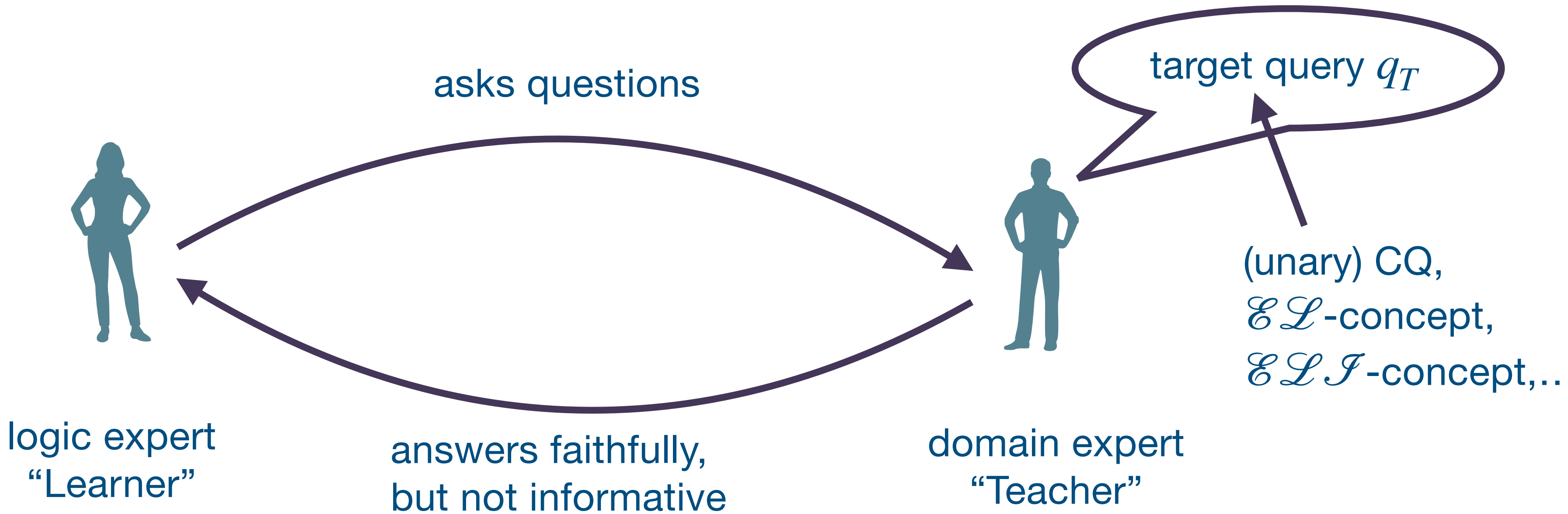
Concept Learning in Description Logics — Part 3:

Exact and PAC Learning

Exact Learning

- developed in 1987 by Dana Angluin in the context of learning finite automata
- our assumption: logic and domain expertise are not in the same hands



Learner and Teacher agree on ontology $\mathcal{O}$

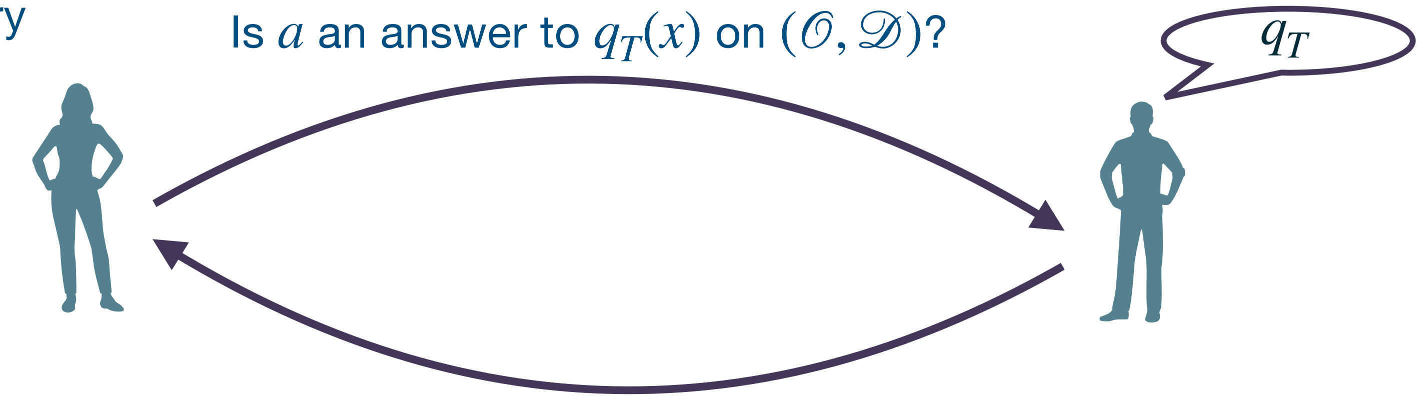
Question: Does Learner have a **strategy** to efficiently identify q_T?

Questions

- Membership query

$(\mathscr{D}, a)$

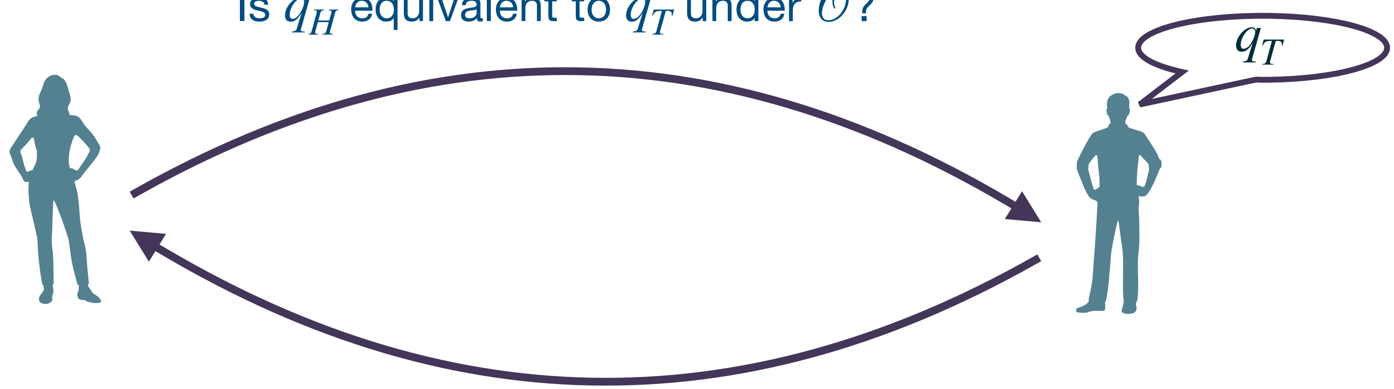
Is a an answer to $q_T(x)$ on $(\mathscr{O}, \mathscr{D})$?



yes/no

- Equivalence query

Is q_H equivalent to q_T under $\mathscr{O}$?

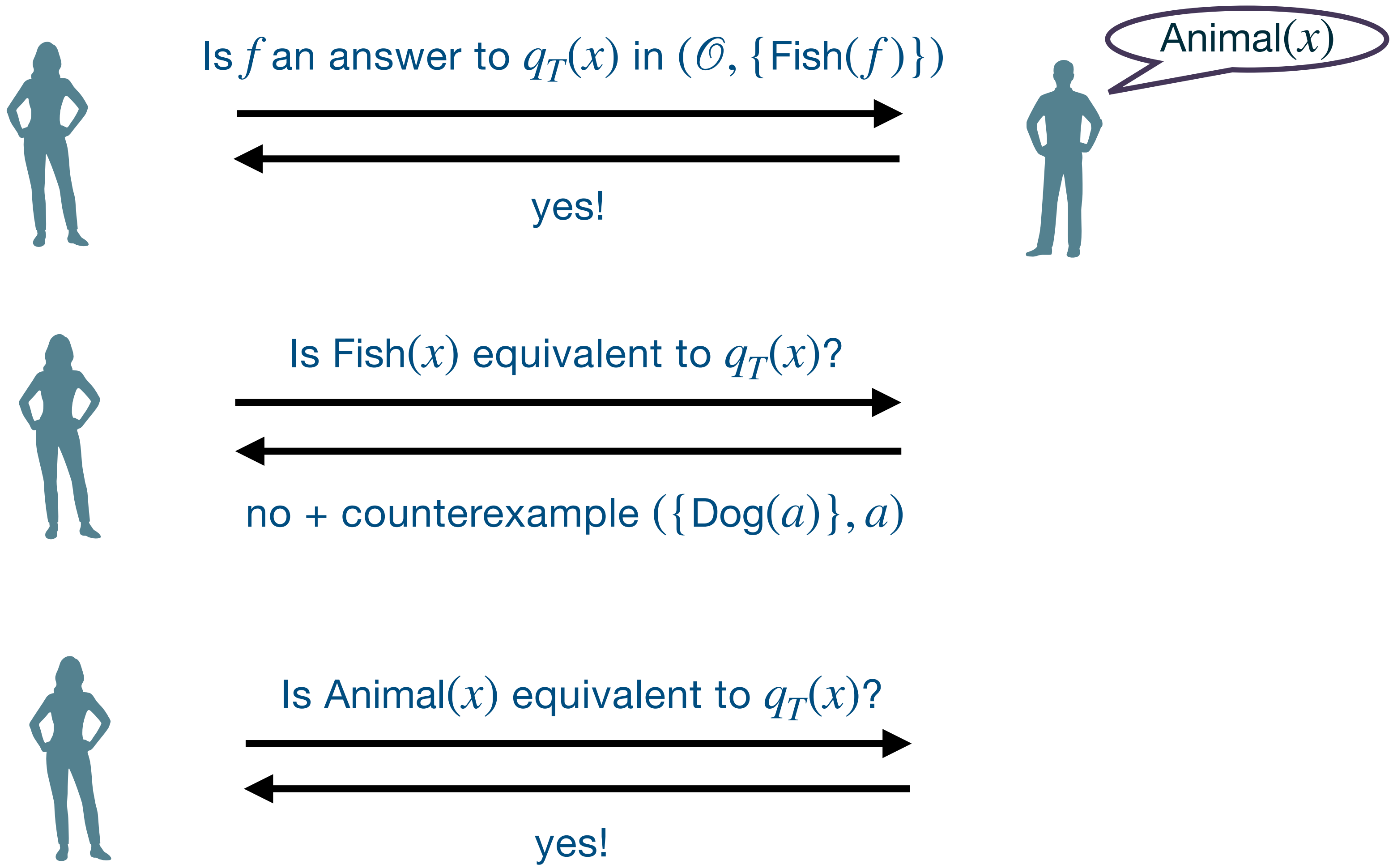


yes $\Rightarrow$ done

no $\Rightarrow$ **counterexample** $(\mathscr{D}, a)$

Example

Ontology $\mathcal{O} = \{$ Fish $\sqsubseteq$ Animal, Dog $\sqsubseteq$ Mammal, Mammal $\sqsubseteq$ Animal $\}$



What is known?

