Approximate (logical) Reasoning

Based on Work of:
Frank van Harmelen
Zhisheng Huang
Perry Groot
Heiner Stuckenschmidt
Annette ten Teije
Holger Wache

Overview

• Why Approximate Reasoning?

• Scalability (Anytime Reasoning):
  – Concept Classification
  – Instance Retrieval

• Robustness:
  – Querying Heterogeneous Sources
  – Reasoning with inconsistency

Logic =

perfect reasoning
under perfect conditions

• unlimited time
• Homogeneous knowledge
• Correct and consistent Knowledge

Robust Knowledge Representation

• Reliance on logic is a strength
  • Strong theoretical basis
  • Well known properties
  • Well known implementation techniques

• Reliance on logic is a weakness
  • Strict (no ‘good enough’ answers)
  • Abrupt (no intermediate answers)
  • Inefficient (no time/quality trade-off)

1: Fault tolerant classification

• Terminologies will be sloppy:
  • made by non-experts
  • made by machines:
    • scraping from
      – file-hierarchies,
      – mail-folders
    – todo-lists & phone-books on PDA’s
  • machine learning from examples

1: Fault tolerant classification

• Sloppy terminologies need robust inference
2: Mapping terminologies

- There will be no standardized vocabulary
- Communication is possible only through shared vocabulary
- All concepts must be approximated in this shared vocabulary

The overall dream….

Logic as a good model of practical reasoning (not just idealised reasoning)

Good model = mathematical + computational

Approximate Classification in Description Logics

Approximate Entailment through Simplification

- Simplify query
- Simple query \( \Rightarrow \) fast query answering
- Simple query \( \Rightarrow \) approximated answers
- Continuously complete query
- Anytime behavior

How to simplify?

First Idea:
Omit some parts (e.g. \( \Phi \), \( \Psi \))

Second Idea:
Rewrite some parts (e.g. \( \Phi \), \( \Psi \))

How to simplify? (II)

\[ \phi \leftrightarrow \psi \]
S1-/S3-entailment

- State of the Art: S1-/S3-entailment
  - sound and complete
  - semantic approach
- S1-entailment: interpret everything outside of S as false
- S3-entailment: interpret everything outside of S as true

S1/S3-Entailment & Anytime

- Anytime behavior when S_i will be enlarged continuously
- Interesting Feature: Reusing proof from previous level

Cadoli-Schaerf-Approximation for DLs

\[
\begin{align*}
C_i^+ & : \exists R.C \rightarrow T \\
C_i^- & : \exists R.C \rightarrow F
\end{align*}
\]

- Depth of subconcept D: number of universal quantifiers that have D in its scope.

Application: Classification

- Central process: Classify Query Q
  - Generating the subsumption hierarchy
  - Instance Retrieval

Classification algorithm

```
Algorithm 1: Classification

Require: A query concept C
Ensure: D ⊆ C

1. LEVEL := 0
2. Compute \( \Gamma \) (\( \Gamma \) is the set of all concepts in the DL)
3. If \( \Gamma \) is unsatisfiable, then output \( \Gamma \)
4. Create a list \( \{C_1, \ldots, C_n\} \) of concepts in \( \Gamma \)
5. For each concept \( C_i \) in the list
   a. If \( C_i \) is a subconcept of \( C \)
      i. LEVEL := LEVEL + 1
      ii. \( Level(C_i) := Level(C) + 1 \)
      iii. \( \text{Result} := \text{true} \)
   b. Else
      i. If \( C_i \) is not a subconcept of \( C \)
         a. If \( C_i \) is a concept in the DL
            i. LEVEL := LEVEL + 1
            ii. \( Level(C_i) := Level(C) + 1 \)
            iii. \( \text{Result} := \text{false} \)
      ii. Else
         i. \( \text{Result} := \text{false} \)

Return \( \text{Result} \)
```

Approximating the Classification

- C is subsumed by D ⇔ \( C \cap \neg D \) is unsatisfiable
- Cadoli-Schaerf ensures:
  \[
  \neg (C \cap \neg D) \text{ at level } \text{Level} \Rightarrow (C \cap \neg D) \text{ is unsatisfiable}
  \]
Approximation of subsumption

Algorithm 3: A simple concept expression is not satisfiable.

1. \( \text{(Query \text{- Concept}) \text{ is not satisfiable}} \)

Algorithm 4: A simple concept expression is not satisfiable.

