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 ...

1. Ontologies and Description Logic
2. Why modularity?
3. A reuse scenario
4. Understanding ontologies via modules
Ontology

= collection of statements about a domain \((\text{axioms})\)

- Language used: usually logic, often \(\textit{description logic (DL)}\)
- \(\textit{Inferences}\) can be drawn from axioms

Domains:
- biology, medicine, chemistry, business processes, natural language, \ldots
Example axioms + inferences

\[ \text{Duck} \sqsubseteq \exists \text{feedsOn} . \text{Grass} \]

\[ \forall x \left( \text{Duck}(x) \rightarrow \exists y \left( \text{feedsOn}(x, y) \land \text{Grass}(y) \right) \right) \]
Example axioms + inferences

- $\text{Duck} \sqsubseteq \exists \text{feedsOn} . \text{Grass}$
  
  $$\forall x \left( \text{Duck}(x) \rightarrow \exists y \left( \text{feedsOn}(x, y) \land \text{Grass}(y) \right) \right)$$

- $\text{Bird} \equiv \text{Duck} \sqcup \text{Chicken}$
  
  $$\forall x \left( \text{Bird}(x) \leftrightarrow (\text{Duck}(x) \lor \text{Chicken}(x)) \right)$$
Example axioms + inferences

- Duck \sqsubseteq \exists \text{feedsOn}. \text{Grass}

  \forall x \left( \text{Duck}(x) \rightarrow \exists y \left( \text{feedsOn}(x, y) \land \text{Grass}(y) \right) \right)

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

  \forall x \left( \text{Bird}(x) \leftrightarrow \left( \text{Duck}(x) \lor \text{Chicken}(x) \right) \right)

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

  \forall x \left( \left( \text{Bird}(x) \land \neg \text{Chicken}(x) \right) \rightarrow \exists y \left( \text{feedsOn}(x, y) \land \text{Grass}(y) \right) \right)
Example axioms + inferences

- **Duck** ⊑ ∃ feedsOn . **Grass**

\[ \forall x \left( \text{Duck}(x) \rightarrow \exists y \left( \text{feedsOn}(x, y) \land \text{Grass}(y) \right) \right) \]

- **Bird** ≡ **Duck** ⊔ **Chicken**

\[ \forall x \left( \text{Bird}(x) \leftrightarrow \left( \text{Duck}(x) \lor \text{Chicken}(x) \right) \right) \]
Example axioms + inferences

- **Duck** $\sqsubseteq \exists \text{feedsOn}. \text{Grass}

- **Bird** $\equiv$ Duck $\sqcup$ Chicken

- **Tweety** : Duck

\[
\forall x \left( \text{Duck}(x) \rightarrow \exists y \left( \text{feedsOn}(x, y) \land \text{Grass}(y) \right) \right)
\[
\forall x \left( \text{Bird}(x) \leftrightarrow \left( \text{Duck}(x) \lor \text{Chicken}(x) \right) \right)
\]

Duck(Tweety)
Example axioms + inferences

- **Duck** ⊑ ∃ feedsOn . **Grass**

\[
\forall x (\text{Duck}(x) \rightarrow \exists y (\text{feedsOn}(x, y) \land \text{Grass}(y)))
\]

- **Bird** ≡ Duck ⊔ Chicken

\[
\forall x (\text{Bird}(x) \leftrightarrow (\text{Duck}(x) \lor \text{Chicken}(x)))
\]

- **Tweety** : Duck

Duck(Tweety)

\[
\models \text{Tweety} : \exists \text{feedsOn. Grass}
\]

\[
\exists y (\text{feedsOn}(\text{Tweety}, y) \land \text{Grass}(y))
\]
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
The Web Ontology Language OWL

- W3C-recommended standard since 2004

**OWL Full**
Consistency?, Reasoning

**OWL DL**
Based on DL *SROIQ*
∃, ∀, counting, role chains and hierarchies, transitivity, inverse roles, nominals

**OWL EL, QL, RL**
Sub-profiles for efficient reasoning and application orientation
And now ...

1. Ontologies and Description Logic

2. Why modularity?

3. A reuse scenario

4. Understanding ontologies via modules
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

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

1,000,000
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. Ontologies and Description Logic
2. Why modularity?
3. A reuse scenario
4. Understanding ontologies via modules
A reuse scenario

