# Putting ABox Updates into Action

Franz Baader, Conrad Drescher, Steffen Guhlemann, Hongkai Liu, Uwe Petersohn, Peter Steinke, Michael Thielscher

Technische Universität Dresden

16.09.2009

# Dynamic DL — ABox Update by Example

DL ABoxes represent knowledge about individuals, e.g.

$$\forall \text{similar\_patient.Drug-tolerant(Mary)}$$

An update might be

$$\neg \text{Drug-tolerant(Jane)}$$

# Description Logic ABox Update

ABox update introduced at KR06

- Deterministic effects
- No domain constraints
- Winslett semantics

# Description Logic ABox Update

ABox update introduced at KR06

- Deterministic effects
- No domain constraints
- Winslett semantics

What's special?

- Open World Semantics
- Quantification
- More expressive than propositional logic
- Both UNA and Non-UNA domains supported

New territory for implemented action languages

# Implementing ABox Update

**Two main challenges:**

- Keep updated ABoxes small
- Reason with updated (i.e. Boolean) ABoxes: DL reasoners support only non-Boolean ABoxes

# Preliminaries

# Description Logics...

- decidable fragments of first order logic
- based on unary/binary predicates (concepts/roles)
- only constants (no function symbols)
- allow only certain formulas (via constructors)

# Description Logics...

- decidable fragments of first order logic
- based on unary/binary predicates (concepts/roles)
- only constants (no function symbols)
- allow only certain formulas (via constructors)

In this work we use

- $\mathcal{ALCO}^{@}$ (PSPACE-complete):
  smallest "real" DL closed under update
- $\mathcal{ALCO}^{+}$ (NEXPTIME-complete):
  admits smaller updated ABoxes

# Basic DL $\mathcal{ALCO}$

The concept constructors of $\mathcal{ALCO}$:

| **Name** | **DL Syntax** | **In FOL** |
|---|---|---|
| negation | $\neg C$ | $\neg C(x)$ |
| conjunction | $C \sqcap D$ | $C(x) \wedge D(x)$ |
| disjunction | $C \sqcup D$ | $C(x) \vee D(x)$ |
| nominal | $\{a\}$ | $x = A$ |
| existential restriction | $\exists r.C$ | $\exists y(r(x,y) \wedge C(y))$ |
| universal restriction | $\forall r.C$ | $\forall y(r(x,y) \supset C(y))$ |

$\mathcal{ALCO}^{@}$ is $\mathcal{ALCO}$ plus the @-constructor

| Name | DL Syntax | In FOL |
|---|---|---|
| @ constructor | $@_a C$ | $C(A)$ |

$\mathcal{ALCO}^{@}$ is $\mathcal{ALCO}$ plus the @-constructor

| Name | DL Syntax | In FOL |
|---|---|---|
| @ constructor | $@_a C$ | $C(A)$ |

$\mathcal{ALCO}^{+}$ is $\mathcal{ALCO}$ plus role constructors

| Name | DL Syntax | In FOL |
|---|---|---|
| role negation | $\neg r$ | $\neg R(x, y)$ |
| role conjunction | $q \sqcap r$ | $Q(x, y) \wedge R(x, y)$ |
| role disjunction | $q \sqcup r$ | $Q(x, y) \vee R(x, y)$ |
| nominal role | $\{(a, b)\}$ | $x = A \wedge y = B$ |

# Assertions, and ABoxes

Assertions are of the form $r(a,b)$ and $C(a)$, where

- concept $C$ may be complex
  **E.g.** $(C \sqcup @_b D)(a)$ $[C(A) \vee D(B)]$
- role assertions are literals ($\mathcal{ALCO}^{@}$) or complex ($\mathcal{ALCO}^{+}$)
  **E.g.** $(r \sqcup \{(a,b)\})(c,d)$ $[R(A,B) \vee (A = C \wedge B = D)]$

