# Efficient MUS Enumeration of Horn Formulae with Applications to Axiom Pinpointing

M. Fareed Arif, Carlos Mencía, **Joao Marques-Silva**

INESC-ID, IST, ULisbon, Portugal
CASL, CSI, UCD, Dublin, Ireland

SAT Conference

UT Austin, Austin, TX, USA

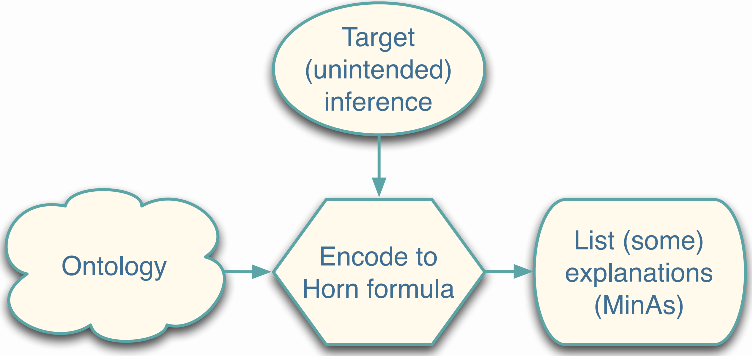
September 2015

# Why MUS enumeration of Horn formulae?

- Axiom pinpointing in lightweight description logics

# Why MUS enumeration of Horn formulae?

- Axiom pinpointing in lightweight description logics




Encode domain knowledge with DL, e.g. medical domain

**Vescovi & Sebastiani'09, EL+SAT**

Many tools: EL+SAT, ...

# Why MUS enumeration of Horn formulae? (cont.)

- Relevant ontologies can be represented with tractable DLs
  - Life sciences ontologies represented with $\mathcal{EL}^+$ [BS08]
    - E.g. SNOMED CT: Systematized Nomenclature Of MEDicine Clinical Terms $\rightsquigarrow$ 311,000 concepts

# Why MUS enumeration of Horn formulae? (cont.)

- Relevant ontologies can be represented with tractable DLs
  - Life sciences ontologies represented with $\mathcal{EL}^+$ [BS08]
    - E.g. SNOMED CT: Systematized Nomenclature Of MEDicine Clinical Terms $\rightsquigarrow$ 311,000 concepts

- Axiom pinpointing: compute minimal set of axioms (**MinAs**) that explain some (unintended) inference, e.g. subsumption relation
  - Goal is to compute many, or even all, minimal explanations

# Why MUS enumeration of Horn formulae? (cont.)

- Relevant ontologies can be represented with tractable DLs
  - Life sciences ontologies represented with $\mathcal{EL}^+$ [BS08]
    - E.g. SNOMED CT: Systematized Nomenclature Of MEDicine Clinical Terms $\rightsquigarrow$ 311,000 concepts
- Axiom pinpointing: compute minimal set of axioms (**MinAs**) that explain some (unintended) inference, e.g. subsumption relation
  - Goal is to compute many, or even all, minimal explanations
- For $\mathcal{EL}^+$, axiom pinpointing encoded with Horn formula [VS09]
- Recent work: for $\mathcal{EL}^+$, a MinA corresponds to an MUS of a Horn formula [AMMS15]

# Why MUS enumeration of Horn formulae? (cont.)

- Relevant ontologies can be represented with tractable DLs
  - Life sciences ontologies represented with $\mathcal{EL}^+$ [BS08]
    - E.g. SNOMED CT: Systematized Nomenclature Of MEDicine Clinical Terms $\rightsquigarrow$ 311,000 concepts
- Axiom pinpointing: compute minimal set of axioms (**MinAs**) that explain some (unintended) inference, e.g. subsumption relation
  - Goal is to compute many, or even all, minimal explanations
- For $\mathcal{EL}^+$, axiom pinpointing encoded with Horn formula [VS09]
- Recent work: for $\mathcal{EL}^+$, a MinA corresponds to an MUS of a Horn formula [AMMS15]
- Our goal: efficient enumeration of MUSes of Horn formulae

# Why MUS enumeration of Horn formulae? (cont.)

- A clarification:
  - There exist polynomial delay algorithms for the enumeration of MUSes of plain Horn formulae [PS10]

# Why MUS enumeration of Horn formulae? (cont.)

- A clarification:
  - There exist polynomial delay algorithms for the enumeration of MUSes of plain Horn formulae [PS10]
  - Axiom pinpointing corresponds to group MUS enumeration of Horn formulae:
    - Group 0 ($G_0$): Set of Horn clauses representing background knowledge, e.g. encode classification of the $\mathcal{EL}^+$ ontology; ignored for the purpose of reporting MUSes
    - Additional groups, each with a single unit clause, representing one axiom

# Why MUS enumeration of Horn formulae? (cont.)

- A clarification:
  - There exist polynomial delay algorithms for the enumeration of MUSes of plain Horn formulae [PS10]
  - Axiom pinpointing corresponds to group MUS enumeration of Horn formulae:
    - Group 0 ($G_0$): Set of Horn clauses representing background knowledge, e.g. encode classification of the $\mathcal{EL}^+$ ontology; ignored for the purpose of reporting MUSes
    - Additional groups, each with a single unit clause, representing one axiom
    - Enumeration of MUSes of group Horn formulae is not output polynomial unless P = NP [BPS07]

# Outline

# Outline

# Description logic $\mathcal{EL}^+$

- $N_C$, $N_R$ denote concept & role names, respectively
- Concept descriptions formed using 3 constructors below
- Ontology $\mathcal{T}$ is a finite set of GCI's and RI's

| | Syntax | Semantics |
|---|---|---|
| top | $\top$ | $\Delta^{\mathcal{I}}$ |
| conjunction | $X \sqcap Y$ | $X^{\mathcal{I}} \cap Y^{\mathcal{I}}$ |
| existential restriction | $\exists r.X$ | $\{x \in \Delta^{\mathcal{I}} \mid \exists y \in \Delta^{\mathcal{I}} : (x,y) \in r^{\mathcal{I}} \wedge y \in X^{\mathcal{I}}\}$ |
| general concept inclusion | $X \sqsubseteq Y$ | $X^{\mathcal{I}} \subseteq Y^{\mathcal{I}}$ |
| role inclusion | $r_1 \circ \cdots \circ r_n \sqsubseteq s$ | $r_1^{\mathcal{I}} \circ \cdots \circ r_n^{\mathcal{I}} \subseteq s^{\mathcal{I}}$ |

# Description logic $\mathcal{EL}^+$

- $N_C$, $N_R$ denote concept & role names, respectively
- Concept descriptions formed using 3 constructors below
- Ontology $\mathcal{T}$ is a finite set of GCI's and RI's
- Interpretation: $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$, where
  - $\Delta^{\mathcal{I}}$ is non-empty set of individuals, and
  - $\cdot^{\mathcal{I}}$ maps
    - Each $C \in N_C$ to $C^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$
    - Each $r \in N_R$ to a binary relation $r^{\mathcal{I}}$ in $\Delta^{\mathcal{I}}$
  - $\cdot^{\mathcal{I}}$ is defined inductively for arbitrary concept descriptions:

| | Syntax | Semantics |
|---|---|---|
| top | $\top$ | $\Delta^{\mathcal{I}}$ |
| conjunction | $X \sqcap Y$ | $X^{\mathcal{I}} \cap Y^{\mathcal{I}}$ |
| existential restriction | $\exists r.X$ | $\{x \in \Delta^{\mathcal{I}} \mid \exists y \in \Delta^{\mathcal{I}} : (x,y) \in r^{\mathcal{I}} \wedge y \in X^{\mathcal{I}}\}$ |
| general concept inclusion | $X \sqsubseteq Y$ | $X^{\mathcal{I}} \subseteq Y^{\mathcal{I}}$ |
| role inclusion | $r_1 \circ \cdots \circ r_n \sqsubseteq s$ | $r_1^{\mathcal{I}} \circ \cdots \circ r_n^{\mathcal{I}} \subseteq s^{\mathcal{I}}$ |