1. \( \text{(Query \text{- Concept}) \text{ is not satisfiable}} \)

Implementation

Query

Classify

Approximate

Subsumption

Taxonomy

DIG Interface:

RACER

FACT

Further Results

\[
\begin{array}{c|c|c|c|c|c|c|c|c|c}
\text{C} & \text{false} & \text{false} & \text{true} & \text{false} & \text{false} & \text{true} & \text{false} \\
\hline
\text{false} & \text{false} & \text{false} & \text{false} & \text{true} & \text{false} & \text{false} & \text{false} \\
\text{true} & \text{false} & \text{true} & \text{true} & \text{false} & \text{false} & \text{true} & \text{true} \\
\end{array}
\]

Mixed Results: Classifying in TAMBIS

- Application: Classification of Concepts

⇒ sequence of subsumption test: \( C \sqsubseteq D \)

\[
\begin{array}{c|c|c|c|c|c|c}
\text{true} & \text{false} & \text{true} & \text{false} & \text{true} & \text{false} \\
\hline
\text{true} & \text{false} & \text{true} & \text{false} & \text{true} & \text{false} \\
\text{false} & \text{false} & \text{false} & \text{false} & \text{false} & \text{false} \\
\end{array}
\]

Problem: Term Collapsing

\( C \sqsubseteq D \leftrightarrow C \sqcap \lnot D \) is unsatisfiable

Query = \( \ldots \cap \Phi \ldots \cap (\ldots \cup \Psi \cup \ldots) \)

- Term C
  - to be classified;
  - very often a conjunction of subterms
  - e.g. conjunctive queries

- Term D
  - From the subsumption hierarchy
  - Very often also conjunction of subterms

Problem: Term Collapsing

\( C \sqsubseteq D \leftrightarrow C \sqcap \lnot D \) is unsatisfiable

Query = \( \ldots \cap \top \ldots \)

- Term C
  - to be classified;
  - very often a conjunction of subterms
  - e.g. conjunctive queries
Problem: Term Collapsing

\[ C \sqsubseteq D \leftrightarrow C \sqcap \neg D \text{ is unsatisfiable} \]

Query = \bot

Classifying in TAMBIS (IV)

| Term Collapsing: | 157 = 100% | 65 = 35.9% | 190 = 62.1% |

Lessons learned

\[ \phi \mapsto \psi \]

- Avoid Term Collapsing
  - Replace \( \psi \) with something else than \( T \) or \( \bot \)
- Find better places to rewrite
  - Ontology-adapted \( \psi \)?
- Look at special cases

Application: Instance Retrieval

What to approximate?

- Approximating the whole expression suffers from the term collapsing problem
- Other options:
  - Approximate instance description
  - Approximate the ontology
  - Approximate only query

Approximating the Query

- Stepwise refinement of the query to exclude instances early on:
  \[ (I \not\subseteq Q') \land (Q \subseteq Q') \rightarrow I \not\subseteq Q \]
- Resulting in a sequence \( Q_1, ..., Q_n \), such that:
  - \( Q = Q_n \)
  - \( i > j \Rightarrow Q_i \subseteq Q_j \)
- Addition of conjuncts from the original query produces the desired properties
- Basic question: order of addition?
Variable Dependency

- Construct Query graph and use depth as a basis for selection:

Ordering Heuristics

- We estimate the size of the tableaux needed for checking satisfiability of the unfolded query
  - High penalties for disjunction and universal quantification, which are known to be the source of complexity

<table>
<thead>
<tr>
<th>Run-Time (ms):</th>
</tr>
</thead>
<tbody>
<tr>
<td>Normal</td>
</tr>
<tr>
<td>----------</td>
</tr>
<tr>
<td>Q2</td>
</tr>
<tr>
<td>Q3</td>
</tr>
<tr>
<td>Q12</td>
</tr>
<tr>
<td>Q14</td>
</tr>
<tr>
<td>Q15</td>
</tr>
<tr>
<td>Q17</td>
</tr>
</tbody>
</table>

Analysis:

Conclusions

- Theoretical approaches have problems
  - Web Ontologies are too simple
  - Approximation often becomes meaningless

- Works for the limited case of conjunctive queries:
  - No term collapsing
  - Leads to improvements for hard cases

Short Break

Approximation for Robustness

After the Break: Approximation for Robustness

See you back in 5 Minutes

Approximation for Robustness

Aligning Terminologies
Reasoning with Inconsistency
Combined ontologies need sloppy inference
Mapping ontologies is almost always messy:
post-doc \approx young-researcher

"almost equal"

A Communication Problem

A Toy Example of Ontology Mismatch

Shared ontology:

Private ontology:

Approximation of Concepts
• Upper bounds (lub):
  - Pet \rightarrow (Domestic-Animal)
  - Farm-Animal \rightarrow (Domestic-Animal & Production-Animal)
  - Zoo-Animal \rightarrow (Foreign-Animal)

