The Benefits of Incremental Reasoning in OWL EL

Yevgeny Kazakov and Pavel Klinov

Institute of Artificial Intelligence, University of Ulm, Germany

Introduction and Motivation

Reasoning in OWL 2 EL is PTime.
ELK is concurrent, optimized and very fast.

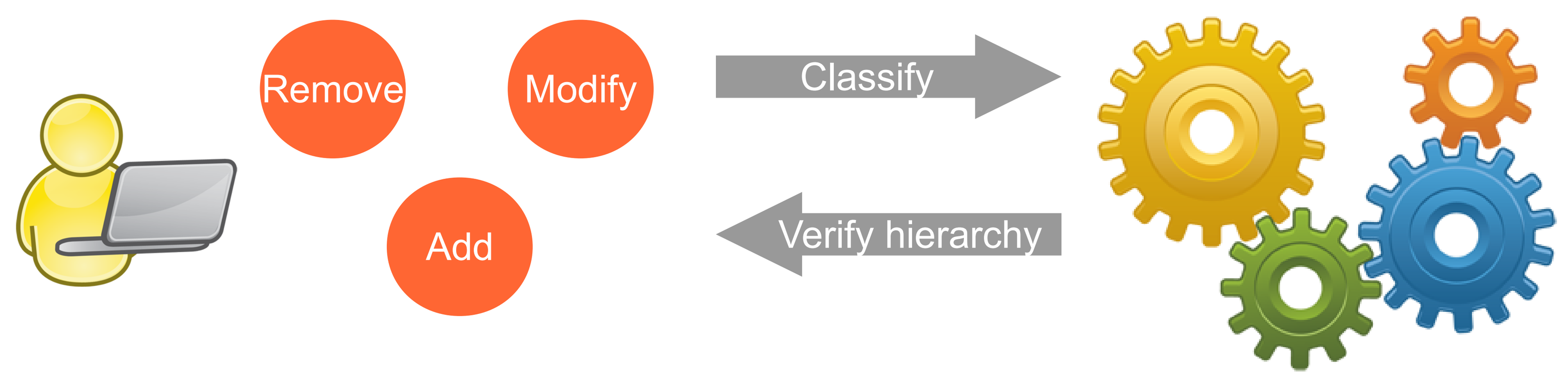
- classifies SNOMED CT (300K concepts) in <10s.

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.



Slow classification leads to the accumulation of changes.
Result: more errors and they are harder to find.
Modern IDEs solve this problem (incremental recompilation).

Incremental Reasoning Procedure

Axiom additions are easy

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

Deletions are tricky

Deleting all conclusions of removed axioms leads to overdeletion.

Consider the ontology:

$\mathbf{A} \sqsubseteq \exists \mathbf{R}.\mathbf{B}$, $\mathbf{B} \sqsubseteq \mathbf{C}$, $\exists \mathbf{R}.\mathbf{C} \sqsubseteq \mathbf{C}$, $\cancel{\mathbf{A} \sqsubseteq \mathbf{B}}$

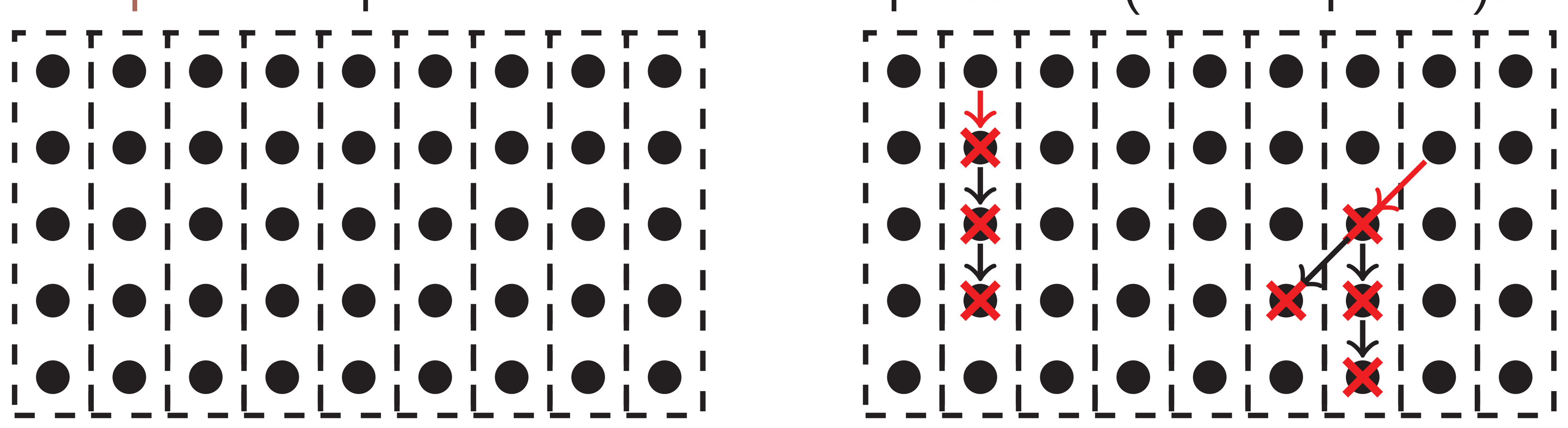
Removed conclusions: $\mathbf{A} \sqsubseteq \mathbf{B}$, $\mathbf{A} \sqsubseteq \mathbf{C}$

However, $\mathbf{A} \sqsubseteq \mathbf{C}$ still follows from the remaining axioms!

