Outline

1. Goal of forgetting & applications
2. Forgetting for description logics
3. Our approach
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
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
\[ \Sigma = \text{set of predicate symbols to be forgotten} \]

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 \( \text{sig}(\alpha) \cap \Sigma = \emptyset \)
2. \( \text{sig}(\mathcal{O}^-\Sigma) \cap \Sigma = \emptyset \)

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

<table>
<thead>
<tr>
<th>Ontology $\mathcal{O}$</th>
<th>First-order encoding of $\mathcal{O}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>VeggiePizza $\sqsubseteq \exists \text{hasTopping. Tomato}$</td>
<td>$\forall x. VP(x) \rightarrow \exists y. h(x, y) \land T(y)$</td>
</tr>
<tr>
<td>VeggiePizza $\sqsubseteq \exists \text{hasTopping. Zucchini}$</td>
<td>$\forall x. VP(x) \rightarrow \exists y. h(x, y) \land Z(y)$</td>
</tr>
<tr>
<td>Tomato $\sqsubseteq \text{Veg} \sqcap \neg \text{Zucchini}$</td>
<td>$\forall x. T(x) \rightarrow (V(x) \land \neg Z(x))$</td>
</tr>
<tr>
<td>Zucchini $\sqsubseteq \text{Veg} \sqcap \neg \text{Tomato}$</td>
<td>$\forall x. Z(x) \rightarrow (V(x) \land \neg T(x))$</td>
</tr>
</tbody>
</table>

Forgetting Zucchini

<table>
<thead>
<tr>
<th>Forgetting Zucchini</th>
<th>Forgetting Z</th>
</tr>
</thead>
<tbody>
<tr>
<td>VeggiePizza $\sqsubseteq \exists \text{hasTopping. Tomato}$</td>
<td>$\forall x. VP(x) \rightarrow \exists y. h(x, y) \land T(y)$</td>
</tr>
<tr>
<td>VeggiePizza $\sqsubseteq \exists \text{hasTopping.}$</td>
<td>$\forall x. VP(x) \rightarrow \exists y. h(x, y) \land$</td>
</tr>
<tr>
<td>(Veg $\sqcap \neg$ Tomato)</td>
<td>$V(y) \land \neg T(y)$</td>
</tr>
<tr>
<td>Tomato $\sqsubseteq \text{Veg}$</td>
<td>$\forall x. T(x) \rightarrow V(x)$</td>
</tr>
</tbody>
</table>
Important property of minimal entailment

Forgetting solutions are minimal entailments of the given ontology

1. \( O \models O^{-\Sigma} \)
2. For any \( O' \),
   \[ O \models O' \; \text{implies} \; O^{-\Sigma} \models O' \; \text{when} \; \text{sig}(O') \cap \Sigma = \emptyset \]

\( O^{-\Sigma} \) = strongest possible entailment of \( O \) 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 $O$

Simple forgetting method

- Compute all possible entailments of $O$
- 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: \( \exists P_1 \ldots \exists P_n \, O \)
- \( \exists P_1 \ldots \exists P_n \, O \) is equivalent to \( O \) up to \( \Sigma = \{P_1, \ldots, P_n\} \)
- Aim of SOQE: find FO formula equivalent to \( \exists \Sigma \, O \)
- 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 \( \leadsto \) ontologies
- Supported by efficient reasoning systems \( \leadsto \) 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

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

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

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

$x \in (\nu X.\exists r.X)^I$ iff there is an infinite $r^I$-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

Possible solutions

- Approximate finitely: \( A \sqsubseteq \exists r.\exists r.\exists r.\exists r.\exists r.\exists r.\exists r.\exists r.\ldots \)
- 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 $\text{sig}(\alpha) \cap \Sigma = \emptyset$
2. $\text{sig}(\mathcal{O}^\Sigma) \cap \Sigma = \emptyset$
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
$\leadsto \mathcal{O}^\Sigma = \text{TBox concept forgetting solution}$
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

**Ontology \( \mathcal{O} \)**

**Forgetting solution \( \mathcal{O}^{-\Sigma} \)**

**Preprocessing**

**Clausal form, w definer symbols**

**Inference phase: Saturate for \( B \)**

**Omit clauses w \( B \)**

**Elimination of definer symbols**

\[ \forall B \in \Sigma \]