# Description logic $\mathcal{EL}^+$ (cont.)

- $\mathcal{I}$ is a model of $\mathcal{T}$ if semantics conditions are satisfied for every concept inclusion (GCI or RI)

# Description logic $\mathcal{EL}^+$ (cont.)

- $\mathcal{I}$ is a model of $\mathcal{T}$ if semantics conditions are satisfied for every concept inclusion (GCI or RI)

- Subsumption: $C$ is subsumed w.r.t. $D$, $C \sqsubseteq_{\mathcal{T}} D$, if $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$ for every model $\mathcal{I}$ of $\mathcal{T}$

# Description logic $\mathcal{EL}^+$ (cont.)

- $\mathcal{I}$ is a model of $\mathcal{T}$ if semantics conditions are satisfied for every concept inclusion (GCI or RI)
- Subsumption: $C$ is subsumed w.r.t. $D$, $C \sqsubseteq_{\mathcal{T}} D$, if $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$ for every model $\mathcal{I}$ of $\mathcal{T}$
- Classification: infer all subsumption relations between atomic concepts
  - **Note**: $\text{EL}^+\text{SAT}$ encodes classification of $\mathcal{T}$ as Horn formula

# Description logic $\mathcal{EL}^+$ (cont.)

- $\mathcal{I}$ is a model of $\mathcal{T}$ if semantics conditions are satisfied for every concept inclusion (GCI or RI)
- Subsumption: $C$ is subsumed w.r.t. $D$, $C \sqsubseteq_{\mathcal{T}} D$, if $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$ for every model $\mathcal{I}$ of $\mathcal{T}$
- Classification: infer all subsumption relations between atomic concepts
  - **Note**: $\text{EL}^+\text{SAT}$ encodes classification of $\mathcal{T}$ as Horn formula
- MinA: minimal set of axioms from which some subsumption relation can be inferred

# Example of medical ontology

$$
\begin{aligned}
&\mathsf{Endocarditis} \sqsubseteq \mathsf{Inflammation} \sqcap \exists \mathsf{hasLoc}.\mathsf{Endocardium},\\
&\mathsf{Inflammation} \sqsubseteq \mathsf{Disease} \sqcap \exists \mathsf{actsOn}.\mathsf{Tissue},\\
&\mathsf{Endocardium} \sqsubseteq \mathsf{Tissue} \sqcap \exists \mathsf{contIn}.\mathsf{HeartValve},\\
&\mathsf{HeartValve} \sqsubseteq \exists \mathsf{contIn}.\mathsf{Heart},\\
&\mathsf{HeartDisease} \equiv \mathsf{Disease} \sqcap \exists \mathsf{hasLoc}.\mathsf{Heart},\\
&\mathsf{contIn} \circ \mathsf{contIn} \sqsubseteq \mathsf{contIn},\\
&\mathsf{hasLoc} \circ \mathsf{contIn} \sqsubseteq \mathsf{hasLoc}
\end{aligned}
$$

- MinA example:
  - $\mathsf{Endocarditis} \sqsubseteq \mathsf{HeartDisease}$ explained by axioms above
- Unintended subsumption in (old version of) SNOMED CT: $\mathsf{AmputationOfFinger} \sqsubseteq_{\mathcal{O}} \mathsf{AmputationOfHand}$

[BS08]

# Context for this work

- EL$^+$SAT state of the art until 2014 [VS09,VS15]

# Context for this work

- EL$^+$SAT state of the art until 2014 [VS09,VS15]
- EL2MCS: our first effort based on **explicit** hitting set dualization
  - Compute **all** MCSes
  - Use hitting set dualization to enumerate MUSes [R87,BL03,BS05,LS08]

# Context for this work

- EL$^+$SAT state of the art until 2014 [VS09,VS15]
- EL2MCS: our first effort based on **explicit** hitting set dualization
  - Compute **all** MCSes
  - Use hitting set dualization to enumerate MUSes [R87,BL03,BS05,LS08]
- In practice, **EL2MCS extensively outperforms EL$^+$SAT** [AMMS15]
  - But, there can be **exponentially many** MCSes
    - It may even be infeasible to start enumeration of MUSes **!**
  - And goal is to enumerate many (or all) MUSes

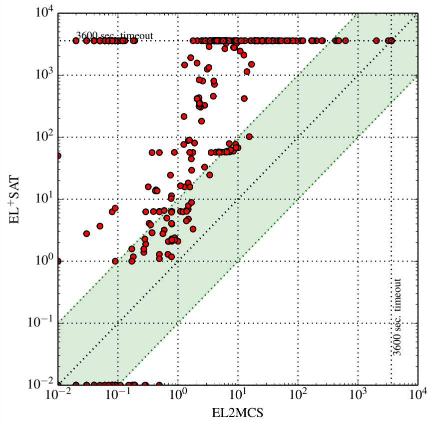
## Context for this work

- $EL^+SAT$ state of the art until 2014 [VS09,VS15]
- EL2MCS: our first effort based on **explicit** hitting set dualization
  - Compute **all** MCSes
  - Use hitting set dualization to enumerate MUSes [R87,BL03,BS05,LS08]
- In practice, **EL2MCS extensively outperforms $EL^+SAT$** [AMMS15]
  - But, there can be **exponentially many** MCSes
    - It may even be infeasible to start enumeration of MUSes **!**
  - And goal is to enumerate many (or all) MUSes
- Alternative is **implicit** hitting set dualization
  - Recently proposed in eMUS / MARCO [PMS13,LM13,LPMMS15]
  - **But**, apparent similarities between $EL^+SAT$ and eMUS / MARCO

# Context for this work

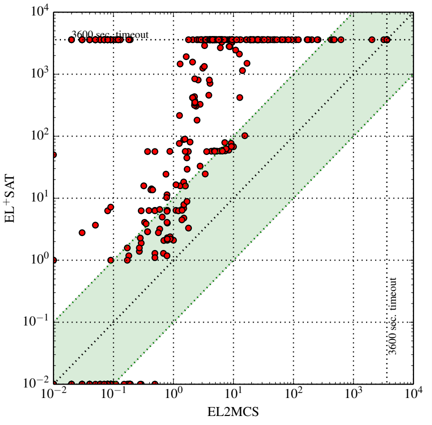
- EL$^+$SAT state of the art until 2014 [VS09,VS15]
- EL2MCS: our first effort based on **explicit** hitting set dualization
  - Compute **all** MCSes
  - Use hitting set dualization to enumerate MUSes [R87,BL03,BS05,LS08]
- In practice, **EL2MCS extensively outperforms EL$^+$SAT** [AMMS15]
  - But, there can be **exponentially many** MCSes
    - It may even be infeasible to start enumeration of MUSes **!**
  - And goal is to enumerate many (or all) MUSes
- Alternative is **implicit** hitting set dualization
  - Recently proposed in eMUS / MARCO [PMS13,LM13,LPMMS15]
  - **But**, apparent similarities between EL$^+$SAT and eMUS / MARCO
  - So, **how come EL2MCS extensively outperforms EL$^+$SAT** ?

