Working Modularly with OWL

Thomas Schneider

School of Computer Science, University of Manchester, UK

27 January 2009

About the project

Title

Composing and decomposing ontologies: a logic-based approach

People involved/interested

- Uli Sattler, Bijan Parsia, Thomas Schneider (Manchester)
- Frank Wolter, Boris Konev, Dirk Walther (Liverpool)
- Ian Horrocks, Bernardo Cuenca Grau (Oxford)
- Carsten Lutz (Bremen)

And now . . .

1. Why modularity?
2. A reuse scenario
3. Logical background
4. Tools
5. Comparison of modularisation approaches
6. Understanding ontologies via modules

Crash course: ontologies and description logics

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 + inference

- $\underbrace{\text{Duck}}_{\text{class}} \sqsubseteq \underbrace{\exists \underbrace{\text{feedsOn}}_{\text{property}} . \underbrace{\text{Grass}}_{\text{class}}}_{\text{class}}$

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

Example axioms + inference

- $\underbrace{\text{Duck}}_{\text{class}} \sqsubseteq \ \exists \ \underbrace{\underbrace{\text{feedsOn}}_{\text{property}} . \underbrace{\text{Grass}}_{\text{class}}}_{\text{class}}$

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

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

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

Example axioms + inference

- $\underbrace{\mathsf{Duck}}_{\text{class}} \sqsubseteq \underbrace{\exists\ \underbrace{\mathsf{feedsOn}}_{\text{property}} . \underbrace{\mathsf{Grass}}_{\text{class}}}_{\text{class}}$

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

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

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

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

Example axioms + inference

- $\underbrace{\text{Duck}}_{\text{class}} \sqsubseteq \exists \underbrace{\underbrace{\text{feedsOn}}_{\text{property}} . \underbrace{\text{Grass}}_{\text{class}}}_{\text{class}}$

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

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

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

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

- $\text{Tweety} : \text{Bird}$
- $\text{Tweety} : \exists \text{feedsOn}.\text{Grass}$

Reasoning tasks

- *Inference:* Does axiom α follow from ontology $\mathcal{O}$?
- *Satisfiability:*
  Is there a model of $\mathcal{O}$ that interprets class C as nonempty?
- *Instance checking:*
  Is individual x an instance of C in every model of $\mathcal{O}$?

Inter-reducible; optimised reasoners available

A case for modularity

Common practice in software engineering

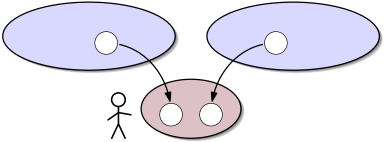
Modular software development allows for:

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

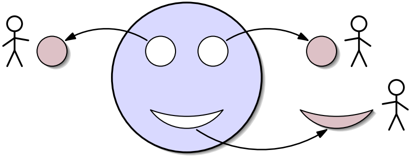
Wouldn't it be nice . . .

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

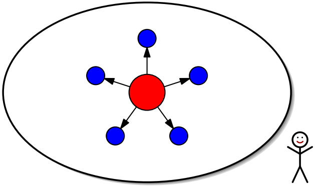
Three scenarios



Import/reuse

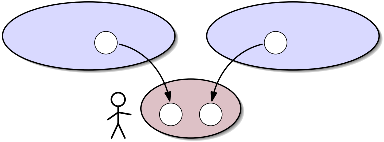


Collaboration

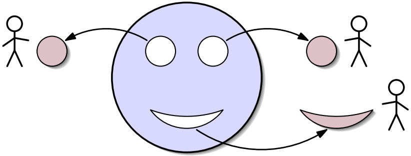


Understanding

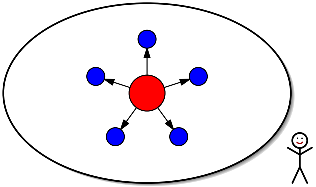
Three scenarios



Import/reuse



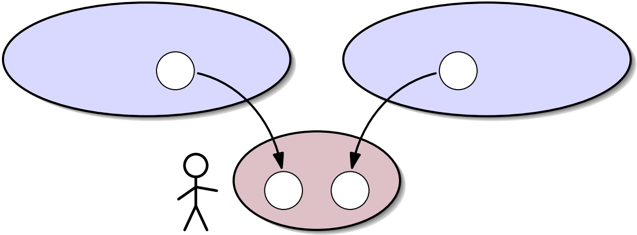
Collaboration



Understanding

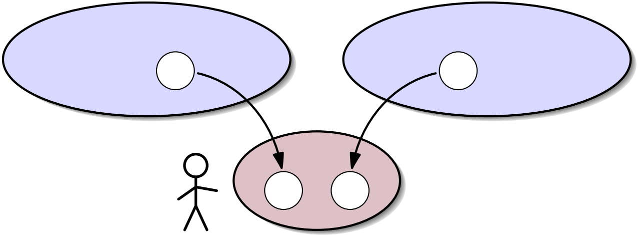
Scenario 1: Import/reuse

“Borrow” knowledge about certain terms from external ontologies



Scenario 1: Import/reuse

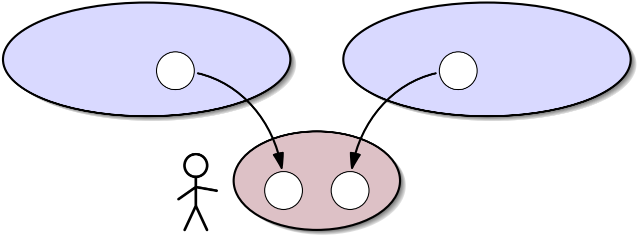
“Borrow” knowledge about certain terms from external ontologies



