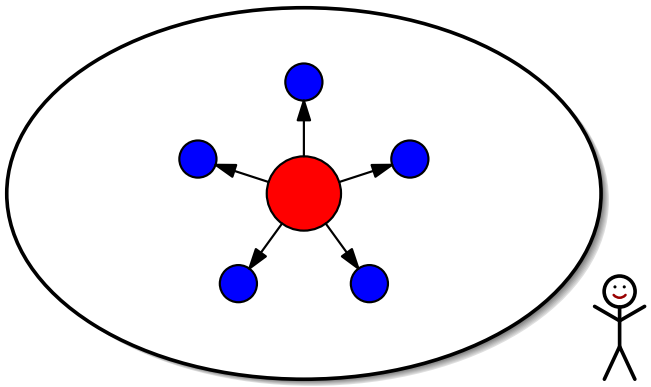
Summing up



Import/reuse

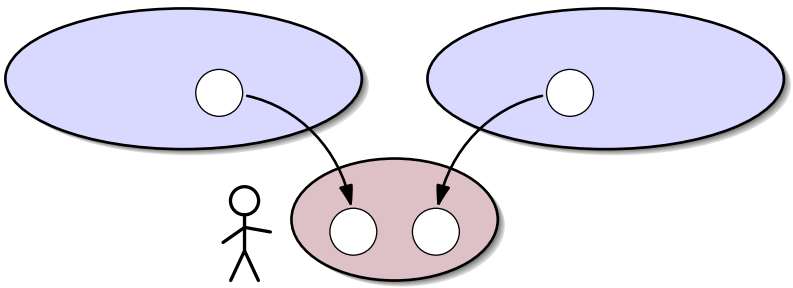


Collaboration

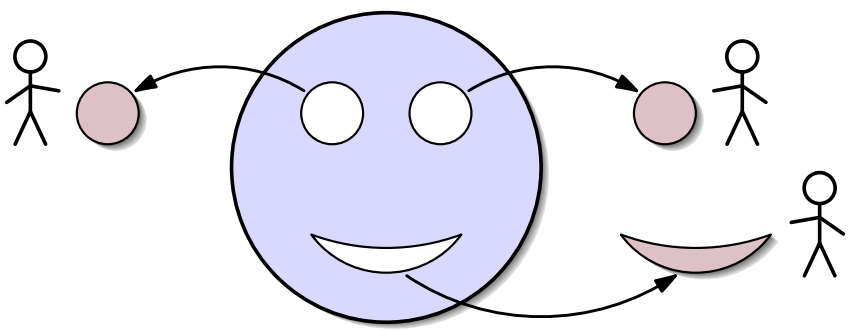


Understanding

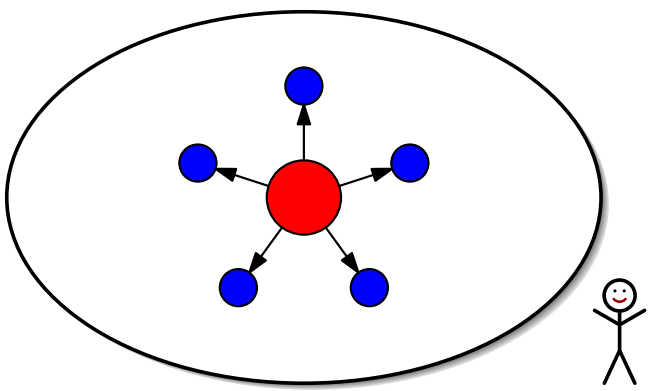
Summing up



Import/reuse

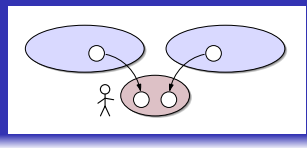


Collaboration

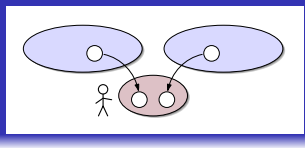


Understanding

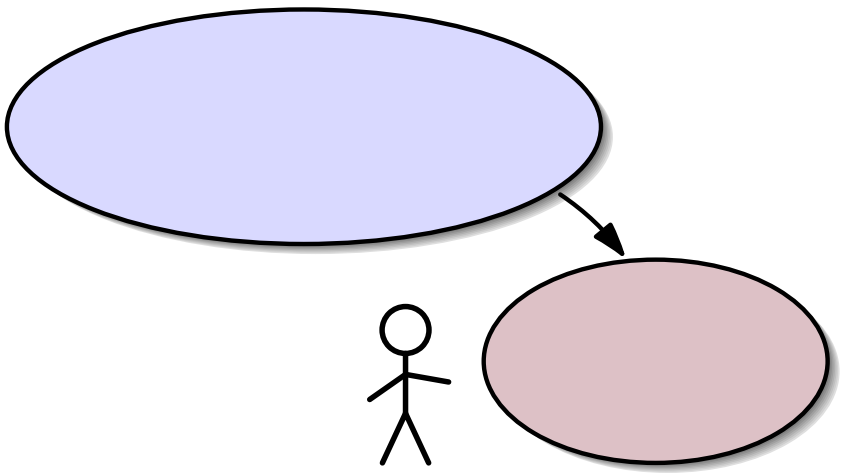
And now . . .



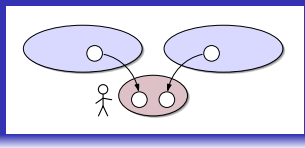
A reuse scenario



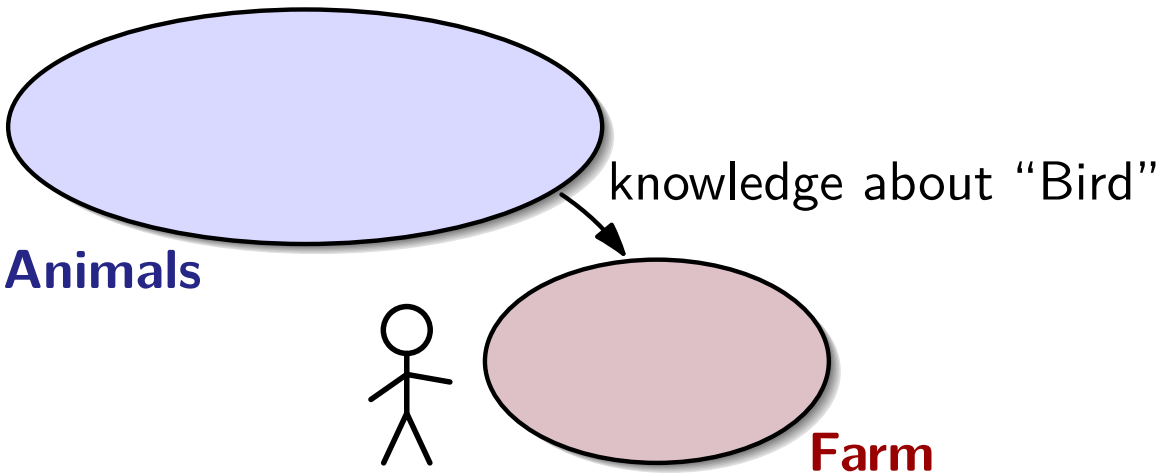
Import/reuse one external ontology



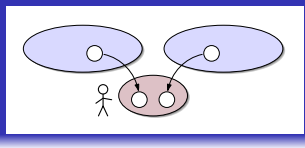
A reuse scenario



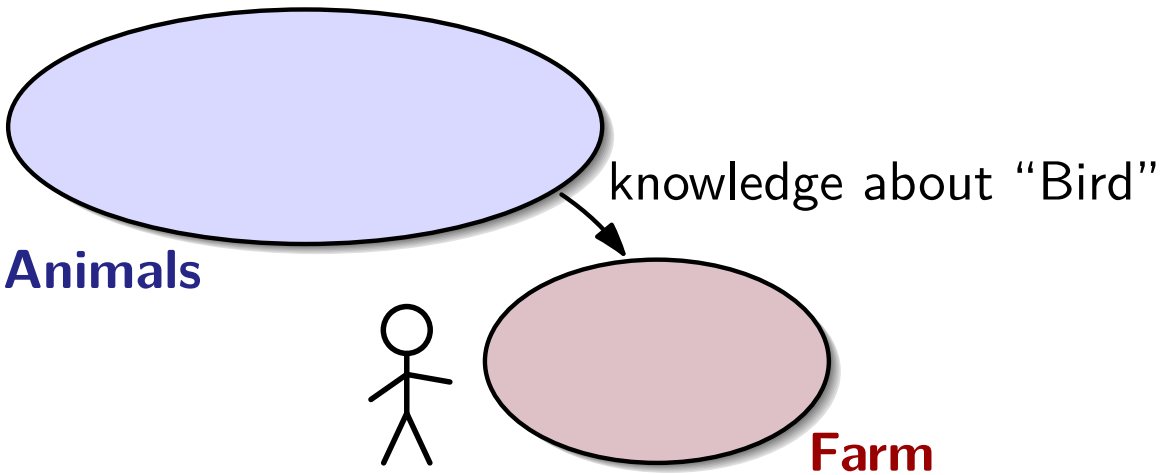
Import/reuse one external ontology



A reuse scenario

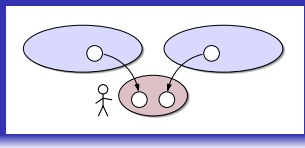


Import/reuse one external ontology

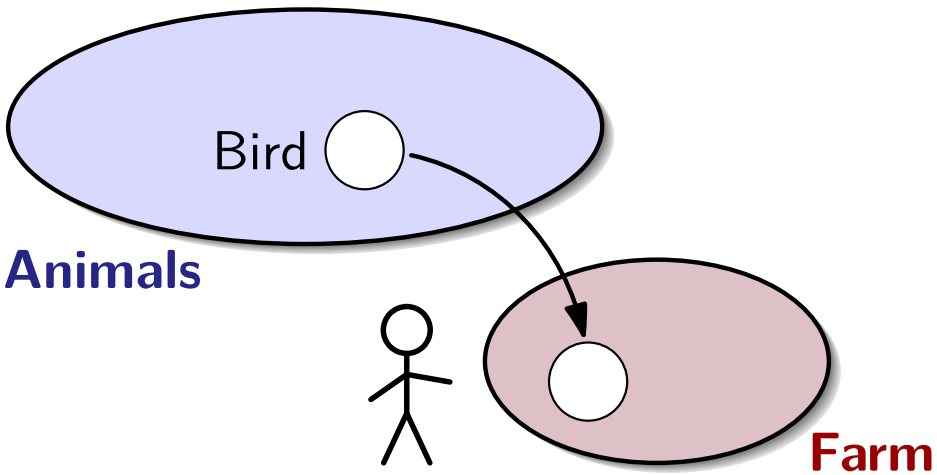


How much of **Animals** do we need?