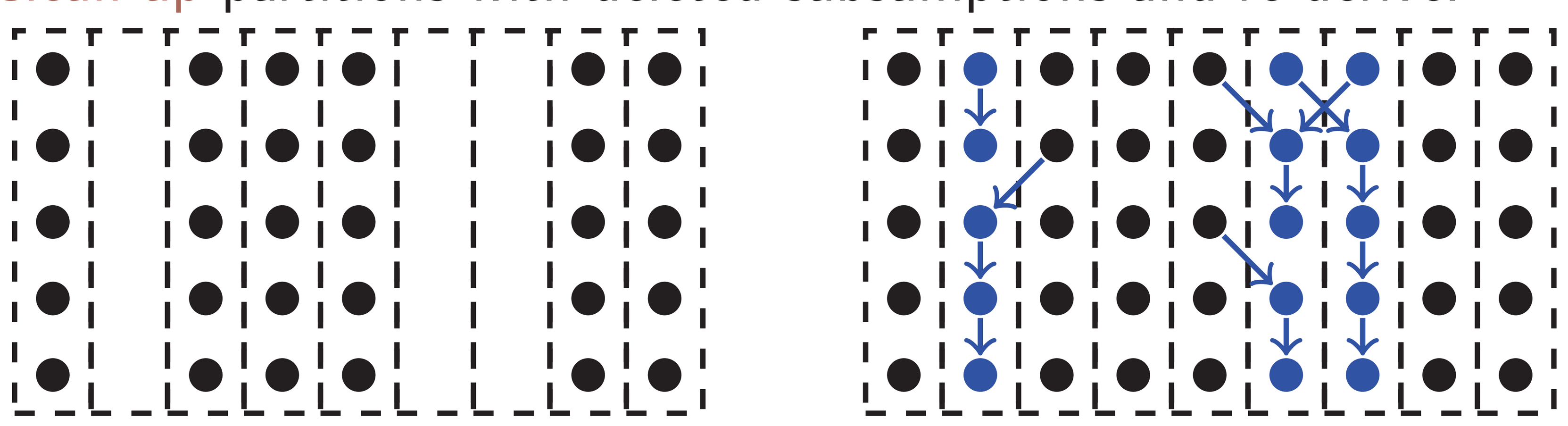
Main problem: how to efficiently recover alternative derivations?

Overdelete-Rederive Strategy

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



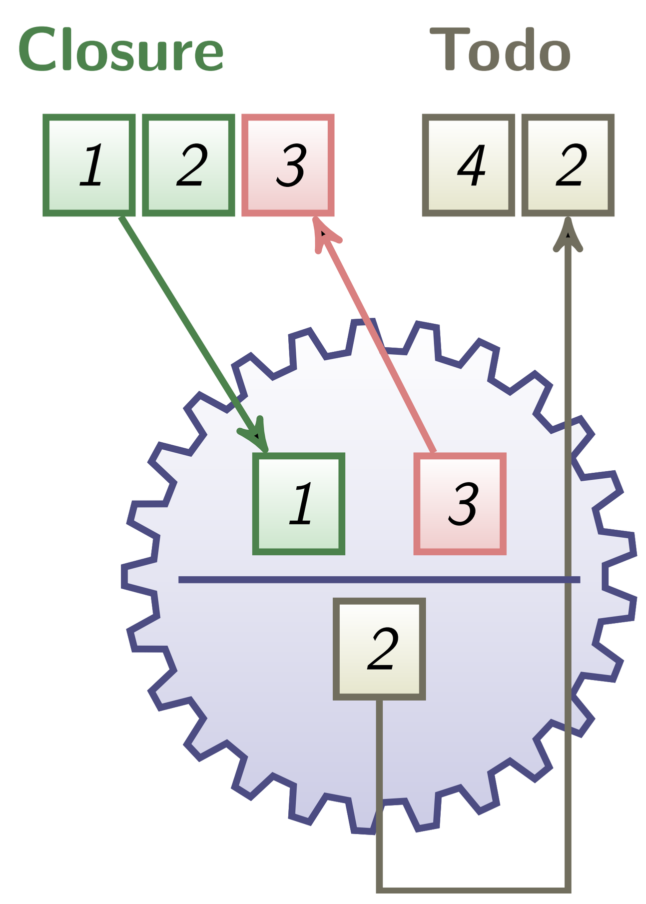
Clean up partitions with deleted subsumptions and re-derive.