- 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

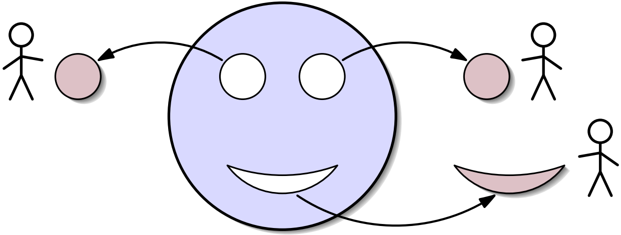


- Provides access to well-established knowledge
- Doesn’t require expertise in external disciplines

This scenario is well-understood and implemented.

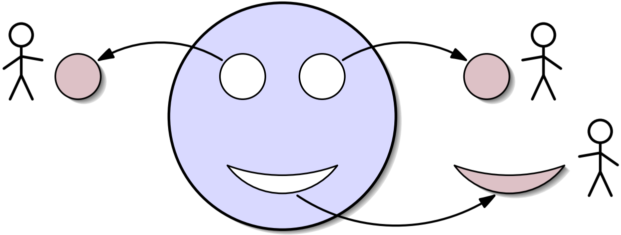
Scenario 2: Collaboration

Collective ontology development



Scenario 2: Collaboration

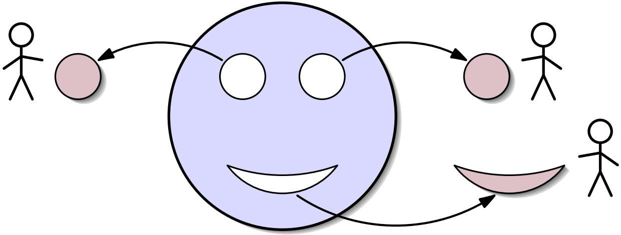
Collective ontology development



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

Scenario 2: Collaboration

Collective ontology development

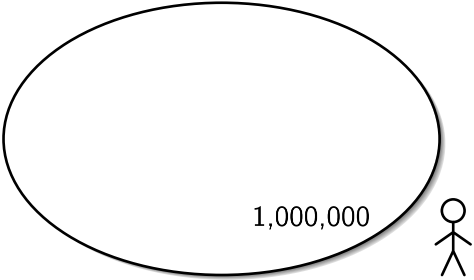


- Developers work (edit, classify) locally
- Extra care at re-combination

This approach is understood, but not implemented yet.

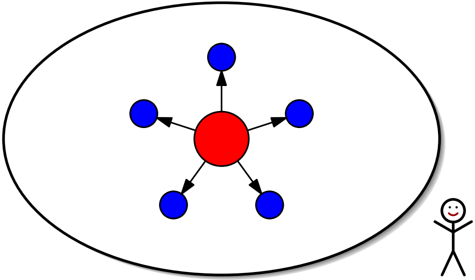
Scenario 3: Understanding

Visualise the modular structure of an ontology



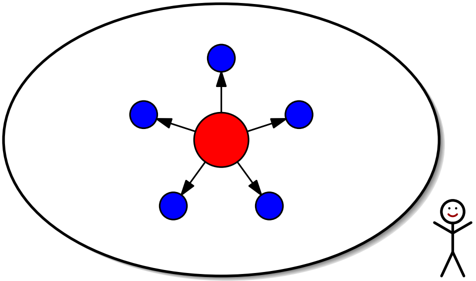
Scenario 3: Understanding

Visualise the modular structure of an ontology



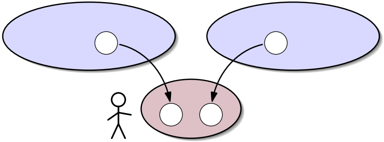
Scenario 3: Understanding

Visualise the modular structure of an ontology

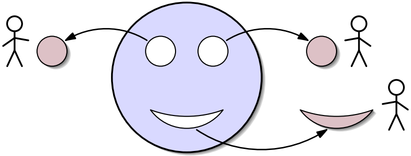


We're still playing with this.

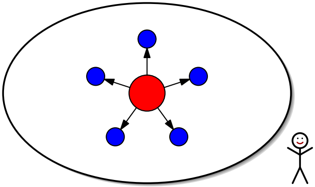
Summing up



Import/reuse

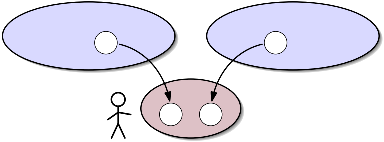


Collaboration

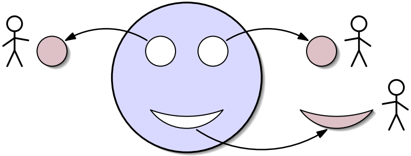


Understanding

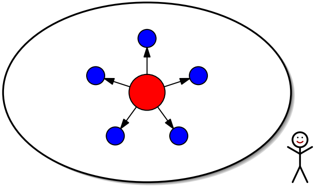
Summing up



Import/reuse



Collaboration

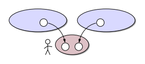


Understanding

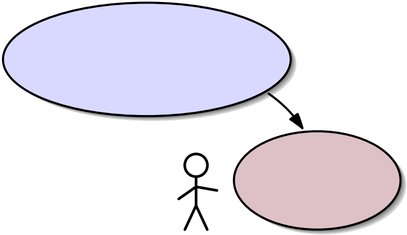
And now . . .