A reuse scenario



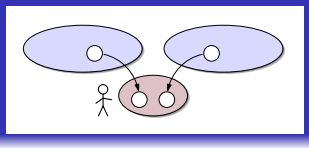
Import/reuse a part of an external ontology



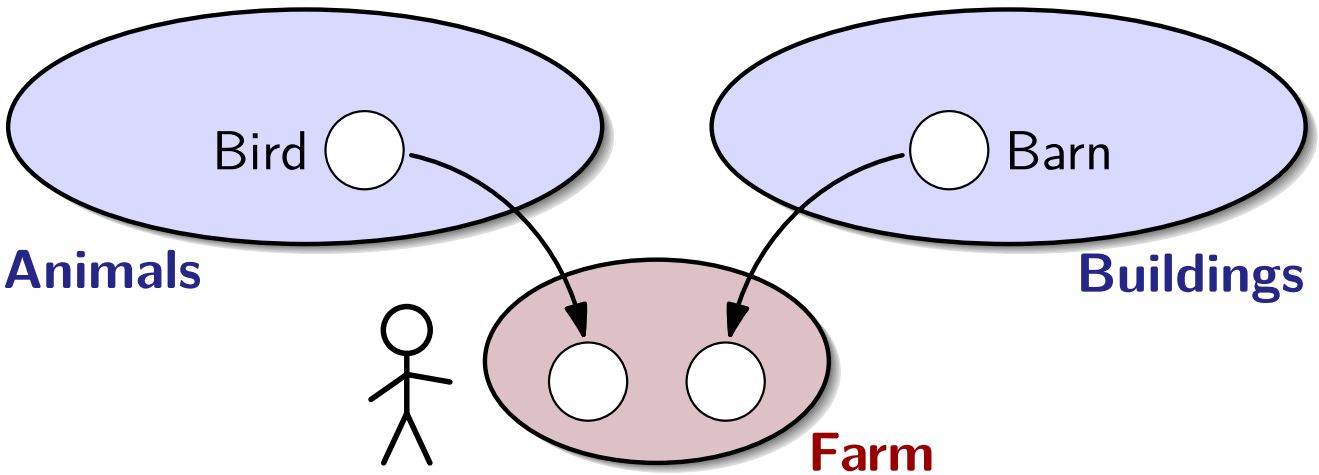
How much of **Animals** do we need?

- **Coverage:** Import *everything* relevant for the chosen terms.
- **Economy:** Import *only* what's relevant for them.
  Compute that part quickly.

