Towards Polynomial-time Forgetting and Instance Query Rewriting in Ontology Languages

Sen Zheng   Renate A. Schmidt

School of Computer Science,
The University of Manchester

April 12, 2018

Overview

Aim:

- polynomial-time forgetting of guards for the guarded fragment (GF)
- polynomial-time instance query rewriting of role symbols in description logics $\mathcal{ALCOI}$

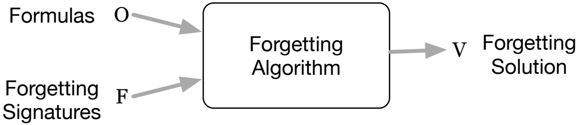
Overview

Aim:

- polynomial-time forgetting of guards for the guarded fragment (GF)
- polynomial-time instance query rewriting of role symbols in description logics $\mathcal{ALCOI}$

- What is forgetting?
- The guarded fragment?
- Instance query rewriting for $\mathcal{ALCOI}$?

Forgetting

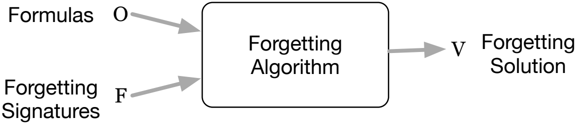


Goal:

Derive $\mathcal{V}$ such that:

- $sig(\mathcal{V}) \subseteq sig(\mathcal{O}) \setminus \mathcal{F}$
- $\mathcal{O}$ and $\mathcal{V}$ are equivalent up to the interpretation of $\mathcal{F}$

Forgetting



Goal:

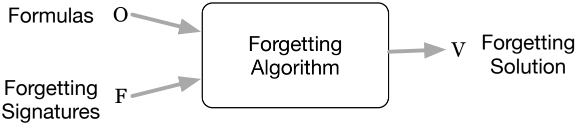
Derive $\mathcal{V}$ such that:

- $sig(\mathcal{V}) \subseteq sig(\mathcal{O}) \setminus \mathcal{F}$
- $\mathcal{O}$ and $\mathcal{V}$ are equivalent up to the interpretation of $\mathcal{F}$

Example:

$\mathcal{O}$: $\text{Postdoc}(x) \rightarrow \text{Researcher}(x)$ $\mathcal{F} = \{\text{Postdoc}\}$
$\text{Postdoc}(\text{Ann})$

Forgetting



Formulas O

Forgetting Signatures F

Forgetting Algorithm

V Forgetting Solution

Goal:

Derive $\mathcal{V}$ such that:

- $sig(\mathcal{V}) \subseteq sig(\mathcal{O}) \setminus \mathcal{F}$
- $\mathcal{O}$ and $\mathcal{V}$ are equivalent up to the interpretation of $\mathcal{F}$

Example:

$\mathcal{O}$: $\text{Postdoc}(x) \rightarrow \text{Researcher}(x)$ $\quad \mathcal{F} = \{\text{Postdoc}\}$
$\quad$ Postdoc(Ann)

$\mathcal{V}$: Researcher(Ann)

Forgetting

Applications

- Uniform interpolation.
- Second-order quantifier elimination.
- Query rewriting in ontology-based data access.
- Ontology debugging.
- Abduction reasoning.

Tools:

- SCAN, first-order logic, resolution
- LETHE, description logic, resolution
- FAME, description logic, Ackermann approach

the Guarded Fragment

The guarded fragment (GF)

- A decidable fragment of first-order logic (FOL).
- FOL translations of description logic $\mathcal{ALCI}$.

Definition:

$\bot \quad | \quad A \quad | \quad \phi \vee \phi \quad | \quad \neg\phi \quad | \quad \forall x(G \rightarrow \phi)$

where free variables of ϕ occur in the guard atom G.

the Guarded Fragment

The guarded fragment (GF)

- A decidable fragment of first-order logic (FOL).
- FOL translations of description logic $\mathcal{ALCI}$.

Definition:

$\bot \quad | \quad A \quad | \quad \phi \vee \phi \quad | \quad \neg\phi \quad | \quad \forall x(G \rightarrow \phi)$

where free variables of ϕ occur in the guard atom G.

Input: A set of non-nested guarded formulas with equality and constants.

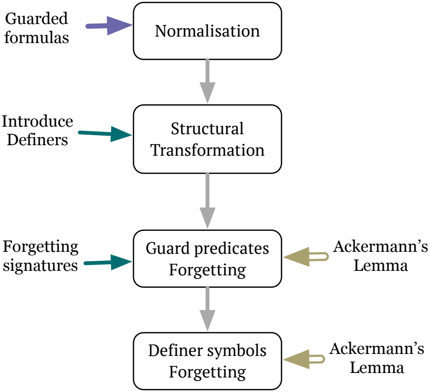
Purpose:

- The output guarded formulas are semantically equivalent to the input formulas up to $\mathcal{F}$.
- The complexity is polynomial.