Import/reuse one external ontology
A reuse scenario

Import/reuse one external ontology

Animals

knowledge about “Bird”

Farm
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

- Bird
- Barn
- Animals
- Buildings
- Farm
The Health-e-Child project

[Diagram showing the classification of arthritis types, including:
- Arthropathy
- Arthritis
- Atrophic Arthritis
- Polyarthritis
- Autoimmune Disease
- Rheumatologic Disorder
- Rheumatoid Arthritis
- Juvenile Chronic Polyarthritis
- Juvenile Rheumatoid Arthritis]
The *Health-e-Child* project

Arthropathy

Arthritis

Autoimmune Disease

Rheumatologic Disorder

Atrophic Arthritis

Polyarthitis

Rheumatoid Arthritis

Juvenile Chronic Polyarthitis

**Juvenile Rheumatoid Arthritis**

**Joints**

**Drugs**

**Galen**

**NCI**

**Arthritis diseases**

**C₁**, **C₇**, *affects *, *isTreatedBy*
A working cycle

1. Edit your ontology
2. Import a module
A working cycle

1. Edit your ontology $O$
2. Load an external ontology $\mathcal{E}$
3. $\mathcal{T} \leftarrow$ Specify terms from $\mathcal{E}$
4. $\mathcal{M} \leftarrow \text{mod}(\mathcal{T}, \mathcal{E})$
5. $O \leftarrow O \cup \mathcal{M}$
A working cycle

- Edit your ontology $O$
- Load an external ontology $E$
- $\mathcal{T} \leftarrow$ Specify terms from $E$
- $\mathcal{M} \leftarrow$ mod($\mathcal{T}, \mathcal{E}$)
- $O \leftarrow O \cup \mathcal{M}$

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:
  - Fox ⊑ ∀ feedsOn. Bird
  - Fox □ Bird ⊑ ∃ feedsOn. T
  - Bird ⊑ ¬Fox
  - Bird ⊑ Bird □ Fox
- Off-topic:
  - Duck ⊑ 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:

Animal $\equiv$ Bird $\sqcup$ Fox
Bird $\equiv$ Duck $\sqcup$ Chicken
Duck $\sqsubseteq$ $\exists$ feedsOn.Grass
Chicken $\sqsubseteq$ $\exists$ feedsOn.Worm
Fox $\sqsubseteq$ $\exists$ feedsOn.Bird

Farm $\sqcup$ Animals
\[
\models Animal \sqsubseteq \exists \text{feedsOn.} \top
\]
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:**

\[
\begin{align*}
\text{Animal} &\equiv \text{Bird} \sqcup \text{Fox} \\
\text{Bird} &\equiv \text{Duck} \sqcup \text{Chicken} \\
\text{Duck} &\sqsubseteq \exists \text{feedsOn}.\text{Grass} \\
\text{Chicken} &\sqsubseteq \exists \text{feedsOn}.\text{Worm} \\
\text{Fox} &\sqsubseteq \exists \text{feedsOn}.\text{Bird} \\
\text{Farm} \cup \text{Animals}_1 &\not= \\
\text{Animals}_1 &
\end{align*}
\]
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:**

- Animal \(\equiv\) Bird \(\sqcup\) Fox
- Bird \(\equiv\) Duck \(\sqcup\) Chicken
- Duck \(\subseteq\) \(\exists\) feedsOn.Grass
- Chicken \(\subseteq\) \(\exists\) feedsOn.Worm
- Fox \(\subseteq\) \(\exists\) feedsOn.Bird

\[\text{Farm} \cup \text{Animals}_2\]
\[\not\subseteq\]
\[\text{Animal} \subseteq\) \(\exists\) 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:**

Animal $\equiv$ Bird $\sqcup$ Fox
Bird $\equiv$ Duck $\sqcup$ Chicken
Duck $\sqsubseteq$ $\exists$ feedsOn.Grass
Chicken $\sqsubseteq$ $\exists$ feedsOn.Worm
Fox $\sqsubseteq$ $\exists$ feedsOn.Bird