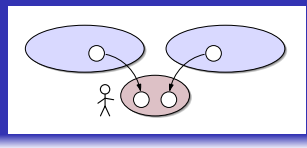
A reuse scenario

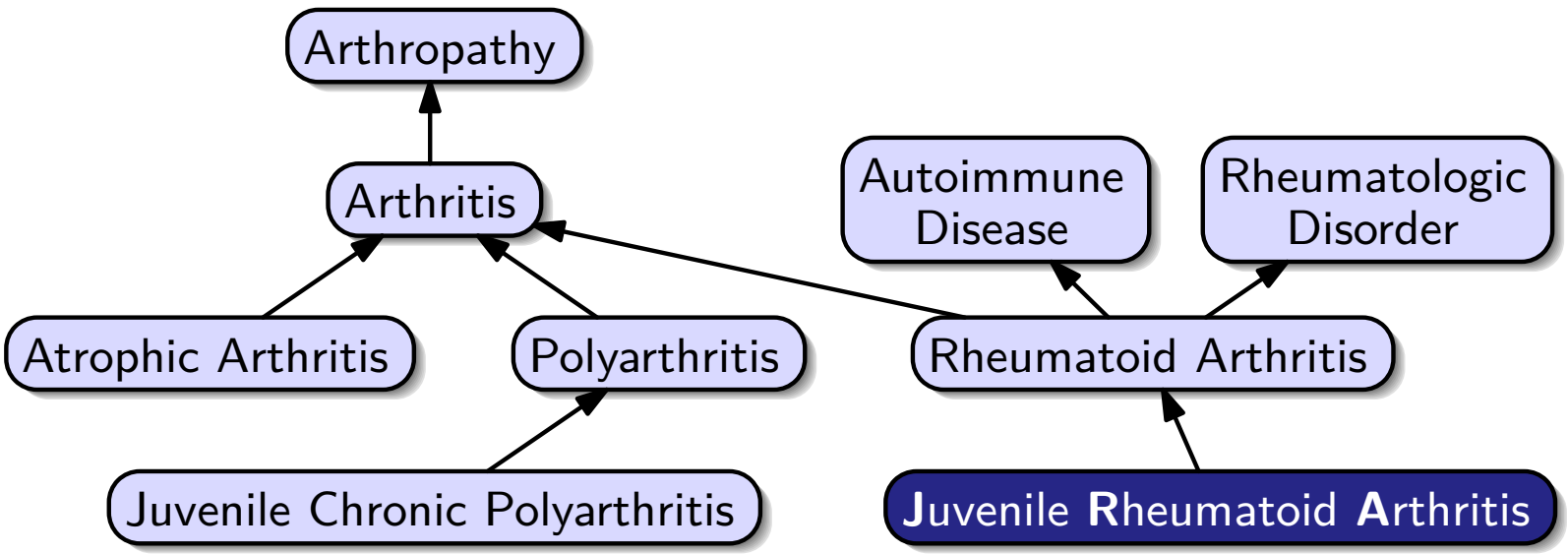


Import/reuse parts of several external ontologies

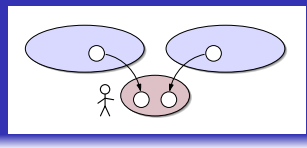


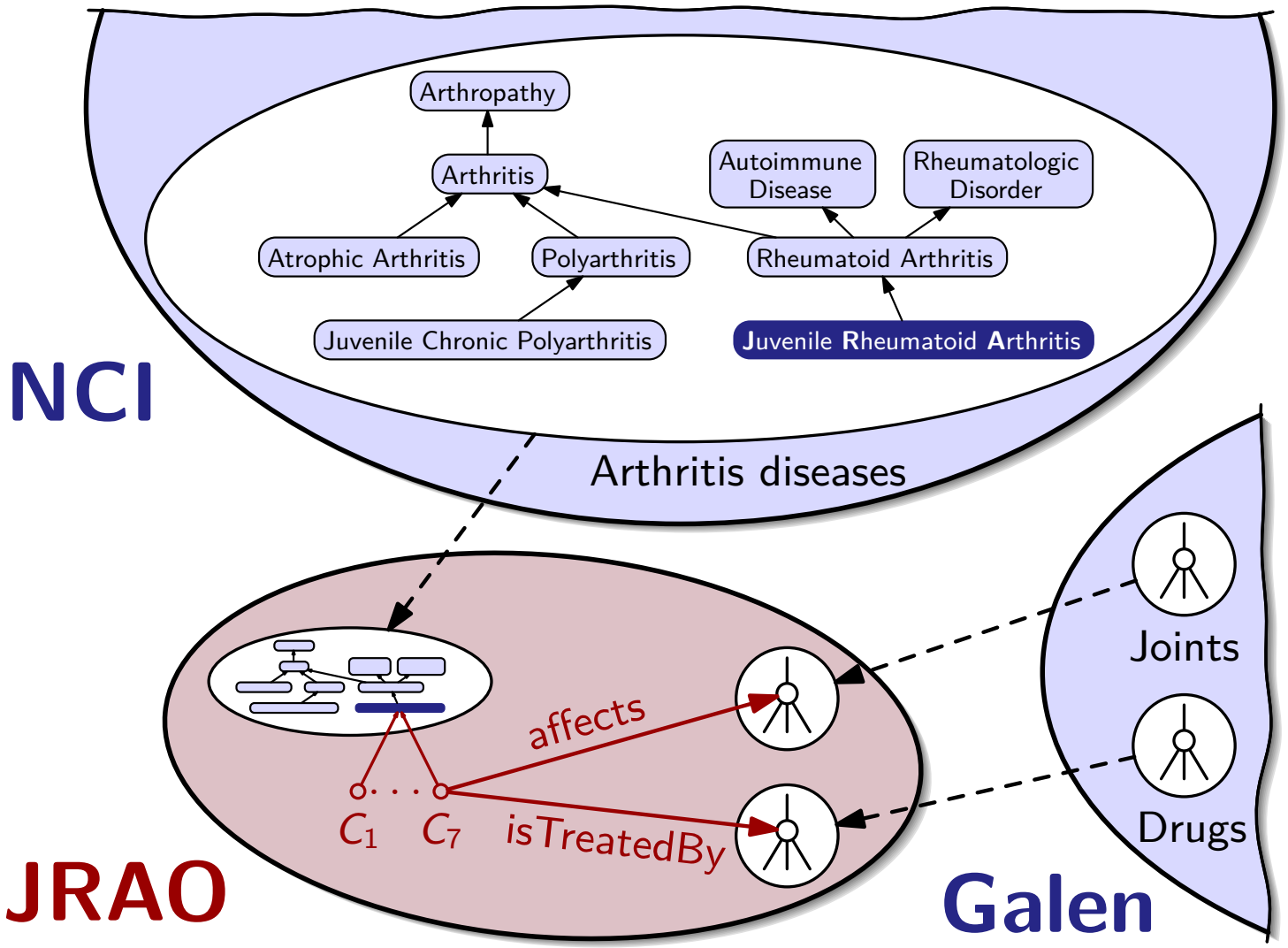
The *Health-e-Child* project



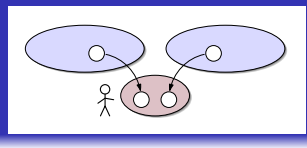


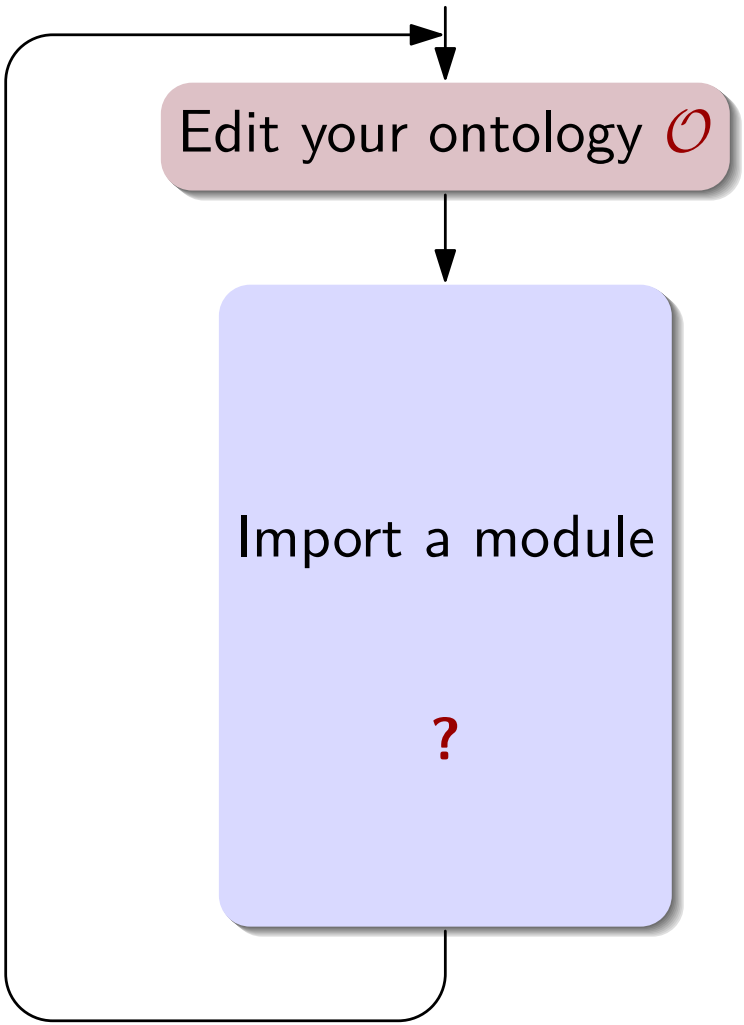
The *Health-e-Child* project



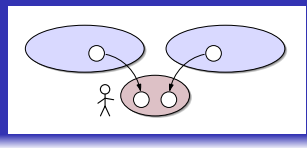


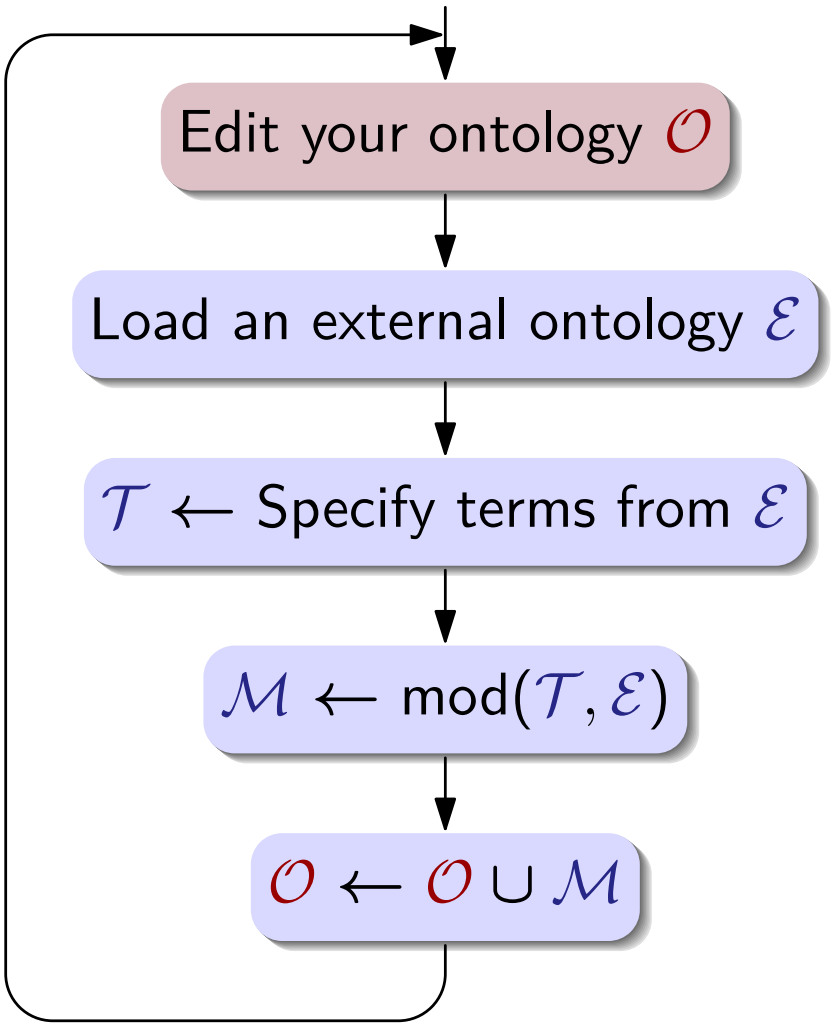
A working cycle



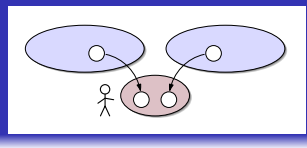


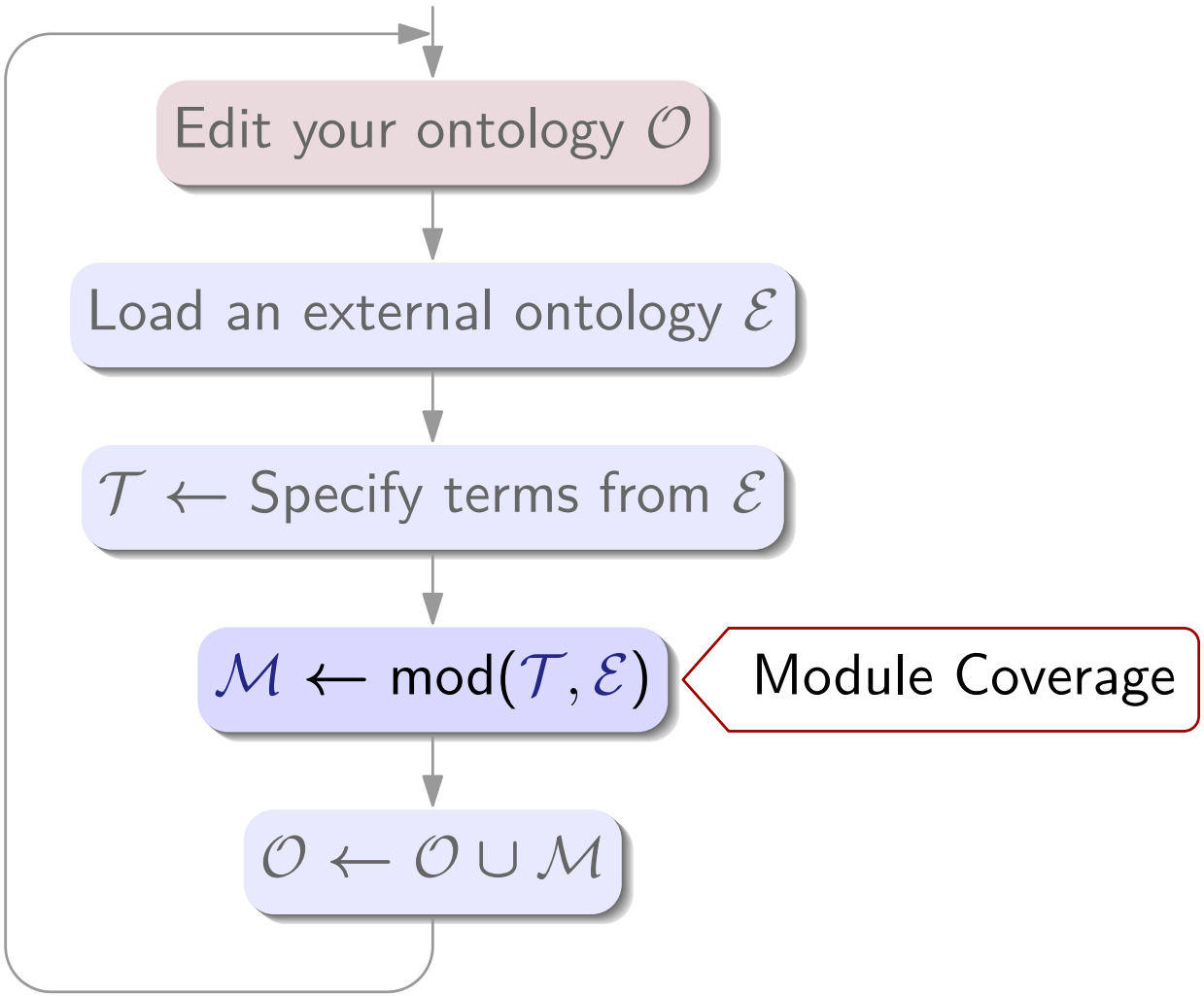
A working cycle



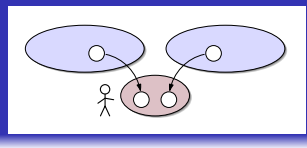


A working cycle



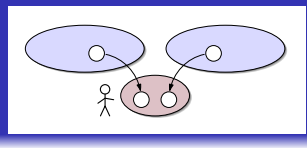


Module coverage



Goal: Import everything the external ontology knows about the topic that consists of the specified terms.

Module coverage



Goal: Import everything the external ontology knows about the topic that consists of the specified terms.

Example 1:

- Topic: Fox, Bird, feedsOn
- On-topic:

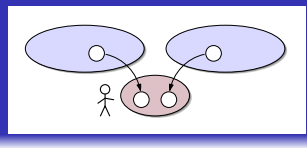
$$\begin{aligned} \text{Fox} &\sqsubseteq \forall\, \text{feedsOn}.\text{Bird} \\ \text{Fox} \sqcup \text{Bird} &\sqsubseteq \exists\, \text{feedsOn}.\top \\ \text{Bird} &\sqsubseteq \neg \text{Fox} \\ \text{Bird} &\sqsubseteq \text{Bird} \sqcup \text{Fox} \end{aligned}$$

Off-topic:

$$\text{Duck} \sqsubseteq \text{Bird}$$

- Goal = preserve all on-topic knowledge

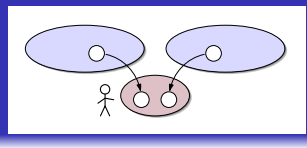
Module coverage



Goal: Import everything the external ontology knows about the topic that consists of the specified terms.

Question: Which axioms do we need to import?