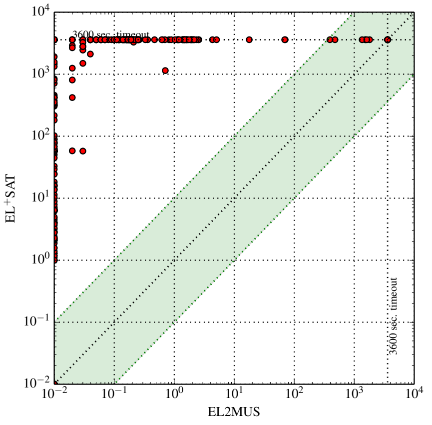
# Context for this work (cont.) – where we started...




- **Note**: results on 500 COI instances (more later)

# Context for this work (cont.) – and where we got!




- **Note**: results on 500 COI instances (more later)

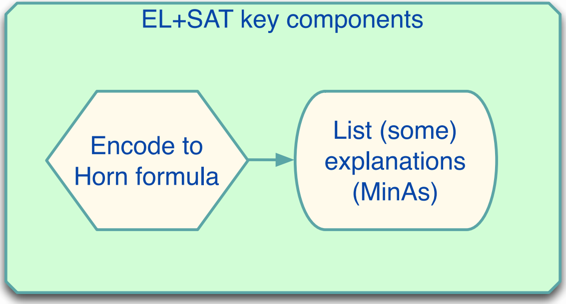
# Outline

# What is $EL^+SAT$?

- Two main components: Horn encoder & MinA enumerator

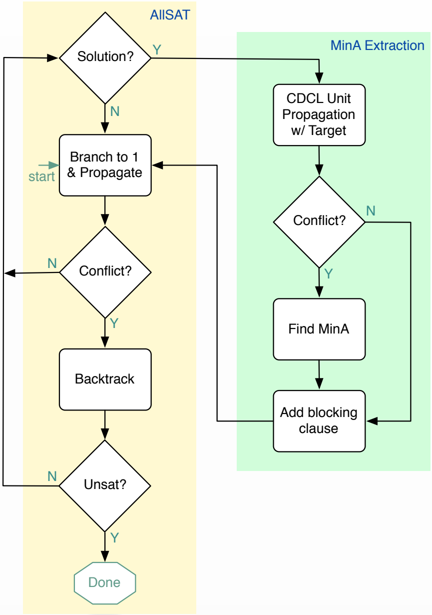
# What is $EL^+SAT$?

- Two main components: Horn encoder & MinA enumerator



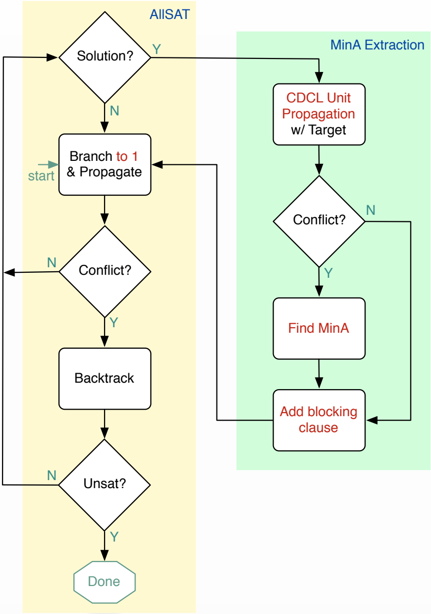
# Organization of EL$^{+}$SAT MinA enumerator

- Enumeration of (maximal) models inspired on AllSMT
- Always assign variables to 1
- Blocking based on decision variables in both cases



# Organization of EL$^+$SAT MinA enumerator

- Enumeration of (maximal) models inspired on AllSMT
- Always assign variables to 1
- Blocking based on decision variables in both cases
- So, **how come EL2MCS extensively outperforms EL$^+$SAT**?



# Performance issues with $EL^+SAT$

- Maximal models computed by assigning decision variables to 1
  - Corresponds to SAT with preferences [GM06,RGM10]
  - We can do better: use MCS extraction [MSHJPB13,MPMS15]

# Performance issues with EL$^+$SAT

- Maximal models computed by assigning decision variables to 1
  - Corresponds to SAT with preferences [GM06,RGM10]
  - We can do better: use MCS extraction [MSHJPB13,MPMS15]
- Horn formulae decided by unit propagation with CDCL SAT solver
  - In MiniSat, unit propagation takes worst-case quadratic time, due to implementation of watched literals [G13]
  - Also, unnecessary propagation of 0-valued variables [DG84,M88]
  - We can do better: use dedicated LTUR algorithm [M88]

# Performance issues with EL$^+$SAT

- Maximal models computed by assigning decision variables to 1
  - Corresponds to SAT with preferences [GM06,RGM10]
  - We can do better: use MCS extraction [MSHJPB13,MPMS15]
- Horn formulae decided by unit propagation with CDCL SAT solver
  - In MiniSat, unit propagation takes worst-case quadratic time, due to implementation of watched literals [G13]
  - Also, unnecessary propagation of 0-valued variables [DG84,M88]
  - We can do better: use dedicated LTUR algorithm [M88]
- Computing each MinA corresponds to deletion-based MUS extraction
  - We can do better: use **insertion**-based MUS extraction (more later)

# Performance issues with $EL^+SAT$

- Maximal models computed by assigning decision variables to 1
  - Corresponds to SAT with preferences [GM06,RGM10]
  - We can do better: use MCS extraction [MSHJPB13,MPMS15]
- Horn formulae decided by unit propagation with CDCL SAT solver
  - In MiniSat, unit propagation takes worst-case quadratic time, due to implementation of watched literals [G13]
  - Also, unnecessary propagation of 0-valued variables [DG84,M88]
  - We can do better: use dedicated LTUR algorithm [M88]
- Computing each MinA corresponds to deletion-based MUS extraction
  - We can do better: use **insertion**-based MUS extraction (more later)
- Blocking of MCSes/MSSes does **not** eliminate supersets of MCSes (or subsets of MSSes)
  - **Why?** Clause learning **only** uses decision variables; learned clauses only contain **negative** literals; supersets of MCSes **not** blocked

# Outline

# Key ideas in HgMUS

# Key ideas in HgMUS

1. Partial MUS enumeration paradigm [PMS13,LM13,LPMMS15]
   - Implements implicit hitting set dualization

# Key ideas in HgMUS

1. Partial MUS enumeration paradigm [PMS13,LM13,LPMMS15]
   - Implements implicit hitting set dualization

2. Blocking of MUSes & MCSes

# Key ideas in HgMUS

1. Partial MUS enumeration paradigm [PMS13,LM13,LPMMS15]
   - Implements implicit hitting set dualization

2. Blocking of MUSes & MCSes

3. Optimized enumeration of maximal models
   - Builds on MCS extraction work – MCSls & LBX [MSHJPB13,MPMS15]

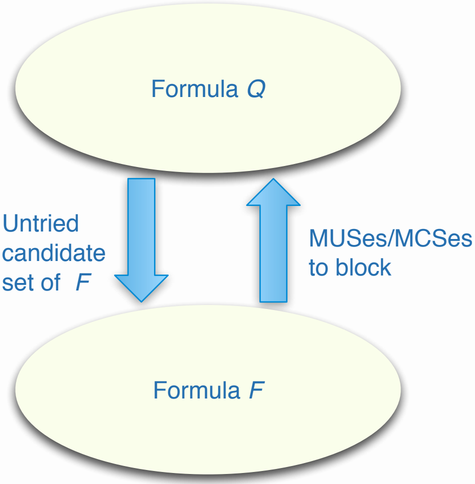
# Key ideas in HgMUS

1. Partial MUS enumeration paradigm [PMS13,LM13,LPMMS15]
   – Implements implicit hitting set dualization
2. Blocking of MUSes & MCSes
3. Optimized enumeration of maximal models
   – Builds on MCS extraction work – MCSls & LBX [MSHJPB13,MPMS15]
4. Dedicated Horn decision procedure – LTUR [M88]