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 handsides of subsumption axioms.
Partitions are NOT stored $\leadsto$ 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.

ELK is multi-threaded (the picture is for one thread only).

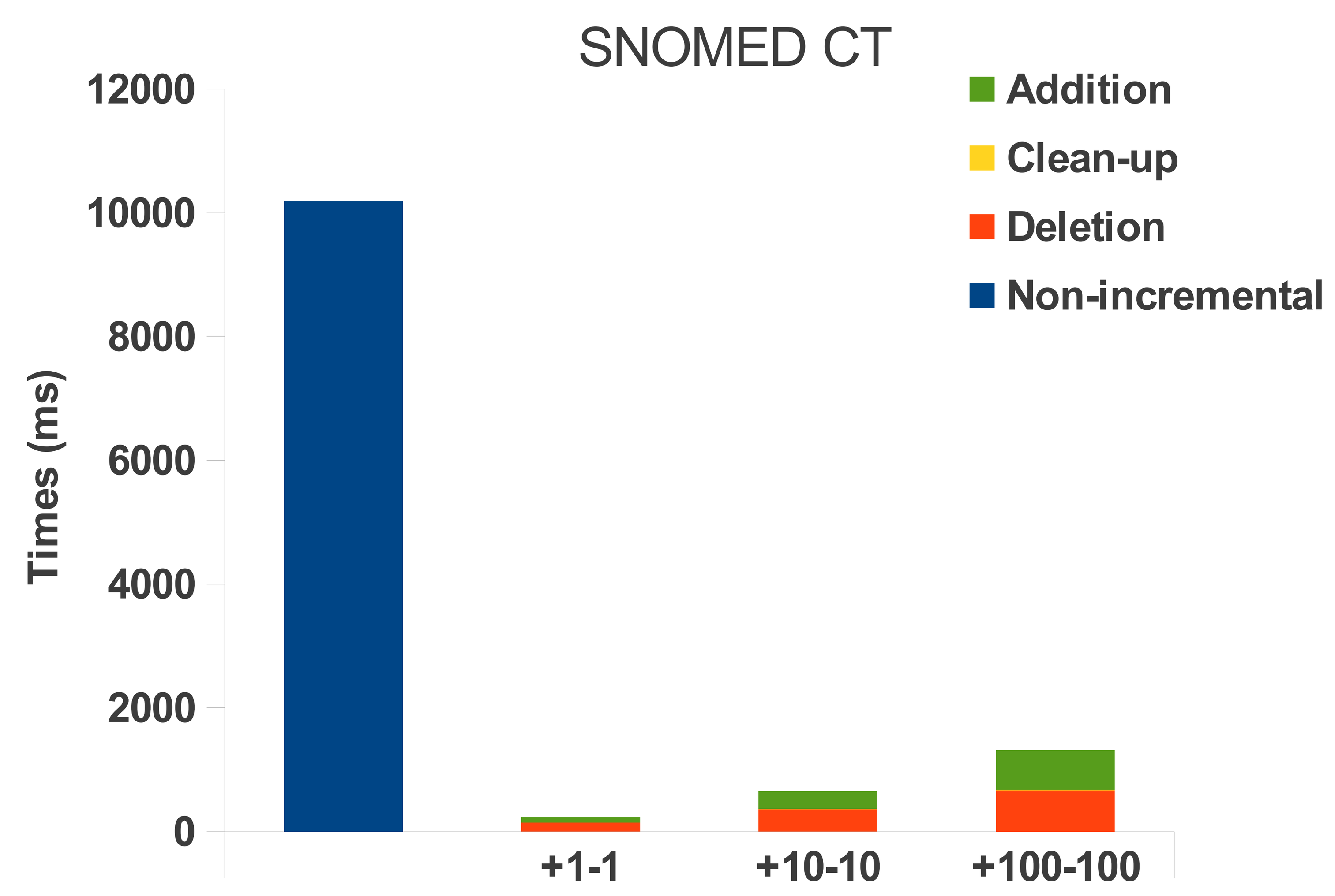
$\mathcal{EL}$ Saturation Rules

$$\mathbf{R}_0\ \frac{}{C \sqsubseteq C} \qquad \mathbf{R}_\sqcap^-\ \frac{C \sqsubseteq D_1 \sqcap D_2}{C \sqsubseteq D_1 \quad C \sqsubseteq D_2} \qquad \mathbf{R}_\sqsubseteq\ \frac{C \sqsubseteq D}{C \sqsubseteq E} : D \sqsubseteq E \in \mathcal{O}$$

