INFR11215 Knowledge Graphs

Tractable Schema Reasoning

Jeff Z. Pan

http://knowledge-representation.org/j.z.pan/

[Reading: Baader et al., Chapter 6]
Lecture Outline

• Motivation
• Overview of EL and reasoning
• Detailed Discussions on EL and reasoning
• Practical
Motivations

• Web Ontology Language (OWL)
  – OWL v2 family
    • OWL 2 DL
    • OWL 2 EL, OWL 2 QL, OWL 2 RL

• ALC not a good starting point
  – its foundation FL₀ (∩ and ∀) is not a good foundation
  – subsumption with GCI is EXPTime-complete
  – EL is PTime-complete
    • TBox reasoning
    • ABox reasoning
    • Query answering
Motivations

- **SNOMED Clinical Terms** is
  - Probably the single most comprehensive clinical terminology
  - Licensed for national use throughout the UK and the US
  - Content that covers most clinical concepts
  - A terminology model that supports retrieval of alternative representations of similar information

- **SNOMED CT** is an EL ontology
  - Clear performance difference between EL algorithm and algorithms for ALC-extended logics
Lecture Outline

• Motivation
• Overview of EL and reasoning
• Detailed Discussions on EL and reasoning
• Practical
What is EL

• **EL Class Description**
  – existential restriction: $\exists r.C$
  – conjunction: $C \sqcap D$
  – the top class: $\top$
  – not including:
    • value restriction: $\forall r.C$
    • disjunction: $C \sqcup D$
    • the bottom class: $\bot$

• **EL Axioms**
  – GCI: $C \sqsubseteq D$
Class Satisfiability Checking in EL

- Every EL class is satisfiable
  - Why?
  - Class satisfiability checking is not an interesting problem

- Challenge
  - Subsumption checking in EL is non-trivial, as it cannot be reduced to class unsatisfiability
  - Why?
  - \( \mathcal{O} \models C \sqsubseteq D \) iff \( C \cap \neg D \) is unsatisfiable
Subsumption Checking in EL

- Subsumption checking in EL (with general Tbox) is \( \text{PTime-complete} \)
  - For \( \text{FL}_0 \), it is \( \text{EXPTime-complete} \)

- Usually this is done in a batch mode: classification
  - A TBox reasoning service that computes subsumption relation among all named classes

- Given an EL TBox \( T \), signature \( \text{Sig} (T) \) contains all class and property names used in \( T \)
Classification

“A class to demonstrate mistakes made with setting a property domain. The property hasTopping has a domain of Pizza. This means that the reasoner can infer that all individuals using the hasTopping property must be of type Pizza. Because of the restriction on this class, all members of IceCream must use the hasTopping property, and therefore must also be members of Pizza. However, Pizza and IceCream are disjoint, so this causes an inconsistency. If they were not disjoint, IceCream would be inferred to be a subclass of Pizza.”

Label:
“Sorvete” @ot
Normalisation

• Idea
  – Simplify the axioms into some certain form so that reasoning algorithms can take advantage of it
  – example: NNF (negated normal form)

• Normal forms for EL
  – $A \sqsubseteq B$
  – $A_1 \sqcap A_2 \sqsubseteq B$
  – $A \sqsubseteq \exists r.B$
  – $\exists r.A \sqsubseteq B$
  – where $A$, $A_1$, $A_2$, $B$ are either named class in $\text{Sig}(T)$ or the top class $\top$
Normalisation Rules

NF0 \[ \hat{D} \subseteq \hat{E} \quad \rightarrow \quad \hat{D} \subseteq A, \quad A \subseteq \hat{E} \]

NF1_r \[ C \cap \hat{D} \subseteq B \quad \rightarrow \quad \hat{D} \subseteq A, \quad C \cap A \subseteq B \]

NF1_l \[ \hat{D} \cap C \subseteq B \quad \rightarrow \quad \hat{D} \subseteq A, \quad A \cap C \subseteq B \]

NF2 \[ \exists r. \hat{D} \subseteq B \quad \rightarrow \quad \hat{D} \subseteq A, \quad \exists r. A \subseteq B \]

NF3 \[ B \subseteq \exists r. \hat{D} \quad \rightarrow \quad A \subseteq \hat{D}, \quad B \subseteq \exists r. A \]

NF4 \[ B \subseteq D \cap E \quad \rightarrow \quad B \subseteq D, \quad B \subseteq E \]

where \( C, D, E \) denote arbitrary \( \mathcal{EL} \) concepts,
\( \hat{D}, \hat{E} \) denote \( \mathcal{EL} \) concepts that are neither concept names nor \( \top \),
\( B \) is a concept name, and
\( A \) is a new concept name.