Efficient Learnability

=

learning strategy is guaranteed to identify target in time polynomial in signature, ontology, target, largest counterexample

query class	**ontology**	**questions**	**learnability**	**today**
$\mathcal{EL}$ / $\mathcal{ELI}$-concepts	no	MQ	efficient	$\Leftarrow$
$\mathcal{EL}$-concepts	$\mathcal{EL}$	MQ+EQ	efficient	$\Leftarrow$
CQ	no	MQ+EQ	efficient	
CQ/$\mathcal{ELI}$-concepts	$\mathcal{ELI}$	MQ+EQ	not efficient	
CQ	DL-Lite/$\mathcal{EL}$	MQ+EQ	**open**	
$\mathcal{ELI}$-concepts	$\mathcal{EL}$	MQ+EQ	**open**	

Sources

ten Cate, Dalmau, & Kolaitis, ToDS, 2013

ten Cate & Dalmau, ToDS, 2022

Funk, Jung, & Lutz, IJCAI, 2021/2022

Learning Strategy

All known learning algorithms follow a **general scheme**: they construct

$$q_0 \quad \subsetneq_{\mathcal{O}} \quad q_1 \quad \subsetneq_{\mathcal{O}} \quad \cdots \quad \subsetneq_{\mathcal{O}} \quad q_n \quad = \quad q_T$$

Start q_0: very strong query that is guaranteed to entail q_T

Step $q_i \to q_{i+1}$: two different strategies for weakening q_i

a) based on **frontiers** $\approx$ minimal weakenings of q_i

b) based on incorporation of counterexample (usually via **product**)

Key ingredient Using MQs, we can (syntactically) **minimize** the q_i

Lemma Sequence $q_0, \ldots, q_n$ as above with all q_i minimal is bounded by a polynomial in signature, ontology, and target

Learning $\mathcal{EL}$-Concepts under Ontologies

Exact learning of concepts:

- Teacher knows target DL concept C_T and answers membership and equivalence queries
- Learner knows DL ontology $\mathcal{O}$ and signature Σ of C_T.

Learning $\mathcal{EL}$-Concepts under Ontologies

Exact learning of concepts:

- Teacher knows target DL concept C_T and answers membership and equivalence queries
- Learner knows DL ontology $\mathcal{O}$ and signature Σ of C_T.

Learnability:
Is there an algorithm that the learner can execute to always identify C_T?

Learning $\mathcal{EL}$-Concepts under Ontologies

Exact learning of concepts:

- Teacher knows target DL concept C_T and answers membership and equivalence queries
- Learner knows DL ontology $\mathcal{O}$ and signature Σ of C_T.

Learnability:
Is there an algorithm that the learner can execute to always identify C_T?

Yes

Learning $\mathcal{EL}$-Concepts under Ontologies

Exact learning of concepts:

- Teacher knows target DL concept C_T and answers membership and equivalence queries
- Learner knows DL ontology $\mathcal{O}$ and signature Σ of C_T.

Learnability:
Is there an algorithm that the learner can execute to always identify C_T?

Yes: enumerate all concepts by size and ask equivalence query for each one

Learning $\mathcal{EL}$-Concepts under Ontologies

Exact learning of concepts:

- Teacher knows target DL concept C_T and answers membership and equivalence queries
- Learner knows DL ontology $\mathcal{O}$ and signature Σ of C_T.

Learnability:
Is there an algorithm that the learner can execute to always identify C_T?

Yes: enumerate all concepts by size and ask equivalence query for each one
running time exponential in $|C_T|$

Learning $\mathcal{EL}$-Concepts under Ontologies

Exact learning of concepts:

- Teacher knows target DL concept C_T and answers membership and equivalence queries
- Learner knows DL ontology $\mathcal{O}$ and signature Σ of C_T.

Learnability:
Is there an algorithm that the learner can execute to always identify C_T?

Yes: enumerate all concepts by size and ask equivalence query for each one
running time exponential in $|C_T|$

Is there a polynomial time learning algorithm?
(polynomial in the size of C_T, $\mathcal{O}$, Σ and largest counterexample)?

Learning $\mathcal{EL}$-Concepts under Ontologies

Exact learning of concepts:

- Teacher knows target DL concept C_T and answers membership and equivalence queries
- Learner knows DL ontology $\mathcal{O}$ and signature Σ of C_T.

Learnability:
Is there an algorithm that the learner can execute to always identify C_T?

Yes: enumerate all concepts by size and ask equivalence query for each one
running time exponential in $|C_T|$

Is there a polynomial time learning algorithm?
(polynomial in the size of C_T, $\mathcal{O}$, Σ and largest counterexample)?

We will look at the polynomial time learnability of $\mathcal{EL}$-concepts under ontologies.
First step: empty ontology ($\mathcal{O} = \emptyset$)

Idea 1: Checking Subsumption

How can we use membership queries to identify C_T?

Learning $\mathcal{EL}$-Concepts

Idea 1: Checking Subsumption

How can we use membership queries to identify C_T?

Every $\mathcal{EL}$-concept C can be represented in a natural way as a data instance $(\mathcal{D}_C, a_C)$

$$C = A \sqcap \exists r.(\exists s.B \sqcap \exists r.A) \quad \Longrightarrow \quad \begin{tikzcd} & a_C \quad \mathrm{A} \arrow[d, "r"] & \\ & a_{\exists s.B \sqcap \exists r.A} \arrow[dl, "s"'] \arrow[dr, "r"] & \\ a_B \quad \mathrm{B} & & a_A \quad \mathrm{A} \end{tikzcd}$$

Learning $\mathcal{EL}$-Concepts — Idea 1: Checking Subsumption

How can we use membership queries to identify C_T?

Every $\mathcal{EL}$-concept C can be represented in a natural way as a data instance $(\mathcal{D}_C, a_C)$

$$C = A \sqcap \exists r.(\exists s.B \sqcap \exists r.A) \qquad \Longrightarrow \qquad \begin{tikzcd}
 & a_C \quad \mathrm{A} \arrow[d, "r"] & \\
 & a_{\exists s.B \sqcap \exists r.A} \arrow[dl, "s"'] \arrow[dr, "r"] & \\
a_B \quad \mathrm{B} & & a_A \quad \mathrm{A}
\end{tikzcd}$$

For all $\mathcal{EL}$-concepts C_1, C_2: $\quad C_1 \sqsubseteq C_2$ if and only if $\mathcal{D}_{C_1} \models C_2(a_{C_2})$

Learning $\mathcal{EL}$-Concepts

Idea 1: Checking Subsumption

How can we use membership queries to identify C_T?

Every $\mathcal{EL}$-concept C can be represented in a natural way as a data instance $(\mathcal{D}_C, a_C)$

$$C = A \sqcap \exists r.(\exists s.B \sqcap \exists r.A) \quad \Longrightarrow \quad \begin{tikzcd} & a_C \quad \mathrm{A} \arrow[d, "r"] & \\ & a_{\exists s.B \sqcap \exists r.A} \arrow[dl, "s"'] \arrow[dr, "r"] & \\ a_B \quad \mathrm{B} & & a_A \quad \mathrm{A} \end{tikzcd}$$

For all $\mathcal{EL}$-concepts C_1, C_2: $\quad C_1 \sqsubseteq C_2$ if and only if $\mathcal{D}_{C_1} \models C_2(a_{C_2})$

Especially: $\quad C \sqsubseteq C_T$ if and only if $\mathcal{D}_C \models C_T(a_C)$

Learning $\mathcal{EL}$-Concepts

Idea 1: Checking Subsumption

How can we use membership queries to identify C_T?

Every $\mathcal{EL}$-concept C can be represented in a natural way as a data instance $(\mathcal{D}_C, a_C)$

$$C = A \sqcap \exists r.(\exists s.B \sqcap \exists r.A) \qquad \Longrightarrow \qquad \begin{tikzcd} & a_C \quad \mathrm{A} \arrow[d, "r"] & \\ & a_{\exists s.B \sqcap \exists r.A} \arrow[dl, "s"'] \arrow[dr, "r"] & \\ a_B \quad \mathrm{B} & & a_A \quad \mathrm{A} \end{tikzcd}$$

For all $\mathcal{EL}$-concepts C_1, C_2: $\quad C_1 \sqsubseteq C_2$ if and only if $\mathcal{D}_{C_1} \models C_2(a_{C_2})$

Especially: $\quad C \sqsubseteq C_T$ if and only if $\mathcal{D}_C \models C_T(a_C)$

If the response to the membership query $\mathcal{D}_C \models C_T(a_C)$ is “Yes”, then $C \sqsubseteq C_T$.

Learning $\mathcal{EL}$-Concepts

Idea 1: Checking Subsumption

How can we use membership queries to identify C_T?

Every $\mathcal{EL}$-concept C can be represented in a natural way as a data instance $(\mathcal{D}_C, a_C)$

$$C = A \sqcap \exists r.(\exists s.B \sqcap \exists r.A) \qquad \Longrightarrow \qquad \begin{tikzcd}[column sep=small]
 & a_C \quad \mathrm{A} \arrow[d, "r"] & \\
 & a_{\exists s.B \sqcap \exists r.A} \arrow[dl, "s"] \arrow[dr, "r"] & \\
a_B \quad \mathrm{B} & & a_A \quad \mathrm{A}
\end{tikzcd}$$

For all $\mathcal{EL}$-concepts C_1, C_2: $\quad C_1 \sqsubseteq C_2$ if and only if $\mathcal{D}_{C_1} \models C_2(a_{C_2})$

Especially: $\quad C \sqsubseteq C_T$ if and only if $\mathcal{D}_C \models C_T(a_C)$

If the response to the membership query $\mathcal{D}_C \models C_T(a_C)$ is “Yes”, then $C \sqsubseteq C_T$.

We can use a membership query to test $C \sqsubseteq C_T$

Idea 2: Approaching C_T from below

If there is a concept C such that
$C \sqsubseteq C_T$ and $C_T \not\sqsubseteq C$,

Idea 2: Approaching C_T from below

If there is a concept C such that
$C \sqsubseteq C_T$ and $C_T \not\sqsubseteq C$,
then there must be a concept D such that
$D \sqsubseteq C_T$, $C \sqsubseteq D$, and $D \not\sqsubseteq C$

Any such D moves us closer to C_T (decreases number of possibilites for C_T)

How can we construct such a D?