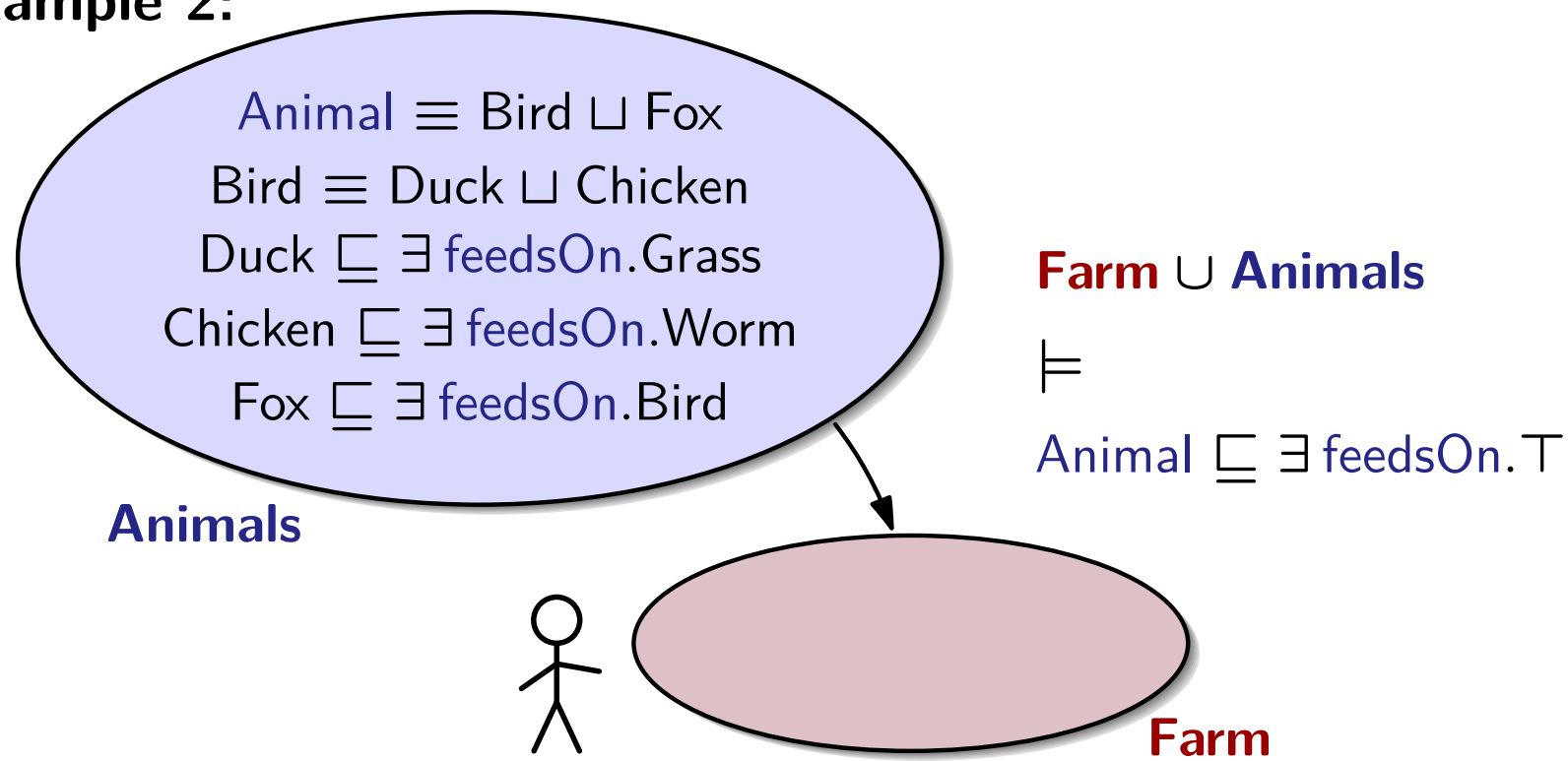
Module coverage



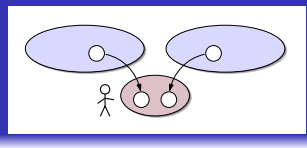
Goal: Import everything the external ontology knows about the topic that consists of the specified terms.

Question: Which axioms do we need to import?

Example 2:



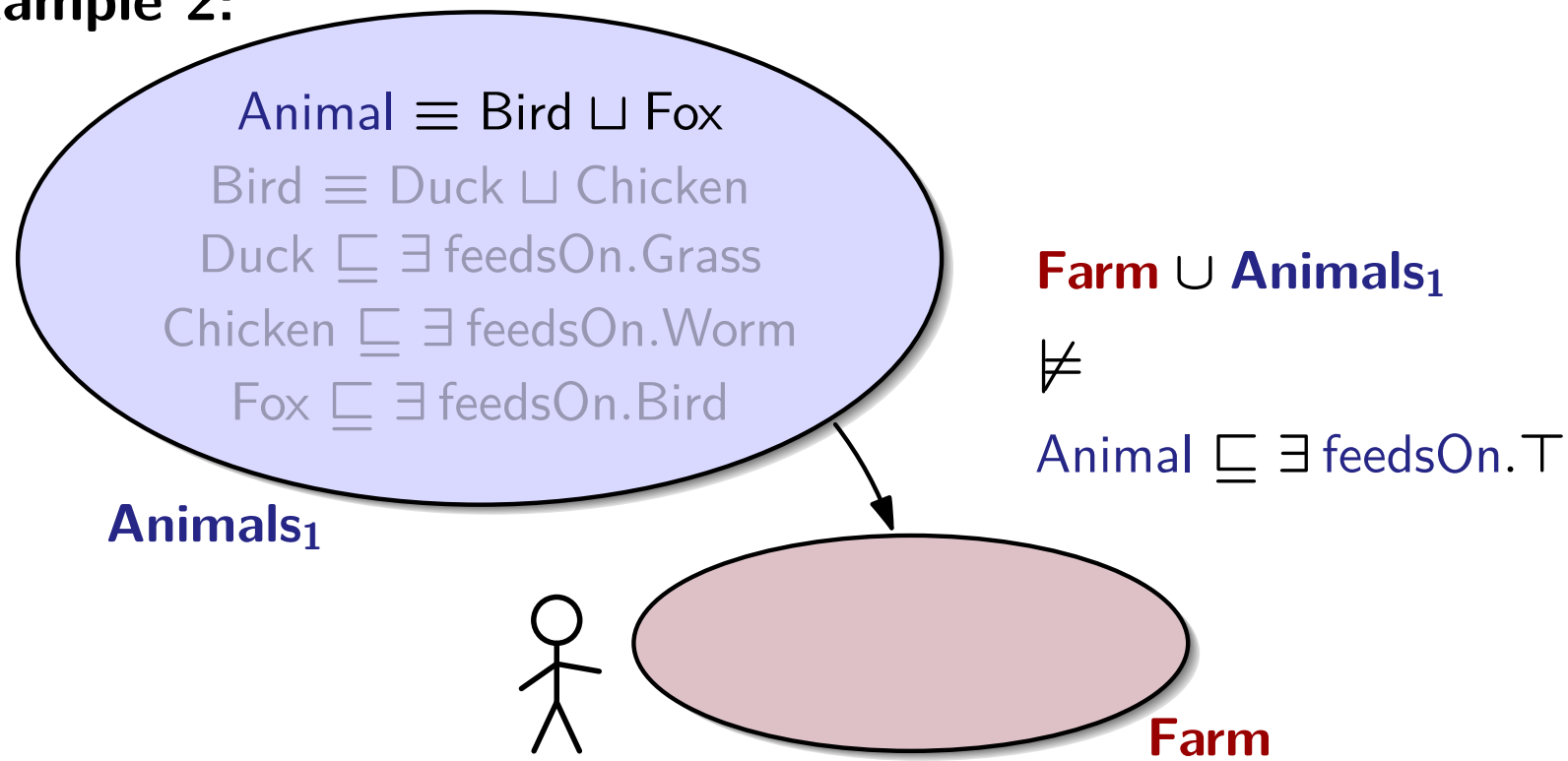
Module coverage



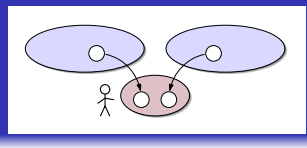
Goal: Import everything the external ontology knows about the topic that consists of the specified terms.

Question: Which axioms do we need to import?

Example 2:



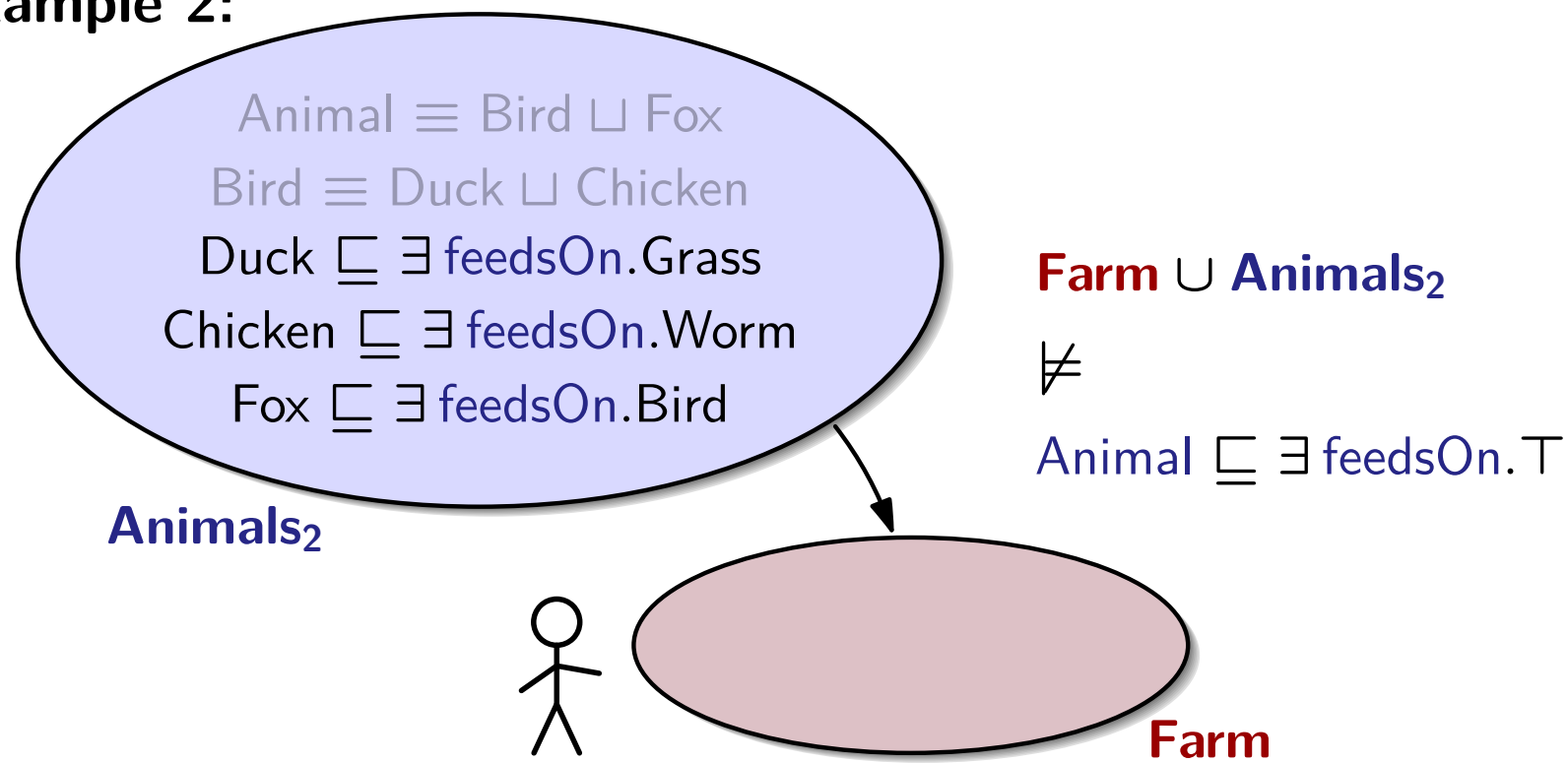
Module coverage



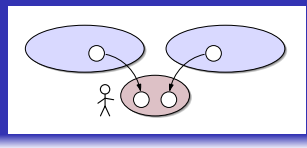
Goal: Import everything the external ontology knows about the topic that consists of the specified terms.