# Key ideas in HgMUS

1. Partial MUS enumeration paradigm [PMS13,LM13,LPMMS15]
   - Implements implicit hitting set dualization
2. Blocking of MUSes & MCSes
3. Optimized enumeration of maximal models
   - Builds on MCS extraction work – MCSls & LBX [MSHJPB13,MPMS15]
4. Dedicated Horn decision procedure – LTUR [M88]
5. Dedicated MUS extraction algorithm for Horn formulae

# Partial MUS enumeration – eMUS / MARCO



# Partial MUS enumeration – eMUS / MARCO (cont.)

**Input**: CNF formula $F$
**Output**: Reports the set of MUSes of $F$

```
I ← {p_i | c_i ∈ F}                          # Variable p_i picks clause c_i
Q ← ∅
while true do
    (st, P) ← MaximalModel(Q)                # Hit MCSes w/o repeating MUSes heig
    if not st then return
    F′ ← {c_i | p_i ∈ P}                      # Pick selected clauses
    M ← ComputeMUS(F′)
    ReportMUS(M)
    b ← {¬p_i | c_i ∈ M}                      # Block computed MUS
    b ← {p_i | p_i ∈ I \ P}                   # Block computed MCS/MSS
    Q ← Q ∪ {b}
```

# MUS/MCS blocking

**Input**: CNF formula $F$
**Output**: Reports the set of MUSes of $F$

$I \leftarrow \{p_i \mid c_i \in F\}$ `# Variable` $p_i$ `picks clause` $c_i$
$Q \leftarrow \emptyset$
**while** true **do**
  $(st, P) \leftarrow \text{MaximalModel}(Q)$ `# Hit MCSes w/o repeating MUSes` heig
  **if not** $st$ **then return**
  $F' \leftarrow \{c_i \mid p_i \in P\}$ `# Pick selected clauses`
  $M \leftarrow \text{ComputeMUS}(F')$
  $\text{ReportMUS}(M)$
  $b \leftarrow \{\neg p_i \mid c_i \in M\}$ `# Negative clause blocking MUS`
  $b \leftarrow \{p_i \mid p_i \in I \setminus P\}$ `# Positive clause blocking MCS/MSS`
  $Q \leftarrow Q \cup \{b\}$

# An example with eMUS / MARCO

$G_0 = \{(\neg a \vee \neg b), (b), (\neg c \vee \neg d \vee \neg e)\}$
$G_1 = \{(a)\}$
$G_2 = \{(c)\}$
$G_3 = \{(d)\}$
$G_4 = \{(e)\}$

# An example with eMUS / MARCO

$G_0 = \{(\neg a \vee \neg b), (b), (\neg c \vee \neg d \vee \neg e)\}$
$G_1 = \{(a)\}$
$G_2 = \{(c)\}$
$G_3 = \{(d)\}$
$G_4 = \{(e)\}$

$\text{MUS} = \{\{(a)\}, \{(c), (d), (e)\}\}$
$\text{MCS} = \{\{(a), (c)\}, \{(a), (d)\}, \{(a), (e)\}\}$

# An example with eMUS / MARCO

$G_0 = \{(\neg a \vee \neg b), (b), (\neg c \vee \neg d \vee \neg e)\}$
$G_1 = \{(a)\}$
$G_2 = \{(c)\}$
$G_3 = \{(d)\}$
$G_4 = \{(e)\}$

$\mathrm{MUS} = \{\{(a)\}, \{(c), (d), (e)\}\}$
$\mathrm{MCS} = \{\{(a), (c)\}, \{(a), (d)\}, \{(a), (e)\}\}$

- First, compute the two MUSes
  - Blocking clauses: $(\neg p_1)$, $(\neg p_2 \vee \neg p_3 \vee \neg p_4)$

# An example with eMUS / MARCO

$G_0 = \{(\neg a \vee \neg b), (b), (\neg c \vee \neg d \vee \neg e)\}$
$G_1 = \{(a)\}$
$G_2 = \{(c)\}$
$G_3 = \{(d)\}$
$G_4 = \{(e)\}$

$\text{MUS} = \{\{(a)\}, \{(c), (d), (e)\}\}$
$\text{MCS} = \{\{(a), (c)\}, \{(a), (d)\}, \{(a), (e)\}\}$

- First, compute the two MUSes
  - Blocking clauses: $(\neg p_1)$, $(\neg p_2 \vee \neg p_3 \vee \neg p_4)$
- Then, compute MCS $\#1$:
  - Pick $p_3 = p_4 = 1$, i.e. clauses $(d)$ and $(e)$ must be satisfied
  - MCS is $\{(a), (c)\}$

# An example with eMUS / MARCO

$G_0 = \{(\neg a \vee \neg b), (b), (\neg c \vee \neg d \vee \neg e)\}$
$G_1 = \{(a)\}$
$G_2 = \{(c)\}$
$G_3 = \{(d)\}$
$G_4 = \{(e)\}$

$\text{MUS} = \{\{(a)\}, \{(c), (d), (e)\}\}$
$\text{MCS} = \{\{(a), (c)\}, \{(a), (d)\}, \{(a), (e)\}\}$

- First, compute the two MUSes
  - Blocking clauses: $(\neg p_1)$, $(\neg p_2 \vee \neg p_3 \vee \neg p_4)$
- Then, compute MCS #1:
  - Pick $p_3 = p_4 = 1$, i.e. clauses $(d)$ and $(e)$ must be satisfied
  - MCS is $\{(a), (c)\}$
- How to block MCS #1?

# An example with eMUS / MARCO

$G_0 = \{(\neg a \vee \neg b), (b), (\neg c \vee \neg d \vee \neg e)\}$
$G_1 = \{(a)\}$
$G_2 = \{(c)\}$
$G_3 = \{(d)\}$
$G_4 = \{(e)\}$

$\text{MUS} = \{\{(a)\}, \{(c), (d), (e)\}\}$
$\text{MCS} = \{\{(a), (c)\}, \{(a), (d)\}, \{(a), (e)\}\}$

- First, compute the two MUSes
  - Blocking clauses: $(\neg p_1)$, $(\neg p_2 \vee \neg p_3 \vee \neg p_4)$
- Then, compute MCS #1:
  - Pick $p_3 = p_4 = 1$, i.e. clauses $(d)$ and $(e)$ must be satisfied
  - MCS is $\{(a), (c)\}$
- How to block MCS #1?
  - Blocking clause $(\neg p_3 \vee \neg p_4)$

[VS09,VS15]

# An example with eMUS / MARCO

$G_0 = \{(\neg a \vee \neg b), (b), (\neg c \vee \neg d \vee \neg e)\}$
$G_1 = \{(a)\}$
$G_2 = \{(c)\}$
$G_3 = \{(d)\}$
$G_4 = \{(e)\}$

$\text{MUS} = \{\{(a)\}, \{(c), (d), (e)\}\}$
$\text{MCS} = \{\{(a), (c)\}, \{(a), (d)\}, \{(a), (e)\}\}$