Learning $\mathcal{EL}$-Concepts

If there is a concept C such that
$C \sqsubseteq C_T$ and $C_T \not\sqsubseteq C$,
then there must be a concept D such that
$D \sqsubseteq C_T$, $C \sqsubseteq D$, and $D \not\sqsubseteq C$

Any such D moves us closer to C_T (decreases number of possibilites for C_T)

How can we construct such a D? Generalize C and check $D \sqsubseteq C_T$

Learning $\mathcal{EL}$-Concepts

Idea 2: Approaching C_T from below

If there is a concept C such that

$C \sqsubseteq C_T$ and $C_T \not\sqsubseteq C$,

then there must be a concept D such that

$D \sqsubseteq C_T$, $C \sqsubseteq D$, and $D \not\sqsubseteq C$

Any such D moves us closer to C_T (decreases number of possibilites for C_T)

How can we construct such a D? Generalize C and check $D \sqsubseteq C_T$

$$\begin{tikzcd}
& a_1 \quad \mathrm{A} \arrow[d, "r"] & \\
& a_2 \arrow[dl, "s"'] \arrow[dr, "r"] & \\
a_3 \quad \mathrm{B} & & a_4 \quad \mathrm{A}
\end{tikzcd}$$

Idea 2: Approaching C_T from below

If there is a concept C such that

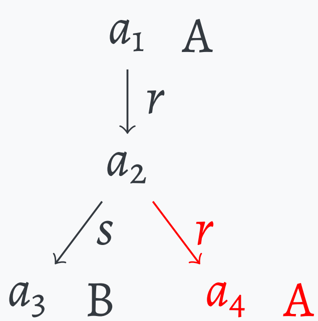
$C \sqsubseteq C_T$ and $C_T \not\sqsubseteq C$,

then there must be a concept D such that

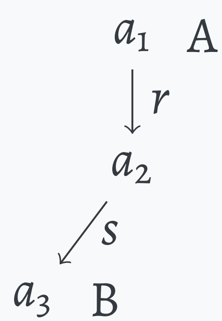
$D \sqsubseteq C_T$, $C \sqsubseteq D$, and $D \not\sqsubseteq C$

Any such D moves us closer to C_T (decreases number of possibilites for C_T)

How can we construct such a D? Generalize C and check $D \sqsubseteq C_T$



Delete $\exists r.A$



Idea 2: Approaching C_T from below

If there is a concept C such that

$C \sqsubseteq C_T$ and $C_T \not\sqsubseteq C$,

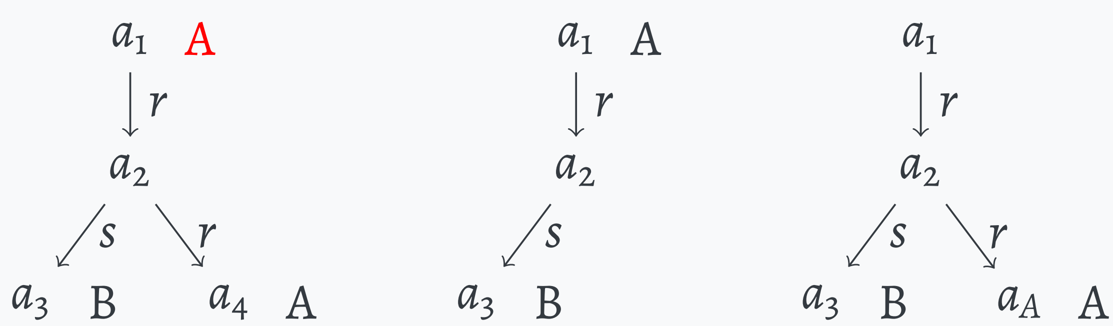
then there must be a concept D such that

$D \sqsubseteq C_T$, $C \sqsubseteq D$, and $D \not\sqsubseteq C$

Any such D moves us closer to C_T (decreases number of possibilites for C_T)

How can we construct such a D? Generalize C and check $D \sqsubseteq C_T$

Delete $\exists r.A$ Delete A



Learning $\mathcal{EL}$-Concepts

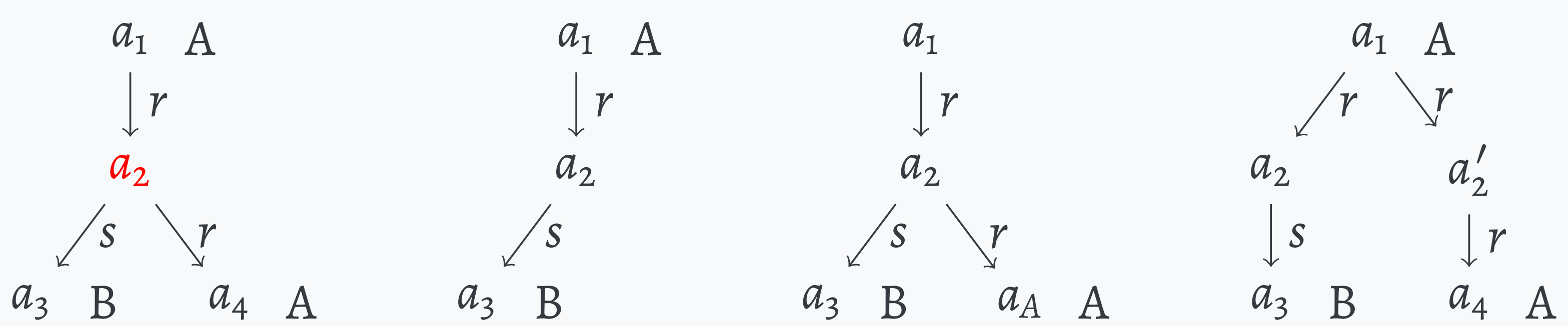
Idea 2: Approaching C_T from below

If there is a concept C such that
  $C \sqsubseteq C_T$ and $C_T \not\sqsubseteq C$,
then there must be a concept D such that
  $D \sqsubseteq C_T$, $C \sqsubseteq D$, and $D \not\sqsubseteq C$

Any such D moves us closer to C_T (decreases number of possibilites for C_T)

How can we construct such a D? Generalize C and check $D \sqsubseteq C_T$

Delete $\exists r.A$ Delete A Split $(\exists s.B \sqcap \exists r.A)$



Learning $\mathcal{EL}$-Concepts

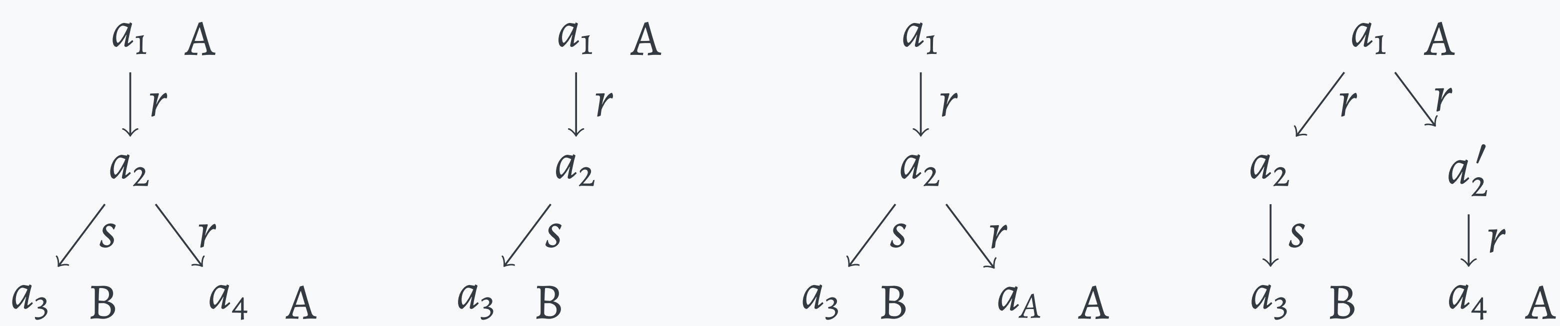
Idea 2: Approaching C_T from below

If there is a concept C such that
$C \sqsubseteq C_T$ and $C_T \not\sqsubseteq C$,
then there must be a concept D such that
$D \sqsubseteq C_T$, $C \sqsubseteq D$, and $D \not\sqsubseteq C$

Any such D moves us closer to C_T (decreases number of possibilites for C_T)

How can we construct such a D? Generalize C and check $D \sqsubseteq C_T$

Delete $\exists r.A$ Delete A Split $(\exists s.B \sqcap \exists r.A)$



We need to check all possible generalizations of C

Definition (Frontier of C)

A set of concepts $\mathfrak{F}$ is a *frontier of C* if

1. $C \sqsubseteq D$ and $D \not\sqsubseteq C$ for all $D \in \mathfrak{F}$

We need to check all possible generalizations of C

Definition (Frontier of C)

A set of concepts $\mathfrak{F}$ is a *frontier of C* if

1. $C \sqsubseteq D$ and $D \not\sqsubseteq C$ for all $D \in \mathfrak{F}$
2. for every concept D' with $C \sqsubseteq D'$ and $D' \not\sqsubseteq C$, there is a $D \in \mathfrak{F}$ such that $D \sqsubseteq D'$.

Learning $\mathcal{EL}$-Concepts

We need to check all possible generalizations of C

Definition (Frontier of C)

A set of concepts $\mathcal{F}$ is a *frontier of C* if

1. $C \sqsubseteq D$ and $D \not\sqsubseteq C$ for all $D \in \mathcal{F}$
2. for every concept D' with $C \sqsubseteq D'$ and $D' \not\sqsubseteq C$, there is a $D \in \mathcal{F}$ such that $D \sqsubseteq D'$.

Theorem (ten Cate and Dalmau 2021/Kriegel 2018)

Let C be an $\mathcal{EL}$-concept. Then a frontier of C can be computed in polynomial time (in $|C|$)

Learning $\mathcal{EL}$-Concepts

We need to check all possible generalizations of C

Definition (Frontier of C)

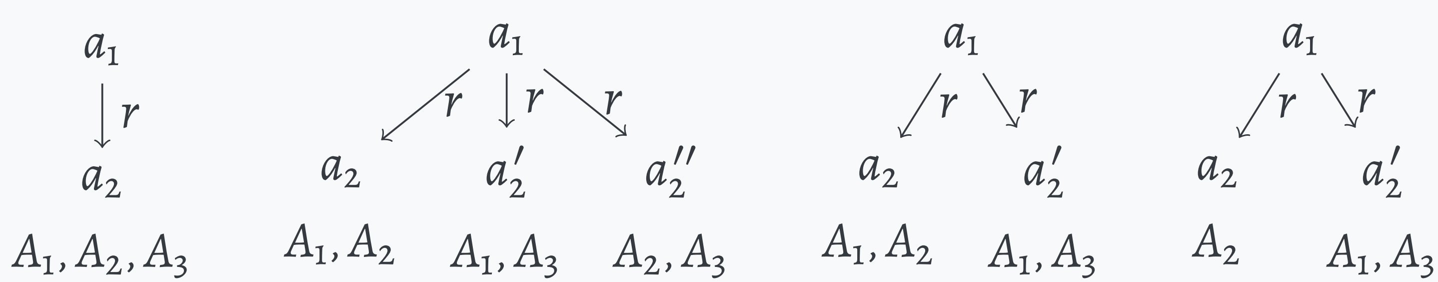
A set of concepts $\mathcal{F}$ is a *frontier of C* if

1. $C \sqsubseteq D$ and $D \not\sqsubseteq C$ for all $D \in \mathcal{F}$
2. for every concept D' with $C \sqsubseteq D'$ and $D' \not\sqsubseteq C$, there is a $D \in \mathcal{F}$ such that $D \sqsubseteq D'$.

