# Practical Forgetting and Uniform Interpolation for Description Logics

Renate A. Schmidt
Joint work with Patrick Koopmann and Yizheng Zhao

School of Computer Science
The University of Manchester

2 August 2015

# Outline

# Forgetting

- Restrict the scope of the vocabulary/signature of a knowledge base (set of formulae)

- $\mathcal{O}$ and $\mathcal{O}^{-\Sigma}$ are equivalent modulo symbols $\Sigma$ forgotten
- All entailments not using $\Sigma$-symbols are preserved

**Input ontology $\mathcal{O}$**

$$\begin{aligned}
\textbf{Margherita} &\sqsubseteq \forall \textbf{hasTopping}.(\mathit{Tomato} \\
&\qquad\qquad\qquad \sqcup \mathit{Mozarella}) \\
\textbf{American} &\sqsubseteq \exists \textbf{hasTopping}.\mathit{Tomato} \\
\textbf{American} &\sqsubseteq \exists \textbf{hasTopping}.\mathit{Mozarella} \\
\textbf{American} &\sqsubseteq \exists \textbf{hasTopping}.\mathit{Pepperoni} \\
\mathit{Tomato} &\sqsubseteq \textbf{Veg} \\
\mathit{Mozarella} &\sqsubseteq \textbf{Veg} \\
\mathit{Pepperoni} &\sqsubseteq \textbf{Meat}
\end{aligned}$$

**Forgetting solution $\mathcal{O}^{-\Sigma}$**

$$\begin{aligned}
\textbf{Margherita} &\sqsubseteq \forall \textbf{hasTopping}.\textbf{Veg} \\
\textbf{American} &\sqsubseteq \exists \textbf{hasTopping}.\textbf{Veg} \\
\textbf{American} &\sqsubseteq \exists \textbf{hasTopping}.\textbf{Meat}
\end{aligned}$$

# Utilisations of forgetting

- Ontology summarisation/generalisation
- Information hiding, access restriction and security
- Ontology reduction into smaller ontologies
- Efficiency
- Designing distributed ontologies
- Ontology analysis
- . . .

# Definition of forgetting

Given: $\mathcal{O}$ = set of formulae, ontology
$\Sigma$ = set of predicate symbols to be forgotten

Goal of forgetting: Compute ontology $\mathcal{O}^{-\Sigma}$ s. t.

1. $\mathcal{O} \models \alpha$ iff $\mathcal{O}^{-\Sigma} \models \alpha$ for any $\alpha$, if $\mathit{sig}(\alpha) \cap \Sigma = \emptyset$
2. $\mathit{sig}(\mathcal{O}^{-\Sigma}) \cap \Sigma = \emptyset$

Uniqueness: Solutions, if they exist, are unique modulo logical equivalence
Thus, $\mathcal{O}^{-\Sigma}$ = the forgetting solution

Equivalent notions:

uniform interpolation, projection,
second-order quantifier elimination

# Example of forgetting

## Ontology $\mathcal{O}$

$$\begin{aligned}
\textbf{VeggiePizza} &\sqsubseteq \exists\textbf{hasTopping}.\textbf{Tomato}\\
\textbf{VeggiePizza} &\sqsubseteq \exists\textbf{hasTopping}.\text{Zucchini}\\
\textbf{Tomato} &\sqsubseteq \textbf{Veg} \sqcap \neg\text{Zucchini}\\
\text{Zucchini} &\sqsubseteq \textbf{Veg} \sqcap \neg\textbf{Tomato}
\end{aligned}$$

## First-order encoding of $\mathcal{O}$

$$\begin{aligned}
&\forall x.\mathbf{VP}(x) \to \exists y.\mathbf{h}(x,y) \wedge \mathbf{T}(y)\\
&\forall x.\mathbf{VP}(x) \to \exists y.\mathbf{h}(x,y) \wedge Z(y)\\
&\forall x.\mathbf{T}(x) \to (\mathbf{V}(x) \wedge \neg Z(x))\\
&\forall x.Z(x) \to (\mathbf{V}(x) \wedge \neg\mathbf{T}(x))
\end{aligned}$$

