Structure-Based Partitioning of Large Concept Hierarchies
Heiner Stuckenschmidt, Michel Klein
Vrije Universiteit Amsterdam

Outline
• Motivation: The Case for Ontology Partitioning
  ⇆ Lots of Pictures
• A Partitioning Method
  – Create a dependency graph
  – Determine the strength of dependencies
  – Compute Partitioning
  – Improve Partitioning
• Experiments

Ontologies and the Semantic Web
• Ontologies are the backbone of semantic web applications
  – Content-Based Retrieval
  – Information Integration
  – Web Service Discovery
• More and More Large Ontologies become available
  – General Purpose: Open Directory, Yahoo!, …
  – Medicine: GALEN, UMLS, FMA, …

The Case for Partitioning
• Distributed Development and Maintenance
  – Experts can update their portion independently of other parts
• Selective Publication and Use of Terminologies
  – Stable subsets can be published in the development phase
  – Users can chose relevant subset of an ontology

An Abstract View of the Problem
• Despite the standardization of Languages there is no agreement on the way ontologies are represented.
  – All ontologies contain classes
  – Most organize them in a hierarchy
  – Many define relations between classes
  – Some provide formal definitions of classes
• We concentrate on partitioning ontologies

Overview of the Process
1. Create Dependency Graph
2. Determine Strength of Dependencies
3. Compute Partitions
Dependencies I: Hierarchy

• There is a strong dependency between classes and their subclasses:

Dependencies I: Hierarchy

• We can include definitions by computing the subsumption hierarchy:

Dependencies II: Shared Relations

Overview of the Process
1. Create Dependency Graph
2. Determine Strength of Dependencies
3. Compute Partitions

Relative Strength Networks

• Compute relative strength [Burt '92] of dependency:

\[ w(c_i, c_j) = \frac{a_{ij} + a_{ji}}{\sum_k a_{ik} + a_{kj}} \]

• Example:

Result for the Example
Overview of the Process
1. Create Dependency Graph
2. Determine Strength of Dependencies
3. Compute Partitions

Computing Islands
- We use maximal line islands [Batagejil 2000] to compute partitions in the dependency graph.
- A set of vertices is a line island in network if and only if it induces a connected subgraph $\{v_1, v_2, \ldots, v_n\}$ such that $\forall i \neq j, v_i \sim v_j$ and there is a maximal spanning tree $T$ over nodes in the island such that:

Understanding Islands

Result for the Example

Overview of the Process
1. Create Dependency Graph
2. Determine Strength of Dependencies
3. Compute Partitions

Improving Partitions (work in progress)
- Islands are often very small (2 - 4 nodes) resulting in unwanted partitions of the ontology.
Improving Partitions

- Islands are often very small (2 - 4 nodes) resulting in unwanted partitions of the ontology.

- Observation: small islands almost always have a large height value (1 or 0.5).

\[ h(i) = h(i) + \sum_{j \in P(i)} h(j) \]

Ontology Partitioning Tool

- Features:
  - OWL and KIF Import
  - Selection of Criteria
  - Computation of line islands

Experiments

- The ACM Computer Science Classification
  - 1000 concepts, fixed hierarchy

- The Standard Upper Model Ontology SUMO
  - 600 concepts, fixed hierarchy

- The NCI cancer ontology
  - Subset with 2400 concepts, fixed hierarchy

- The DICE ontology
  - About 2000 concepts, classified hierarchy

Distributed Representation and Reasoning

Based on Work of:
- Alex Borgida
- Paolo Bouquet
- Fausto Giunchiglia
- Frank van Harmelen
- Luciano Serafini
- Heiner Stuckenschmidt
- Andre Tamkin
- Holger Wache

Distributed Representation and Reasoning

Based on Work of:
- Alex Borgida
- Paolo Bouquet
- Fausto Giunchiglia
- Frank van Harmelen
- Luciano Serafini
- Heiner Stuckenschmidt
- Andre Tamkin
- Holger Wache
Ontology Mappings

Problem: Semantics of Mappings
- What is the semantic of this mapping?
- Share different approaches the semantic?
- If not where are the differences?

The Logic of Mappings

Global Interpretation

An Alternative Semantics

Local Domain Semantics
Goal: Reasoning with and about mappings

The Case Study:

- Select three topics with sufficient overlap
  - Substances
  - Structures
  - Processes
- Analyse Subhierarchies in the different Terminologies
- Define partial ad-hoc mappings between individual concepts
- Use semantics to verify and complete mappings

Example: Substances I

Example: Substances II

Verification of Mappings

Infering new Mappings
Reasoning using Mappings

The formal Basis: Distributed Description Logics

DDL syntax
- **DDL** is a family of description logics \(\{\text{DL}_i\}_{i \in I}\).
  - A bridge rule from \(i\) to \(j\) is an expression of the form:
    - \(i : X \rightarrow j : Y\) (into-bridge rule)
    - \(i : X\hookrightarrow j : Y\) (onto-bridge rule)
  where \(X\) and \(Y\) are concepts of \(\text{DL}_i\) and \(\text{DL}_j\).
  - A distributed T-box (DTB) is a pair \(\langle \{T_i\}_{i \in I}, \{B_{ij}\}_{i \neq j \in I} \rangle\) where \(B_{ij}\) is a collection of bridge rules from \(i\) to \(j\).