**Resolution + Role Combination rules**

**SOQE techniques**

Convert to inclusions
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

\[
C_1 \uplus Qr.C_2 \iff C_1 \uplus Qr.D, \neg D \uplus C_2
\]

\[
C_1 \uplus \nu X.C_2[X] \iff C_1 \uplus Qr.D, \neg D \uplus C_2[D]
\]
Example

\[ C_1 \sqcup \exists r. B \]
\[ \neg D_1 \sqcup B \]
\[ C_1 \sqcup \exists r. D_1 \]

\[ C_2 \sqcup \forall r. (\neg B \sqcup A) \]
\[ \neg D_2 \sqcup \neg B \sqcup A \]
\[ C_2 \sqcup \forall r. D_2 \]

(Input)

(NF)
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:

\[
\begin{align*}
C_1 \sqcup B, \quad & \neg B \sqcup C_2 \\
\hline
C_1 \sqcup C_2
\end{align*}
\]

\( \exists \forall \) Role combination:

\[
\begin{align*}
C_1 \sqcup \exists r.D_1, \quad & \forall r.D_2 \sqcup C_2 \\
\hline
C_1 \sqcup C_2 \sqcup \exists r.D_{12}
\end{align*}
\]

\( \forall \forall \) Role combination:

\[
\begin{align*}
C_1 \sqcup \forall r.D_1, \quad & \forall r.D_2 \sqcup C_2 \\
\hline
C_1 \sqcup C_2 \sqcup \forall r.D_{12}
\end{align*}
\]

- \( B \in \Sigma \), 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**

**ALC-ontologies with TBoxes and ABoxes**

Target language: allow \( \nu \) and ABox disjunction
\[ \text{Tomato}(t) \lor \text{Tomato}(z) \]
ABox disjunctions can be avoided, if nominals are allowed
\[ \{ t \} \]

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 $\alpha \in \text{log. diff.}$

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

<table>
<thead>
<tr>
<th></th>
<th>(ALCH), forget 50 symbols</th>
<th>(ALCH), forget 100 symbols</th>
</tr>
</thead>
<tbody>
<tr>
<td>Success Rate:</td>
<td>91.10%</td>
<td>88.10%</td>
</tr>
<tr>
<td>Without Fixpoints:</td>
<td>95.29%</td>
<td>93.27%</td>
</tr>
<tr>
<td>Duration Mean:</td>
<td>7.68 sec.</td>
<td>18.03 sec.</td>
</tr>
<tr>
<td>Duration Median:</td>
<td>2.74 sec.</td>
<td>3.81 sec.</td>
</tr>
<tr>
<td>Duration 90th percentile:</td>
<td>12.45 sec.</td>
<td>21.17 sec.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>(ALC) w. ABoxes, forget 50 symbols</th>
<th>(ALC) w. ABoxes, forget 100 symbols</th>
</tr>
</thead>
<tbody>
<tr>
<td>Success Rate:</td>
<td>94.79%</td>
<td>91.37%</td>
</tr>
<tr>
<td>Without Fixpoints:</td>
<td>92.91%</td>
<td>92.48%</td>
</tr>
<tr>
<td>Duration Mean:</td>
<td>23.94 sec.</td>
<td>57.87 sec.</td>
</tr>
<tr>
<td>Duration Median:</td>
<td>3.01 sec.</td>
<td>6.43 sec.</td>
</tr>
<tr>
<td>Duration 90th percentile:</td>
<td>29.00 sec.</td>
<td>99.26 sec.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>(SHQ), forget 50 concept symbols</th>
<th>(SHQ), forget 100 concept symbols</th>
</tr>
</thead>
<tbody>
<tr>
<td>Success Rate:</td>
<td>95.83%</td>
<td>90.77%</td>
</tr>
<tr>
<td>Without Fixpoints:</td>
<td>93.40%</td>
<td>91.99%</td>
</tr>
<tr>
<td>Duration Mean:</td>
<td>7.62 sec.</td>
<td>13.51 sec.</td>
</tr>
<tr>
<td>Duration Median:</td>
<td>1.04 sec.</td>
<td>1.60 sec.</td>
</tr>
<tr>
<td>Duration 90th percentile:</td>
<td>4.89 sec.</td>
<td>11.65 sec.</td>
</tr>
</tbody>
</table>

**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 $SIF$ and $SHI$
- Yizheng Zhao has developed a method for $ALCOI$
- Ackermann-based approach with novel techniques
- incomplete; nevertheless success rates $\geq 95\%$

Evaluation of abduction and logical difference

Other use cases of forgetting
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