- First, compute the two MUSes
  - Blocking clauses: $(\neg p_1)$, $(\neg p_2 \vee \neg p_3 \vee \neg p_4)$
- Then, compute MCS #1:
  - Pick $p_3 = p_4 = 1$, i.e. clauses $(d)$ and $(e)$ must be satisfied
  - MCS is $\{(a), (c)\}$
- How to block MCS #1?
  - Blocking clause $(\neg p_3 \vee \neg p_4)$ [VS09,VS15]
    - There is another maximal model with $p_3 = 0$, i.e. pick $p_4 = 1$
    - There is another maximal model with $p_4 = 0$, i.e. pick $p_3 = 1$

# An example with eMUS / MARCO

$G_0 = \{(\neg a \vee \neg b), (b), (\neg c \vee \neg d \vee \neg e)\}$
$G_1 = \{(a)\}$
$G_2 = \{(c)\}$
$G_3 = \{(d)\}$
$G_4 = \{(e)\}$

$\text{MUS} = \{\{(a)\}, \{(c), (d), (e)\}\}$
$\text{MCS} = \{\{(a), (c)\}, \{(a), (d)\}, \{(a), (e)\}\}$

- First, compute the two MUSes
  - Blocking clauses: $(\neg p_1)$, $(\neg p_2 \vee \neg p_3 \vee \neg p_4)$
- Then, compute MCS #1:
  - Pick $p_3 = p_4 = 1$, i.e. clauses $(d)$ and $(e)$ must be satisfied
  - MCS is $\{(a), (c)\}$
- How to block MCS #1?
  - Blocking clause $(\neg p_3 \vee \neg p_4)$ [VS09,VS15]
    - There is another maximal model with $p_3 = 0$, i.e. pick $p_4 = 1$
    - There is another maximal model with $p_4 = 0$, i.e. pick $p_3 = 1$
  - **Alternative**: blocking clause $(p_1 \vee p_2)$ [PMS13,LM13,LPMMS15]

# An example with eMUS / MARCO (cont.)

$G_0 = \{(\neg a \vee \neg b), (b), (\neg c \vee \neg d \vee \neg e)\}$
$G_1 = \{(a)\}$
$G_2 = \{(c)\}$
$G_3 = \{(d)\}$
$G_4 = \{(e)\}$

$\text{MUS} = \{\{(a)\}, \{(c), (d), (e)\}\}$
$\text{MCS} = \{\{(a), (c)\}, \{(a), (d)\}, \{(a), (e)\}\}$

- Blocking with negative clauses [VS09,VS15]

| $Q$ | MxM model | MUS | MCS | Blocking clause | OK? |
|---|---|---|---|---|---|
| $\emptyset$ | $p_1 = \ldots = 1$ | $\{G_1\}$ | | $B_1 = (\neg p_1)$ | ✓ |
| $\{B_1\}$ | $p_2 = \ldots = 1$ | $\{G_2, G_3, G_4\}$ | | $B_2 = (\neg p_2 \vee \neg p_3 \vee \neg p_4)$ | ✓ |
| $\{B_1, B_2\}$ | $p_3 = p_4 = 1$ | | $\{G_1, G_2\}$ | $B_3 = (\neg p_3 \vee \neg p_4)$ | ✓ |
| $\{B_1, B_2, B_3\}$ | $p_2 = p_3 = 1$ | | $\{G_1, G_4\}$ | $B_4 = (\neg p_2 \vee \neg p_3)$ | ✓ |
| $\{B_1, \ldots, B_4\}$ | $p_2 = p_4 = 1$ | | $\{G_1, G_3\}$ | $B_5 = (\neg p_2 \vee \neg p_4)$ | ✓ |
| $\{B_1, \ldots, B_5\}$ | $p_3 = 1$ | | $\{G_1, G_2, G_4\}$ | $B_6 = (\neg p_3)$ | ✗ |
| $\{B_1, \ldots, B_6\}$ | $p_4 = 1$ | | $\{G_1, G_2, G_3\}$ | $B_7 = (\neg p_4)$ | ✗ |
| $\{B_1, \ldots, B_7\}$ | $p_1 = \ldots = 0$ | | | | |

# An example with eMUS / MARCO (cont.)

$G_0 = \{(\neg a \vee \neg b), (b), (\neg c \vee \neg d \vee \neg e)\}$
$G_1 = \{(a)\}$
$G_2 = \{(c)\}$
$G_3 = \{(d)\}$
$G_4 = \{(e)\}$

$\text{MUS} = \{\{(a)\}, \{(c), (d), (e)\}\}$
$\text{MCS} = \{\{(a), (c)\}, \{(a), (d)\}, \{(a), (e)\}\}$

- Distinct blocking of MUSes and MCSes [PMS13,LM13,LPMMS13]

| $Q$ | MxM model | MUS | MCS | Blocking clause | OK? |
|---|---|---|---|---|---|
| $\emptyset$ | $p_1 = \ldots = 1$ | $\{G_1\}$ | | $B_1 = (\neg p_1)$ | ✓ |
| $\{B_1\}$ | $p_2 = \ldots = 1$ | $\{G_2, G_3, G_4\}$ | | $B_2 = (\neg p_2 \vee \neg p_3 \vee \neg p_4)$ | ✓ |
| $\{B_1, B_2\}$ | $p_3 = p_4 = 1$ | | $\{G_1, G_2\}$ | $B_3 = (p_1 \vee p_2)$ | ✓ |
| $\{B_1, B_2, B_3\}$ | $p_2 = p_3 = 1$ | | $\{G_1, G_4\}$ | $B_4 = (p_1 \vee p_4)$ | ✓ |
| $\{B_1, \ldots, B_4\}$ | $p_2 = p_4 = 1$ | | $\{G_1, G_3\}$ | $B_5 = (p_1 \vee p_3)$ | ✓ |
| $\{B_1, \ldots, B_5\}$ | $\emptyset$ | | $\{G_1, G_2, G_3\}$ | | |

# Computing maximal models

**Input**: $Q$ a CNF formula
**Output**: $(st, P)$: with $st$ a Boolean and $P$ a MxM (if it exists)
 # $P$: under-approximation of maximal model
 # $U$: remaining target set of literals
 # $B$: backbone literals
$(P, U, B) \leftarrow (\{\{x\} \mid \neg x \notin L(Q)\}, \{\{x\} \mid \neg x \in L(Q)\}, \emptyset)$
$(st, P, U) \leftarrow \text{InitialAssignment}(Q \cup P)$
**if not** $st$ **then return** (false, $\emptyset$)
**while** $U \neq \emptyset$ **do**
  $l \leftarrow \text{SelectLiteral}(U)$ heig
  $(st, \mu) = \text{SAT}(Q \cup P \cup B \cup \{l\})$
  ( # Else, update $B$ with backbone literal from $U$)
  $(U, B) \leftarrow (U \setminus \{l\}, B \cup \{\neg l\})$
**return** (true, $P$) # P is a MxM of $Q$

# LTUR

- Simplified implementation of Dowling&Gallier's algorithm [DG84,M88]
- Variables assigned value 0, by default
- Can be viewed as one-sided unit propagation, i.e. propagate only variables that must be assigned value 1
- There is no branching
- Either no conflicts or a single conflict
- Data structures:
  - Use adjacency lists (for negative literals)
  - Use clause counters for the negative literals
- Run time: $\mathcal{O}(||F||)$
- **And**, **LTUR can be used incrementally**

# MUS extraction in Horn formulae

- **Deletion**-based:
  - $\mathcal{O}(|F|)$ calls to SAT oracle
  - For Horn formulae, with LTUR: $\mathcal{O}(|F| \cdot ||F||)$

# MUS extraction in Horn formulae

- **Deletion**-based:
  - $\mathcal{O}(|F|)$ calls to SAT oracle
  - For Horn formulae, with LTUR: $\mathcal{O}(|F| \cdot ||F||)$
- **Insertion**-based:
  - $\mathcal{O}(|F| \cdot |M|)$ calls to SAT oracle
  - For Horn formulae, with LTUR: $\mathcal{O}(|F| \cdot |M| \cdot ||F||)$?