Question: Which axioms do we need to import?

Example 2:



Module coverage



Goal: Import everything the external ontology knows about the topic that consists of the specified terms.

Question: Which axioms do we need to import?

Example 2:

$\mathsf{Animal} \equiv \mathsf{Bird} \sqcup \mathsf{Fox}$

$\mathsf{Bird} \equiv \mathsf{Duck} \sqcup \mathsf{Chicken}$

$\mathsf{Duck} \sqsubseteq \exists\, \mathsf{feedsOn}.\mathsf{Grass}$

$\mathsf{Chicken} \sqsubseteq \exists\, \mathsf{feedsOn}.\mathsf{Worm}$

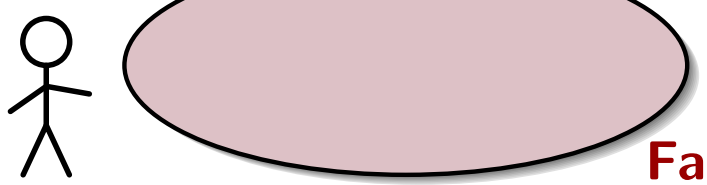
$\mathsf{Fox} \sqsubseteq \exists\, \mathsf{feedsOn}.\mathsf{Bird}$

$\mathsf{Animals}_3$

$\mathsf{Farm} \cup \mathsf{Animals}_3$

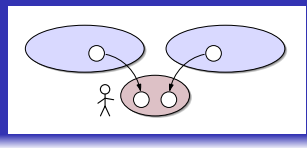
$\not\models$

$\mathsf{Animal} \sqsubseteq \exists\, \mathsf{feedsOn}.\top$



Farm

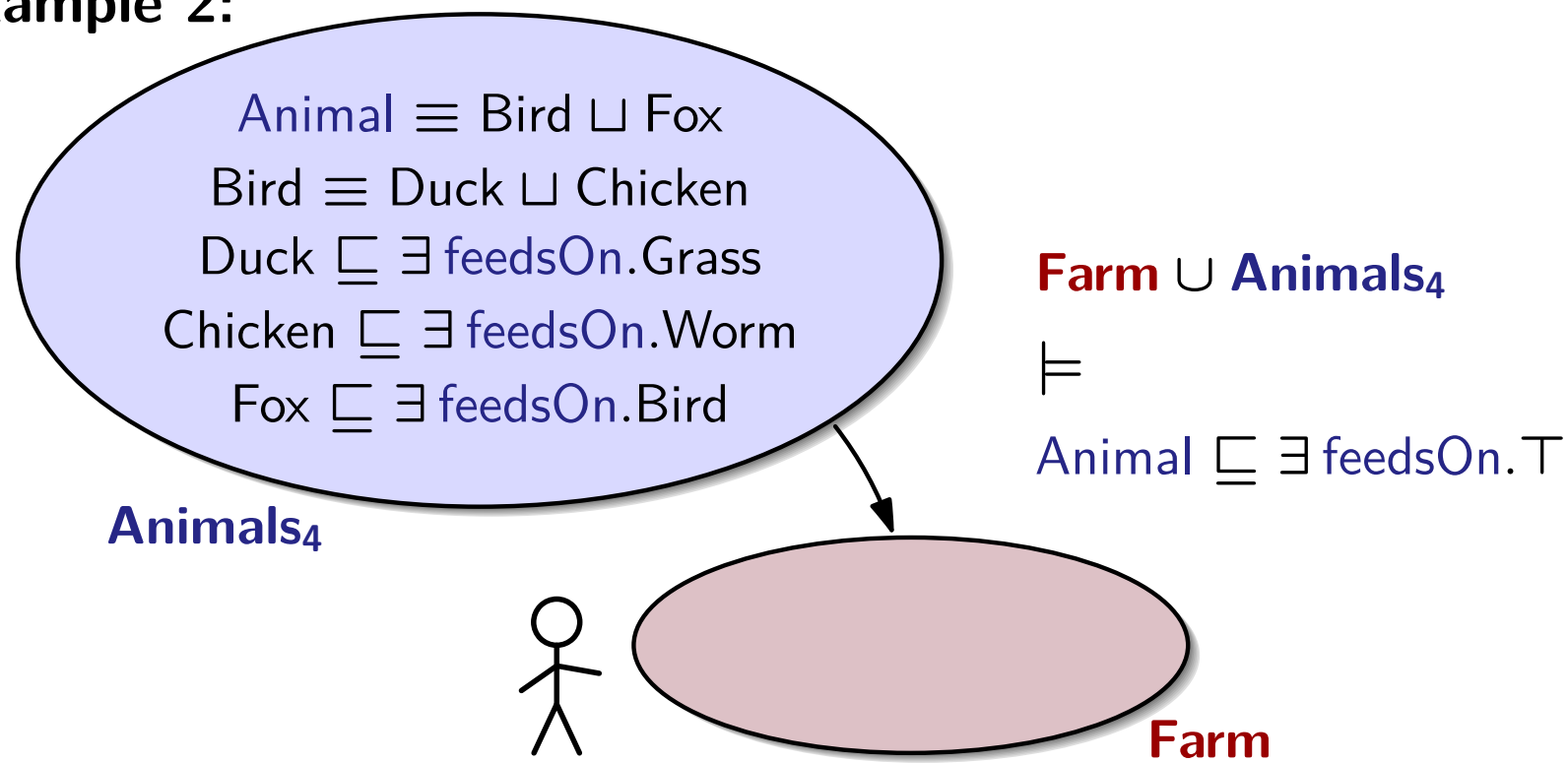
Module coverage



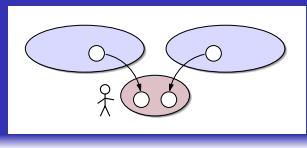
Goal: Import everything the external ontology knows about the topic that consists of the specified terms.

Question: Which axioms do we need to import?

Example 2:

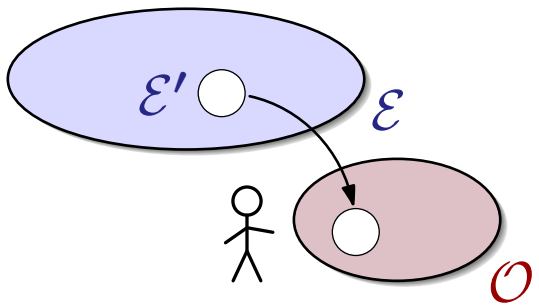


Module coverage

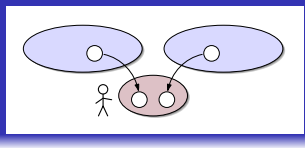


- Module $\mathcal{E}'$ covers ontology $\mathcal{E}$ for the specified topic $\mathcal{T}$ if for all concepts C, D built from terms in $\mathcal{T}$:

  if $\quad \mathcal{O} \cup \mathcal{E} \models \mathsf{C} \sqsubseteq \mathsf{D}$,
  then $\quad \mathcal{O} \cup \mathcal{E}' \models \mathsf{C} \sqsubseteq \mathsf{D}$.

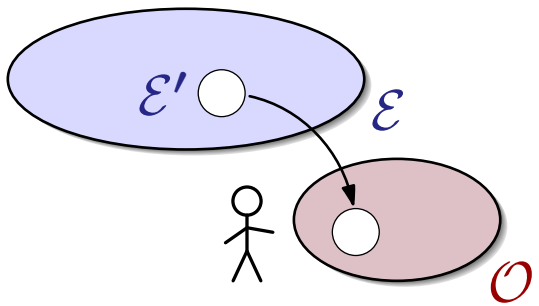


Module coverage



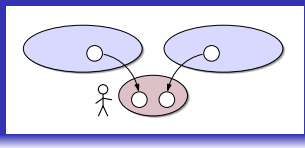
- Module $\mathcal{E}'$ covers ontology $\mathcal{E}$ for the specified topic $\mathcal{T}$ if for all concepts C, D built from terms in $\mathcal{T}$:

  if $\quad \mathcal{O} \cup \mathcal{E} \models C \sqsubseteq D,$
  then $\quad \mathcal{O} \cup \mathcal{E}' \models C \sqsubseteq D.$



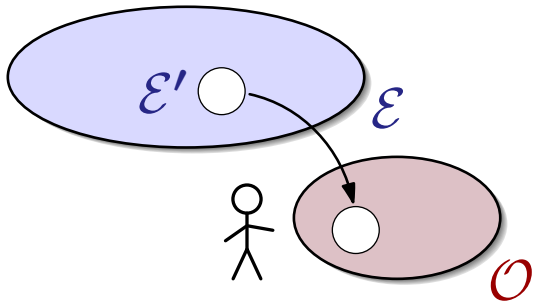
- Coverage $\hat{=}$ preserving entailments
- $\mathcal{O} \cup \mathcal{E}$ is called *conservative extension (CE)* of $\mathcal{O} \cup \mathcal{E}'$

Module coverage



- Module $\mathcal{E}'$ covers ontology $\mathcal{E}$ for the specified topic $\mathcal{T}$ if for all concepts C, D built from terms in $\mathcal{T}$:

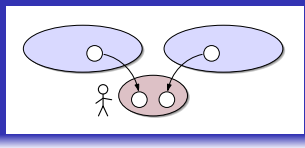
  if $\quad \mathcal{O} \cup \mathcal{E} \models \mathsf{C} \sqsubseteq \mathsf{D}$,
  then $\quad \mathcal{O} \cup \mathcal{E}' \models \mathsf{C} \sqsubseteq \mathsf{D}$.