[credit: F Baader]
Example: Normalisation

- **Input axiom**
  - $\exists r.A \cap \exists r.\exists s.A \subseteq A \cap B$

- **Normalisation**
  1. $\exists r.A \cap \exists r.\exists s.A \subseteq A_0, A_0 \subseteq A \cap B$ (NF0)
  2. $\exists r.A \subseteq A_1, A_1 \cap \exists r.\exists s.A \subseteq A_0$ (NF1l)
  3. $\exists r.\exists s.A \subseteq A_2, A_1 \cap A_2 \subseteq A_0$ (NF1r)
  4. $\exists s.A \subseteq A_3, \exists r.A_3 \subseteq A_2$ (NF2)
  5. $A_0 \subseteq A, A_0 \subseteq B$ (NF4)

[credit: F Baader]
Conservative Extension

- Given two EL TBoxes $T_1$ and $T_2$, $T_2$ is a conservative extension of $T_1$ if
  - $\text{Sig}(T_1) \subseteq \text{Sig}(T_2)$
  - every model of $T_2$ is a model of $T_1$
  - for every model $I_1$ of $T_1$, there exists a model $I_2$ of $T_2$ such as $I_1$ and $I_2$ coincide on $\text{sig}(T_1) \cup \top$, i.e.,
    - $\Delta^{I_1} = \Delta^{I_2}$
    - $A^{I_1} = A^{I_2}$ for every named class in $A \in \text{Sig}(T_1)$, and
    - $r^{I_1} = r^{I_2}$ for every named property in $r \in \text{Sig}(T_1)$
Conservative Extension and EL

- Given two EL TBoxes T1 and T2, such that T2 is a conservative extension of T1, and C, D are EL class descriptions containing only class and property names from Sig(T1)
  - Then T1 ⊨ C ⊑ D iff T2 ⊨ C ⊑ D

- Given two EL TBoxes T1 and T2, such that T2 is the normalised TBox obtained from T1
  - Then T2 is a **conservative extension** of T1
  - T2 is **linear** in the size of T1
Classification Procedure

• We assume that the input TBox axioms are all in normal form
  – The overall number of the normalised GCIs is polynomial in the size of the TBox

• Idea
  – start from the inputs GCIs and add implied GCIs using classification rules
Classification Rules

- To get the concrete we need to
  - replace meta-variables $A, A_1, A_2, A_3, B, B_1$ by concrete named classes and replace meta-variable $r$ by a concrete named property

- Rule application
  - $T'$ start as the TBox
  - If axioms appear on top of the line are in $T'$, then add the axioms below into $T'$ (unless they are already in)

\[
\begin{align*}
\text{CR1: } & A \sqsubseteq A \\
\text{CR2: } & A \sqsubseteq \top \\
\text{CR3: } & A_1 \sqsubseteq A_2, A_2 \sqsubseteq A_3 \\
\text{CR4: } & A \sqsubseteq A_1, A \sqsubseteq A_2, A_1 \sqcap A_2 \sqsubseteq B \\
\text{CR5: } & A \sqsubseteq \exists r.A_1, A_1 \sqsubseteq B_1, \exists r.B_1 \sqsubseteq B \\
\end{align*}
\]
Example: Classification Rules