• Lower bounds (glb):
  - Pet \rightarrow (Cat v Dog)
  - Farm Animal \rightarrow (Cow v Pig)
  - Zoo Animal \rightarrow (Camel v Elephant v Tiger v Lion)

Determining Bounds

Deciding Membership
• We can define an approximate classifier that:
  - Returns ‘Yes’ if the object definitely is a member of the approximated concept, i.e. its lower boundary
  - Returns ‘No’ if the object does definitely not belong to the concept, i.e. its upper boundary
  - Returns ‘?’ if the object is between the bounds
Theory Approximation

• answers are computed as follows:

\[ x \in t^A \Downarrow x \in \bigcup t_{\text{lb}} \Downarrow x \notin \bigcap t_{\text{ub}} \Downarrow \]

[Selman & Kautz 1996]

Re-writing Queries

• The General Idea:
  – replace non-negated concepts by their lower bound
  – Replace negated concepts by their upper bound

• Example Re-Writing:
  (Animal & ¬(Pet v Farm-Animal))
  ➢ (Animal & ¬Pet & ¬Farm-Animal)
  ➢ (Animal & ¬(Cat v Dog) & ¬(Cow v Pig))

Re-Writing Algorithm

```
Algorithm 1 TranslateMessage

Require: The Message to be translated: C
Require: A list of shared concepts: T
require: A terminological knowledge base T
for all t in C do
  if t is negated then
    \[ B(t) = \text{not}(\text{directSubsumes}(t)) \]
    \[ C(t) = \neg (c_1 \land \cdots \land c_n) \text{ for } c_i \in B \]
  else
    \[ B(t) = \text{not}(\text{directSubsumes}(t)) \]
    \[ C(t) = \neg (c_1 \lor \cdots \lor c_n) \text{ for } c_i \in B \]
  end if
  \[ C' = \text{proc replace } t \text{ in } C \text{ by } C(t) \]
end for
return C'
```

Inconsistency and Explosion

• The classical entailment is explosive:

\[ P, \neg P \vdash Q \text{ (Any formula is a logical consequence of a contradiction.)} \]

• The conclusions derived from an inconsistent ontology using the standard reasoning may be completely meaningless.

Why DL reasoning cannot escape the explosion

– The derivation checking is usually achieved by the satisfiability checking.
– \( \Sigma \models \varphi \iff \Sigma \cup \{\neg \varphi\} \) is not satisfiable.
– Tableau algorithms are approaches based on the satisfiability checking
– \( \Sigma \) is inconsistent \( \Rightarrow \) \( \Sigma \) is not satisfiable
  \( \Rightarrow \Sigma \cup \{\neg \varphi\} \) is not satisfiable.
Two main approaches to deal with inconsistent Ontologies

• Reasoning with Inconsistent Ontologies (RIO) by using non-standard (in particular approximate) reasoning

• Inconsistency Diagnose and Repair (will be discussed tomorrow)

Reasoning with Inconsistency: pre-processing

I will ask someone else.

I do not know

Reasoning with inconsistent ontologies: Main Idea

• select some consistent sub-theory from an inconsistent ontology
  – using a selection function, which can be defined on the syntactic or semantic relevance

• apply standard reasoning on the selected sub-theory to find meaningful answers

Inconsistency by Default rules

• Bird ⊆ Animal
• Bird ⊆ Fly
• Eagle ⊆ Bird
• Penguin ⊆ Bird
• Penguin ⊆ ¬Fly

Inconsistency Reasoning Processing: Linear Extension

Selections for the Bird Ontology

• Bird ⊆ Animal
• Bird ⊆ Fly
• Eagle ⊆ Bird
• Penguin ⊆ Bird
• Penguin ⊆ ¬Fly

Selected set for queries on penguins

Selected set for other queries
Inconsistency by Modelling errors

- Brain $\subseteq$ CentralNervousSystem
- Brain $\subseteq$ BodyPart
- CentralNervousSystem $\subseteq$ NervousSystem
- BodyPart $\subseteq$ ¬NervousSystem

Selections on the Brain Example

- Brain $\subseteq$ CentralNervousSystem
- Brain $\subseteq$ BodyPart
- CentralNervousSystem $\subseteq$ NervousSystem
- BodyPart $\subseteq$ ¬NervousSystem

Selected set for queries on body parts

Selection Functions

- Measured by relevance
- Syntactic relevance vs. semantic relevance
- Semantic distance, semantic relateness, semantic similarity in the computational linguistics

The RIO System (available online)