Introduction and Motivation

Reasoning in OWL 2 EL is \( \text{PTime} \).

ELK is concurrent, optimized and very fast.

- classifies SNOMED CT (300K concepts) in <10s.

Still annoying (and stupid) to do this after every change.

Most changes affect only a small part of the class hierarchy.

The goal: recompute only subsumptions affected by the change.

Typical Ontology Editing Life Cycle

1. Edit: create, remove, or modify axioms (concept definitions).
2. Classify to observe the results and check for errors.
3. Fix, if necessary, and repeat.

Incremental Reasoning Procedure

**Axiom additions are easy**

1. Add expressions to which rules are applicable in Todo.
2. Exhaustively apply rules till fixpoint.

**Deleting all conclusions of removed axioms leads to overdeletion.**

Consider the ontology:

\[
A \sqsubseteq 3R.B, B \subseteq C, 3R.C \sqsubseteq C, A \sqsubseteq B
\]

Removed conclusions: \( A \sqsubseteq B, A \sqsubseteq C \)

However, \( A \sqsubseteq C \) still follows from the remaining axioms!

Main problem: how to efficiently recover alternative derivations?

**Overdelete-Rederive Strategy**

Principal idea: partition the set of expressions (subsumptions).

Clean up partitions with deleted subsumptions and re-derive.

Post clean-up: do not apply rules to remaining subsumptions.

Any partitioning works but it impacts how much stuff is cleaned. \( \mathcal{EL}^+ \) partitions are left hand sides of subsumption axioms. Partitions are NOT stored \( \sim \) no overhead!

Conclusion goes to the same partition as the premise.

Efficient: cleaning overhead negligible, see the evaluation results.

ELK Reasoner: http://elk.semanticweb.org

**Abstract Rule-based Saturation Procedure**

Two collections of expressions:

- **Closure**: expressions between which all rules are applied.
  (initially empty)

- **Todo**: expressions to which rules are yet to be applied.

Apply inferences:

- Poll from Todo.
- Insert into Closure.
- If new, apply all rules with elements from Closure.
- Add the result into Todo.

\( \mathcal{EL} \) Saturation Rules

\[
R_0: C \sqsubseteq C
\]

\[
R_1: C \sqcap D \sqsubseteq E
\]

\[
R_2: C \sqsubseteq E \implies C \sqsubseteq D
\]

\[
R_3: C \sqsubseteq 3R.C \implies C \sqsubseteq D
\]

Evaluation: SNOMED CT, EL-GALEN, GO

SNOMED CT: random changes (±1, ±10, ±100 axioms).

\( \mathcal{EL}^+ \) version of GALEN: random changes.

GO-EXT: revisions obtained from the project’s SVN.

GO and GALEN

The method is simple and extensible to other logics.

More efficient on larger ontologies and smaller changesets. Implemented in ELK 0.4+, available in Protiége.