## Forgetting Zucchini

$$\begin{aligned}
\textbf{VeggiePizza} &\sqsubseteq \exists\textbf{hasTopping}.\textbf{Tomato}\\
\textbf{VeggiePizza} &\sqsubseteq \exists\textbf{hasTopping}.\\
&\qquad (\textbf{Veg} \sqcap \neg\textbf{Tomato})\\
\textbf{Tomato} &\sqsubseteq \textbf{Veg}
\end{aligned}$$

## Forgetting Z

$$\begin{aligned}
&\forall x.\mathbf{VP}(x) \to \exists y.\mathbf{h}(x,y) \wedge \mathbf{T}(y)\\
&\forall x.\mathbf{VP}(x) \to \exists y.\mathbf{h}(x,y) \wedge\\
&\qquad \mathbf{V}(y) \wedge \neg\mathbf{T}(y)\\
&\forall x.\mathbf{T}(x) \to \mathbf{V}(x)
\end{aligned}$$

# Important property of minimal entailment

Forgetting solutions are minimal entailments of the given ontology

1. $\mathcal{O} \models \mathcal{O}^{-\Sigma}$
2. For any $\mathcal{O}'$,

$$\mathcal{O} \models \mathcal{O}' \quad \text{implies} \quad \mathcal{O}^{-\Sigma} \models \mathcal{O}' \qquad \text{when } sig(\mathcal{O}') \cap \Sigma = \emptyset$$

$\mathcal{O}^{-\Sigma}$ = strongest possible entailment of $\mathcal{O}$ without symbols in $\Sigma$

Provides an alternative definition

# Consequently

Forgetting is different to standard reasoning

- Forgetting cannot be reduced to a satisfiability or validity test
- The aim is to find formulae that are entailed by the given $\mathcal{O}$

Simple forgetting method

- Compute all possible entailments of $\mathcal{O}$
- Omit those containing $\Sigma$-symbols

But: In general, not feasible

# Forgetting with SCAN for FOL and FO clause logic

(Gabbay, Ohlbach, Engel 1992–)

First automated forgetting method

- Resolution method with constraints

Presented as a second-order quantifier elimination method

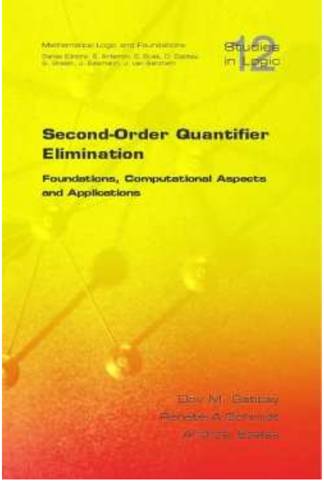
- In SOL, forgetting solution is: $\exists P_1 \ldots \exists P_n\ \mathcal{O}$
- $\exists P_1 \ldots \exists P_n\ \mathcal{O}$ is equivalent to $\mathcal{O}$ up to $\Sigma = \{P_1, \ldots, P_n\}$
- Aim of SOQE: find FO formula equivalent to $\exists\Sigma\ \mathcal{O}$
- No guarantee of success; SOQE for FOL is not always solvable

Automated modal correspondence theory

- SCAN solves frame correspondence problem of all modal logic axioms belonging to Sahlqvist class