Bridge graph
- A bridge graph of a DTB

DDL semantics
A distributed interpretation (DI) of a DTB
\(\langle \{I_i\}_{i \in I}, \{r_{ij}\}_{i \neq j \in I} \rangle\)
- \(I_i\) is a local interpretation of \(T_i\) on a local domain \(\Delta_i\)
  \(T_1, T_2, T_3, T_4, T_5, T_6, T_7\)
  \(I_1, I_2, I_3, I_4, I_5, I_6, I_7\)
- \(r_{ij}\) is a domain relations from \(I\) to \(j\)
  \(r_{ij} : \Delta_i \times \Delta_j\)

DDL satisfiability
A distributed interpretation (DI) satisfies DTB
\(\langle \{I_i\}_{i \in I}, \{r_{ij}\}_{i \neq j \in I} \rangle\) satisfies DTB\(\langle \{T_i\}_{i \in I}, \{B_{ij}\}_{i \neq j \in I} \rangle\)
DI \(\models\) DTB
- If
  - all \(T_i\) are satisfied
  - all bridge rules \(B_{ij}\) are satisfied
Into-bridge rule satisfiability

\[ \Delta^i \models i : x \rightarrow j : y \]

\[ r_j(x^i) \subseteq Y^j \]

Onto-bridge rule satisfiability

\[ \Delta^i \models i : x \rightarrow j : y \]

\[ r_j(x^i) \supseteq Y^j \]

Subsumption propagation in DDL

\[ \text{DTB} = \langle T_1, T_2, B_{12} \rangle \]

Directionality property:
Knowledge propagates \textit{ONLY} along the direction of bridge rules!

Generalized subsumption propagation

\[ \text{DTB} = \langle \{T_i\}_{i \in I}, \{B_{ij}\}_{i \neq j \in I} \rangle \]

Soundness and completeness

Let \( \text{DTB}_{12} = \langle T_1, T_2, B_{12} \rangle \) be a distributed T-box

Bridge operator encapsulates propagated axioms

\[ B_{12}(T_1) = \langle G \sqcup \bigcup_{k=1}^{n} H_k \mid 1:A \rightarrow 2:G = B_{12} \rangle \]

for \( 1 \leq k \leq n, n \geq 0 \)

Theorem

\[ \text{DTB}_{12} \models 2 : X \sqsubseteq Y \iff T_2 \cup B_{12}(T_1) \models X \sqsubseteq Y \]

Distributed tableau algorithm
Basic reasoning service of DDL

\[ \text{DTB} \models i:C \subseteq D \quad \text{DTB} \models i:X \]

i:X is **satisfiable** with respect to DTB
- if there exist a DI such that DI \( \models \) DTB
- and \( X \models \neg \exists 0 \)

**Restrictions:**
1. bridge graph is cycle-free
2. bridge rules connect atomic concepts
3. no nominals

Distributed tableau intuition

\[ \text{DTab}_j(X) = \text{Tab}_j(X) + \text{"lazy computation of bridge operator"} \]

Distributed tableau intuition

Algorithm formalization

\[ \text{DTab}_j \]

- **SHIQ**-tableau expansion rules +
- "bridge" expansion rule:
  
  \[
  \begin{align*}
  &\text{If} \\
  &\quad 1. \ G \in L(x), \ i:A \rightarrow j;G \rightarrow B, \ and \\
  &\quad 2. \ B \subseteq \{B \} \quad i:B \rightarrow j;B, \ and \\
  &\quad \text{DTab}_j(A \sqcap \neg (B, \cup B)) = \text{Unsatisfiable} \text{ for some } H \subseteq L(x), \ then \\
  &\quad L(x) \rightarrow L(x) \cup \{H\}
  \end{align*}
  \]

Algorithm properties

**Theorem (Termination)** For any acyclic distributed T-box and for any **SHIQ** concept X, \( \text{DTab}_j(X) \) terminates.

**Theorem (Soundness and completeness)** j:X is satisfiable in distributed T-box if and only if \( \text{DTab}_j(X) \) can generate a complete and clash-free completion tree.

DRAGO reasoning architecture
Distributed reasoning architecture

Implementation

- OWL ontologies
- C-OWL semantic mappings
- Distributed Reasoner is an extension to open source OWL Reasoner Pellet