CATS Solver:

The Rise of Hybrid Abduction Algorithms

ABox Abduction in DL

Given a finite set of ABox assertions Abd (*abducibles*), a DL knowledge base $\mathcal{K}$ and an ABox assertion $\mathcal{O}$ (*observation*), **ABox abduction** finds *explanations* $\mathcal{E} \subseteq$ Abd such that $\mathcal{K} \cup \mathcal{E} \models \mathcal{O}$.

Explanations should be *explanatory* ($\mathcal{K} \not\models \mathcal{O}$), *consistent* ($\mathcal{K} \cup \mathcal{E} \not\models \bot$), *relevant* ($\mathcal{E} \not\models \mathcal{O}$), and *minimal* ($\mathcal{K} \cup \mathcal{E}_1 \not\models \mathcal{O}$ for all $\mathcal{E}_1 \subsetneq \mathcal{E}$).

ABox abduction is useful in *ontology engineering*, *ontology debugging* and *repair*, *medical diagnosis*, *system diagnosis*, *multimedia interpretation*, *e-commerce*, and other areas.

$\mathcal{K}$: DentalTrauma $\sqsubseteq$ Toothache
Cavity $\sqsubseteq$ SensitiveTeeth
SensitiveTeeth $\sqcap$ DrankColdDrink $\sqsubseteq$ Toothache

$\mathcal{O}$: Toothache(john)

$\mathcal{E}_1$ = {DentalTrauma(john)}
$\mathcal{E}_2$ = {DrankColdDrink(john), Cavity(john)}
$\mathcal{E}_3$ = {DrankColdDrink(john), SensitiveTeeth(john)}

CATS (Comenius Abduction Team Solver)

CATS is an **ABox abduction solver for DLs** written in Java, using the OWL API to work with DL objects.

It offers **8 abduction algorithms:**

A. **Classic: Minimal Hitting Set, HS-Tree, RC-Tree;**
B. **Divide-and-conquer: QuickXplain, MergeXplain;**
C. **Hybrid: MHS-MXP, HST-MXP, RCT-MXP.**

It is a **vastly improved** version of the previous MHS-MXP solver, which supported only MHS and MHS-MXP.

Other changes include:

- reworked **modular architecture**, which allows to easily implement additional algorithms and combine their properties;
- algorithmic and programming **optimisations** and **bug fixes** that improve the solver's performance and memory consumption;
- new log files that track **detailed statistics** about the internal behaviour of the algorithms.

CATS supports:

- any **OWL 2 ontology** as the **background knowledge** $\mathcal{K}$,
- any **concept or role assertion** (possibly multiple ones) as the **observation** $\mathcal{O}$.

It provides **explanations** as sets of **atomic (concept and role) assertions** and their complements, involving the named individuals from $\mathcal{O}$ or from $\mathcal{K}$.

The solver allows **(optional) additional settings**:

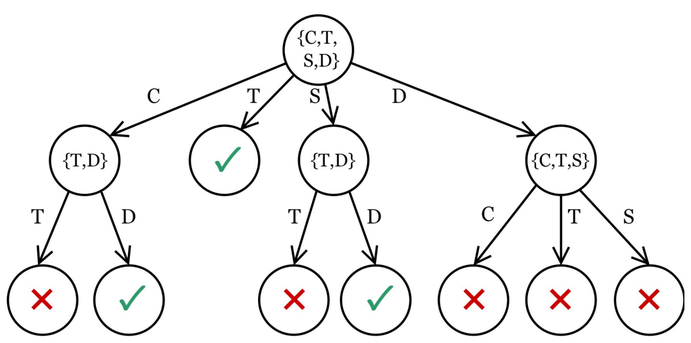
- abduction algorithm;
- timeout;
- depth limit for HS-tree exploration;
- abducibles, either as a *set of symbols* or *assertions*;
- complements in explanations toggle;
- ...