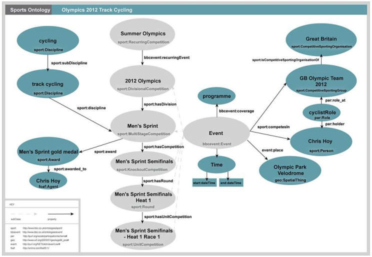
# Reference on SOQE

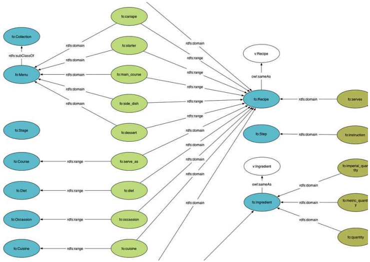
- Forgetting as SOQE
- Resolution-based methods, SCAN
- Ackermann-based method, DLS, DLS*
- Many applications



# Description logic based ontologies

- Description logics form part of the web ontology standard
- Used to define of common terms associated with an application domain
- Information represented as hierarchies of concepts and relationships between them
  $\rightsquigarrow$ ontologies
- Supported by efficient reasoning systems
  $\rightsquigarrow$ classification; consistency testing
- Emphasis has moved to: ontology-based query answering, rewritability, ontology management, repair, revision, merging, ...
- Automated forgetting/UI tools are going to be important



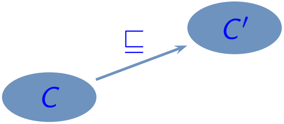




Source: `http://www.bbc.co.uk/ontologies/`

# Description logic $\mathcal{ALC}$

**Ontology statements**

$C \sqsubseteq C' \mid C(a) \mid r(a,b)$ possibly also $r \sqsubseteq r'$




$C, C'$ concepts $\rightsquigarrow$ sets
$r, r'$ (atomic) roles $\rightsquigarrow$ binary relations
$a, b$ individuals $\rightsquigarrow$ domain elements

**Concept expressions in $\mathcal{ALC}$**

$A \mid \neg C \mid C \sqcup C' \mid C \sqcap C' \mid \exists r.C \mid \forall r.C$

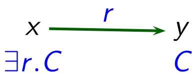
$A$ = concept name; $r$ = role name

**In $\mathcal{ALC}\nu$ also**

$\nu X.C$

$X$ = concept variable

$x \in (\exists r.C)^{\mathcal{I}}$ iff
$x$ is $r^{\mathcal{I}}$-related to some $y$ in $C^{\mathcal{I}}$




$x \in (\nu X.\exists r.X)^{\mathcal{I}}$ iff there is an infinite $r^{\mathcal{I}}$-path from $x$

# Challenges for forgetting for description logics

Applying SCAN to FO encoding of DL ontologies

- May produce a solution inexpressible in the relevant DL
- May fail, even though a DL solution exists

Using standard DL approaches

- Not suitable; designed for satisfiability testing and classification

Therefore:

- A new approach is needed

# Cyclic ontologies

Problem: Forgetting solution may not be finite

**Ontology $\mathcal{O}$**

$A \sqsubseteq B$

$B \sqsubseteq \exists r.B$

**Forgetting $B$**

$A \sqsubseteq \exists r.\exists r.\exists r.\exists r.\exists r.\exists r.\exists r.\exists r.\exists r.\ldots$

Possible solutions

- Approximate finitely: $A \sqsubseteq \exists r.\exists r.\exists r.\top$
- Use greatest fixpoint: $A \sqsubseteq \nu X.\exists r.X$
- Use definer symbols: $A \sqsubseteq D, \quad D \sqsubseteq \exists r.D$

By Ackermann's Generalised Lemma:

$$A \sqsubseteq \nu X.\exists r.X \quad \text{is equivalent to} \quad \exists D\{\, A \sqsubseteq D, \quad D \sqsubseteq \exists r.D \,\}$$

# Liberalised definition of forgetting for description logics

Given: $\mathcal{O}$ = set of formulae, ontology
$\Sigma$ = set of concept or role symbols to be forgotten

Goal of forgetting: Compute ontology $\mathcal{O}^{-\Sigma}$ s. t.