# Assertions, and ABoxes

Assertions are of the form $r(a, b)$ and $C(a)$, where

- concept $C$ may be complex
  **E.g.** $(C \sqcup @_b D)(a)$ $\quad [C(A) \vee D(B)]$
- role assertions are literals ($\mathcal{ALCO}^{@}$) or complex ($\mathcal{ALCO}^{+}$)
  **E.g.** $(r \sqcup \{(a, b)\})(c, d)$ $\quad [R(A, B) \vee (A = C \wedge B = D)]$

An ABox is a conjunction of assertions

A Boolean ABox is a Boolean combination of assertions
(Negation can be pushed inside assertions)

# ABox Update

- Based on Winslett semantics: deterministic update only
- Update only with concept/role literals
- Updated $\mathcal{ALCO}^{@}$ ABoxes exponential in ABox *and* update
- Updated $\mathcal{ALCO}^{+}$ ABoxes exponential in update

# ABox Update

- Based on Winslett semantics: deterministic update only
- Update only with concept/role literals
- Updated $\mathcal{ALCO}^{@}$ ABoxes exponential in ABox *and* update
- Updated $\mathcal{ALCO}^{+}$ ABoxes exponential in update

We only consider singleton updates $\mathcal{U} = \{\delta(\vec{t})\}$:
sufficient, easier presentation

# ABox Update II

Assertion $A(a)$ updated by $\neg A(b)$:

$$(\neg A(b) \wedge A(a)) \vee (\neg A(b) \wedge A \sqcup \{b\}(a))$$

# ABox Update II

Assertion $A(a)$ updated by $\neg A(b)$:

$$(\neg A(b) \wedge A(a)) \vee (\neg A(b) \wedge A \sqcup \{b\}(a))$$

Updating ABox $\mathcal{A}$ with $\mathcal{U}$ to $\mathcal{A}$' is defined as

$$\mathcal{A}' = \bigwedge(\mathcal{A} \cup \mathcal{U}) \vee \bigwedge(\mathcal{A}^{\mathcal{U}} \cup \mathcal{U})$$

- $\mathcal{A}^{\mathcal{U}}$ denotes restriction of $\mathcal{A}$ by $\mathcal{U}$ (will use frequently)
- The form of $\mathcal{A}^{\mathcal{U}}$ depends on DL used
- Updated ABoxes are Boolean (and in DNF)

# Keeping Updated ABoxes Smaller

# Huge Updated ABoxes

Naive implementation of the algorithms from KR-06 is unworkable:

- Updated ABoxes are highly redundant and HUGE

How can we get smaller updated ABoxes?

- We use equivalence-preserving transformations

# Logical Transformations for Smaller ABoxes

We introduce five transformations:

- CNF representation for updated ABoxes
- Exploit determinate updates
- Exploit the Unique Name Assumption
- Remove Subsuming Disjuncts
- Identify independent assertions

# Use CNF ABox Representation (Technique 1)

**First step:**
From $\mathcal{A}' = \bigwedge(\mathcal{A} \cup \mathcal{U}) \vee \bigwedge(\mathcal{A}^{\mathcal{U}} \cup \mathcal{U})$ to $\mathcal{A}' = \mathcal{U} \wedge (\bigwedge \mathcal{A} \vee \bigwedge \mathcal{A}^{\mathcal{U}})$

# Use CNF ABox Representation (Technique 1)

**First step:**
From $\mathcal{A}' = \bigwedge(\mathcal{A} \cup \mathcal{U}) \vee \bigwedge(\mathcal{A}^{\mathcal{U}} \cup \mathcal{U})$ to $\mathcal{A}' = \mathcal{U} \wedge (\bigwedge \mathcal{A} \vee \bigwedge \mathcal{A}^{\mathcal{U}})$

**Second step:**
Use $\bigwedge\{(\alpha^{\mathcal{U}} \vee \alpha) | \alpha \in \mathcal{A}\}$ instead of $\bigwedge \mathcal{A} \vee \bigwedge \mathcal{A}^{\mathcal{U}}$

Updated ABox is in CNF

# Exploit Determinate Updates (Technique 2)

- If assertion $\alpha$ entails update $\mathcal{U} = \{\delta\}$ then

$$(\alpha \vee \alpha^{\mathcal{U}}) \equiv \alpha$$

- If assertion $\alpha \models \neg\delta$ then

$$(\alpha \vee \alpha^{\mathcal{U}}) \equiv \alpha^{\mathcal{U}}$$

Can be detected only by reasoning steps (expensive)

# Exploit Unique Name Assumption (Technique 3)

Assume assertion $A(i)$, update $\mathcal{U} = \{\neg A(j)\}$:

Update to $A \sqcup \{j\}(i)$?