CATS can be used:

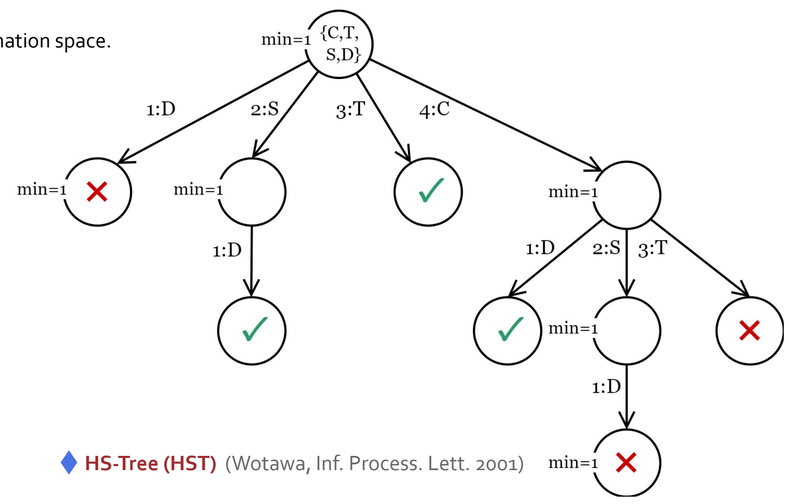
1. through a **command-line** (default);
2. via a **graphical interface** in the DL Abduction App;
3. as a **Java library** through the DL Abduction API.

A. Minimal Hitting Set (Classic) Algorithms

- use **different strategies** to build a **tree structure** that navigates through the explanation space.

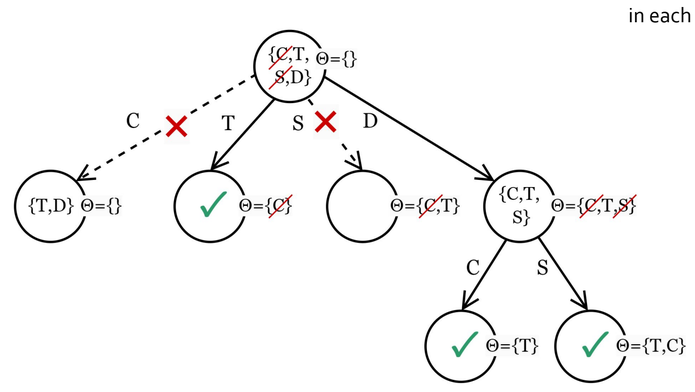


♦ **Minimal Hitting Set (MHS)** (Reiter, Artif. Intell. 1983)



♦ **HS-Tree (HST)** (Wotawa, Inf. Process. Lett. 2001)

obtaining the assertions satisfied in a model of KB ← **consistency check & model extraction** of $\mathcal{K} \cup \{\neg\mathcal{O}\} \cup$ *path* in each node



♦ **RC-Tree (RCT)** (Pill and Quaritsch, ISSRE Workshops 2015)

B. Divide-And-Conquer Algorithms

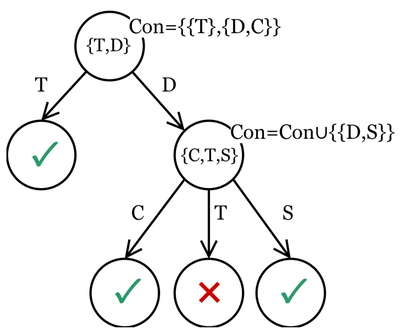
- **fast** and resource-efficient, but **incomplete**;

♦ **QuickXplain (QXP)** (Junker, AAAI 2004) finds **at most 1** explanation;
♦ **MergeXplain (MXP)** (Shchekotykhin et al., JCAI 2015) finds **all** explanations **of size 1**, & **one bigger** if at least one exists.

C. Hybrid Algorithms

- combination of classic algorithms with MergeXplain.