Theorem (ten Cate and Dalmau 2021/Kriegel 2018)

Let C be an $\mathcal{EL}$-concept. Then a frontier of C can be computed in polynomial time (in $|C|$)

Long chains:



Idea 3: Minimization

Problem: Frontier-chains are long and concepts are too large

Learning $\mathcal{EL}$-Concepts — Idea 3: Minimization

Problem: Frontier-chains are long and concepts are too large

$$
\begin{tikzcd}
a_1 \arrow[d, "r"] \\
a_2 \\
A_1, A_2, A_3
\end{tikzcd}
\qquad
\begin{array}{c}
C \\
\begin{tikzcd}
 & a_1 \arrow[dl, "r"'] \arrow[d, "r"] \arrow[dr, "r"] & \\
a_2 & a_2' & a_2'' \\
A_1, A_2 & A_1, A_3 & A_2, A_3
\end{tikzcd}
\end{array}
\qquad\qquad
C_T
$$

Learning $\mathcal{EL}$-Concepts

Idea 3: Minimization

Problem: Frontier-chains are long and concepts are too large

$$
\begin{tikzcd}
a_1 \arrow[d, "r"] \\
a_2 \\
A_1, A_2, A_3
\end{tikzcd}
$$

C

$$
\begin{tikzcd}
 & a_1 \arrow[dl, "r"] \arrow[d, "r"] \arrow[dr, "r"] & \\
a_2 & a_2' & a_2'' \\
A_1, A_2 & A_1, A_3 & A_2, A_3
\end{tikzcd}
$$

C_T

$$
\begin{tikzcd}
a_1 \arrow[d, "r"] \\
a_2 \\
A_1
\end{tikzcd}
$$

Learning $\mathcal{EL}$-Concepts

Idea 3: Minimization

Problem: Frontier-chains are long and concepts are too large

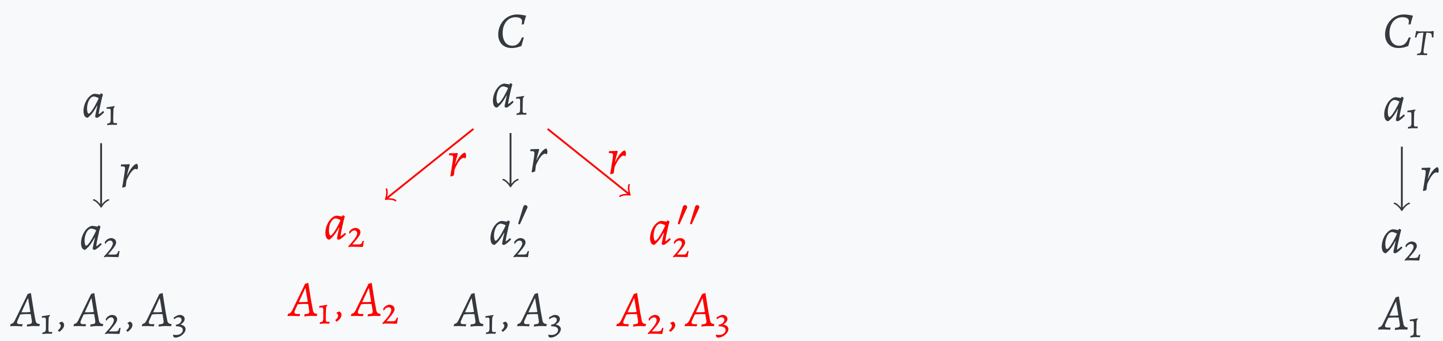
$$
\begin{tikzcd}
a_1 \arrow[d, "r"] \\
a_2 \\
A_1, A_2, A_3
\end{tikzcd}
\qquad
\begin{tikzcd}
 & C & \\
 & a_1 \arrow[dl, "r"'] \arrow[d, "r"] \arrow[dr, "r"] & \\
a_2 & a_2' & a_2'' \\
A_1, A_2 & A_1, A_3 & A_2, A_3
\end{tikzcd}
\qquad
\begin{tikzcd}
C_T \\
a_1 \arrow[d, "r"] \\
a_2 \\
A_1
\end{tikzcd}
$$

Only small part of C needed for $C \sqsubseteq C_T$ ($\leqslant |C_T|$)

Learning $\mathcal{EL}$-Concepts

Idea 3: Minimization

Problem: Frontier-chains are long and concepts are too large



Only small part of C needed for $C \sqsubseteq C_T$ ($\leqslant |C_T|$)

Learning $\mathcal{E}\mathcal{L}$-Concepts

Idea 3: Minimization

Problem: Frontier-chains are long and concepts are too large

$$\begin{tikzcd}[row sep=small] a_1 \arrow[d, "r"] \\ a_2 \\ A_1, A_2, A_3 \end{tikzcd}
\qquad
\begin{tikzcd}[row sep=small] & C & \\ & a_1 \arrow[dl, "r"'] \arrow[d, "r"] \arrow[dr, "r"] & \\ a_2 & a_2' & a_2'' \\ A_1, A_2 & A_1, A_3 & A_2, A_3 \end{tikzcd}
\qquad
\begin{tikzcd}[row sep=small] \mathsf{minimize}(C) \\ \color{red}{a_1} \arrow[d, red, "\color{red}{r}"] \\ \color{red}{a_2'} \\ \color{red}{A_1, A_3} \end{tikzcd}
\qquad
\begin{tikzcd}[row sep=small] C_T \\ a_1 \arrow[d, "r"] \\ a_2 \\ A_1 \end{tikzcd}$$

Only small part of C needed for $C \sqsubseteq C_T$ ($\leqslant |C_T|$)

$\mathsf{minimize}(C)$: for each existential restriction, remove if it is unecessary

Learning $\mathcal{EL}$-Concepts — Idea 3: Minimization

Problem: Frontier-chains are long and concepts are too large

$$
\begin{tikzcd}
a_1 \arrow[d, "r"] \\
a_2 \\
A_1, A_2, A_3
\end{tikzcd}
\qquad
\begin{array}{c} C \\
\begin{tikzcd}
 & a_1 \arrow[dl, "r"'] \arrow[d, "r"] \arrow[dr, "r"] & \\
a_2 & a_2' & a_2'' \\
A_1, A_2 & A_1, A_3 & A_2, A_3
\end{tikzcd}
\end{array}
\qquad
\begin{array}{c} \text{minimize}(C) \\
\begin{tikzcd}
a_1 \arrow[d, "r"] \\
a_2' \\
A_1, A_3
\end{tikzcd}
\end{array}
\qquad
\begin{array}{c} C_T \\
\begin{tikzcd}
a_1 \arrow[d, "r"] \\
a_2 \\
A_1
\end{tikzcd}
\end{array}
$$

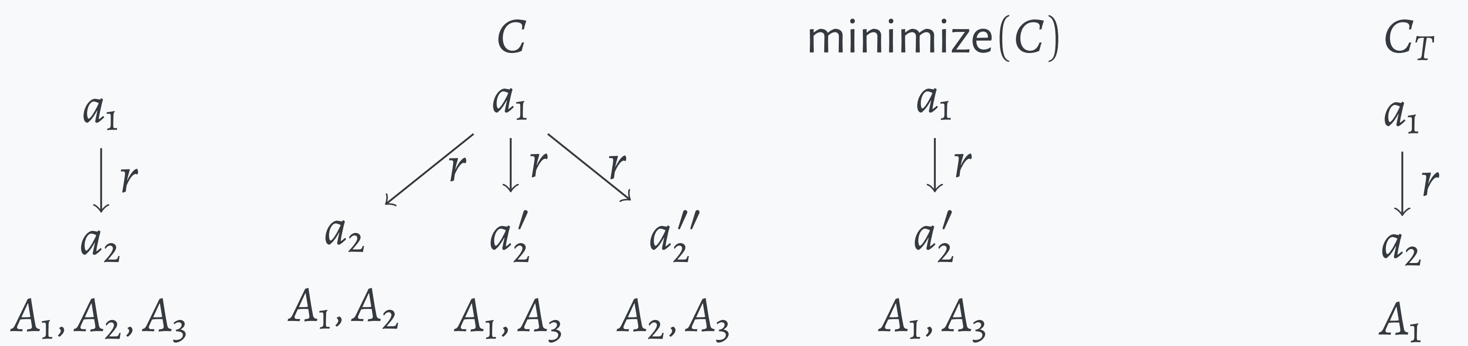
Only small part of C needed for $C \sqsubseteq C_T$ ($\leqslant |C_T|$)

minimize(C): for each existential restriction, remove if it is unecessary

$$C \sqsubseteq \text{minimize}(C) \sqsubseteq C_T \text{ and } |\text{minimize}(C)| \leqslant |C_T$$

Learning $\mathcal{EL}$-Concepts — Idea 3: Minimization

Problem: Frontier-chains are long and concepts are too large



Only small part of C needed for $C \sqsubseteq C_T$ ($\leqslant |C_T|$)

minimize(C): for each existential restriction, remove if it is unecessary

$$C \sqsubseteq \text{minimize}(C) \sqsubseteq C_T \text{ and } |\text{minimize}(C)| \leqslant |C_T$$

Lemma

A sequence of minimized concepts that approaches C_T has at most polynomial length (in $|C_T|$)

Learning $\mathcal{EL}$-Concepts

Input An $\mathcal{EL}$-concept $C_\circ$ such that $C_\circ \sqsubseteq C_T$
Output An $\mathcal{EL}$-concept C_H such that $C_H \equiv C_T$

$C_H \coloneqq C_\circ$
while there is a D in the frontier of C_H with $D \sqsubseteq C_T$ **do**
    $C_H \coloneqq \mathsf{minimize}(D)$
end while
return C_H

Theorem (ten Cate and Dalmau 2021)

$\mathcal{EL}$-concepts are polynomial time learnable using only membership queries (under the empty ontology)

How can we obtain the input $C_\circ$? (with $C_\circ \sqsubseteq C_T$)

Learning $\mathcal{EL}$-Concepts

Idea 4: Initial Concept

How can we obtain the input C_0? (with $C_0 \sqsubseteq C_T$)

For a given Σ, there is $(\mathcal{D}, a)$ such that $\mathcal{D} \models C_T(a)$ for all C_T over Σ

$(\mathcal{D}, a_1)$

$$\begin{tikzcd} \overset{A_1, A_2, A_3}{a_1} \arrow[loop below, "r"] \end{tikzcd}$$

C_T

$$\begin{tikzcd} a_1 \arrow[d, "r"] \\ \underset{A_1, A_2, A_3}{a_2} \end{tikzcd}$$

Learning $\mathcal{EL}$-Concepts — Idea 4: Initial Concept

How can we obtain the input $C_\circ$? (with $C_\circ \sqsubseteq C_T$)

For a given Σ, there is $(\mathcal{D}, a)$ such that $\mathcal{D} \models C_T(a)$ for all C_T over Σ

Repeatedly double cycles and minimize (membership queries) to obtain C with $C \sqsubseteq C_T$

$(\mathcal{D}, a_1)$

$$\begin{tikzcd} \overset{A_1, A_2, A_3}{a_1} \arrow[loop below, "r"] \end{tikzcd}$$

C_T

$$\begin{tikzcd} a_1 \arrow[d, "r"] \\ \underset{A_1, A_2, A_3}{a_2} \end{tikzcd}$$

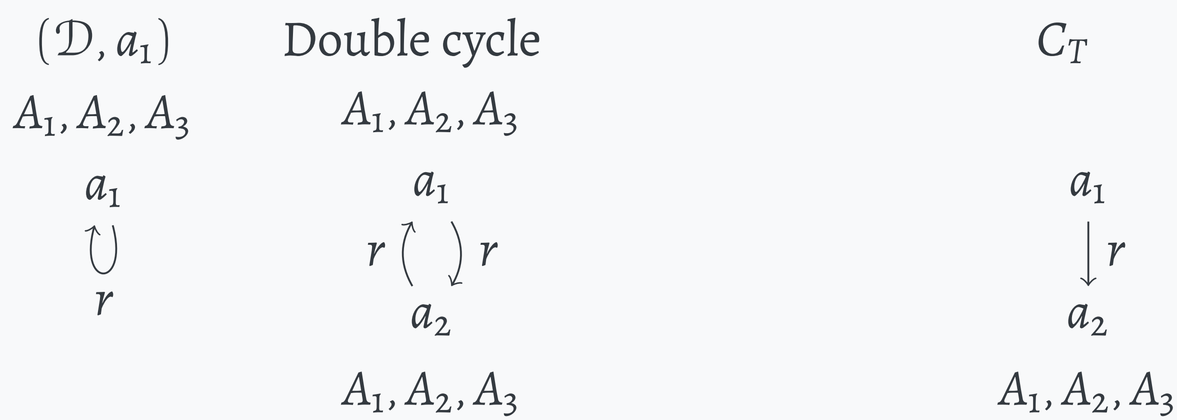
Learning $\mathcal{EL}$-Concepts

Idea 4: Initial Concept

How can we obtain the input $C_\circ$? (with $C_\circ \sqsubseteq C_T$)