Farm $\sqcup$ Animals$_3$

$\not\equiv$

Animal $\sqsubseteq$ $\exists$ feedsOn.$\top$

Animals$_3$

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:

\[
\begin{align*}
\text{Animal} & \equiv \text{Bird} \sqcup \text{Fox} \\
\text{Bird} & \equiv \text{Duck} \sqcup \text{Chicken} \\
\text{Duck} & \sqsubseteq \exists \text{feedsOn. Grass} \\
\text{Chicken} & \sqsubseteq \exists \text{feedsOn. Worm} \\
\text{Fox} & \sqsubseteq \exists \text{feedsOn. Bird}
\end{align*}
\]

\[
\text{Farm} \sqcup \text{Animals}_4
\]

\[
\models
\]

\[
\text{Animal} \sqsubseteq \exists \text{feedsOn. T}
\]
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 $\mathcal{O} \cup \mathcal{E} \models C \sqsubseteq D,$
then $\mathcal{O} \cup \mathcal{E}' \models C \sqsubseteq D.$
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 $\mathcal{O} \cup \mathcal{E} \models C \sqsubseteq D$,
  
  then $\mathcal{O} \cup \mathcal{E}' \models C \sqsubseteq D$.

- Coverage $\models$ preserving entailments

- $\mathcal{O} \cup \mathcal{E}$ is called \textit{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 $\mathcal{T}$:
  
  \[
  \text{if } \quad O \cup \mathcal{E} \models C \sqsubseteq D, \quad \text{then } \quad O \cup \mathcal{E}' \models C \sqsubseteq D.
  \]

- Coverage $\hat{=} \text{ preserving entailments}$

- $O \cup \mathcal{E}$ is called \textit{conservative extension (CE)} of $O \cup \mathcal{E}'$

- No coverage $\leadsto$ no encapsulation $\leadsto$ no \textit{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 $\mathcal{T}$:

  
  \[
  \text{if } \mathcal{O} \cup \mathcal{E} \models C \sqsubseteq D, \text{ then } \mathcal{O} \cup \mathcal{E}' \models C \sqsubseteq D.
  \]

- Coverage $\hat{\models}$ preserving entailments

- $\mathcal{O} \cup \mathcal{E}$ is called *conservative extension (CE)* of $\mathcal{O} \cup \mathcal{E}'$

- Minimal covering modules via CEs
- CEs hard to impossible to decide
- Tractable approximation: syntactic locality
A working cycle

1. **Edit your ontology** $O$
2. **Load an external ontology** $E$
3. **Specify terms from** $E$
4. **Module Coverage** $M \leftarrow \text{mod}(T, E)$
5. **Safety** $O \leftarrow O \cup M$
**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?
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:

Tweety : Duck, ¬Flies
Duck ⊑ Bird

Animals

Bird ⊑ Flies

Farm
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:

Tweety : Duck, ¬Flies
Duck ⊑ Bird

Animals

Bird ⊑ Flies

Farm
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:

Tweety: Duck, ¬Flies
Duck ⊑ Bird

Animals

Farm ∪ Animals ⊨ Bird ⊑ Flies
but Animals ⊭ Bird ⊑ Flies
Safety

- Our ontology $\mathcal{O}$ uses the imported terms safely if for all concepts $C, D$ built from the imported terms:
  
  if $\mathcal{E}' \not\models C \sqsubseteq D,$
  then $\mathcal{O} \cup \mathcal{E}' \not\models C \sqsubseteq D,$

- Safety $\widehat{\models}$ preserving non-entailments
## Comparison of different approaches

<table>
<thead>
<tr>
<th>Kind of “module”</th>
<th>Covrg.</th>
<th>Min.</th>
<th>Covered DLs</th>
<th>Complexity</th>
</tr>
</thead>
<tbody>
<tr>
<td>All ax’s referencing $\mathcal{T}$</td>
<td>$\times$</td>
<td>any</td>
<td>easy</td>
<td></td>
</tr>
<tr>
<td>Seidenberg/Rector</td>
<td>$\times$</td>
<td>any</td>
<td>easy</td>
<td></td>
</tr>
<tr>
<td>Prompt</td>
<td>$\times$</td>
<td>?</td>
<td>easy</td>
<td></td>
</tr>
</tbody>
</table>
## Comparison of different approaches