1. $\mathcal{O} \models \alpha$ iff $\mathcal{O}^{-\Sigma} \models \alpha$ for any $\alpha$, if $sig(\alpha) \cap \Sigma = \emptyset$
2. $sig(\mathcal{O}^{-\Sigma}) \cap \Sigma = \emptyset$
3. $\mathcal{O}^{-\Sigma}$ is expressed using a language extension, e.g., $\nu$ or definer symbols

The $\alpha$ can be of the form $C \sqsubseteq D$ or $C(a)$ or $r(a, b)$ and are expressed in the source logic

This talk: We mostly assume $\mathcal{O}$ contains only inclusions, the $\alpha$ are of the form $C \sqsubseteq D$ and $\Sigma$ is set of concept symbols
$\rightsquigarrow$ $\mathcal{O}^{-\Sigma}$ = TBox concept forgetting solution

# High complexity

- Testing satisfiability of $\mathcal{ALC}$ ontologies is EXPTIME-complete
- Reasoning in $\mathcal{ALC}\nu$ is EXPTIME-complete
- For $\mathcal{ALC}$, worst-case size of finite forgetting solutions in $\mathcal{ALC}$ is triple-exponential wrt. the input ontology (Lutz & Wolter 2011)

Theorem:

For $\mathcal{ALC}$, with $\nu$ operator (or definer symbols) in the target language,

1. Finite forgetting solutions always exist
2. The worst-case size is double-exponential wrt. the input ontology

# Key ingredients of our forgetting approach

Resolution-based, saturation method

Compute enough entailments, but not too many

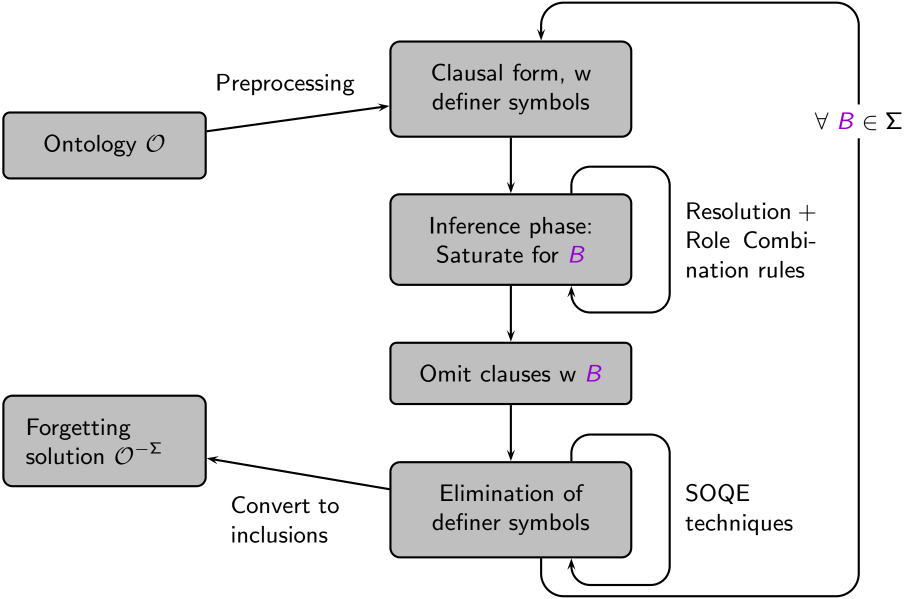
Goal-oriented

Represent entailments finitely

- internally: w definer symbols
- output: with $\nu$ or
  definer symbols

Redundancy elimination & optimisations

# Our approach



# Aspects of note

Quantifier restriction expressions are flat

Invariant: Max. 1 negative definer per clause

- Both ensured by introducing fresh definer symbols during clausification and the inference phase (number finitely bounded)
- Ensure any set of clauses can be converted into an $\mathcal{ALC}\nu$ ontology

$$C_1 \sqcup \mathrm{Q}r.C_2 \iff C_1 \sqcup \mathrm{Q}r.D,\ \neg D \sqcup C_2$$
$$C_1 \sqcup \nu X.C_2[X] \iff C_1 \sqcup \mathrm{Q}r.D,\ \neg D \sqcup C_2[D]$$