1. Why modularity?
2. A reuse scenario
3. Logical background
4. Tools
5. Comparison of modularisation approaches
6. Understanding ontologies via modules

A reuse scenario

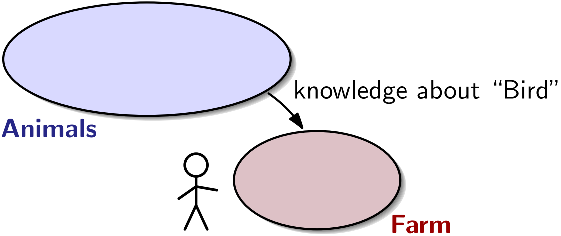


Import/reuse one external ontology



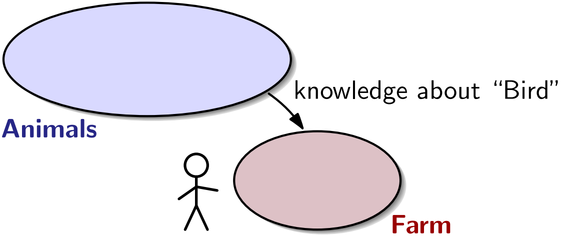
A reuse scenario

Import/reuse one external ontology



A reuse scenario

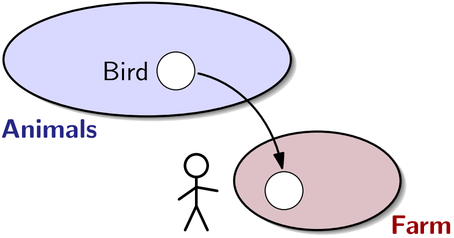
Import/reuse one external ontology



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

A reuse scenario

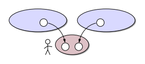
Import/reuse a part of an external ontology



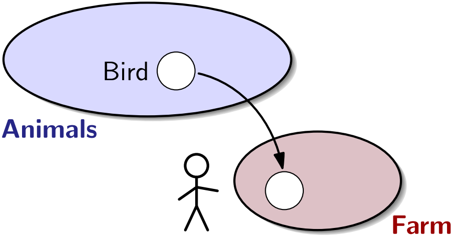
How much of **Animals** do we need?

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

A reuse scenario



Import/reuse a part of an external ontology



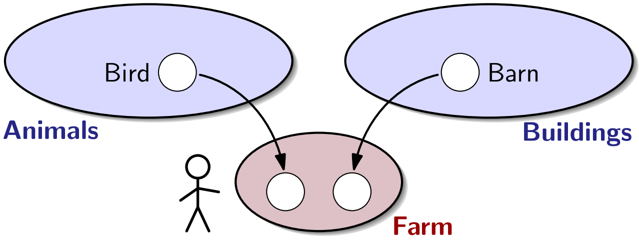
How much of **Animals** do we need?

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

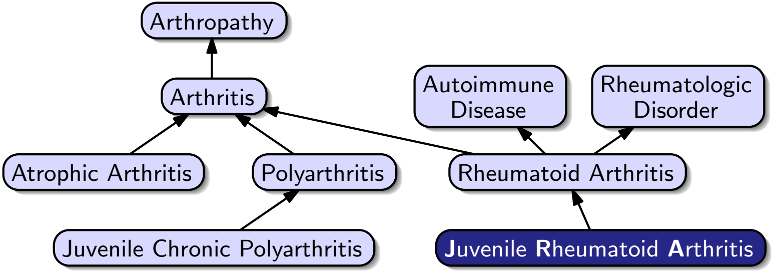
How to achieve coverage and economy?

A reuse scenario

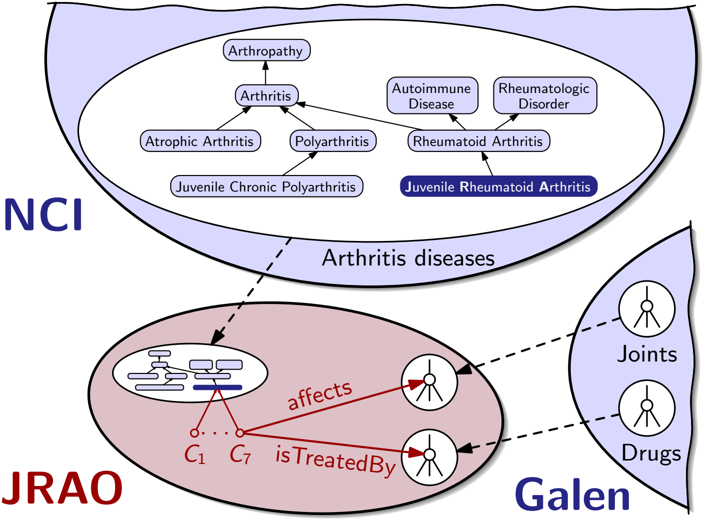
Import/reuse parts of several external ontologies