1. $A \sqsubseteq A, B \sqsubseteq B, B_1 \sqsubseteq B_1, B_2 \sqsubseteq B_2, C \sqsubseteq C$ (CR1)
2. $A \sqsubseteq T, B_1 \sqsubseteq T, B_2 \sqsubseteq T, C \sqsubseteq T, B \sqsubseteq T$ (CR2)
3. $A \sqsubseteq T, T \sqsubseteq B \Rightarrow A \sqsubseteq B$ (CR3)
4. $B_1 \sqsubseteq T, T \sqsubseteq B \Rightarrow B_1 \sqsubseteq B$ (CR3)
5. $B_2 \sqsubseteq T, T \sqsubseteq B \Rightarrow B_2 \sqsubseteq B$ (CR3)
6. $C \sqsubseteq T, T \sqsubseteq B \Rightarrow C \sqsubseteq B$ (CR3)
7. $A \sqsubseteq \exists r.A, A \sqsubseteq B, \exists r.B \sqsubseteq B_1 \Rightarrow A \sqsubseteq B_1$ (CR5)
8. $A \sqsubseteq B_1, A \sqsubseteq B_2, B_1 \cap B_2 \sqsubseteq C \Rightarrow A \sqsubseteq C$ (CR4)
Lecture Outline

• Motivation
• Overview of EL and reasoning
• Detailed Discussions on EL and reasoning
• Practical
Subsumption Checking

- Subsumption checking between two class descriptions $C \sqsubseteq D$ can be reduced to that between two named classes $A_1 \sqsubseteq A_2$

- More precisely
  - An EL TBox $T \models C \sqsubseteq D$ iff $T \cup \{A_1 \sqsubseteq C, D \sqsubseteq A_2\} \models A_1 \sqsubseteq A_2$
Example: Subsumption Checking

1. Extend the KB with \{A' \sqsubseteq A \cap C, \exists r.B \sqsubseteq B'\}, which is normalised as \{A' \sqsubseteq A, A' \sqsubseteq C, \exists r.B \sqsubseteq B'\} (NF4)

2. \(A \sqsubseteq A, B \sqsubseteq B, A' \sqsubseteq A', B' \sqsubseteq B', B1 \sqsubseteq B1, B2 \sqsubseteq B2, C \sqsubseteq C\) (CR1)

3. \(A \sqsubseteq T, A' \sqsubseteq T, B1 \sqsubseteq T, B2 \sqsubseteq T, C \sqsubseteq T, B \sqsubseteq T, B' \sqsubseteq T\) (CR2)

4. \(A \sqsubseteq T, T \sqsubseteq B \Rightarrow A \sqsubseteq B\) (CR3)

5. \(B1 \sqsubseteq T, T \sqsubseteq B \Rightarrow B1 \sqsubseteq B\) (CR3)

6. \(B2 \sqsubseteq T, T \sqsubseteq B \Rightarrow B2 \sqsubseteq B\) (CR3)

7. \(C \sqsubseteq T, T \sqsubseteq B \Rightarrow C \sqsubseteq B\) (CR3)

8. \(A \sqsubseteq \exists r.A, A \sqsubseteq B, \exists r.B \sqsubseteq B1 \Rightarrow A \sqsubseteq B1\) (CR5)

9. \(A \sqsubseteq \exists r.A, A \sqsubseteq B, \exists r.B \sqsubseteq B' \Rightarrow A \sqsubseteq B'\) (CR5)

10. \(A' \sqsubseteq A, A \sqsubseteq B' \Rightarrow A' \sqsubseteq B'\) (CR3)

11. Since \(A' \sqsubseteq B'\) holds, we have \(A \cap C \sqsubseteq \exists r.B\)
EL Family

- **EL+** extends EL with
  - property chain inclusion: \( r_1 \circ \ldots \circ r_k \sqsubseteq r \)
  - concrete domain (n-ary datatype predicate): \( D(f_1, \ldots, f_n) \)

- **EL++** extends EL+ with
  - the bottom class: \( \bot \)
  - nominal: \{a\}
Classification: OWL 2 EL vs OWL 2 DL

- **OWL 2 DL**
  - Subsumption checking is N2EXPTime-Complete
  - GCI-rule is expensive
  - Many new optimisations but still challenging when there are large number of classes (SNOMED CT has over 300K)

- **OWL 2 EL**
  - Batch mode
  - Good base for approximation (such as those used by the TrOWL reasoner)
Conjunctive Queries

