In the Search of Improvements to the $\mathcal{EL}^+$ Classification Algorithm

Barış Sertkaya

July 16, 2011
**$\mathcal{EL}^+$ Classification Algorithm** [Baader et al., 2005]

1. Normalize the axioms
2. Simultaneously compute the concept hierarchy (without making single subsumption tests)

**The Normal Form:**
- All GCI's of the form
  - $A_1 \sqcap \ldots \sqcap A_n \sqsubseteq B$
  - $A_1 \sqsubseteq \exists r.A_2$
  - $\exists r.A_1 \sqsubseteq B$
- All RIs are of the form
  - $r \sqsubseteq s$, or $r_1 \circ r_2 \sqsubseteq s$
**$\mathcal{EL}^+$ Classification Algorithm**

Given a normalized $\mathcal{EL}^+$ TBox $\mathcal{T}$ the algorithm computes:

1. a mapping $S$ from $\mathit{CN}_{\mathcal{T}}$ to $\mathit{CN}_{\mathcal{T}}$
2. a mapping $R$ from $\mathit{RN}_{\mathcal{T}}$ to $\mathit{CN}_{\mathcal{T}} \times \mathit{CN}_{\mathcal{T}}$

with the meanings:

- $B \in S(A)$ implies $A \sqsubseteq_{\mathcal{T}} B$,
- $(A, B) \in R(r)$ implies $A \sqsubseteq_{\mathcal{T}} \exists r . B$

exhaustively applies a set of completion rules

<table>
<thead>
<tr>
<th>Rule</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>R1</td>
<td>If $A_1, \ldots, A_n \in S(X)$, $A_1 \sqcap \cdots \sqcap A_n \sqsubseteq B \in \mathcal{O}$, and $B \notin S(X)$ then $S(X) := S(X) \cup {B}$</td>
</tr>
<tr>
<td>R2</td>
<td>If $A \in S(X)$, $A \sqsubseteq \exists r . B \in \mathcal{O}$, and $(X, B) \notin R(r)$ then $R(r) := R(r) \cup {(X, B)}$</td>
</tr>
<tr>
<td>R3</td>
<td>If $(X, Y) \in R(r)$, $A \in S(Y)$, $\exists r . A \sqsubseteq B \in \mathcal{O}$, and $B \notin S(X)$ then $S(X) := S(X) \cup {B}$</td>
</tr>
<tr>
<td>R4</td>
<td>If $(X, Y) \in R(r)$, $r \sqsubseteq s \in \mathcal{O}$, and $(X, Y) \notin R(s)$ then $R(s) := R(s) \cup {(X, Y)}$</td>
</tr>
<tr>
<td>R5</td>
<td>If $(X, Y) \in R(r)$, $(Y, Z) \in R(s)$, $r \circ s \sqsubseteq t \in \mathcal{O}$, and $(X, Z) \notin R(t)$ then $R(t) := R(t) \cup {(X, Z)}$</td>
</tr>
</tbody>
</table>
Classification based on Closure

In terms of closure from relational databases [?]  
- \( S(A) \) is the *closure* of \( A \) under \( T \)  
- classifying \( T \) is computing the closure of every concept name

In relational databases  
- *functional dependencies* (FD) specify constraints on data  
- relation \( r \) satisfies the FD \( X \rightarrow Y \) if the tuples with equal \( X \)-values also have equal \( Y \)-values  
- closure of \( X \subseteq R \) under the FDs \( F \) is \( X^+ \) s.t:  
  - for \( A \rightarrow B \in F \) if \( A \subseteq X^+ \), then \( B \subseteq X^+ \)  
  - \( X^+ \) is minimal

\((r \text{ a relation, } X, Y \subseteq R \text{ a set of attributes})\)
The Linear Closure Algorithm [Beeri & Bernstein, 1979]

- **input**: attribute set $X$, set of FDs $\mathcal{F}$
- **output**: closure of $X$ under $\mathcal{F}$

**Initialization:**
- a counter for each FD, initially $\text{count}[W \rightarrow X] := |W|$
- an index for each attribute: $\text{index}[A] := \{ W \rightarrow Z \mid A \in W \}$
- queue of attributes to be processed, initially $q := X$
- closure, initially $c := X$