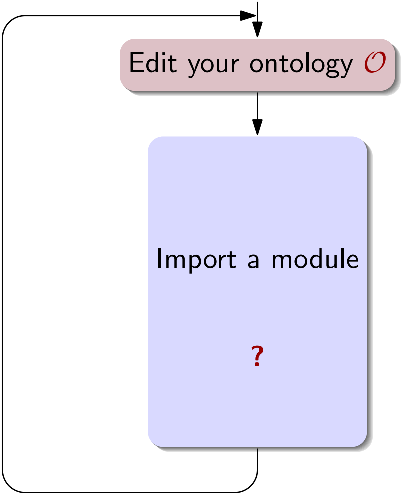
The *Health-e-Child* project



The *Health-e-Child* project



A working cycle

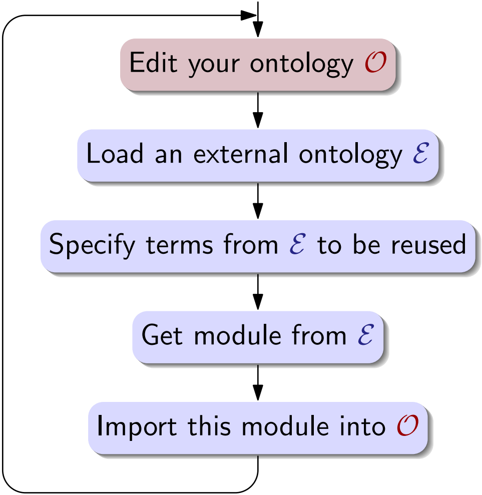


Edit your ontology $\mathcal{O}$

Import a module

?

A working cycle



Edit your ontology $\mathcal{O}$

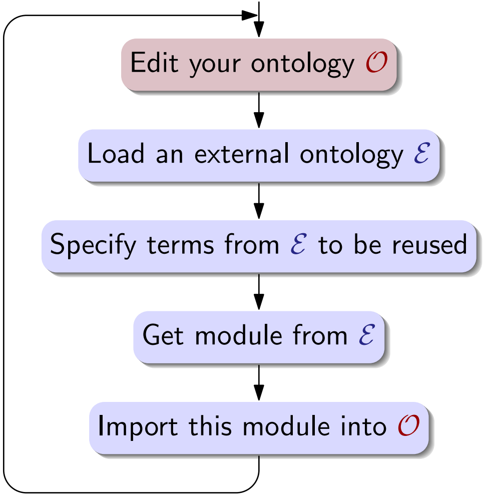
Load an external ontology $\mathcal{E}$

Specify terms from $\mathcal{E}$ to be reused

Get module from $\mathcal{E}$

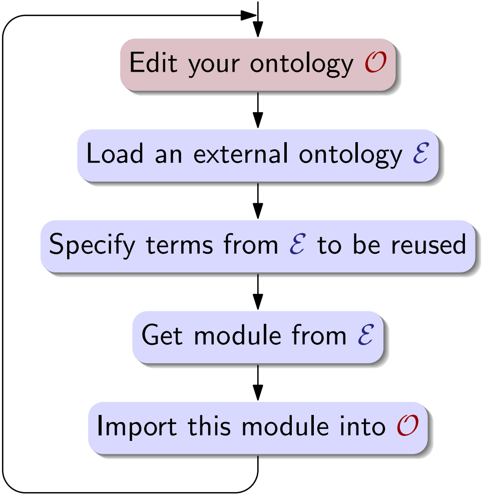
Import this module into $\mathcal{O}$

A working cycle



Edit your ontology $\mathcal{O}$	**Farm**
Load an external ontology $\mathcal{E}$	**Animals**
Specify terms from $\mathcal{E}$ to be reused	Animal, feedsOn
Get module from $\mathcal{E}$	**Animals′**
Import this module into $\mathcal{O}$	**Farm** ∪ **Animals′**

A working cycle


Edit your ontology $\mathcal{O}$
Load an external ontology $\mathcal{E}$
Specify terms from $\mathcal{E}$ to be reused
Get module from $\mathcal{E}$
Import this module into $\mathcal{O}$

Farm ∪ **Animals′**

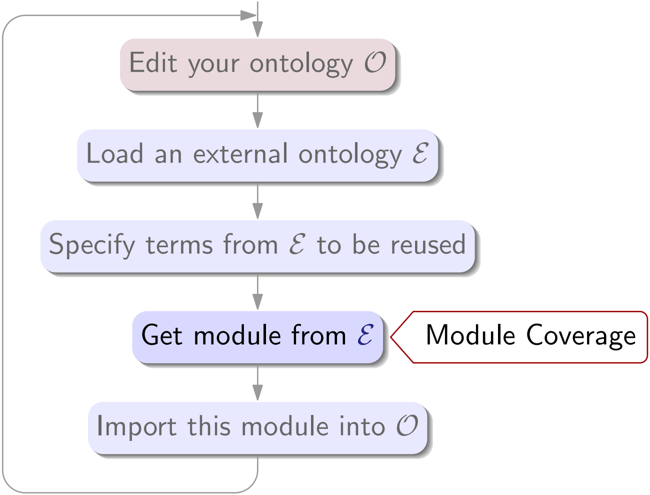
Buildings

DuckHousing, Silo

Buildings′

Farm ∪ **Animals′** ∪ **Buildings′**

A working cycle



Edit your ontology $\mathcal{O}$

Load an external ontology $\mathcal{E}$

Specify terms from $\mathcal{E}$ to be reused

Get module from $\mathcal{E}$

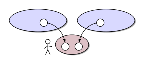
Module Coverage

Import this module into $\mathcal{O}$

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:

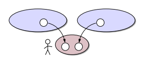
- Topic: Fox, Bird, feedsOn
- On-topic: Off-topic:

$$\mathsf{Fox} \sqsubseteq \forall\,\mathsf{feedsOn}.\mathsf{Bird}$$
$$\mathsf{Fox} \sqcup \mathsf{Bird} \sqsubseteq \exists\,\mathsf{feedsOn}.\top$$
$$\mathsf{Bird} \sqsubseteq \neg\mathsf{Fox}$$
$$\mathsf{Bird} \sqsubseteq \mathsf{Bird} \sqcup \mathsf{Fox}$$

$$\mathsf{Duck} \sqsubseteq \mathsf{Bird}$$

- Goal = preserve all on-topic knowledge

Module coverage



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

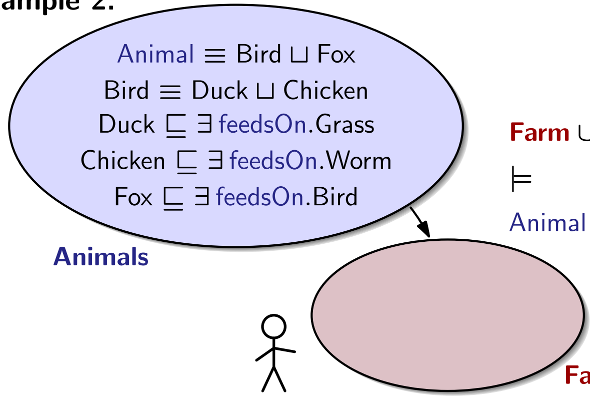
Question: Which axioms do we need to import?

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:



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

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

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

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

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

Farm ∪ **Animals**

$\models$

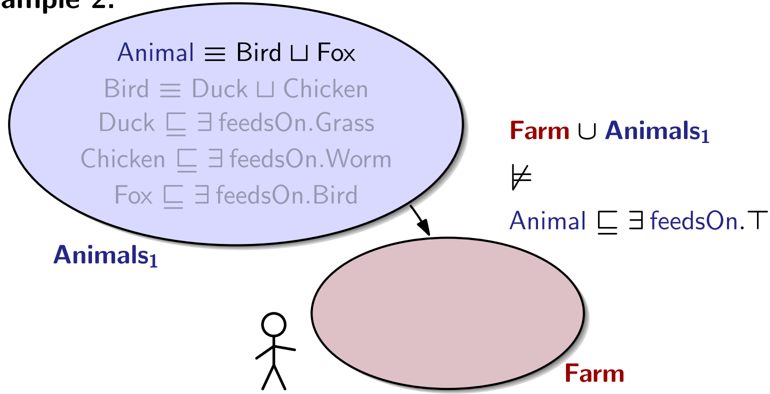
$\text{Animal} \sqsubseteq \exists\, \text{feedsOn}.\top$

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:



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

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

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

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

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

$\textbf{Farm} \cup \textbf{Animals}_1$

$\not\models$

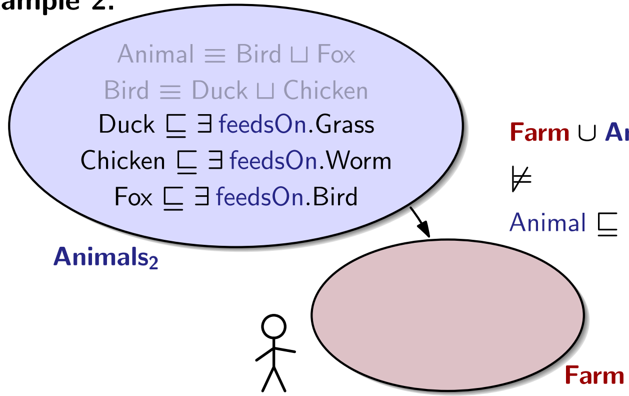
$\text{Animal} \sqsubseteq \exists\,\text{feedsOn}.\top$

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:



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

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

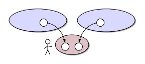
$\mathsf{Duck} \sqsubseteq \exists\, \mathsf{feedsOn}.\mathsf{Grass}$

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

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

$$\mathbf{Farm} \cup \mathbf{Animals_2}$$
$$\not\models$$
$$\mathsf{Animal} \sqsubseteq \exists\, \mathsf{feedsOn}.\top$$