- Coverage $\hat{=}$ preserving entailments
- $\mathcal{O} \cup \mathcal{E}$ is called *conservative extension (CE)* of $\mathcal{O} \cup \mathcal{E}'$

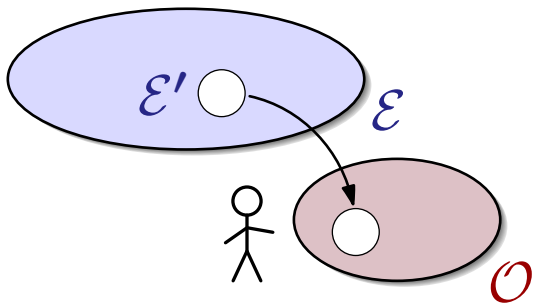
- No coverage $\rightsquigarrow$ no encapsulation $\rightsquigarrow$ no *module*
- With coverage: trade-off minimality $\leftrightarrow$ computation time

Module coverage



- Module $\mathcal{E}'$ covers ontology $\mathcal{E}$ for the specified topic $\mathcal{T}$ if for all concepts C, D built from terms in $\mathcal{T}$:

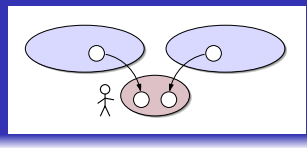
  if $\quad \mathcal{O} \cup \mathcal{E} \models C \sqsubseteq D,$
  then $\quad \mathcal{O} \cup \mathcal{E}' \models C \sqsubseteq D.$

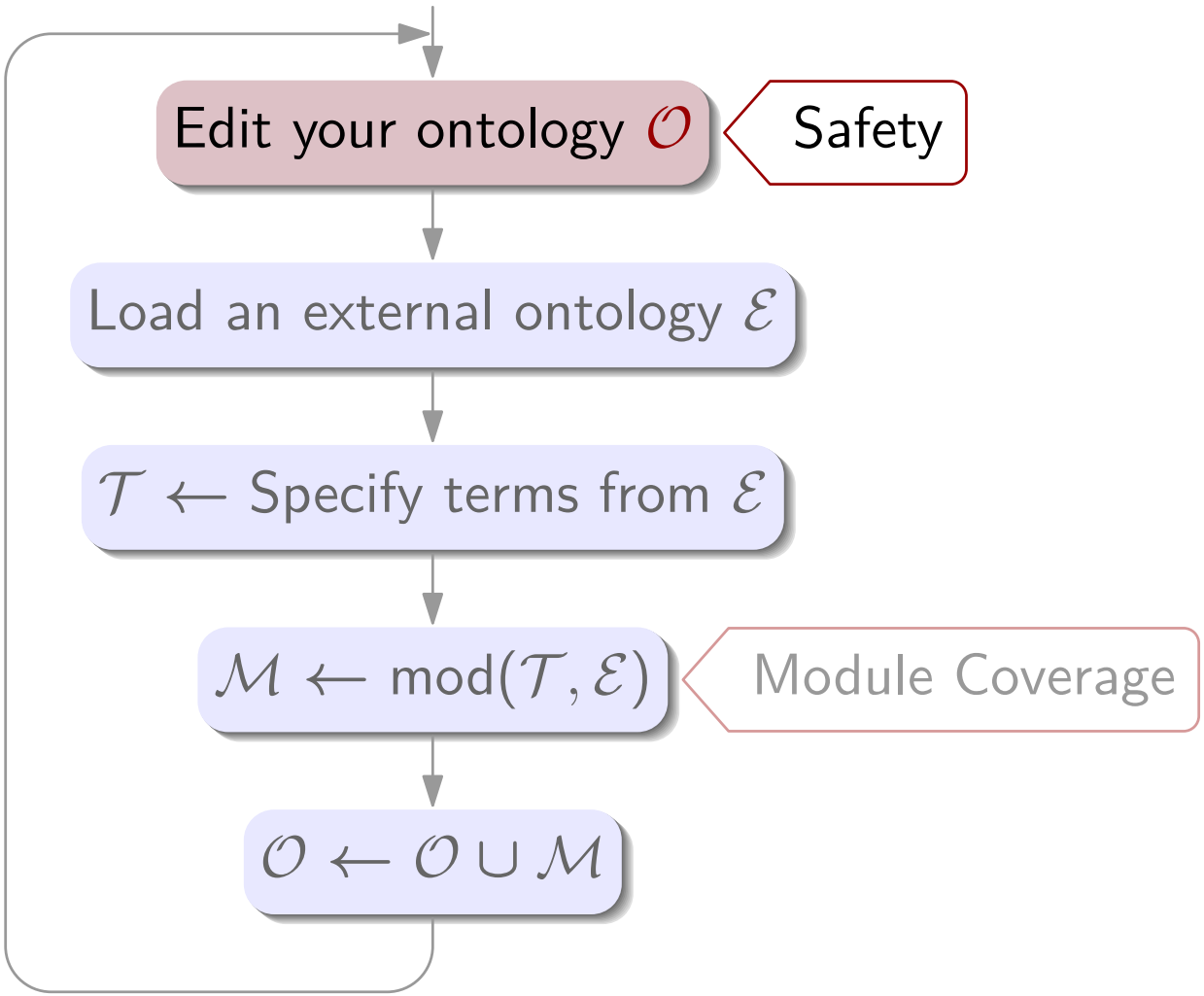


- Coverage $\hat{=}$ preserving entailments
- $\mathcal{O} \cup \mathcal{E}$ is called *conservative extension (CE)* of $\mathcal{O} \cup \mathcal{E}'$

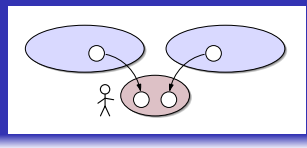
- Minmal covering modules via CEs
- CEs hard to impossible to decide
- Tractable approximation: syntactic locality

A working cycle





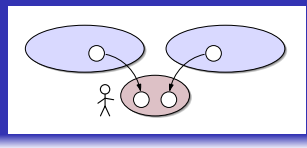
Safety



Goal: Don't change the meaning of imported terms.
= Don't add new knowledge about the imported topic.

Question: Which axioms are we allowed to write?

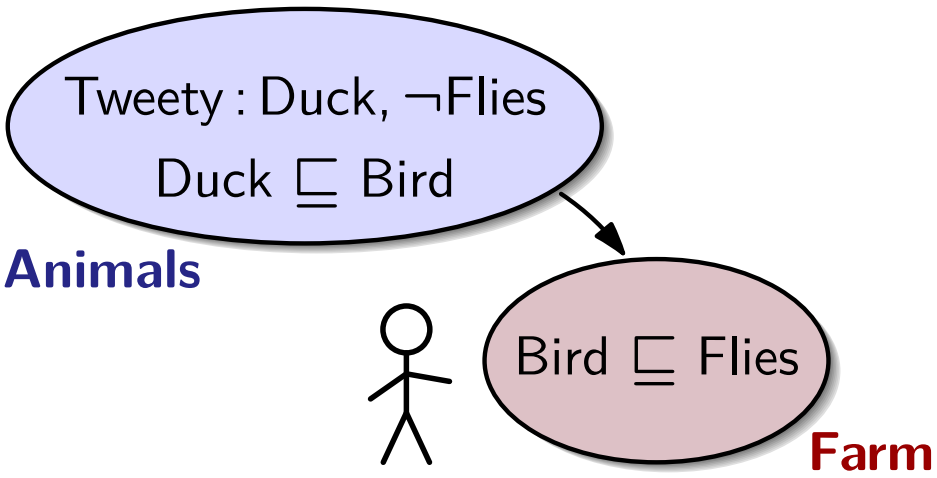
Safety



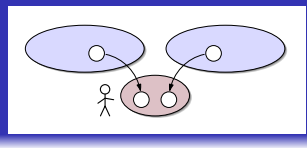
Goal: Don't change the meaning of imported terms.
= Don't add new knowledge about the imported topic.

Question: Which axioms are we allowed to write?

Example:



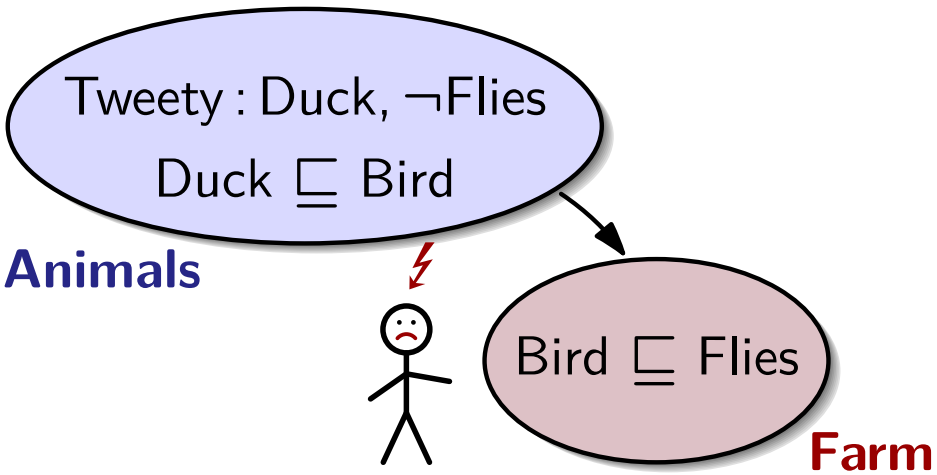
Safety



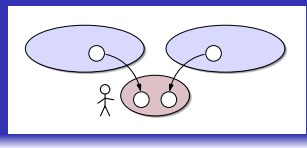
Goal: Don't change the meaning of imported terms.
= Don't add new knowledge about the imported topic.

Question: Which axioms are we allowed to write?

Example:



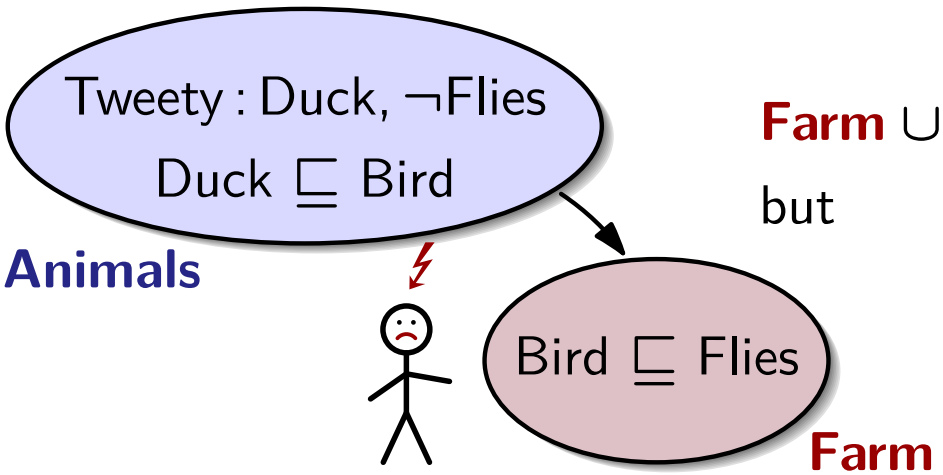
Safety



Goal: Don't change the meaning of imported terms.
= Don't add new knowledge about the imported topic.

Question: Which axioms are we allowed to write?