# Example

$C_1 \sqcup \exists r.B$ $\qquad$ $C_2 \sqcup \forall r.(\neg B \sqcup A)$ $\qquad$ (Input)

$\neg D_1 \sqcup B$ $\qquad$ $\neg D_2 \sqcup \neg B \sqcup A$ $\qquad$ (NF)

$C_1 \sqcup \exists r.D_1$ $\qquad$ $C_2 \sqcup \forall r.D_2$

# Example

Cannot resolve due to invariant

$\neg D_1 \sqcup B$
$C_1 \sqcup \exists r.D_1$

$\neg D_2 \sqcup \neg B \sqcup A$
$C_2 \sqcup \forall r.D_2$

# Example

Cannot resolve due to invariant

$\neg D_1 \sqcup B$
$C_1 \sqcup \exists r.D_1$

$\neg D_2 \sqcup \neg B \sqcup A$
$C_2 \sqcup \forall r.D_2$

Role combination

$C_1 \sqcup C_2 \sqcup \exists r.D_{12}$
$\neg D_{12} \sqcup B$
$\neg D_{12} \sqcup \neg B \sqcup A$

# Example

Cannot resolve due to invariant

$\neg D_1 \sqcup B$
$C_1 \sqcup \exists r.D_1$

$\neg D_2 \sqcup \neg B \sqcup A$
$C_2 \sqcup \forall r.D_2$

Role combination

$C_1 \sqcup C_2 \sqcup \exists r.D_{12}$
$\neg D_{12} \sqcup B$
$\neg D_{12} \sqcup \neg B \sqcup A$

Resolves to $\neg D_{12} \sqcup A$

# Forgetting calculus for $\mathcal{ALC}$

Resolution: $$\frac{C_1 \sqcup B, \quad \neg B \sqcup C_2}{C_1 \sqcup C_2}$$

$\exists\forall$ Role combination: $$\frac{C_1 \sqcup \exists r.D_1, \quad \forall r.D_2 \sqcup C_2}{C_1 \sqcup C_2 \sqcup \exists r.D_{12}}$$

$\forall\forall$ Role combination: $$\frac{C_1 \sqcup \forall r.D_1, \quad \forall r.D_2 \sqcup C_2}{C_1 \sqcup C_2 \sqcup \forall r.D_{12}}$$

- $B \in \Sigma$, or $B$ a definer symbol
- All resolvents contain at most 1 negated definer symbol
- $D_{12}$ defines $D_1 \sqcap D_2$ uniquely

# Results for $\mathcal{ALC}$

**Theorem**

- For any $\mathcal{ALC}$-TBox, a finite $\mathcal{ALC}\nu$-representation of the forgetting solution is computed
- A double exponential number of clauses are derived

- Results hold for the elimination of concept and role symbols
- The method decides satisfiability of $\mathcal{ALC}\nu$-ontologies

# Method and results have been extended to

**Role inclusions**

$\text{isFatherOf} \sqsubseteq \text{isParentOf}$

**Transitive roles**

**Number restrictions**

$\text{VeggiePizza} \sqsubseteq \geq 2\ \text{hasTopping.Veg}$

Target language, in all cases: allow $\nu$ or definer symbols

**$\mathcal{ALC}$-ontologies with TBoxes and ABoxes**

Target language: allow $\nu$ and ABox disjunction

$\text{Tomato}(t) \vee \text{Tomato}(z)$

ABox disjunctions can be avoided, if nominals are allowed

$\{\text{t}\}$

In all cases our forgetting methods also provide decision procedures for satisfiability checking

'River of Forgetfulness'