For a given Σ, there is $(\mathcal{D}, a)$ such that $\mathcal{D} \models C_T(a)$ for all C_T over Σ

Repeatedly double cycles and minimize (membership queries) to obtain C with $C \sqsubseteq C_T$



How can we obtain the input C_0? (with $C_0 \sqsubseteq C_T$)

For a given Σ, there is $(\mathcal{D}, a)$ such that $\mathcal{D} \models C_T(a)$ for all C_T over Σ

Repeatedly double cycles and minimize (membership queries) to obtain C with $C \sqsubseteq C_T$

$(\mathcal{D}, a_1)$

$$\begin{tikzcd} A_1, A_2, A_3 \\ a_1 \arrow[loop below, "r"] \end{tikzcd}$$

Double cycle

$$\begin{tikzcd} A_1, A_2, A_3 \\ a_1 \arrow[d, bend left, "r"] \\ a_2 \arrow[u, bend left, "r"] \\ A_1, A_2, A_3 \end{tikzcd}$$

C_T

$$\begin{tikzcd} a_1 \arrow[d, "r"] \\ a_2 \\ A_1, A_2, A_3 \end{tikzcd}$$

Learning $\mathcal{EL}$-Concepts

Idea 4: Initial Concept

How can we obtain the input C_0? (with $C_0 \sqsubseteq C_T$)

For a given Σ, there is $(\mathcal{D}, a)$ such that $\mathcal{D} \models C_T(a)$ for all C_T over Σ

Repeatedly double cycles and minimize (membership queries) to obtain C with $C \sqsubseteq C_T$

extract-el: from a data instance $(\mathcal{D}, a)$ with $\mathcal{D} \models C_T(a)$,
extract an $\mathcal{EL}$-concept C such that $\mathcal{D} \models C(a)$ and $C \sqsubseteq C_T$

$(\mathcal{D}, a_1)$	Double cycle	C	C_T
A_1, A_2, A_3	A_1, A_2, A_3	A_1, A_2, A_3	
a_1	a_1	a_1	a_1
r	$r \quad r$	$\downarrow r$	$\downarrow r$
	a_2	a_2	a_2
	A_1, A_2, A_3	A_1, A_2, A_3	A_1, A_2, A_3

- More expressive concepts
  Also works for $\mathcal{ELI}$-concepts ($\mathcal{EL}$ with inverses) and "c-acyclic" conjunctive queries (ten Cate and Dalmau 2022)

Learning $\mathcal{EL}$-Concepts

- More expressive concepts
  Also works for $\mathcal{ELI}$-concepts ($\mathcal{EL}$ with inverses) and "c-acyclic" conjunctive queries (ten Cate and Dalmau 2022)
- Non-empty ontologies
  Also works under some lightweight ontology languages like *DL-Lite*$_{core}$ (F., Jung, Lutz 2022)
  Frontiers w.r.t. *DL-Lite*$_{core}$ ontologies can be computed in polynomial time

Remarks

- More expressive concepts
  Also works for $\mathcal{ELI}$-concepts ($\mathcal{EL}$ with inverses) and "c-acyclic" conjunctive queries (ten Cate and Dalmau 2022)
- Non-empty ontologies
  Also works under some lightweight ontology languages like *DL-Lite*$_{core}$ (F., Jung, Lutz 2022)
  Frontiers w.r.t. *DL-Lite*$_{core}$ ontologies can be computed in polynomial time
- Disjointness constraints
  If the ontology contains disjointness constraints like $A \sqcap B \sqsubseteq \bot$ then obtaining the initial concept becomes more complicated

Learning $\mathcal{EL}$-Concepts

- More expressive concepts
  Also works for $\mathcal{ELI}$-concepts ($\mathcal{EL}$ with inverses) and "c-acyclic" conjunctive queries (ten Cate and Dalmau 2022)
- Non-empty ontologies
  Also works under some lightweight ontology languages like $\mathit{DL\text{-}Lite}_{core}$ (F., Jung, Lutz 2022)
  Frontiers w.r.t. $\mathit{DL\text{-}Lite}_{core}$ ontologies can be computed in polynomial time
- Disjointness constraints
  If the ontology contains disjointness constraints like $A \sqcap B \sqsubseteq \bot$ then obtaining the initial concept becomes more complicated
- Practicality
  asks a lot of membership queries

Now we move on to learning $\mathcal{EL}$-concepts under $\mathcal{EL}$-ontologies.

How many of our idea do still work?

Learning $\mathcal{EL}$-Concepts under Ontologies

Now we move on to learning $\mathcal{EL}$-concepts under $\mathcal{EL}$-ontologies.

How many of our idea do still work?

- Testing subsumption with membership queries works
  $\mathcal{O}, \mathcal{D}_C \models C_T(a_C)$ if and only if $\mathcal{O} \models C \sqsubseteq C_T$

Now we move on to learning $\mathcal{EL}$-concepts under $\mathcal{EL}$-ontologies.

How many of our idea do still work?

- Testing subsumption with membership queries works
  $\mathcal{O}, \mathcal{D}_C \models C_T(a_C)$ if and only if $\mathcal{O} \models C \sqsubseteq C_T$
- Minimization works

Now we move on to learning $\mathcal{EL}$-concepts under $\mathcal{EL}$-ontologies.

How many of our idea do still work?

- Testing subsumption with membership queries works
  $\mathcal{O}, \mathcal{D}_C \models C_T(a_C)$ if and only if $\mathcal{O} \models C \sqsubseteq C_T$
- Minimization works
- extract-el works

Now we move on to learning $\mathcal{EL}$-concepts under $\mathcal{EL}$-ontologies.

How many of our idea do still work?

- Testing subsumption with membership queries works
  $\mathcal{O}, \mathcal{D}_C \models C_T(a_C)$ if and only if $\mathcal{O} \models C \sqsubseteq C_T$
- Minimization works
- extract-el works
- Frontiers can no longer be computed in polynomial time

Learning $\mathcal{EL}$-Concepts under Ontologies

Now we move on to learning $\mathcal{EL}$-concepts under $\mathcal{EL}$-ontologies.

How many of our idea do still work?

- Testing subsumption with membership queries works
  $\mathcal{O}, \mathcal{D}_C \models C_T(a_C)$ if and only if $\mathcal{O} \models C \sqsubseteq C_T$
- Minimization works
- extract-el works
- Frontiers can no longer be computed in polynomial time

Theorem (F., Jung, Lutz, 2021)

$\mathcal{EL}$-concepts are not polynomial time learnable under $\mathcal{EL}$-ontologies using only membership queries

Learning $\mathcal{EL}$-Concepts under Ontologies

Consider an $\mathcal{EL}$ ontology $\mathcal{O}$ with the CIs:

$$A_i \sqcap B_i \sqsubseteq A_1 \sqcap B_1 \sqcap \cdots \sqcap A_n \sqcap B_n \quad \text{for } 1 \leqslant i \leqslant n$$

Learning $\mathcal{EL}$-Concepts under Ontologies

Consider an $\mathcal{EL}$ ontology $\mathcal{O}$ with the CIs:

$$A_i \sqcap B_i \sqsubseteq A_1 \sqcap B_1 \sqcap \cdots \sqcap A_n \sqcap B_n \quad \text{for } 1 \leqslant i \leqslant n$$

and the set of concepts $S = \{\alpha_1 \sqcap \cdots \sqcap \alpha_n \mid \alpha_i \in \{A_i, B_i\}\}$

Learning $\mathcal{EL}$-Concepts under Ontologies

Using only membership queries

Consider an $\mathcal{EL}$ ontology $\mathcal{O}$ with the CIs:

$$A_i \sqcap B_i \sqsubseteq A_1 \sqcap B_1 \sqcap \cdots \sqcap A_n \sqcap B_n \quad \text{for } 1 \leqslant i \leqslant n$$

and the set of concepts $S = \{\alpha_1 \sqcap \cdots \sqcap \alpha_n \mid \alpha_i \in \{A_i, B_i\}\}$

$C_T \in S$ is hard to identify:

If $\mathcal{O}, \mathcal{D} \models C_1(a)$ and $\mathcal{O}, \mathcal{D} \models C_2(a)$ for $C_1, C_2 \in S$ with $C_1 \neq C_2$,
then $\mathcal{O}, \mathcal{D} \models C(a)$ for all $C \in S$

Learning $\mathcal{EL}$-Concepts under Ontologies

Using only membership queries

Consider an $\mathcal{EL}$ ontology $\mathcal{O}$ with the CIs:

$$A_i \sqcap B_i \sqsubseteq A_1 \sqcap B_1 \sqcap \cdots \sqcap A_n \sqcap B_n \quad \text{for } 1 \leqslant i \leqslant n$$

and the set of concepts $S = \{\alpha_1 \sqcap \cdots \sqcap \alpha_n \mid \alpha_i \in \{A_i, B_i\}\}$

$C_T \in S$ is hard to identify:

If $\mathcal{O}, \mathcal{D} \models C_1(a)$ and $\mathcal{O}, \mathcal{D} \models C_2(a)$ for $C_1, C_2 \in S$ with $C_1 \neq C_2$,
then $\mathcal{O}, \mathcal{D} \models C(a)$ for all $C \in S$

Worst case: learning algorithm needs $|S| = 2^n$ membership queries to identify C_T in S

Learning $\mathcal{EL}$-Concepts under Ontologies

Using only membership queries

Consider an $\mathcal{EL}$ ontology $\mathcal{O}$ with the CIs:

$$A_i \sqcap B_i \sqsubseteq A_1 \sqcap B_1 \sqcap \cdots \sqcap A_n \sqcap B_n \quad \text{for } 1 \leqslant i \leqslant n$$

and the set of concepts $S = \{\alpha_1 \sqcap \cdots \sqcap \alpha_n \mid \alpha_i \in \{A_i, B_i\}\}$

$C_T \in S$ is hard to identify:

If $\mathcal{O}, \mathcal{D} \models C_1(a)$ and $\mathcal{O}, \mathcal{D} \models C_2(a)$ for $C_1, C_2 \in S$ with $C_1 \neq C_2$,
then $\mathcal{O}, \mathcal{D} \models C(a)$ for all $C \in S$

Worst case: learning algorithm needs $|S| = 2^n$ membership queries to identify C_T in S

Learning algorithm for $\mathcal{EL}$-ontologies must use equivalence queries and counterexamples

Learner asks equivalence query with hypothesis C_H

If $\mathcal{O} \not\models C_H \equiv C_T$, then teacher returns counter example $(\mathcal{D}, a)$ such that

- $\mathcal{O}, \mathcal{D} \models C_H(a)$ and $\mathcal{O}, \mathcal{D} \not\models C_T(a)$, or

Learner asks equivalence query with hypothesis C_H

If $\mathcal{O} \not\models C_H \equiv C_T$, then teacher returns counter example $(\mathcal{D}, a)$ such that

- $\mathcal{O}, \mathcal{D} \models C_H(a)$ and $\mathcal{O}, \mathcal{D} \not\models C_T(a)$, or
- $\mathcal{O}, \mathcal{D} \not\models C_H(a)$ and $\mathcal{O}, \mathcal{D} \models C_T(a)$.