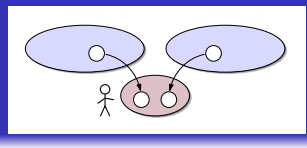
Example:



$\mathbf{Farm} \cup \mathbf{Animals} \models \mathrm{Bird} \sqsubseteq \mathrm{Flies}$

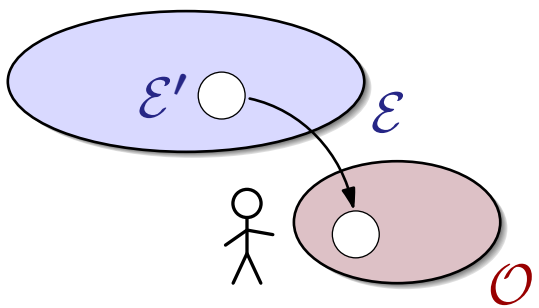
but $\mathbf{Animals} \not\models \mathrm{Bird} \sqsubseteq \mathrm{Flies}$

Safety



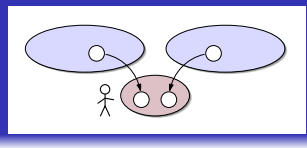
- *Our* ontology $\mathcal{O}$ uses the imported terms safely if for all concepts C, D built from the imported terms:

  if $\quad \mathcal{E}' \not\models \mathsf{C} \sqsubseteq \mathsf{D}$,
  then $\quad \mathcal{O} \cup \mathcal{E}' \not\models \mathsf{C} \sqsubseteq \mathsf{D}$,



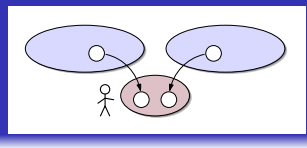
- Safety $\hat{=}$ preserving non-entailments

Comparison of different approaches



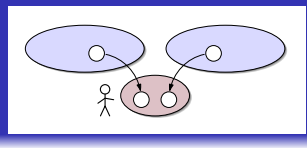
Kind of "module"	Covrg.	Min.	Covered DLs	Complexity
All ax's referencing $\mathcal{T}$	✘		any	easy
Seidenberg/Rector	✘		any	easy
Prompt	✘		?	easy

Comparison of different approaches



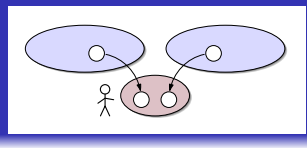
Kind of "module"	Covrg.	Min.	Covered DLs	Complexity
All ax's referencing $\mathcal{T}$	✘		any	easy
Seidenberg/Rector	✘		any	easy
Prompt	✘		?	easy
The whole ontology	✓	✘✘	any	easy
conserv.-based mod.	✓	✓	few	hard
MEX (Liverpool)	✓	✓	acyclic $\mathcal{EL}$	easy
locality-based mod.	✓	✘	$\approx$ OWL 2	easy
E-connections	✓	✘	OWL 1	easy

Comparison of different approaches



Kind of "module"	Covrg.	Min.	Covered DLs	Complexity
All ax's referencing $\mathcal{T}$	✘		any	easy
Seidenberg/Rector	✘		any	easy
Prompt	✘		?	easy
The whole ontology	✓	✘✘	any	easy
conserv.-based mod.	✓	✓	few	hard
MEX (Liverpool)	✓	✓	acyclic $\mathcal{EL}$	easy
locality-based mod.	✓	✘	$\approx$ OWL 2	easy
E-connections	✓	✘	OWL 1	easy
interpolants-based (no subsets!)	✓	✓✓	few	hard

Module extraction in Protégé 4



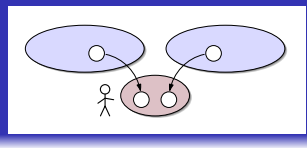
Nightly build:

http://owl.cs.manchester.ac.uk/2008/iswc-modtut/equinox.zip

- Realises import scenario
- Provides coverage via locality-based modules
- We're working on safety ...
- To be released as Protégé 4 plugin soon

(Thanks to Matthew Horridge.)

Web service for module extraction



http://owl.cs.manchester.ac.uk/modularity

The University of Manchester

MANCHESTER 1824

OWL Module Extractor

Ontology source

Paste your **ontology**, or enter a **URL** of a document, into the text box below.

http://www.co-ode.org/ontologies/pizza/pizza.owl

Signature

Enter a signature. Put each entity name on a new line. (Accepts full URIs or URI fragments)

Pizza

Modularity type

Select the module type

- ○ Top (lower) module
- ○ Bottom (upper) module
- ○ Bottom-of-top (upper-of-lower) module
- ◉ Top-of-bottom (lower-of-upper) module

☑ Show axioms view (instead of outputting RDF/XML)

Extract module

The University of Manchester

MANCHESTER 1824

Module: http://www.co-ode.org/ontologies/pizza/pizza.owl_module.owl

Selected signature

Pizza (http://www.co-ode.org/ontologies/pizza/pizza.owl#Pizza)

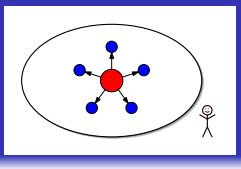
Module metrics

Number of axioms: 112
Number of logical axioms: 112
Number of classes: 35
Number of object properties: 7
Number of data properties: 0
Number of individuals: 5

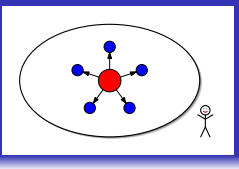
Module axioms

CheeseTopping SubClassOf PizzaTopping
CheeseTopping DisjointWith FishTopping
CheeseTopping DisjointWith FruitTopping
CheeseTopping DisjointWith HerbSpiceTopping
CheeseTopping DisjointWith MeatTopping
CheeseTopping DisjointWith NutTopping
CheeseTopping DisjointWith SauceTopping
CheeseTopping DisjointWith VegetableTopping
CheeseyPizza EquivalentTo Pizza **and** (hasTopping **some** CheeseTopping)
Country EquivalentTo DomainConcept **and** ({America , England , France , Germany , Italy})
DeepPanBase SubClassOf PizzaBase
DeepPanBase DisjointWith ThinAndCrispyBase
DomainConcept DisjointWith ValuePartition
FishTopping SubClassOf PizzaTopping
FishTopping SubClassOf hasSpiciness **some** Mild
FishTopping DisjointWith FruitTopping
FishTopping DisjointWith HerbSpiceTopping
FishTopping DisjointWith MeatTopping
FishTopping DisjointWith NutTopping
FishTopping DisjointWith SauceTopping
FishTopping DisjointWith VegetableTopping
Food SubClassOf DomainConcept
FruitTopping SubClassOf PizzaTopping
FruitTopping DisjointWith HerbSpiceTopping
FruitTopping DisjointWith MeatTopping
FruitTopping DisjointWith NutTopping
FruitTopping DisjointWith SauceTopping
FruitTopping DisjointWith VegetableTopping
HerbSpiceTopping SubClassOf PizzaTopping

And now ...

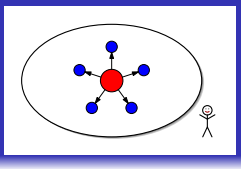


We bet Robert Stevens . . .

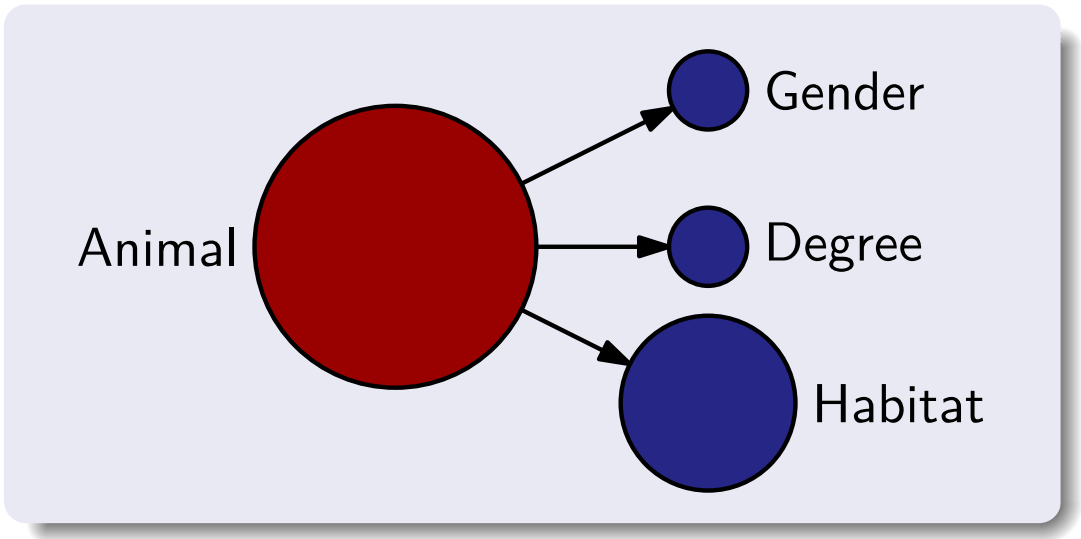


- Ontology about periodic table of the chemical elements
- What is its modular structure?
- What is “the meat” of it?
- We can find it using locality-based modules.

Impetus for the "Meat" idea



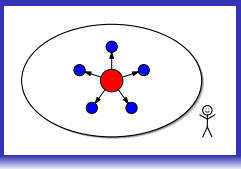
Partition of `koala.owl` via E-connections in Swoop

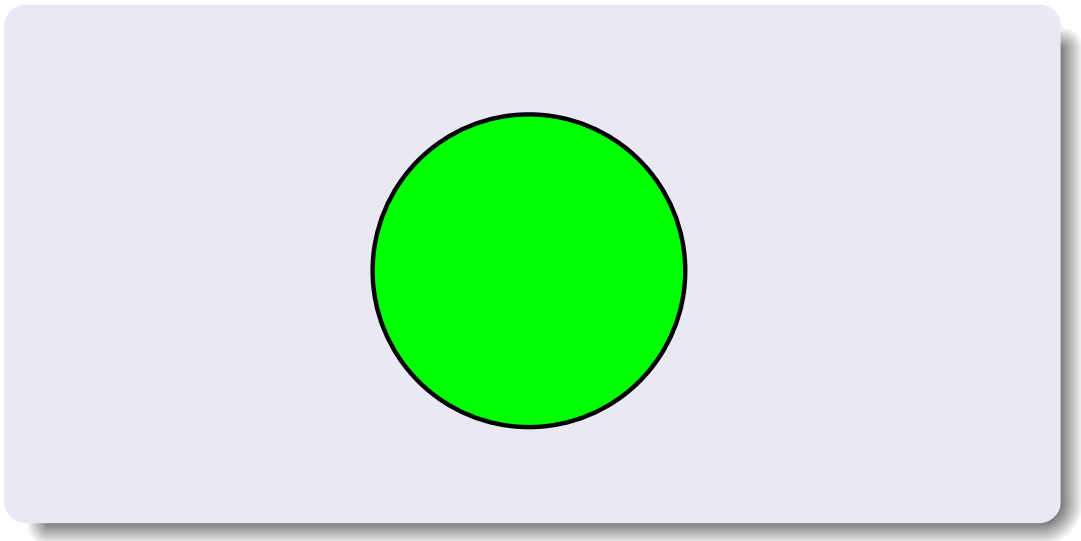


- importing part
- imported but non-importing part
- isolated part

→ "imports vocabulary from"

Partition for the periodic table ontology

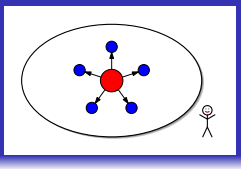




- importing part
- imported but non-importing part
- isolated part

→ "imports vocabulary from"

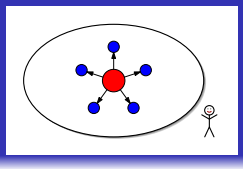
"Meat" via locality-based modules



Hopes:

- Finer-grained analysis
- Guidance for users to choose the right topic(s)
  (module signature $\neq \mathcal{T}$)
- Draw conclusions on characteristics of an ontology:
  topicality, connectedness, axiomatic richness, superfluous parts, modelling

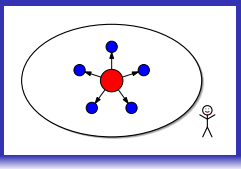
"Meat" via locality-based modules



Problem:

- Ontologies of size n can have up to 2^n modules
- But do real-life ontologies fall into the worst case?