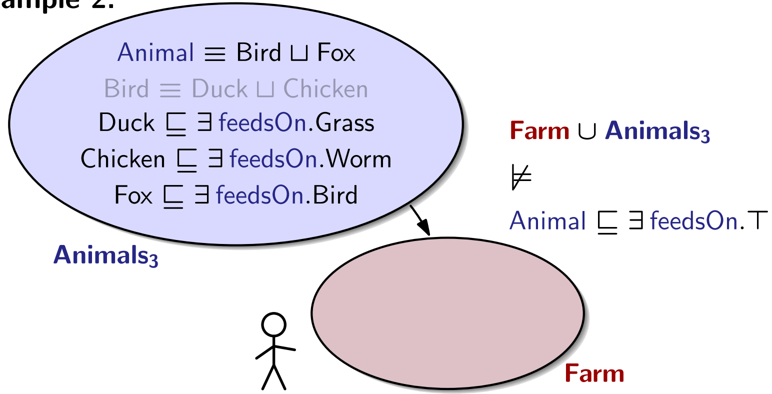
Module coverage



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:



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

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

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

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

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

Animals$_3$

Farm $\cup$ **Animals$_3$**

$\not\models$

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

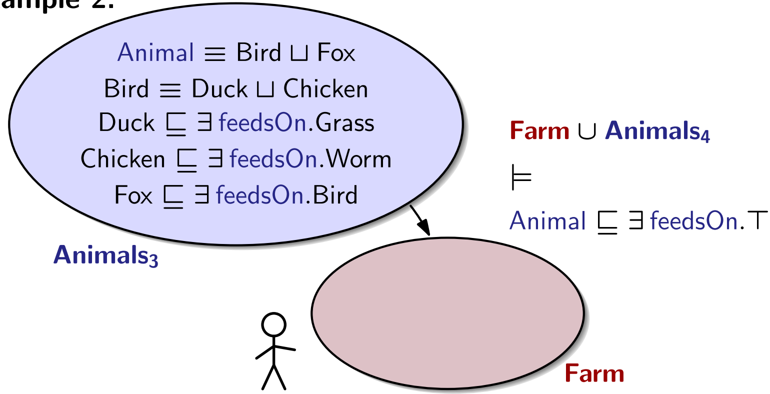
Farm

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:



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

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

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

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

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

Animals$_3$

Farm $\cup$ **Animals$_4$**

$\models$

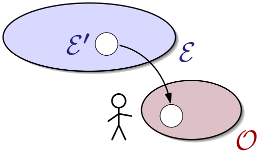
$\mathsf{Animal} \sqsubseteq \exists\, \mathsf{feedsOn}.\top$

Farm

Module coverage

- The module $\mathcal{E}'$ covers the ontology $\mathcal{E}$ for the specified topic $\mathcal{T}$ if for all classes A, B built from terms in $\mathcal{T}$:

  $$\begin{array}{lll} \text{if} & \mathcal{O} \cup \mathcal{E} & \models \mathsf{A} \sqsubseteq \mathsf{B}, \\ \text{then} & \mathcal{O} \cup \mathcal{E}' & \models \mathsf{A} \sqsubseteq \mathsf{B}. \end{array}$$

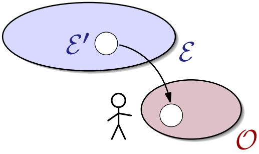


- Coverage $\hat{=}$ preserving entailments

Module coverage

- The module $\mathcal{E}'$ covers the ontology $\mathcal{E}$ for the specified topic $\mathcal{T}$ if for all classes A, B built from terms in $\mathcal{T}$:

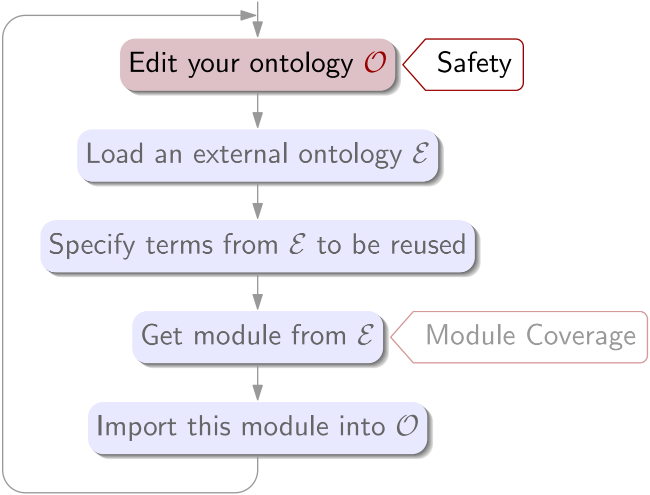
  if $\quad \mathcal{O} \cup \mathcal{E} \models A \sqsubseteq B$,
  then $\quad \mathcal{O} \cup \mathcal{E}' \models A \sqsubseteq B$.



- Coverage $\hat{=}$ preserving entailments

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

A working cycle



Edit your ontology $\mathcal{O}$

Safety

Load an external ontology $\mathcal{E}$

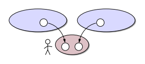
Specify terms from $\mathcal{E}$ to be reused

Get module from $\mathcal{E}$

Module Coverage

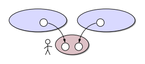
Import this module into $\mathcal{O}$

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?