# MUS extraction in Horn formulae

- **Deletion**-based:
  - $\mathcal{O}(|F|)$ calls to SAT oracle
  - For Horn formulae, with LTUR: $\mathcal{O}(|F| \cdot ||F||)$

- **Insertion**-based:
  - $\mathcal{O}(|F| \cdot |M|)$ calls to SAT oracle
  - For Horn formulae, with LTUR: $\mathcal{O}(|F| \cdot |M| \cdot ||F||)$?

- But, LTUR can be used incrementally!
  - Iterative non-conflicting calls to SAT solver have **amortized** $\mathcal{O}(||F||)$ run time
  - For Horn formulae, with LTUR: $\mathcal{O}(|M| \cdot ||F||)$

# MUS extraction in Horn formulae

- **Deletion**-based:
  - $\mathcal{O}(|F|)$ calls to SAT oracle
  - For Horn formulae, with LTUR: $\mathcal{O}(|F| \cdot ||F||)$
- **Insertion**-based:
  - $\mathcal{O}(|F| \cdot |M|)$ calls to SAT oracle
  - For Horn formulae, with LTUR: $\mathcal{O}(|F| \cdot |M| \cdot ||F||)$?
- But, LTUR can be used incrementally!
  - Iterative non-conflicting calls to SAT solver have **amortized** $\mathcal{O}(||F||)$ run time
  - For Horn formulae, with LTUR: $\mathcal{O}(|M| \cdot ||F||)$
- Thus, for Horn formulae, with an incremental decision procedure, insertion-based is preferable to deletion-based**!**

# MUS extraction in Horn formulae

- **Deletion**-based:
  - $\mathcal{O}(|F|)$ calls to SAT oracle
  - For Horn formulae, with LTUR: $\mathcal{O}(|F| \cdot ||F||)$
- **Insertion**-based:
  - $\mathcal{O}(|F| \cdot |M|)$ calls to SAT oracle
  - For Horn formulae, with LTUR: $\mathcal{O}(|F| \cdot |M| \cdot ||F||)$?
- But, LTUR can be used incrementally!
  - Iterative non-conflicting calls to SAT solver have **amortized** $\mathcal{O}(||F||)$ run time
  - For Horn formulae, with LTUR: $\mathcal{O}(|M| \cdot ||F||)$
- Thus, for Horn formulae, with an incremental decision procedure, insertion-based is preferable to deletion-based**!**
  - Also, QuickXplain/Progression cannot improve over **insertion**-based

# Insertion-based MUS extraction

**Input**: $H$ denotes the $G_0$ clauses; $I$ denotes the set of (individual) group clauses
**Output**: $M$ denotes the computed MUS

```
# LTUR_prop / LTUR_undo:  incremental LTUR propagation / undo
(M, c_r) ← (H, 0)
LTUR_prop(M, M)                              # Start by propagating G_0 clauses
while true do
    if c_r > 0 then
        M ← M ∪ {c_r}                        # Add transition clause c_r to M
        if not LTUR_prop(M, {c_r}) then
            LTUR_undo(M, M)
            return M \ H                     # Remove G_0 clauses from computed MUS
    S ← ∅
    while true do
        c_r ← SelectRemoveClause(I)          # Target transition clause
        S ← S ∪ {c_r}
        if not LTUR_prop(M ∪ S, {c_r}) then
            I ← S \ {c_r}                    # Update working set of groups
            LTUR_undo(M, S)
            break                            # c_r represents a transition clause
```

# Outline

# Experimental setup

- Medical ontologies:
  - GALEN, with two variants: FULL-GALEN and NOT-GALEN
  - GENE
  - NCI
  - SNOMED CT

# Experimental setup

- Medical ontologies:
  - GALEN, with two variants: FULL-GALEN and NOT-GALEN
  - GENE
  - NCI
  - SNOMED CT – **large size!**

# Experimental setup

- Medical ontologies:
  - GALEN, with two variants: FULL-GALEN and NOT-GALEN
  - GENE
  - NCI
  - SNOMED CT – **large size!**

- Instance generation for each medical ontology:
  - 50 random
  - 50 sorted – expected to have large number of minimal explanations
  - **Total**: **500** instances

# Experimental setup

- Medical ontologies:
  - GALEN, with two variants: FULL-GALEN and NOT-GALEN
  - GENE
  - NCI
  - SNOMED CT – **large size!**
- Instance generation for each medical ontology:
  - 50 random
  - 50 sorted – expected to have large number of minimal explanations
  - **Total**: **500** instances
- Instance simplifications: [VS09,VS15]
  - COI: Cone-of-influence reduction
  - x2: apply COI, re-encode to ontology and encode to Horn formula

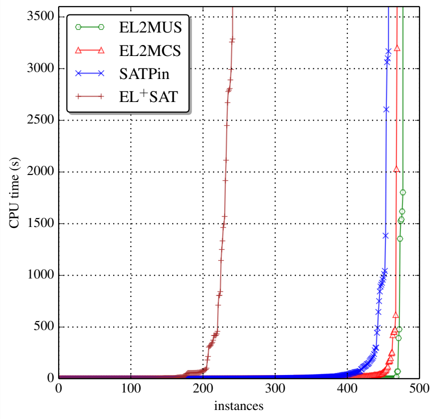
# Experimental setup

- Medical ontologies:
  - GALEN, with two variants: FULL-GALEN and NOT-GALEN
  - GENE
  - NCI
  - SNOMED CT – **large size!**
- Instance generation for each medical ontology:
  - 50 random
  - 50 sorted – expected to have large number of minimal explanations
  - **Total**: **500** instances
- Instance simplifications: [VS09,VS15]
  - COI: Cone-of-influence reduction
  - x2: apply COI, re-encode to ontology and encode to Horn formula
- Compute cluster, with 3600s time limit and 4 GByte memory limit

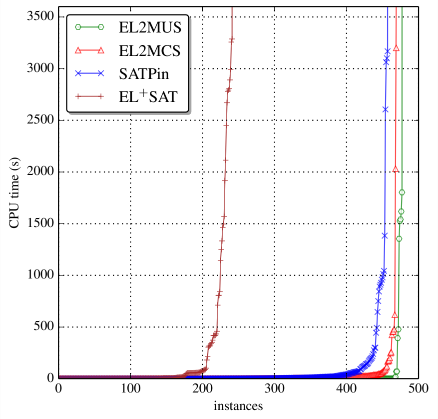
# Experimental setup – goal & tools

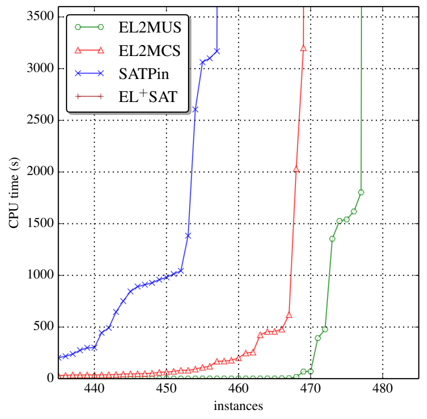
- Goal:
  - Compute **all** MinAs/MUSes or as many MinAs/MUSes as possible within the given timeout
- Our tool:
  - **EL2MUS**: uses $\text{EL}^{+}$SAT encoder as front-end to **HgMUS**
- Other SAT-based tools:
  - $\text{EL}^{+}$SAT: developed in 2009, updated in 2014, 2015 [VS09,VS15]
  - EL2MCS: developed in 2015 [AMMS15]
  - SATPin: developed in 2015 [MP15]
- Other (non SAT-based) tools:
  - CEL – only computes 10 MinAs, developed in 2006 [BLS06]
  - JUST – cannot handle all $\mathcal{EL}^{+}$ constructs, developed in 2014 [L14]