Ackermann's Lemma-based forgetting approach

There are 4 major steps:

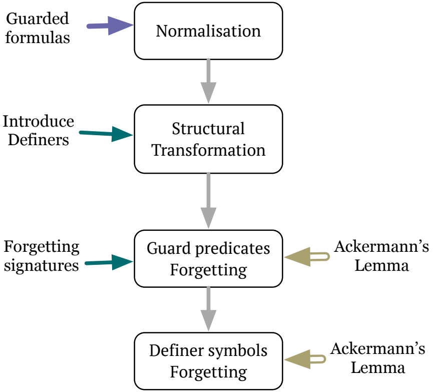
1. Add quantifiers for free variables and negation normal form transformation.
2. Introduce definers to flatten formulas.
3. Incrementally forget guard predicates in $\mathcal{F}$.
4. Incrementally forget definers.



Ackermann's Lemma-based forgetting approach

There are 4 major steps:

1. Add quantifiers for free variables and negation normal form transformation.
2. Introduce definers to flatten formulas.
3. Incrementally forget guard predicates in $\mathcal{F}$.
4. Incrementally forget definers.



Guarded formulas

Normalisation

Introduce Definers

Structural Transformation

Forgetting signatures

Guard predicates Forgetting

Ackermann's Lemma

Definer symbols Forgetting

Ackermann's Lemma

Conclusions:

- Polynomial-time complexity.
- The first approach to forget the guard predicates.

An instance query rewriting example

Instance query: A query that contains only one atom (a role).

An instance query rewriting example

Instance query: A query that contains only one atom (a role).

Knowledge base:

$\mathsf{Researcher} \sqsubseteq \exists \mathsf{worksFor}$

worksFor(Alice, WebCure)

Researcher(Cook)

Who works for any project?

$q(x) = \exists y\ \mathsf{worksFor}(x, y)$

An instance query rewriting example

Instance query: A query that contains only one atom (a role).

Knowledge base:
Researcher $\sqsubseteq$ $\exists$worksFor
worksFor(Alice, WebCure)
Researcher(Cook)

Who works for any project?
$q(x) = \exists y$ worksFor(x, y)

$\mathcal{F} = \{\text{worksFor}\}$.
ANS: an answer predicate tracing the variable x in $q(x)$.

KB:	$\neg Researcher(x) \vee \exists y\ worksFor(x, y)$	1
q:	$\neg worksFor(x, y) \vee ANS(x)$	2
AL on 1, 2:	$\neg Researcher(x) \vee ANS(x)$	3

An instance query rewriting example

Instance query: A query that contains only one atom (a role).

Knowledge base:
Researcher $\sqsubseteq \exists$worksFor
worksFor(Alice, WebCure)
Researcher(Cook)

Who works for any project?
$q(x) = \exists y \text{ worksFor}(x, y)$

$\mathcal{F} = \{\text{worksFor}\}$.
ANS: an answer predicate tracing the variable x in $q(x)$.

KB:	$\neg Researcher(x) \vee \exists y\ worksFor(x, y)$	1
q:	$\neg worksFor(x, y) \vee ANS(x)$	2
AL on 1, 2:	$\neg Researcher(x) \vee ANS(x)$	3

$q_1 = \exists y \text{ worksFor}(x, y)$, $q_2 = Researcher(x)$

An instance query rewriting example

Instance query: A query that contains only one atom (a role).

Knowledge base:
Researcher $\sqsubseteq$ $\exists$worksFor
worksFor(Alice, WebCure)
Researcher(Cook)

Who works for any project?
$q(x) = \exists y \text{ worksFor}(x, y)$

$\mathcal{F} = \{\text{worksFor}\}$.
ANS: an answer predicate tracing the variable x in $q(x)$.

KB:	$\neg Researcher(x) \vee \exists y\ worksFor(x, y)$	1
q:	$\neg worksFor(x, y) \vee ANS(x)$	2
AL on 1, 2:	$\neg Researcher(x) \vee ANS(x)$	3

$q_1 = \exists y \text{ worksFor}(x, y)$, $q_2 = Researcher(x)$

Answer set: {Alice, Cook}

Conclusions and ongoing work

- It is a polynomial-time method to forget guard predicates in non-nested guarded formulas.
- This approach is a partial instance query rewriting method for $\mathcal{ALCOI}$.

- The current work focus on expressing the $\mathcal{ALCOI}$ forgetting results into queries.

Thank You!