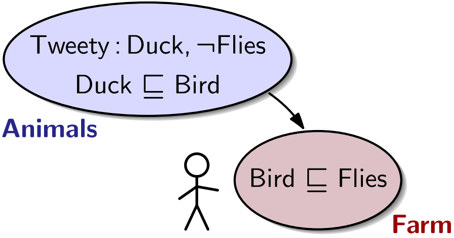
Safety



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:

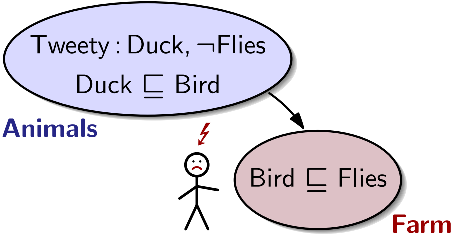


Safety

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:

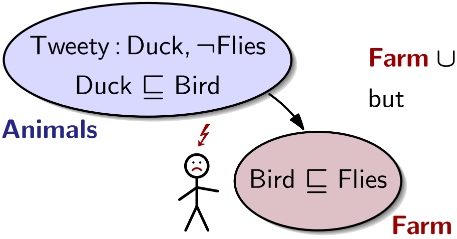


Safety

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?

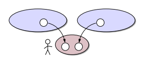
Example:



Farm ∪ **Animals** $\models$ Bird ⊑ Flies

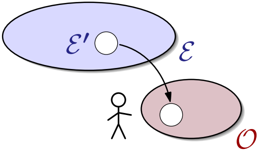
but **Animals** $\not\models$ Bird ⊑ Flies

Safety



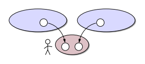
- *Our* ontology $\mathcal{O}$ uses the imported terms safely if for all classes A, B built from the imported terms:

$$\begin{array}{lrcl} \text{If} & \mathcal{E}' & \not\models & \mathsf{A} \sqsubseteq \mathsf{B}, \\ \text{then} & \mathcal{O} \cup \mathcal{E}' & \not\models & \mathsf{A} \sqsubseteq \mathsf{B}, \end{array}$$



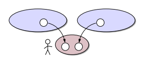
- Safety $\hat{=}$ preserving non-entailments

And now . . .



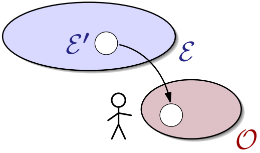
1. Why modularity?
2. A reuse scenario
3. Logical background
4. Tools
5. Comparison of modularisation approaches
6. Understanding ontologies via modules

Module coverage



- The module $\mathcal{E}'$ covers the ontology $\mathcal{E}$ for the specified topic $\mathcal{T}$ if for all classes A, B built from terms in $\mathcal{T}$:

$$\begin{array}{lccc} \text{if} & \mathcal{O} \cup \mathcal{E} & \models & A \sqsubseteq B, \\ \text{then} & \mathcal{O} \cup \mathcal{E}' & \models & A \sqsubseteq B. \end{array}$$

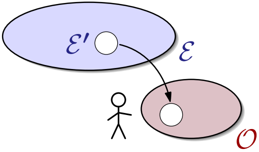


- Coverage $\hat{=}$ preserving entailments

Module coverage

- The module $\mathcal{E}'$ covers the ontology $\mathcal{E}$ for the specified topic $\mathcal{T}$ if for all classes A, B built from terms in $\mathcal{T}$:

$$\begin{array}{lllll} \text{if} & \mathcal{O} \cup \mathcal{E} & \models & \mathsf{A} \sqsubseteq \mathsf{B}, \\ \text{then} & \mathcal{O} \cup \mathcal{E}' & \models & \mathsf{A} \sqsubseteq \mathsf{B}. \end{array}$$



- Coverage $\hat{=}$ preserving entailments

- $\mathcal{O}$ may allow "more" interpretations of imported terms than $\mathcal{E}$.
- If so, include more "restricting" axioms into $\mathcal{E}'$.
- Finish when all terms $\notin \mathcal{E}'$ can be interpreted as $\bot$ or $\top$.
- *Locality* says whether this is possible.

Notions of covering modules

- Minimal coverage-providing modules

  based on conservative extensions
  hard to compute (intractable/undecidable)

- Locality-based modules

  based on the above considerations
  not minimal, hard to compute

- Modules based on *syntactic locality*

  not minimal, *easy* to compute (tractable)

Notions of covering modules

- Minimal coverage-providing modules

  based on conservative extensions
  hard to compute (intractable/undecidable)

- Locality-based modules

  based on the above considerations
  not minimal, hard to compute

- Modules based on *syntactic locality*

  not minimal, *easy* to compute (tractable)

- Computation:

```
$\mathcal{T} \leftarrow$ topic;   $M \leftarrow \emptyset$
While there is non-local axiom $\alpha$ w.r.t. $\underbrace{\mathcal{T} \cup \mathrm{sig}(M)}_{\textit{extended topic}}$ do:
    $M \leftarrow M \cup \{\alpha\}$
```

Notions of covering modules

- Minimal coverage-providing modules

  based on conservative extensions
  hard to compute (intractable/undecidable)

- Locality-based modules

  based on the above considerations
  not minimal, hard to compute

- Modules based on *syntactic locality*

  not minimal, *easy* to compute (tractable)

- Computation:

```
$\mathcal{T} \leftarrow$ topic;   $M \leftarrow \emptyset$
While there is non-local axiom $\alpha$ w.r.t. $\underbrace{\mathcal{T} \cup \mathrm{sig}(M)}_{\textit{extended topic}}$ do:
    $M \leftarrow M \cup \{\alpha\}$
```

- We often extract the $\top$-module of the $\bot$-module of $\mathcal{E}$.