Learner asks equivalence query with hypothesis C_H

If $\mathcal{O} \not\models C_H \equiv C_T$, then teacher returns counter example $(\mathcal{D}, a)$ such that

- $\mathcal{O}, \mathcal{D} \models C_H(a)$ and $\mathcal{O}, \mathcal{D} \not\models C_T(a)$, or (Not possible if we ensure that $\mathcal{O} \models C_H \sqsubseteq C_T$)
- $\mathcal{O}, \mathcal{D} \not\models C_H(a)$ and $\mathcal{O}, \mathcal{D} \models C_T(a)$.

Learner asks equivalence query with hypothesis C_H

If $\mathcal{O} \not\models C_H \equiv C_T$, then teacher returns counter example $(\mathcal{D}, a)$ such that

- $\mathcal{O}, \mathcal{D} \models C_H(a)$ and $\mathcal{O}, \mathcal{D} \not\models C_T(a)$, or (Not possible if we ensure that $\mathcal{O} \models C_H \sqsubseteq C_T$)
- $\mathcal{O}, \mathcal{D} \not\models C_H(a)$ and $\mathcal{O}, \mathcal{D} \models C_T(a)$.

Need: generalize C_H such that $\mathcal{O}, \mathcal{D} \models C_H(a)$

Learning $\mathcal{EL}$-Concepts under Ontologies

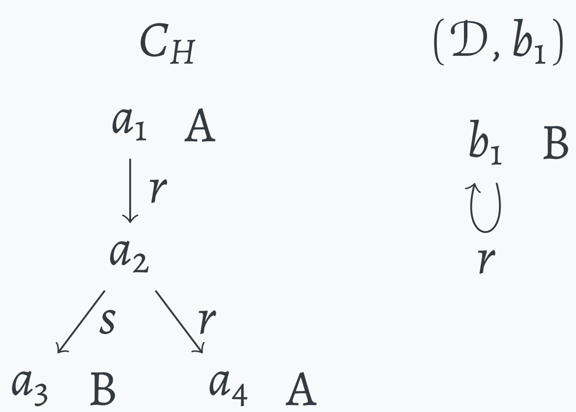
Counter examples and Products

Learner asks equivalence query with hypothesis C_H

If $\mathcal{O} \not\models C_H \equiv C_T$, then teacher returns counter example $(\mathcal{D}, a)$ such that

- $\mathcal{O}, \mathcal{D} \models C_H(a)$ and $\mathcal{O}, \mathcal{D} \not\models C_T(a)$, or (Not possible if we ensure that $\mathcal{O} \models C_H \sqsubseteq C_T$)
- $\mathcal{O}, \mathcal{D} \not\models C_H(a)$ and $\mathcal{O}, \mathcal{D} \models C_T(a)$.

Need: generalize C_H such that $\mathcal{O}, \mathcal{D} \models C_H(a)$

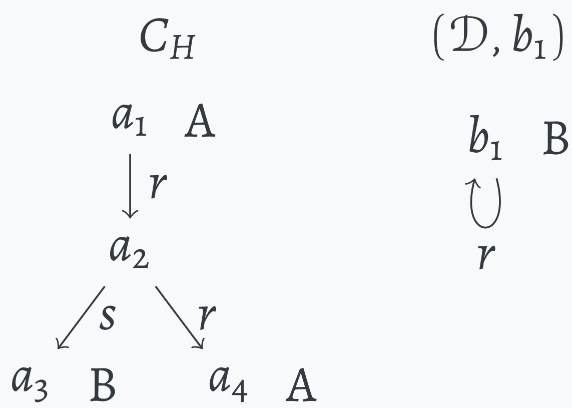


Learner asks equivalence query with hypothesis C_H

If $\mathcal{O} \not\models C_H \not\equiv C_T$, then teacher returns counter example $(\mathcal{D}, a)$ such that

- $\mathcal{O}, \mathcal{D} \models C_H(a)$ and $\mathcal{O}, \mathcal{D} \not\models C_T(a)$, or   (Not possible if we ensure that $\mathcal{O} \models C_H \sqsubseteq C_T$)
- $\mathcal{O}, \mathcal{D} \not\models C_H(a)$ and $\mathcal{O}, \mathcal{D} \models C_T(a)$.

Need: generalize C_H such that $\mathcal{O}, \mathcal{D} \models C_H(a)$ $\implies$ direct product $\times$



Learning $\mathcal{EL}$-Concepts under Ontologies

Counter examples and Products

Learner asks equivalence query with hypothesis C_H

If $\mathcal{O} \not\models C_H \not\equiv C_T$, then teacher returns counter example $(\mathcal{D}, a)$ such that

- $\mathcal{O}, \mathcal{D} \models C_H(a)$ and $\mathcal{O}, \mathcal{D} \not\models C_T(a)$, or (Not possible if we ensure that $\mathcal{O} \models C_H \sqsubseteq C_T$)
- $\mathcal{O}, \mathcal{D} \not\models C_H(a)$ and $\mathcal{O}, \mathcal{D} \models C_T(a)$.

Need: generalize C_H such that $\mathcal{O}, \mathcal{D} \models C_H(a)$ $\implies$ direct product $\times$

C_H

$$\begin{tikzcd}
 & a_1 \;\; \mathrm{A} \arrow[d, "r"] & \\
 & a_2 \arrow[dl, "s"'] \arrow[dr, "r"] & \\
a_3 \;\; \mathrm{B} & & a_4 \;\; \mathrm{A}
\end{tikzcd}$$

$(\mathcal{D}, b_1)$

$$\begin{tikzcd}
b_1 \;\; \mathrm{B} \arrow[loop below, "r"]
\end{tikzcd}$$

$C_H \times (\mathcal{D}, b_1)$

$$\begin{tikzcd}
 & (a_1, b_1) \arrow[d, "r"] & \\
 & (a_2, b_1) \arrow[dr, "r"] & \\
(a_3, b_1) \;\; \mathrm{B} & & (a_4, b_1)
\end{tikzcd}$$

Learning $\mathcal{EL}$-Concepts under Ontologies

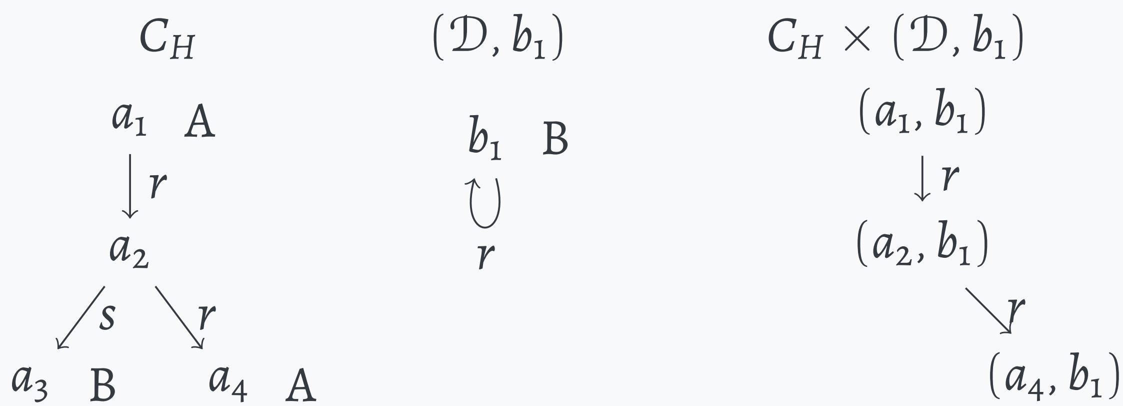
Counter examples and Products

Learner asks equivalence query with hypothesis C_H

If $\mathcal{O} \not\models C_H \not\equiv C_T$, then teacher returns counter example $(\mathcal{D}, a)$ such that

- $\mathcal{O}, \mathcal{D} \models C_H(a)$ and $\mathcal{O}, \mathcal{D} \not\models C_T(a)$, or (Not possible if we ensure that $\mathcal{O} \models C_H \sqsubseteq C_T$)
- $\mathcal{O}, \mathcal{D} \not\models C_H(a)$ and $\mathcal{O}, \mathcal{D} \models C_T(a)$.

Need: generalize C_H such that $\mathcal{O}, \mathcal{D} \models C_H(a)$ $\implies$ direct product $\times$



Learning $\mathcal{EL}$-Concepts under Ontologies

Input An $\mathcal{EL}$-ontology $\mathcal{O}$ and an $\mathcal{EL}$-concept $C_\circ$ such that $\mathcal{O} \models C_\circ \sqsubseteq C_T$
Output An $\mathcal{EL}$-concept C_H such that $\mathcal{O} \models C_H \equiv C_T$

$C_H \coloneqq C_\circ$
while the equivalence query $\mathcal{O} \models C_H \equiv C_T$ returns a counterexample $(\mathcal{D}, a)$ **do**
    $C'_H \coloneqq C_H \times (\mathcal{D}, a)$
    $C_H \coloneqq \mathsf{minimize}(C'_H)$
end while
return C_H

Input An $\mathcal{EL}$-ontology $\mathcal{O}$ and an $\mathcal{EL}$-concept C_0 such that $\mathcal{O} \models C_0 \sqsubseteq C_T$
Output An $\mathcal{EL}$-concept C_H such that $\mathcal{O} \models C_H \equiv C_T$

$C_H \coloneqq C_0$
while the equivalence query $\mathcal{O} \models C_H \equiv C_T$ returns a counterexample $(\mathcal{D}, a)$ **do**
    $C'_H \coloneqq C_H \times (\mathcal{D}, a)$
    $C_H \coloneqq \mathsf{minimize}(C'_H)$
end while
return C_H

For a counterexample $(\mathcal{D}, a)$ with

1. $C_H \sqsubseteq C_T$ and $\mathcal{D} \models C_T(a)$
2. $\mathcal{D} \not\models C_H(a)$,

it follows that $C_H \sqsubseteq C_H \times (\mathcal{D}, a) \sqsubseteq C_T$

Learning $\mathcal{EL}$-Concepts under Ontologies

Algorithm, first try

Input An $\mathcal{EL}$-ontology $\mathcal{O}$ and an $\mathcal{EL}$-concept C_0 such that $\mathcal{O} \models C_0 \sqsubseteq C_T$
Output An $\mathcal{EL}$-concept C_H such that $\mathcal{O} \models C_H \equiv C_T$

$C_H \coloneqq C_0$
while the equivalence query $\mathcal{O} \models C_H \equiv C_T$ returns a counterexample $(\mathcal{D}, a)$ **do**
  $C'_H \coloneqq C_H \times (\mathcal{D}, a)$
  $C_H \coloneqq \mathsf{minimize}(C'_H)$
end while
return C_H

For a counterexample $(\mathcal{D}, a)$ with

1. $C_H \sqsubseteq C_T$ and $\mathcal{D} \models C_T(a)$
2. $\mathcal{D} \not\models C_H(a)$,

it follows that $C_H \sqsubseteq C_H \times (\mathcal{D}, a) \sqsubseteq C_T$ and $C_H \times (\mathcal{D}, a) \not\sqsubseteq C_H$

$C_H \times (\mathcal{D}, a)$ does not work under $\mathcal{EL}$-ontologies. Let $\mathcal{O} = \{A \sqsubseteq \exists r.\top, \quad B \sqsubseteq \exists r.\top\}$

$C_H \times (\mathcal{D}, a)$ does not work under $\mathcal{EL}$-ontologies. Let $\mathcal{O} = \{A \sqsubseteq \exists r.\top, \quad B \sqsubseteq \exists r.\top\}$

C_H	$(\mathcal{D}, a)$	$C_H \times (\mathcal{D}, a)$	C_T
a_1 A	b_1 B		$a_1 \xrightarrow{r} a_2$

Learning $\mathcal{EL}$-Concepts under Ontologies

$C_H \times (\mathcal{D}, a)$ does not work under $\mathcal{EL}$-ontologies. Let $\mathcal{O} = \{A \sqsubseteq \exists r.\top, \quad B \sqsubseteq \exists r.\top\}$

C_H	$(\mathcal{D}, a)$	$C_H \times (\mathcal{D}, a)$	C_T
$a_1 \quad \mathrm{A}$	$b_1 \quad \mathrm{B}$	(a_1, b_1)	$a_1 \xrightarrow{r} a_2$

$\mathcal{O} \models C_H \sqsubseteq C_T$ and $\mathcal{O}, \mathcal{D} \models C_T(b_1)$, but $\mathcal{O} \not\models C_H \times (\mathcal{D}, a) \sqsubseteq C_T$.