# Cactus plots – COI instances

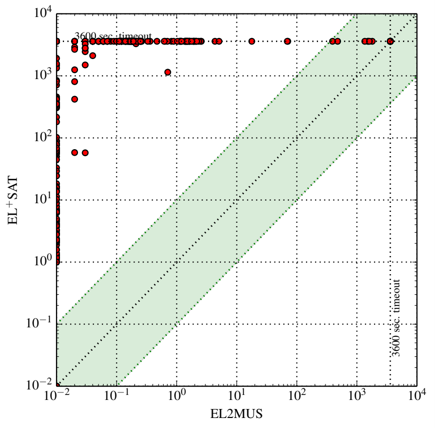


# Cactus plots – COI instances

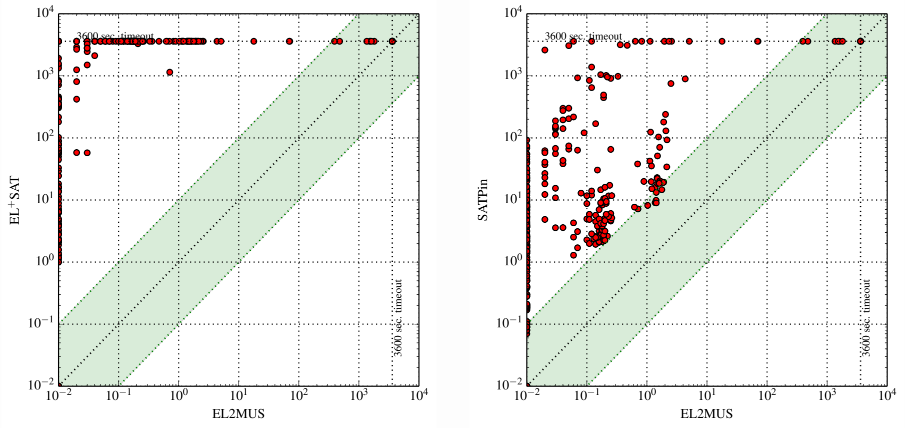


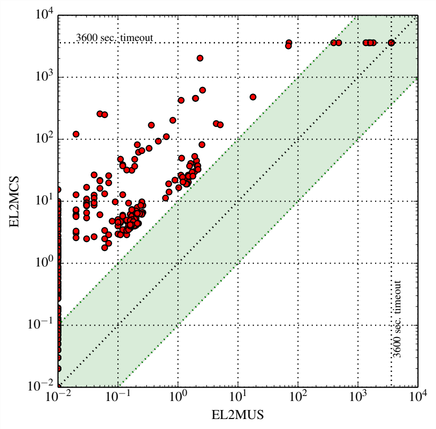
# Scatter plots – COI instances



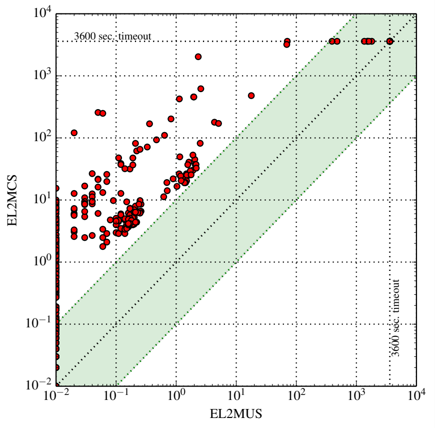
# Scatter plots – COI instances



# Scatter plots – COI instances (cont.)

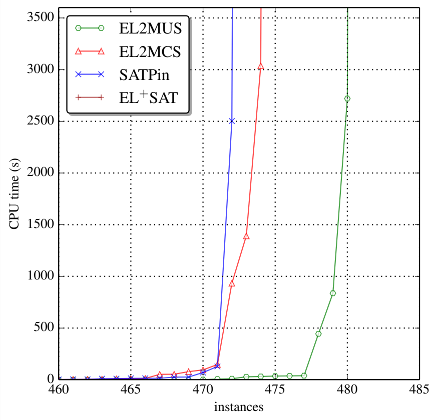


# Scatter plots – COI instances (cont.)

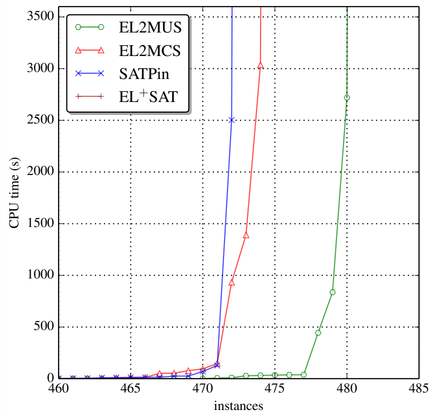


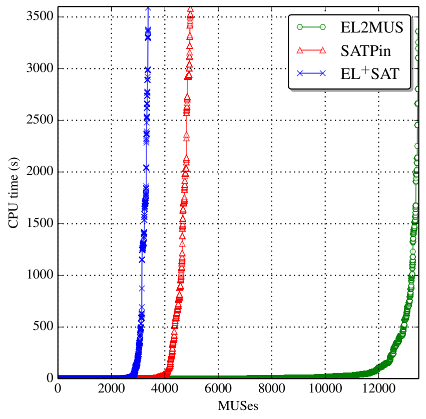

| % wins | EL$^+$SAT | SATPin | EL2MCS |
|---|---|---|---|
| EL$^+$SAT | – | 20.29% | 17.66% |
| SATPin | 79.71% | – | 19.13% |
| EL2MCS | 82.34% | 80.41% | – |
| EL2MUS | 100.0% | 100.0% | 100.0% |
| $> 10^1$x | 98.09% | 96.78% | 98.41% |
| $> 10^2$x | 97.55% | 72.07% | 58.07% |
| $> 10^3$x | 96.46% | 47.75% | 14.09% |
| $> 10^4$x | 74.05% | 06.49% | 00.00% |
| $> 10^5$x | 31.10% | 00.45% | 00.00% |