<table>
<thead>
<tr>
<th>Kind of “module”</th>
<th>Covrg.</th>
<th>Min.</th>
<th>Covered DLs</th>
<th>Complexity</th>
</tr>
</thead>
<tbody>
<tr>
<td>All ax’s referencing $\mathcal{T}$</td>
<td>✗</td>
<td></td>
<td>any</td>
<td>easy</td>
</tr>
<tr>
<td>Seidenberg/Rector</td>
<td>✗</td>
<td></td>
<td>any</td>
<td>easy</td>
</tr>
<tr>
<td>Prompt</td>
<td>✗</td>
<td></td>
<td>?</td>
<td>easy</td>
</tr>
<tr>
<td>The whole ontology</td>
<td>✓</td>
<td>XX</td>
<td>any</td>
<td>easy</td>
</tr>
<tr>
<td>conserv.-based mod.</td>
<td>✓</td>
<td>✓</td>
<td>few</td>
<td>hard</td>
</tr>
<tr>
<td>MEX (Liverpool)</td>
<td>✓</td>
<td>✓</td>
<td>acyclic $\mathcal{EL}$</td>
<td>easy</td>
</tr>
<tr>
<td>locality-based mod.</td>
<td>✓</td>
<td>✗</td>
<td>$\approx$ OWL 2</td>
<td>easy</td>
</tr>
<tr>
<td>E-connections</td>
<td>✓</td>
<td>✗</td>
<td>OWL 1</td>
<td>easy</td>
</tr>
</tbody>
</table>
## Comparison of different approaches

<table>
<thead>
<tr>
<th>Kind of “module”</th>
<th>Covrg.</th>
<th>Min.</th>
<th>Covered DLs</th>
<th>Complexity</th>
</tr>
</thead>
<tbody>
<tr>
<td>All ax’s referencing $\mathcal{T}$</td>
<td>✗</td>
<td></td>
<td>any</td>
<td>easy</td>
</tr>
<tr>
<td>Seidenberg/Rector</td>
<td>✗</td>
<td></td>
<td>any</td>
<td>easy</td>
</tr>
<tr>
<td>Prompt</td>
<td>✗</td>
<td></td>
<td>?</td>
<td>easy</td>
</tr>
<tr>
<td>The whole ontology</td>
<td>✓</td>
<td>✗</td>
<td>any</td>
<td>easy</td>
</tr>
<tr>
<td>conserv.-based mod.</td>
<td>✓</td>
<td>✓</td>
<td>few</td>
<td>hard</td>
</tr>
<tr>
<td>MEX (Liverpool)</td>
<td>✓</td>
<td>✓</td>
<td>acyclic $\mathcal{EL}$</td>
<td>easy</td>
</tr>
<tr>
<td>locality-based mod.</td>
<td>✓</td>
<td>✗</td>
<td>$\approx$ OWL 2</td>
<td>easy</td>
</tr>
<tr>
<td>E-connections</td>
<td>✓</td>
<td>✗</td>
<td>OWL 1</td>
<td>easy</td>
</tr>
<tr>
<td>interpolants-based (no subsets!)</td>
<td>✓</td>
<td>✓✓</td>
<td>few</td>
<td>hard</td>
</tr>
</tbody>
</table>
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
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 $\neq T$)
- 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?
Highly optimised algorithm to extract all modules

<table>
<thead>
<tr>
<th>Ontology</th>
<th>#Ax</th>
<th>#Terms</th>
<th>#mods</th>
<th>Theor. Max.</th>
<th>time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Koala</td>
<td>42</td>
<td>25</td>
<td>3660</td>
<td>33 554 432</td>
<td>9s</td>
</tr>
<tr>
<td>Mereology</td>
<td>44</td>
<td>25</td>
<td>1952</td>
<td>33 554 432</td>
<td>3min</td>
</tr>
</tbody>
</table>
Highly optimised algorithm to extract all modules

<table>
<thead>
<tr>
<th>Ontology</th>
<th>#Ax</th>
<th>#Terms</th>
<th>#mods</th>
<th>Theor. Max.</th>
<th>time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Koala</td>
<td>42</td>
<td>25</td>
<td>3660</td>
<td>33 554 432</td>
<td>9s</td>
</tr>
<tr>
<td>Mereology</td>
<td>44</td>
<td>25</td>
<td>1952</td>
<td>33 554 432</td>
<td>3min</td>
</tr>
</tbody>
</table>

- 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

\[
\text{Weight}(M) = \text{PullingPower}(M) \cdot \text{Cohesion}(M)
\]

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

*How many terms are needed to “pull” all the terms into \(M\)?*

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

*How strongly are terms in \(M\) 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.