**Computation:**
- pop an attribute $A$ from $q$
- traverse $\text{index}[A]$, decrement the counters of those FDs
- if $\text{count}[W \rightarrow Z] == 0$ then $q := q \cup (Z \setminus c)$ and $c := c \cup Z$
- repeat until $q == \emptyset$
\( \mathcal{EL}^+ \) Classification based on LinClosure

- make use of the idea with counters
- avoid subset tests when checking whether an axiom applies

```
procedure process(A, X)
begin
    if \( X = B_1, \ldots, B_n \rightarrow B \) and \( B \notin S(A) \) then
        if \( \{B_1, \ldots, B_n\} \subseteq S(A) \) then
            (P1) \( S(A) := S(A) \cup \{B\} \);
            (Q1) \( \text{queue}(A) := \text{queue}(A) \cup \hat{O}(B) \);
        endif
    endif
end
```

- avoid in the worst-case an \( n \) step
Classification based on LinClosure

- slightly modified normal form
- all GCIs of the form:
  - $C_1 \sqcap \ldots \sqcap C_n \sqsubseteq D_1 \sqcap \ldots \sqcap D_n$ where $C_i, D_i$ concept name, or of the form $\exists r.A$.
- “stripped down” version of the original normalization rules
- the linear upper bound on the size of the normalized TBox still holds
- the algorithm computes closure of every $A \in CN_{\mathcal{T}}$ under $\mathcal{T}$
\( \mathcal{EL}^+ \) Classification based on LinClosure

**Initialization**
- counter per concept name per GCI, \( \text{count}[A][\mathbf{\sqcap} C_i \sqsubseteq \mathbf{\sqcap} D_i] := |\mathbf{\sqcap} C_i| \)
- index for each concept name or existential restriction
  \( \text{index}[C] := \{\mathbf{\sqcap} C_i \sqsubseteq \mathbf{\sqcap} D_i \mid C \text{ occurs in } \mathbf{\sqcap} C_i\} \)
- queue of concepts to be processed for each \( A \in \text{CN}_T \), \( q(A) := \{A, \top\} \)
- subsumer list for each \( A \in \text{CN}_T \), \( S(A) := \{A, \top\} \)

**Computation**
- fetch a concept \( C \) from \( q(A) \)
- if \( C \) concept name \( \text{process} - \text{name}(C) \), else \( \text{process} - \text{existential}(C) \)
- traverse \( \text{index}[C] \), decrement the counters of those GCIs
- if \( \text{count}[A][\mathbf{\sqcap} C_i \sqsubseteq \mathbf{\sqcap} D_i] == 0 \), extend \( q(A) \) and \( S(A) \) with new concepts in \( \mathbf{\sqcap} D_i \)
- repeat until all queues are empty
\( \mathcal{EL}^+ \) Classification based on LinClosure

**Initialization**
- counter per concept name per GCI, \( \text{count}[A][\bigcap C_i \sqsubseteq \bigcap D_i] := |\bigcap C_i| \)
- index for each concept name or existential restriction
  \( \text{index}[C] := \{\bigcap C_i \sqsubseteq \bigcap D_i \mid C \text{ occurs in } \bigcap C_i\} \)
- queue of concepts to be processed for each \( A \in \text{CN}_T \), \( q(A) := \{A, \top\} \)
- subsumer list for each \( A \in \text{CN}_T \), \( S(A) := \{A, \top\} \)

**Computation**
- fetch a concept \( C \) from \( q(A) \)
  - if \( C \) concept name \( \text{process - name}(C) \), else \( \text{process - existential}(C) \)
  - traverse \( \text{index}[C] \), decrement the counters of those GCIs
  - if \( \text{count}[A][\bigcap C_i \sqsubseteq \bigcap D_i] == 0 \), extend \( q(A) \) and \( S(A) \) with new concepts in \( \bigcap D_i \)
  - repeat until all queues are empty
\( \mathcal{EL}^+ \) Classification based on LinClosure

**Initialization**

- counter per concept name per GCI, \( \text{count}[A][\sqcap C_i \sqsubseteq \sqcap D_i] := |\sqcap C_i| \)
- index for each concept name or existential restriction
  \( \text{index}[C] := \{\sqcap C_i \sqsubseteq \sqcap D_i \mid C \text{ occurs in } \sqcap C_i\} \)
- queue of concepts to be processed for each \( A \in CN_T \), \( q(A) := \{A, \top\} \)
- subsumer list for each \( A \in CN_T \), \( S(A) := \{A, \top\} \)