- A conjunctive query \( q(\overrightarrow{x}) \) has the form
  - \( \exists y_1, \ldots, y_m. (\alpha_1 \land \ldots \land \alpha_n) \), where \( m \geq 0 \), \( n \geq 1 \)
  - each atom \( \alpha_i \) is a concept atom \( A(x) \) or a property atom \( r(x,y) \)
  - \( y_1, \ldots, y_m \) are called quantified variables
  - quantified variables that appear only in one atom are called unbounded variables

- CQs without constants are called pure CQs
- CQs can be reduced to pure CQs in polynomial time
- An FO query is called a Boolean query if its arity is 0.
Example: Conjunctive Queries

- Assuming that we have three tables Professor, supervises and Student
- Return all pairs of supervisors and students
  - \( q_1(x_1,x_2) = \text{Professor}(x_1) \land \text{supervise}(x_1,x_2) \land \text{Student}(x_2) \)
  - also written as \( q_1(x_1,x_2) \leftarrow \text{Professor}(x_1) \land \text{supervise}(x_1,x_2) \land \text{Student}(x_2) \)
- Return all students whom are supervised by some professors
  - \( q_2(x) = \exists y. \text{Professor}(y) \land \text{supervises}(y,x) \land \text{Student}(x) \)
Ontology Based QA: Example 1

We assume that each concept/relationship of the ontology is mapped directly to a database table.

But the database tables may be **incompletely specified**, or even missing for some concepts/relationships.

**DB:**

Coordinator ⊇ \{ serge, marie \}
Project ⊇ \{ webdam, diadem \}
worksFor ⊇ \{ (serge,webdam), (georg,diadem) \}

**Query:** \( q(x) \leftarrow \text{Researcher}(x) \)

**Answer:** \{ serge, marie, georg \}

[credit: G Xiao]
Ontology Based QA: Example 2

Each person has a father, who is a person.

DB: Person \( \supseteq \{ \text{john, nick, toni} \} \)

hasFather \( \supseteq \{ (\text{john,nick}), (\text{nick,toni}) \} \)

Queries:
- \( q_1(x, y) \leftarrow \text{hasFather}(x, y) \)
- \( q_2(x) \leftarrow \exists y. \text{hasFather}(x, y) \)
- \( q_3(x) \leftarrow \exists y_1, y_2, y_3. \text{hasFather}(x, y_1) \land \text{hasFather}(y_1, y_2) \land \text{hasFather}(y_2, y_3) \)
- \( q_4(x, y_3) \leftarrow \exists y_1, y_2. \text{hasFather}(x, y_1) \land \text{hasFather}(y_1, y_2) \land \text{hasFather}(y_2, y_3) \)

Answers:
- to \( q_1 \): \( \{ (\text{john,nick}), (\text{nick,toni}) \} \)
- to \( q_2 \): \( \{ \text{john, nick, toni} \} \)
- to \( q_3 \): \( \{ \text{john, nick, toni} \} \)
- to \( q_4 \): \( \{ \} \)

[credit: G Xiao]
Ontology Based QA : Example 3

Manager $\equiv$ PrincInv $\sqcup$ Coordinator

- Researcher $\subseteq$ \{andrea, paul, mary, john\}
- Manager $\subseteq$ \{andrea, paul, mary\}
- PrincInv $\subseteq$ \{paul\}
- Coordinator $\subseteq$ \{mary\}
- supervisedBy $\subseteq$ \{(john, andrea), (john, mary)\}
- officeMate $\subseteq$ \{(mary, andrea), (andrea, paul)\}

\[
q(x) \leftarrow \exists y, z. \supseteq \text{supervisedBy}(x, y), \text{Coordinator}(y), \text{officeMate}(y, z), \text{PrincInv}(z)
\]

Answer: \{john\}

To obtain this answer, we need to reason by cases.

[credit: A Schaerf]
Lecture Outline

• Motivation: efficient and scalable reasoning
• Introduction: the EL description logic
• Focus: subsumption checking in EL
• Tutorial
  – Normalisation
  – Classification
  – Subsumption