Learning $\mathcal{EL}$-Concepts under Ontologies

$C_H \times (\mathcal{D}, a)$ does not work under $\mathcal{EL}$-ontologies. Let $\mathcal{O} = \{A \sqsubseteq \exists r.\top, \quad B \sqsubseteq \exists r.\top\}$

$$
\begin{array}{cccc}
C_H & (\mathcal{D}, a) & C_H \times (\mathcal{D}, a) & C_T \\
a_1 \;\; \mathrm{A} & b_1 \;\; \mathrm{B} & (a_1, b_1) & a_1 \\
\downarrow r & \downarrow r & & \downarrow r \\
a_2 & b_2 & & a_2
\end{array}
$$

$\mathcal{O} \models C_H \sqsubseteq C_T$ and $\mathcal{O}, \mathcal{D} \models C_T(b_1)$, but $\mathcal{O} \not\models C_H \times (\mathcal{D}, a) \sqsubseteq C_T$. Need to include consequences of $\mathcal{O}$

Learning $\mathcal{EL}$-Concepts under Ontologies

$C_H \times (\mathcal{D}, a)$ does not work under $\mathcal{EL}$-ontologies. Let $\mathcal{O} = \{A \sqsubseteq \exists r.\top, \quad B \sqsubseteq \exists r.\top\}$

$$
\begin{array}{cccc}
C_H & (\mathcal{D}, a) & C_H \times (\mathcal{D}, a) & C_T \\
a_1 \quad A & b_1 \quad B & (a_1, b_1) & a_1 \\
\downarrow r & \downarrow r & \color{red}{\downarrow r} & \downarrow r \\
a_2 & b_2 & \color{red}{(a_2, b_2)} & a_2
\end{array}
$$

$\mathcal{O} \models C_H \sqsubseteq C_T$ and $\mathcal{O}, \mathcal{D} \models C_T(b_1)$, but $\mathcal{O} \not\models C_H \times (\mathcal{D}, a) \sqsubseteq C_T$. Need to include consequences of $\mathcal{O}$

Learning $\mathcal{EL}$-Concepts under Ontologies

Compact Models

$C_H \times (\mathcal{D}, a)$ does not work under $\mathcal{EL}$-ontologies. Let $\mathcal{O} = \{A \sqsubseteq \exists r.\top, \quad B \sqsubseteq \exists r.\top\}$

$$\begin{array}{cccc}
C_H & (\mathcal{D}, a) & C_H \times (\mathcal{D}, a) & C_T \\
a_1 \;\; \text{A} & b_1 \;\; \text{B} & (a_1, b_1) & a_1 \\
\big\downarrow r & \big\downarrow r & \big\downarrow r & \big\downarrow r \\
a_2 & b_2 & (a_2, b_2) & a_2
\end{array}$$

$\mathcal{O} \models C_H \sqsubseteq C_T$ and $\mathcal{O}, \mathcal{D} \models C_T(b_1)$, but $\mathcal{O} \not\models C_H \times (\mathcal{D}, a) \sqsubseteq C_T$. Need to include consequences of $\mathcal{O}$

In $\mathcal{EL}$ there can be infinite consequences ($A \sqsubseteq \exists r.A$)

Learning $\mathcal{EL}$-Concepts under Ontologies

$C_H \times (\mathcal{D}, a)$ does not work under $\mathcal{EL}$-ontologies. Let $\mathcal{O} = \{A \sqsubseteq \exists r.\top, \quad B \sqsubseteq \exists r.\top\}$

$$
\begin{array}{cccc}
C_H & (\mathcal{D}, a) & C_H \times (\mathcal{D}, a) & C_T \\
a_1 \quad \mathrm{A} & b_1 \quad \mathrm{B} & (a_1, b_1) & a_1 \\
\downarrow r & \downarrow r & \downarrow r & \downarrow r \\
a_2 & b_2 & (a_2, b_2) & a_2
\end{array}
$$

$\mathcal{O} \models C_H \sqsubseteq C_T$ and $\mathcal{O}, \mathcal{D} \models C_T(b_1)$, but $\mathcal{O} \not\models C_H \times (\mathcal{D}, a) \sqsubseteq C_T$. Need to include consequences of $\mathcal{O}$

In $\mathcal{EL}$ there can be infinite consequences ($A \sqsubseteq \exists r.A$)

Fortunately, for $\mathcal{EL}$ there are compact universal models $\mathcal{G}_{C_H, \mathcal{O}}$ of ontologies (with size polynomial in $|C_H|$ and $|\mathcal{O}|$)

Input An $\mathcal{EL}$-ontology $\mathcal{O}$ and an $\mathcal{EL}$-concept $C_\circ$ such that $\mathcal{O} \models C_\circ \sqsubseteq C_T$
Output An $\mathcal{EL}$-concept C_H such that $\mathcal{O} \models C_H \equiv C_T$

$C_H \coloneqq C_\circ$
while the equivalence query $\mathcal{O} \models C_H \equiv C_T$ returns a counterexample $(\mathcal{D}, a)$ **do**
  $C'_H \coloneqq \mathcal{G}_{C_H,\mathcal{O}} \times \mathcal{G}_{\mathcal{D},\mathcal{O}}$
  $C_H \coloneqq \mathsf{minimize}(C'_H)$
end while
return C_H

Input An $\mathcal{EL}$-ontology $\mathcal{O}$ and an $\mathcal{EL}$-concept $C_\circ$ such that $\mathcal{O} \models C_\circ \sqsubseteq C_T$
Output An $\mathcal{EL}$-concept C_H such that $\mathcal{O} \models C_H \equiv C_T$

$C_H \coloneqq C_\circ$
while the equivalence query $\mathcal{O} \models C_H \equiv C_T$ returns a counterexample $(\mathcal{D}, a)$ **do**
  $C'_H \coloneqq \text{extract-el}(\mathcal{G}_{C_H,\mathcal{O}} \times \mathcal{G}_{\mathcal{D},\mathcal{O}})$
  $C_H \coloneqq \mathsf{minimize}(C'_H)$
end while
return C_H

Input An $\mathcal{EL}$-ontology $\mathcal{O}$ and an $\mathcal{EL}$-concept $C_\circ$ such that $\mathcal{O} \models C_\circ \sqsubseteq C_T$
Output An $\mathcal{EL}$-concept C_H such that $\mathcal{O} \models C_H \equiv C_T$

$C_H \coloneqq C_\circ$
while the equivalence query $\mathcal{O} \models C_H \equiv C_T$ returns a counterexample $(\mathcal{D}, a)$ **do**
  $C'_H \coloneqq \text{extract-el}(\mathcal{G}_{C_H,\mathcal{O}} \times \mathcal{G}_{\mathcal{D},\mathcal{O}})$
  $C_H \coloneqq \text{minimize}(C'_H)$
end while
return C_H

Theorem (F., Jung, Lutz 2021)

$\mathcal{EL}$-concepts are polynomial time learnable under $\mathcal{EL}$-ontologies

- Conjunctive Queries
  $\times$-based learning algorithm for conjunctive queries (ten Cate, Dalmau, Kolaitis 2013)

Learning $\mathcal{EL}$-Concepts under Ontologies

- Conjunctive Queries
  $\times$-based learning algorithm for conjunctive queries (ten Cate, Dalmau, Kolaitis 2013)
- More expressive concepts
  Also works for "symmetry-free" $\mathcal{ELI}$-concepts and "symmetry-free, chordal" $\mathcal{ELI}$-concepts (under $\mathcal{EL}$-ontologies, compact models exist) (F., Jung, Lutz 2021)

Learning $\mathcal{EL}$-Concepts under Ontologies

- Conjunctive Queries
  $\times$-based learning algorithm for conjunctive queries (ten Cate, Dalmau, Kolaitis 2013)
- More expressive concepts
  Also works for "symmetry-free" $\mathcal{ELI}$-concepts and "symmetry-free, chordal" $\mathcal{ELI}$-concepts (under $\mathcal{EL}$-ontologies, compact models exist) (F., Jung, Lutz 2021)
- More expressive ontology languages
  A similar approach works for $\textit{DL-Lite}_{horn}$-ontologies (F., Jung, Lutz 2022)

Learning $\mathcal{EL}$-Concepts under Ontologies

- Conjunctive Queries
  $\times$-based learning algorithm for conjunctive queries (ten Cate, Dalmau, Kolaitis 2013)
- More expressive concepts
  Also works for "symmetry-free" $\mathcal{ELI}$-concepts and "symmetry-free, chordal" $\mathcal{ELI}$-concepts (under $\mathcal{EL}$-ontologies, compact models exist) (F., Jung, Lutz 2021)
- More expressive ontology languages
  A similar approach works for $\mathit{DL\text{-}Lite}_{horn}$-ontologies (F., Jung, Lutz 2022)
  Does not work for $\mathcal{ELI}$-ontologes (No compact models)

Learning $\mathcal{EL}$-Concepts under Ontologies

- Conjunctive Queries
  $\times$-based learning algorithm for conjunctive queries (ten Cate, Dalmau, Kolaitis 2013)
- More expressive concepts
  Also works for "symmetry-free" $\mathcal{ELI}$-concepts and "symmetry-free, chordal" $\mathcal{ELI}$-concepts (under $\mathcal{EL}$-ontologies, compact models exist) (F., Jung, Lutz 2021)
- More expressive ontology languages
  A similar approach works for *DL-Lite*$_{horn}$-ontologies (F., Jung, Lutz 2022)
  Does not work for $\mathcal{ELI}$-ontologes (No compact models)

Theorem (F., Jung, Lutz 2021)

$\mathcal{EL}$-Concepts are not polynomal time learnable under $\mathcal{ELI}$-ontologies

References

[FJL21] Maurice Funk, Jean Christoph Jung, and Carsten Lutz. "Actively Learning Concepts and Conjunctive Queries under $\mathcal{EL}^r$-Ontologies". In: *Proc. of IJCAI*. 2021.

[FJL22a] Maurice Funk, Jean Christoph Jung, and Carsten Lutz. "Exact Learning of $\mathcal{ELI}$ Queries in the Presence of DL-Lite-Horn Ontologies". In: *Proc. of DL*. 2022.