**Computation**

- fetch a concept \( C \) from \( q(A) \)
- if \( C \) concept name \( \text{process} \) \( \text{name}(C) \), else \( \text{process} \) \( \text{existential}(C) \)
- traverse \( \text{index}[C] \), decrement the counters of those GCIs
- if \( \text{count}[A][\sqcap C_i \sqsubseteq \sqcap D_i] = 0 \), extend \( q(A) \) and \( S(A) \) with new concepts in \( \sqcap D_i \)
- repeat until all queues are empty
\textbf{\(\mathcal{EL}^+\) Classification based on LinClosure}

\textbf{Initialization}

- counter per concept name per GCI, \(\text{count}[A][\sqcap C_i \sqsubseteq \sqcap D_i] := |\sqcap C_i|\)
- index for each concept name or existential restriction
  \(\text{index}[C] := \{\sqcap C_i \sqsubseteq \sqcap D_i \mid \text{C occurs in } \sqcap C_i\}\)
- queue of concepts to be processed for each \(A \in \text{CN}_T\), \(q(A) := \{A, \top\}\)
- subsumer list for each \(A \in \text{CN}_T\), \(S(A) := \{A, \top\}\)

\textbf{Computation}

- fetch a concept \(C\) from \(q(A)\)
- if \(C\) concept name \(\text{process} – \text{name}(C)\), else \(\text{process} – \text{existential}(C)\)
- traverse \(\text{index}[C]\), decrement the counters of those GCIs
- if \(\text{count}[A][\sqcap C_i \sqsubseteq \sqcap D_i] = 0\), extend \(q(A)\) and \(S(A)\) with new concepts in \(\sqcap D_i\)
- repeat until all queues are empty
\textbf{\( \mathcal{EL}^+ \) Classification based on LinClosure}

\textbf{Initialization}

- counter per concept name per GCI, \( \text{count}[A][\sqcap C_i \sqsubseteq \sqcap D_i] := |\sqcap C_i| \)
- index for each concept name or existential restriction
  \( \text{index}[C] := \{\sqcap C_i \sqsubseteq \sqcap D_i \mid C \text{ occurs in } \sqcap C_i\} \)
- queue of concepts to be processed for each \( A \in \text{CN}_T \), \( q(A) := \{A, \top\} \)
- subsumer list for each \( A \in \text{CN}_T \), \( S(A) := \{A, \top\} \)

\textbf{Computation}

- fetch a concept \( C \) from \( q(A) \)
- if \( C \) concept name \( \text{process} \text{ -- name}(C) \), else \( \text{process} \text{ -- existential}(C) \)
- traverse \( \text{index}[C] \), decrement the counters of those GCIs
- if \( \text{count}[A][\sqcap C_i \sqsubseteq \sqcap D_i] = 0 \), extend \( q(A) \) and \( S(A) \) with \emph{new} concepts in \( \sqcap D_i \)
- repeat until all queues are empty
\( \mathcal{EL}^+ \) Classification based on LinClosure

**Initialization**

- counter per concept name per GCI, \( \text{count}[A][\sqcap C_i \sqsubseteq \sqcap D_i] := |\sqcap C_i| \)
- index for each concept name or existential restriction
  \( \text{index}[C] := \{\sqcap C_i \sqsubseteq \sqcap D_i \mid C \text{ occurs in } \sqcap C_i\} \)
- queue of concepts to be processed for each \( A \in \text{CN}_T \), \( q(A) := \{A, \top\} \)
- subsumer list for each \( A \in \text{CN}_T \), \( S(A) := \{A, \top\} \)

**Computation**

- fetch a concept \( C \) from \( q(A) \)
- if \( C \) concept name process \( \text{name}(C) \), else process \( \text{existential}(C) \)
- traverse \( \text{index}[C] \), decrement the counters of those GCIs
- if \( \text{count}[A][\sqcap C_i \sqsubseteq \sqcap D_i] = 0 \), extend \( q(A) \) and \( S(A) \) with new concepts in \( \sqcap D_i \)
- repeat until all queues are empty
References

In Proceedings of the Methods for Modalities Workshop (M4M-05).

ACM Transactions on Database Systems 4, 30–59.