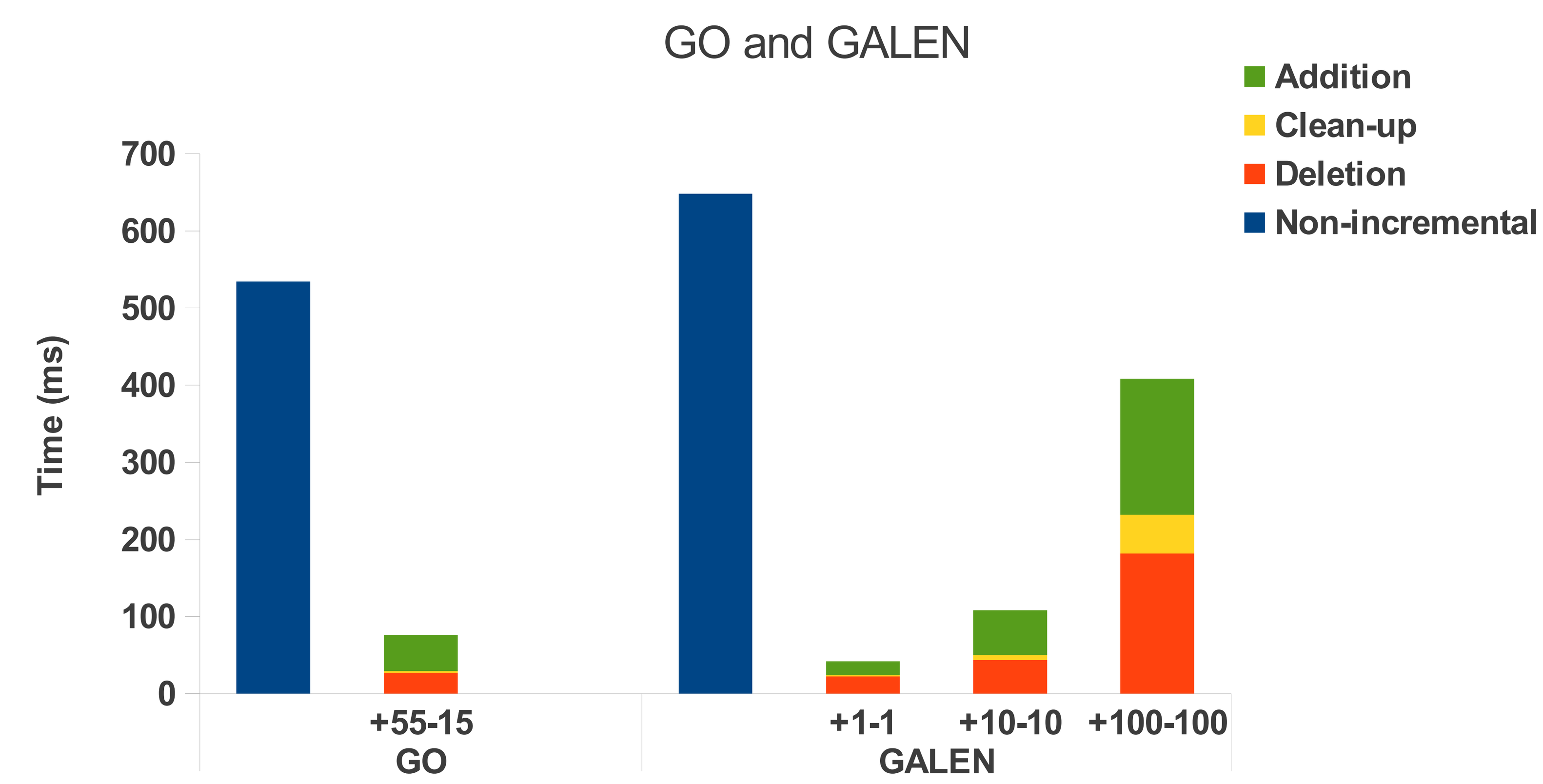
$$\mathbf{R}_\top\ \frac{}{C \sqsubseteq \top} \qquad \mathbf{R}_\sqcap^+\ \frac{C \sqsubseteq D_1 \quad C \sqsubseteq D_2}{C \sqsubseteq D_1 \sqcap D_2} \qquad \mathbf{R}_\exists\ \frac{E \sqsubseteq \exists R.C \quad C \sqsubseteq D}{E \sqsubseteq \exists R.D}$$

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.



The method is simple and extensible to other logics.
More efficient on larger ontologies and smaller changesets.
Implemented in ELK 0.4+, available in Protégé.

Y. Kazakov and P. Klinov, ''Incremental Reasoning without Bookkeeping,'' ISWC 2013

yevgeny.kazakov@uni-ulm.de, pavel.klinov@uni-ulm.de