Core functionality

- **Forgetting and UI** for expressive DLs (from $\mathcal{ALC}$ up to $\mathcal{SHQ}$)

Additional functionality via reduction to forgetting

- **Ontology abduction:** For a given ontology $\mathcal{O}$ and a goal $G$, generate the weakest hypothesis $H$ such that $\mathcal{O}, H \models G$
  $\Sigma$ specifies the symbols not allowed
  Useful for ontology development
- **Logical Difference:** not using symbols in $\Sigma$
  'diff' for two ontologies; useful for ontology analysis and comparison
  Forget $\Sigma$ from $\mathcal{O}_2$; if $\alpha \in \mathcal{O}_2^{-\Sigma}$ but $\mathcal{O}_1 \not\models \alpha$ then $\alpha \in$ log. diff.

Usage from command line, as Java library, or via GUI

# Evaluation

| $\mathcal{ALCH}$, forget 50 symbols | |
|---|---|
| Success Rate: | 91.10% |
| Without Fixpoints: | 95.29% |
| Duration Mean: | 7.68 sec. |
| Duration Median: | 2.74 sec. |
| Duration 90th percentile: | 12.45 sec. |

| $\mathcal{ALCH}$, forget 100 symbols | |
|---|---|
| Success Rate: | 88.10% |
| Without Fixpoints: | 93.27% |
| Duration Mean: | 18.03 sec. |
| Duration Median: | 3.81 sec. |
| Duration 90th percentile: | 21.17 sec. |

| $\mathcal{ALC}$ w. ABoxes, forget 50 symbols | |
|---|---|
| Success Rate: | 94.79% |
| Without Fixpoints: | 92.91% |
| Duration Mean: | 23.94 sec. |
| Duration Median: | 3.01 sec. |
| Duration 90th percentile: | 29.00 sec. |

| $\mathcal{ALC}$ w. ABoxes, forget 100 symbols | |
|---|---|
| Success Rate: | 91.37% |
| Without Fixpoints: | 92.48% |
| Duration Mean: | 57.87 sec. |
| Duration Median: | 6.43 sec. |
| Duration 90th percentile: | 99.26 sec. |

| $\mathcal{SHQ}$, forget 50 concept symbols | |
|---|---|
| Success Rate: | 95.83% |
| Without Fixpoints: | 93.40% |
| Duration Mean: | 7.62 sec. |
| Duration Median: | 1.04 sec. |
| Duration 90th percentile: | 4.89 sec. |

| $\mathcal{SHQ}$, forget 100 concept symbols | |
|---|---|
| Success Rate: | 90.77% |
| Without Fixpoints: | 91.99% |
| Duration Mean: | 13.51 sec. |
| Duration Median: | 1.60 sec. |
| Duration 90th percentile: | 11.65 sec. |

Corpus: Respective fragments of 306 ontologies from BioPortal having at most 100,000 axioms

Timeout: 30 minutes

# Conclusion

- For expressive description logics, despite the high complexity, forgetting is often solvable with language enhancements
- Our methods always produce finite representations of forgetting solutions
- Solutions are more compact
- Experiments show practicality and often no fixpoints are needed

# Future work

Extend to more expressive description logics: allow inverse & nominals

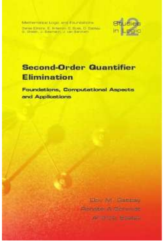
- We have already got methods and results for $\mathcal{SIF}$ and $\mathcal{SHI}$
- Yizheng Zhao has developed a method for $\mathcal{ALCOI}$
- Ackermann-based approach with novel techniques
- incomplete; nevertheless success rates $\geq 95\%$

Evaluation of abduction and logical difference

Other use cases of forgetting

# References

Second-order quantifier elimination




For description logics using resolution-based approach
FroCoS13, WoMo13, LPAR13, IJCAR14, DL14, AAAI15, DL15, ORE15

For description logics using Ackermann-based approach
DL15, ISWC15