# Cactus plots – x2 instances

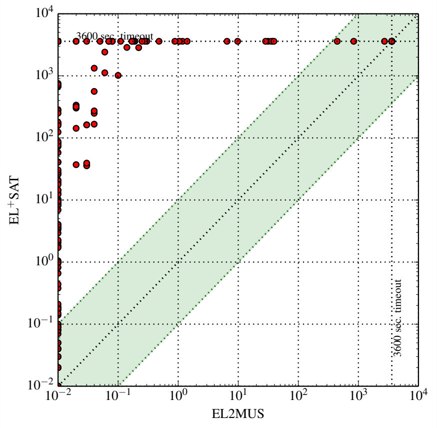


# Cactus plots – x2 instances

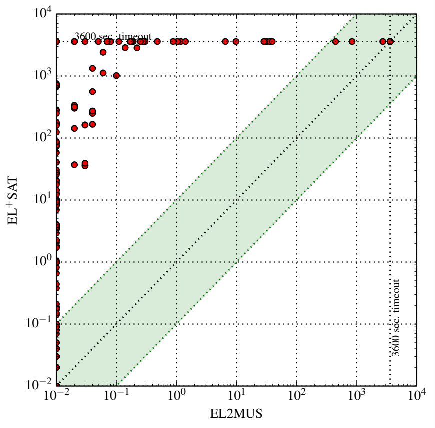


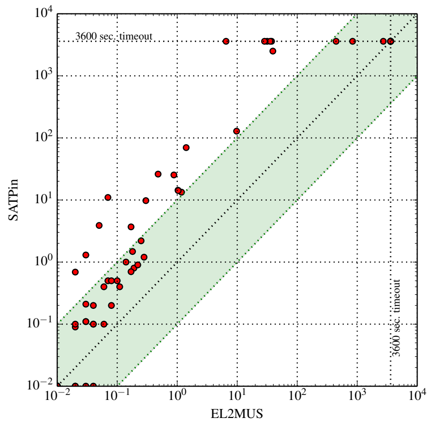



# Scatter plots – x2 instances

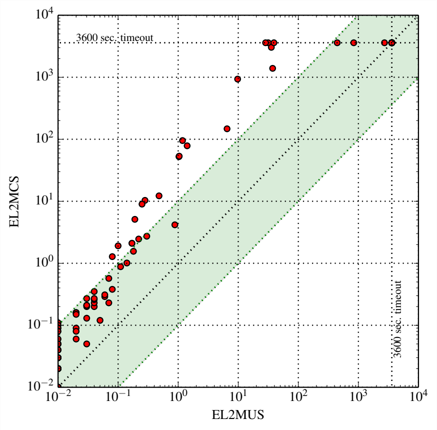


# Scatter plots – x2 instances

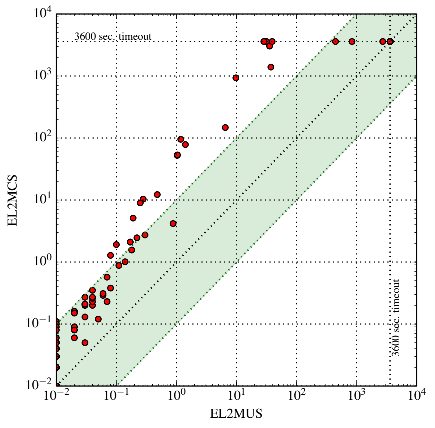


# Scatter plots – x2 instances (cont.)



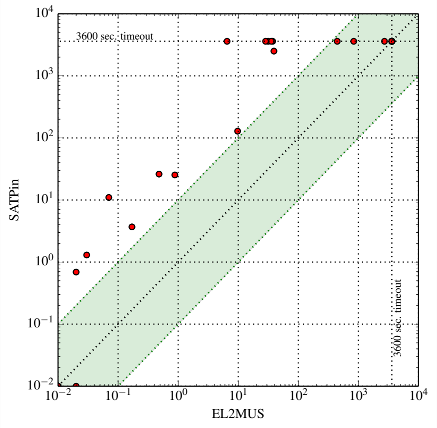
# Scatter plots – x2 instances (cont.)



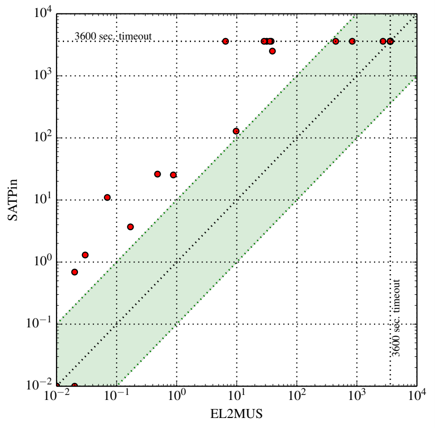

| % wins | $EL^{+}SAT$ | SATPin | EL2MCS |
|---|---|---|---|
| $EL^{+}SAT$ | – | 00.00% | 00.00% |
| SATPin | 100.0% | – | 91.55% |
| EL2MCS | 100.0% | 08.45% | – |
| EL2MUS | 100.0% | 67.69% | 99.32% |

| 19 TO inst | $EL^{+}SAT$ | SATPin | EL2MCS |
|---|---|---|---|
| # MUSes | 788 | 1484 | 0 |
| Δ MUSes | 9160 | 8864 | 9948 |

# SNOMED CT x2 instances – EL2MUS vs. SATPin

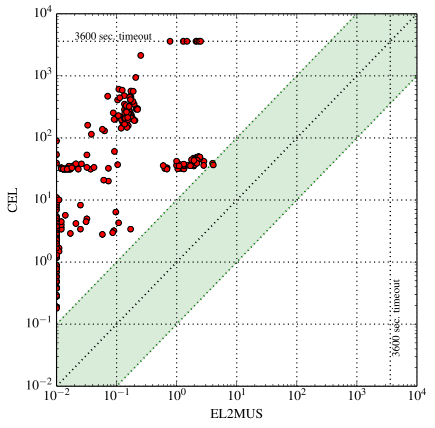


# SNOMED CT x2 instances – EL2MUS vs. SATPin

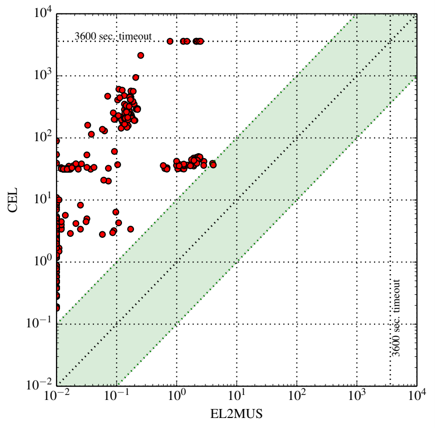


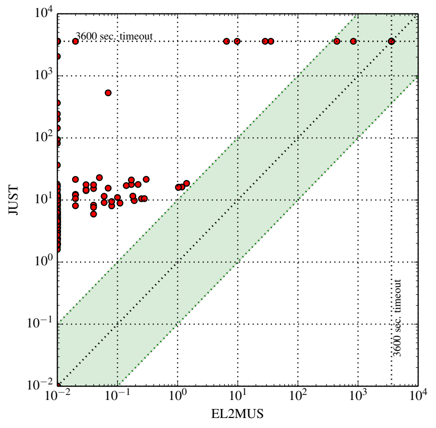

| | |
|---|---|
| # instances with run time ≥ 3600s | 19 |
| # instances with run time ≤ 0.1s | 65 |
| # EL2MUS wins | 16 |
| Total instances | 100 |

# Non-SAT based tools – vs CEL on COI instances



# Non-SAT based tools – vs Just on x2 instances



# Outline

# Conclusions & research directions

- Novel MUS enumeration algorithm for group Horn formulae
  - Exploits partial MUS enumeration, MCS extraction, (old) LTUR, insertion-based MUS extraction, etc.
- Application in axiom pinpointing of $\mathcal{EL}^+$ ontologies
- Orders of magnitude speedups for the larger instances (i.e. COI)
- Clear performance gains for the smaller instances (i.e. x2)
- Can enumerate **much** larger number of MUSes/MinAs than any other approach

# Conclusions & research directions

- Novel MUS enumeration algorithm for group Horn formulae
  - Exploits partial MUS enumeration, MCS extraction, (old) LTUR, insertion-based MUS extraction, etc.
- Application in axiom pinpointing of $\mathcal{EL}^+$ ontologies
- Orders of magnitude speedups for the larger instances (i.e. COI)
- Clear performance gains for the smaller instances (i.e. x2)
- Can enumerate **much** larger number of MUSes/MinAs than any other approach

- Develop standalone axiom pinpointing tool
  - Collaboration with SATPin authors
- Integrate recent advances in MUS & MCS extraction

Thank You