[FJL22b] Maurice Funk, Jean Christoph Jung, and Carsten Lutz. "Frontiers and Exact Learning of $\mathcal{ELI}$ Queries under DL-Lite Ontologies". In: *Proc. of IJCAI*. 2022.

[Kri18] Francesco Kriegel. "The Distributive, Graded Lattice of EL Concept Descriptions and Its Neighborhood Relation". In: *Proc. of CLA*. Vol. 2123. 2018, pp. 267–278.

[tD21] Balder ten Cate and Victor Dalmau. "Conjunctive Queries: Unique Characterizations and Exact Learnability". In: *Proc. of ICDT*. Vol. 186. LIPIcs. 2021, 9:1–9:24.

[tDK13] Balder ten Cate, Víctor Dalmau, and Phokion G. Kolaitis. "Learning schema mappings". In: *ACM Trans. Database Syst.* 38.4 (2013), 28:1–28:31. DOI: 10.1145/2539032.2539035.

PAC Learning

PAC Learning — Motivation

So far concentrated on **fitting/separability problem**:
given positive/negative examples, find a concept/query that fits

Neglected the aspect of **generalization**
we want the fitting concept to generalize well to **unseen examples**

Leslie Valiant introduced PAC learning in a seminal paper in 1984

Notion of PAC (**probably — approximately — correct**) tries to capture generalization

Plan

1. Definition
2. Boundaries
3. Occams Razor & Bounded Fitting
4. SPELL demo

Statistical Machine Learning

unknown probability distribution D

draw examples i.i.d.

$e_1, \ldots, e_n$

label with unknown function f^*

$(e_1, f^*(e_1)), \ \ldots, \ (e_n, f^*(e_n))$

learning algorithm **A**

hypothesis h for f^*

Papayas on some unknown island

random papayas $P_1, \ldots, P_n$

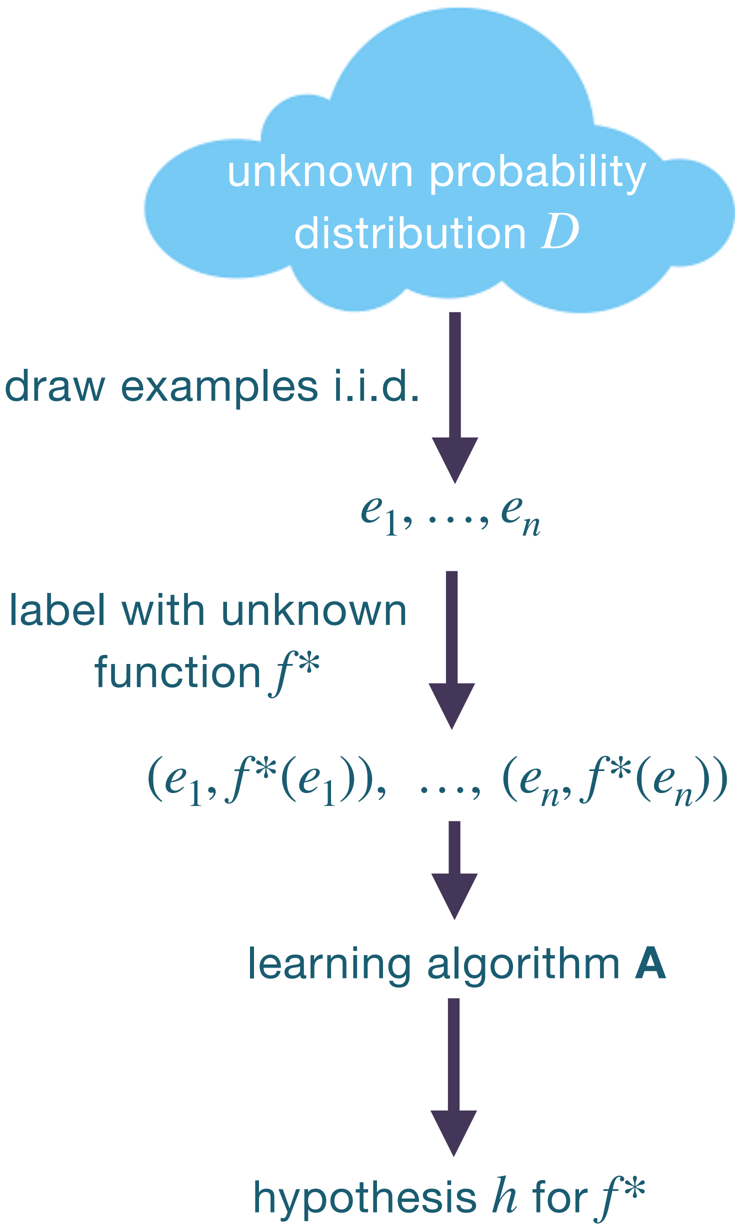
f^* = “nature” labels with “tasty” (+) or “not tasty (-)

$(P_1, +), \ \ldots, \ (P_n, -)$

rule predicting tastyness e.g.:

“if papaya is yellow and has size 10-13 cm and is not too soft, then it’s tasty”

Statistical Machine Learning



No access to all examples $\Rightarrow$ two kinds of errors

- h approximates f^* $\quad \Rightarrow \quad$ approximation parameter ϵ
- **A** works only probably $\quad \Rightarrow \quad$ confidence parameter δ

PAC learnability

Class Q is PAC learnable if there are (**A**,m_Q) s.t.:

- **A** is learning algorithm
- if given $m_Q(\epsilon, \delta, n)$ examples labeled by q of size n,
  A outputs with high probability $\geq 1 - \delta$
  a hypothesis $h \in Q$ with small error $\leq \epsilon$

$$\text{error}_{D,f^*}(h) = Pr_{e \sim D}(f^*(e) \neq h(e))$$

No Papayas today

Examples = databases, Q = class of queries

PAC Learnability

PAC learnability
Class Q is PAC learnable if there are (**A**,m_Q) s.t.:

- **A** is learning algorithm
- if given $m_Q(\epsilon, \delta, n)$ examples labeled by q of size n, **A** outputs with high probability $\geq 1 - \delta$ a hypothesis $h \in Q$ with small error $\leq \epsilon$

Theorem Every class of queries $Q \subseteq$ FO is PAC learnable.

Proof Every class of queries is union $Q = Q_1 \cup Q_2 \cup Q_3 \cup \ldots$ where Q_i is the size(i)-fragment of Q

Each Q_i is finite and any countable union of finite classes is PAC-learnable $\square$

Two notions of Efficiency:

a) polynomial time $\Rightarrow$ **efficient** PAC learning

b) m_Q polynomial in $n, 1/\varepsilon, 1/\delta$ $\Rightarrow$ **sample-efficient** PAC learning

Known:
a) $\Rightarrow$ b)

Failure of Efficient PAC

Q has **polynomial size fitting property if**
whenever a fitting query exists, there exists one of size polynomial in E^+, E^-

Q has **polynomial time evaluation property if**
there is a polynomial time algorithm

Theorem (Pitt & Valiant, J. ACM 1984)
Let Q be polynomial time evaluable and have the polynomial size fitting property.
Then: If Q is efficiently PAC learnable, then the fitting problem for Q is in RP.

Consequence Any class $Q \subseteq$ CQs containing all Path Queries is **not** efficiently PAC learnable (unless NP=RP).

Proof Assume Q is efficiently PAC learnable [ten Cate, Funk, J, Lutz 2023]
Q is also PAC learnable over the instances from the NP-hardness proof
1) Over these instances we have polynomial size fitting property
2) Instances are tree-shaped $\Rightarrow$ polynomial time evaluation
Apply Pitt & Valiant.

Efficient PAC with Membership Queries

Tight relation between Exact Learnability and PAC Learnability

Efficient exact learnability with EQ $\Rightarrow$ Efficient PAC learnability
(very often, converse also true)

Efficient exact learnability with EQ+MQ $\Rightarrow$ Efficient PAC learnability with MQs

Transfer results

query class	**ontology**	**questions**	**learnability**
$\mathcal{EL}$/$\mathcal{ELI}$-concepts	no	MQ	efficient
$\mathcal{EL}$-concepts	$\mathcal{EL}$	MQ+EQ	efficient
CQ	no	MQ+EQ	efficient
CQ/$\mathcal{ELI}$-concepts	$\mathcal{ELI}$	MQ+EQ	not efficient
CQ	DL-Lite/$\mathcal{EL}$	MQ+EQ	**open**
$\mathcal{ELI}$-concepts	$\mathcal{EL}$	MQ+EQ	**open**

Sample Efficiency of Product Algorithm

Recall the „product algorithm“ for $\mathscr{E}\mathscr{L}$:

Given concepts $C_1, \ldots, C_n$ and $D_1, \ldots, D_k$:
1. compute product concept $C := C_1 \times \ldots \times C_n$
2. if $D_i \not\sqsubseteq C$ for all i, return C
3. otherwise return „no fitting concept“

Product algorithm returns always the **most specific fitting concept**

Bad News Any fitting algorithm that returns a most specific fitting concept (if it exists), is **not** sample-efficient! [ten Cate, Funk, J, Lutz IJCAI’23]

So the „natural“ algorithm does not enjoy **generalization abilities**

Same holds for algorithms that
a) return the most general fitting concept
b) return the fitting concept with minimal quantifier depth

Occam's Razor (William of Ockham, 14th century)

„The simplest explanation is usually the best one"

Original formulation "Entities must not be multiplied beyond necessity"

(„multiplied" is here in the sense of „combined")

Computational Learning Theory has poured this intuition into the following definition

Learning algorithm **A** is an **Occam algorithm** if there are a polynomial p and $\alpha \in (0,1)$ such that **A** outputs a concept of size $p(s) \cdot m^{\alpha}$, where s is the size of the target and m is the number of examples.

Theorem Every Occam algorithm **A** is a **sample-efficient** PAC learning algorithm.

[Blumer, Ehrenfeucht, Haussler, & Warmuth J. ACM 1989]

$\Rightarrow$ Occam algorithms are one way to obtain sample-efficient PAC learning algorithms.

(For many hypothesis classes, a converse of this is also true)

Bounded Fitting

[ten Cate, Funk, J, Lutz IJCAI'23]

Input: Ontology $\mathcal{O}$, examples E^+, E^-

Bounded Fitting proceeds in rounds:

Round 1: search for a fitting concept of size 1
Round 2: search for a fitting concept of size 2
⋮
Round i: search for a fitting concept of size i
⋮

Return the first fitting concept that is found in this way

Similarity with **Bounded Model Checking** [Biere, Cimatti, Clarke, Zhu TACAS 1999]

Bounded fitting is Occam Algorithm (**independent** of ontology/query language)

sample efficient with complexity: $O(\frac{1}{\varepsilon} \cdot \log \frac{1}{\varepsilon} \cdot \frac{1}{\delta} \cdot \log |\Sigma| \cdot ||q_T||)$

Flexibility

- different size measures work
- different sequences such as 1, 2, 4, 8, … work

SPELL: Bounded Fitting for $\mathscr{EL}$

Observation Problem in

Round i: search for a fitting concept of size i

is **NP-complete** for ontology and query language $\mathscr{EL}$

Our system SPELL (⇒ https://github.com/spell-system/SPELL)

implements Bounded Fitting for $\mathscr{EL}$ leveraging a SAT solver

⇒ Talk tomorrow @DL with more information and detailed benchmarks

SPELL Demo