MergeXplain & model extraction of $\mathcal{K} \cup \{\neg\mathcal{O}\} \cup$ *path* in each node



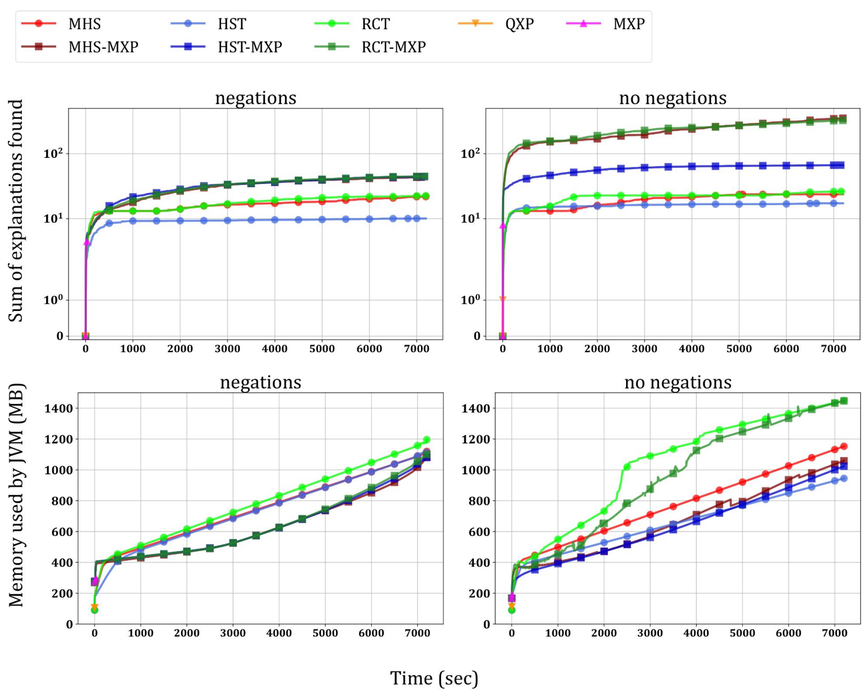
- **brand new** algorithms:
  ♦ **HST-MXP** (based on HS-Tree)
  ♦ **RCT-MXP** (based on RC-Tree)

♦ **MHS-MXP** (Homola et al., Logics in Artif. Intell. 2023)

Classics vs. Hybrids

	Classic algorithms	Hybrid algorithms
Explanations found per node	max.1	possibly **multiple**
All explanations of size x found	after xth level is finished	after **x–1th level** is finished
Can detect if all possible explanations have been found	never	**yes** (may terminate sooner)

Evaluation



Compared to their classic counterparts, **hybrid algorithms:**

- are able to find **more explanations faster** (especially in cases without negations),
- they also consume **less memory.**

In most cases, **RCT-MXP** algorithm **outperforms MHS-MXP**.

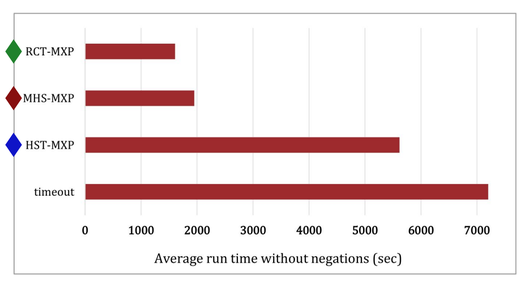
Methodology

The algorithms were tested on an abduction problem dataset used in previous evaluations:

- $\mathcal{K}$ – **LUBM** ontology,
- $\mathcal{O}$ – $A_1 \sqcap \ldots \sqcap A_n(a)$.

Each test case had two variants: **with negations** allowed in the explanations and **without negations**. Problems with negations are harder to solve, especially for hybrid algorithms.

The runs had a 2 hour (7200 seconds) time limit.



Cases with negations:

- all algorithms reached the 7200-second limit.

Cases without negations:

- the hybrids could sometimes detect that all explanations have already been found, and terminate earlier (**RCT-MXP** was the most successful, with the **shortest average runtime**).

This research was funded by the EU NextGenerationEU through the Recovery and Resilience Plan for Slovakia under the project No. 09105-03-V02-00064.

COMENIUS UNIVERSITY BRATISLAVA

Read paper

Check CATS on GitHub

Jakub Kloc, Janka Boborová, Martin Homola and Júlia Pukancová

Comenius University in Bratislava, Slovakia