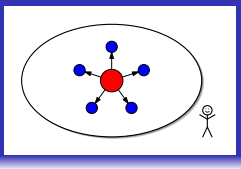
Results so far



- Highly optimised algorithm to extract all modules

Ontology	#Ax	#Terms	**#mods**	Theor. Max.	time
Koala	42	25	**3660**	33 554 432	9s
Mereology	44	25	**1952**	33 554 432	3min

Results so far

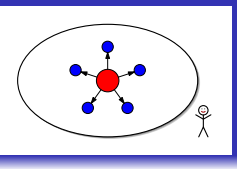


- Highly optimised algorithm to extract all modules

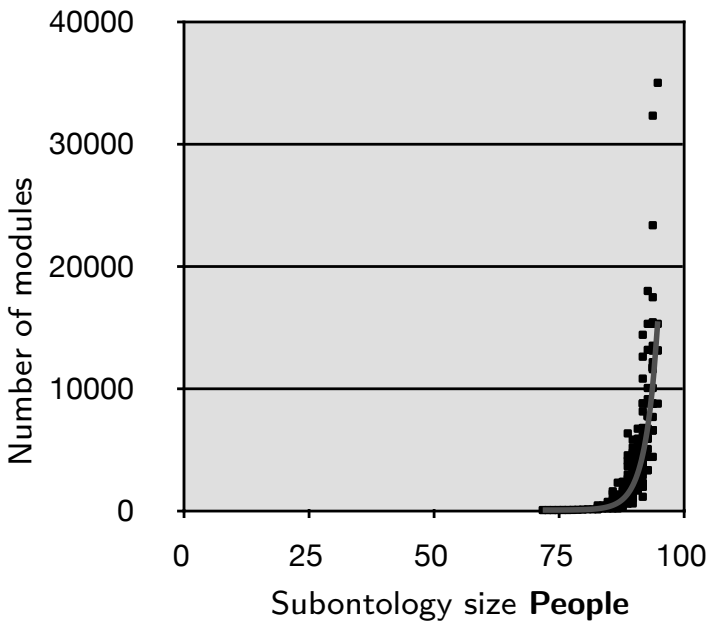
Ontology	#Ax	#Terms	**#mods**	Theor. Max.	time
Koala	42	25	**3660**	33 554 432	9s
Mereology	44	25	**1952**	33 554 432	3min

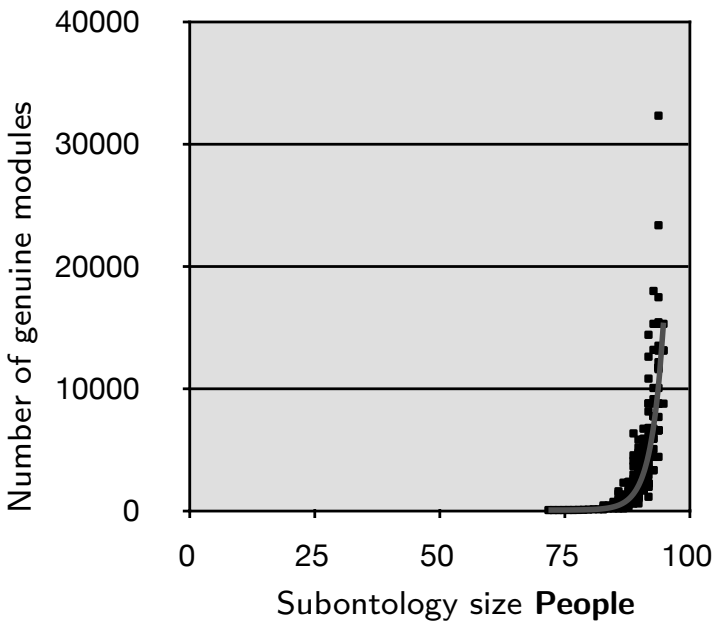
- Not scalable
- Single module numbers don't say much

Subset sampling



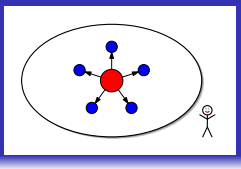
- For 8 ontologies, we modularised randomly generated subontologies
- Mostly "negative" results





Trendline equation: $y = O(1.5^x)$, confidence 0.96

Weight analysis



- Ordered all 3660 modules of Koala by weight

$$\text{Weight}(\mathcal{M}) = \text{PullingPower}(\mathcal{M}) \cdot \text{Cohesion}(\mathcal{M})$$

$$\text{PullingPower}(\mathcal{M}) = \frac{\#\text{terms in } \mathcal{M}}{|\text{smallest seed signature for } \mathcal{M}|}$$

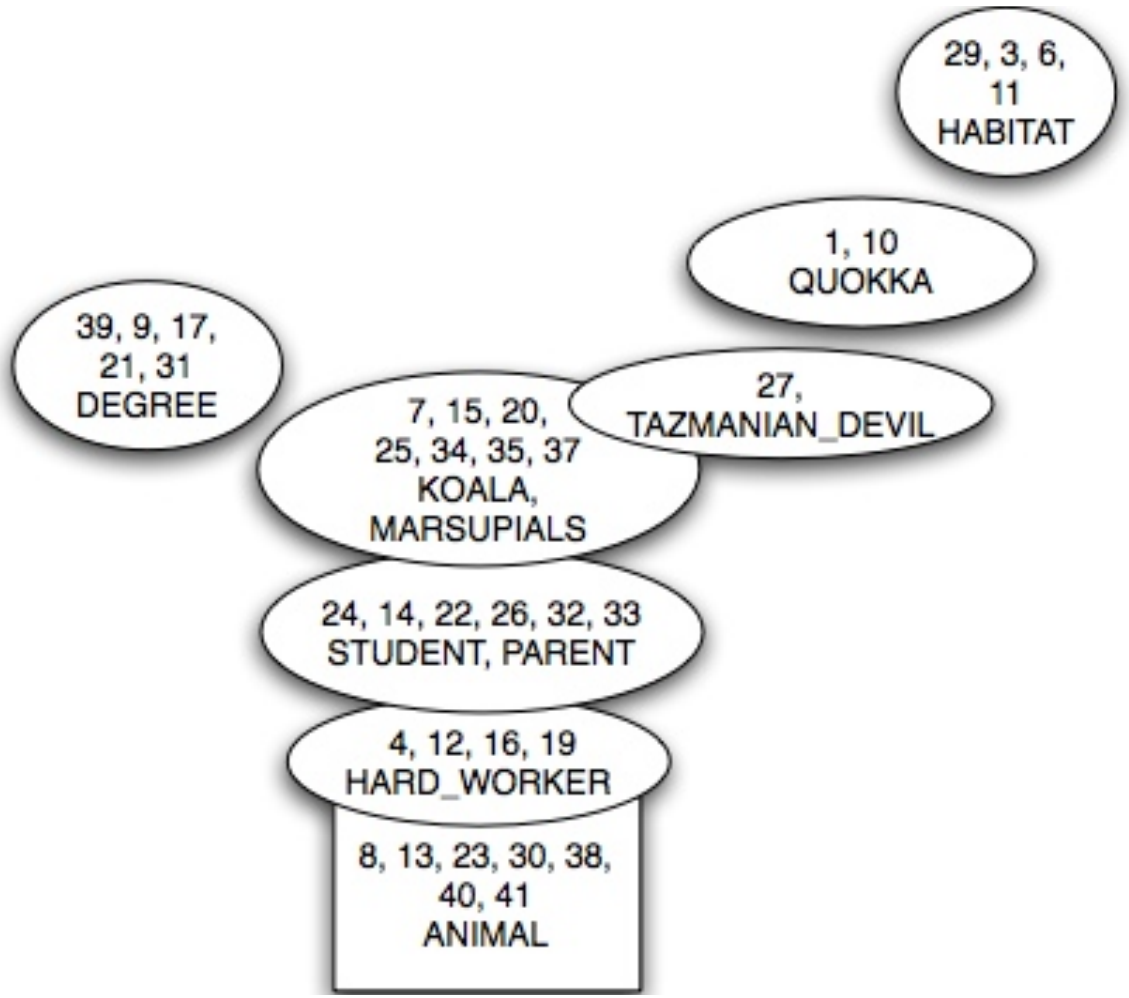
How many terms are needed to "pull" all the terms into $\mathcal{M}$?

$$\text{Cohesion}(\mathcal{M}) = \frac{\#\text{minimal seed signatures of } \mathcal{M}}{|\text{smallest seed signature for } \mathcal{M}|}$$

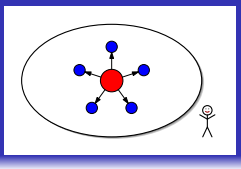
How strongly are terms in $\mathcal{M}$ held together?

- Inspected heaviest modules

Weight analysis

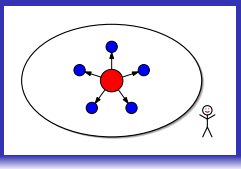


Outlook



- Find heaviest modules without computing all modules
- Relation between module number and justificatory structure of an ontology

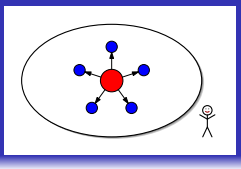
Outlook



- Find heaviest modules without computing all modules
- Relation between module number and justificatory structure of an ontology

- Collaborative ontology development using modules
- Modules that are no subsets
- Modularity for belief revision

Outlook



- Find heaviest modules without computing all modules
- Relation between module number and justificatory structure of an ontology

- Collaborative ontology development using modules
- Modules that are no subsets
- Modularity for belief revision

Thank you.