And now . . .

1. Why modularity?
2. A reuse scenario
3. Logical background
4. **Tools**
5. Comparison of modularisation approaches
6. Understanding ontologies via modules

Module extraction in Protégé 4

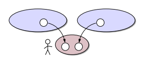
Nightly build:

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

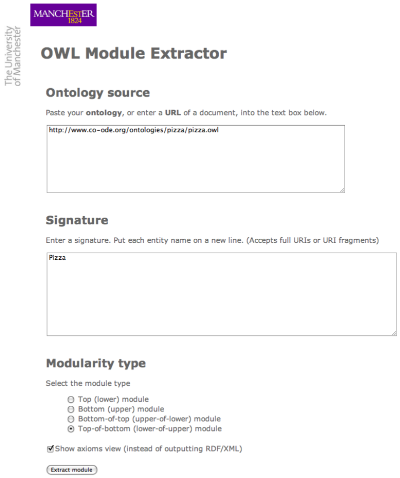
- Realises import scenario
- Provides coverage via locality-based modules
- Will soon provide safety too . . .
- To be released as Protégé 4 plugin in the near future

(Thanks to Matthew Horridge.)

Web-based module extraction



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



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)

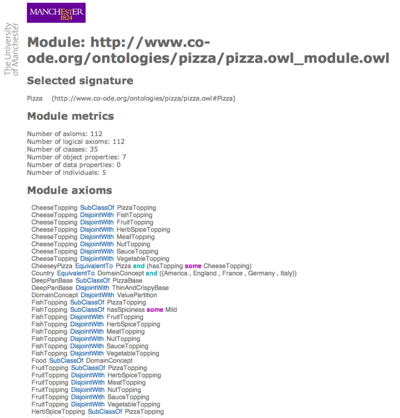
Pizza

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)



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

Web-based module extraction

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

Try it! ☺

- Ontology: http://www.co-ode.org/ontologies/pizza/pizza.owl
- Signature “Pizza”, “VegetarianPizza”, or “Country”
- Select “Show axioms view”

(Thanks to Matthew Horridge.)

This tool currently ignores non-logical axioms (annotations etc.).

And now . . .

1. Why modularity?
2. A reuse scenario
3. Logical background
4. Tools
5. Comparison of modularisation approaches
6. Understanding ontologies via modules

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
MEX (Liverpool)	✓	✓	acyclic $\mathcal{EL}$	easy
conserv.-based mod.	✓	✓	few	hard
locality-based mod.	✓	✗	$\approx$ OWL 1 DL	easy
E-connections	✓	✗	OWL 1 DL	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
MEX (Liverpool)	✓	✓	acyclic $\mathcal{EL}$	easy
conserv.-based mod.	✓	✓	few	hard
locality-based mod.	✓	✗	$\approx$ OWL 1 DL	easy
E-connections	✓	✗	OWL 1 DL	easy
interpolants-based (no subsets!)	✓	✓✓	few	hard

And now . . .

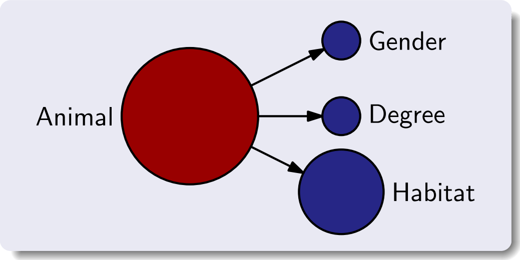
1. Why modularity?
2. A reuse scenario
3. Logical background
4. Tools
5. Comparison of modularisation approaches
6. Understanding ontologies via modules

We bet Robert . . .

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

Impetus for the “Meat” idea

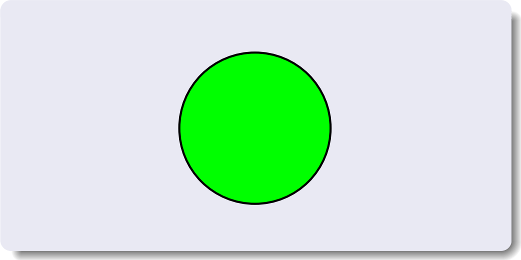
Partition of koala.owl via E-connections in Swoop



- 🔴 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

- Hope: finer-grained analysis
- Difficulties: Computation harder, interpretation unclear

"Meat" via locality-based modules

- Hope: finer-grained analysis
- Difficulties: Computation harder, interpretation unclear

- Results so far
  - 416 modules for all ≈ 800 singleton topics
  - Sizes $0, \ldots, 2800$; average 1600 ($\approx 4\,\%$)
  - Found small modelling irregularity

“Meat” via locality-based modules

- Hope: finer-grained analysis
- Difficulties: Computation harder, interpretation unclear

- Results so far
  - 416 modules for all $\approx$ 800 singleton topics
  - Sizes 0, . . . , 2800; average 1600 ($\approx$ 4 %)
  - Found small modelling irregularity

- Struggle with visualisation
- Blowup-free methodology for bigger modules?
- What does the collection of *all* modules tell us?
- Modules for topics of axioms?