Only easy outside the scope of quantifiers

$\Rightarrow$ *Syntactic method vs. reasoning*

# Remove Subsuming Disjuncts (Technique 4)

Assume updated ABox $\mathcal{A} \vee \mathcal{B}$. If $\mathcal{A} \models \mathcal{B}$ then $(\mathcal{A} \vee \mathcal{B}) \equiv \mathcal{B}$.

*Identifying subsuming disjuncts requires reasoning*

Assume ABox $\mathcal{A}$, update $\mathcal{U} = \{\delta(\vec{t})\}$, where $\delta$ is unnegated. Then

- if $\delta$ occurs only positively in $\mathcal{A}$ then $\mathcal{A}^{\mathcal{U}} \models \mathcal{A}$; and
- if $\delta$ occurs only negatively in $\mathcal{A}$ then $\mathcal{A} \models \mathcal{A}^{\mathcal{U}}$

Symmetric condition for negative update

$\Rightarrow$ *Identify some subsuming disjuncts without reasoning*

# Detect Independent Assertions (Technique 5)

Assertion $\alpha \in \mathcal{A}$ is independent from update $\mathcal{U}$ iff

$$\mathcal{A} * \mathcal{U} \equiv \alpha \wedge [(\mathcal{A} \setminus \{\alpha\}) * \mathcal{U}],$$

where $\mathcal{A} * \mathcal{U}$ denotes updating $\mathcal{A}$ by $\mathcal{U}$

How to find out?

$\Rightarrow$ *Syntactic method vs. reasoning*

# Reasoning about Updated (Boolean) ABoxes

# Reasoning about Updated ABoxes

Updated ABoxes are Boolean (either CNF or DNF):

- *Boolean ABox reasoning not supported by DL reasoners*
- $\mathcal{ALCO}^{+}$ *and* $\mathcal{ALCO}^{@}$ *not supported by DL reasoners*

We present four reasoners for Boolean ABoxes

# Reasoning about Updated ABoxes

Updated ABoxes are Boolean (either CNF or DNF):

- *Boolean ABox reasoning not supported by DL reasoners*
- $\mathcal{ALCO}^{+}$ *and* $\mathcal{ALCO}^{@}$ *not supported by DL reasoners*

We present four reasoners for Boolean ABoxes

**Reasoning Tasks:** Logical Consequence, Query-Answering

# DL Reasoning for DNF ABoxes (Approach 1)

**For $\mathcal{ALCO}^{@}$:**

Simulate @ operator by "universal role" (linear)
Maybe compile to DNF (exponential)
$\mathcal{A} = \mathcal{A}_1 \vee \mathcal{A}_2 \vee \ldots \vee \mathcal{A}_n$ is consistent iff. some $\mathcal{A}_i$ is:

*DL reasoners can decide this for each $\mathcal{A}_i$*

# From Boolean to Non-Boolean ABoxes (Approach 2)

**For $\mathcal{ALCO}^{@}$:**

Linearly compile Boolean ABox to Non-Boolean @-ABox

$$C(a) \vee D(b) \longrightarrow (C \sqcup @_b D)(a)$$

Linearly compile @-ABox to “universal role” ABox

$$(C \sqcup @_b D)(a) \longrightarrow (C \sqcup \exists \mathrm{uR}.(D \sqcap \{b\}))(a)$$

Call DL reasoner on result (Reduction Approach)

# DPLL(T) on CNF ABoxes (Approach 3)

**For $\mathcal{ALCO}^{@}$:**

Simulate @ by universal role

DPLL(T): combine SAT-solver with theory solver

Pellet is DL theory solver:

- supports explanation of inconsistency
- thus can build backjump clauses

# Automated Theorem Proving (Approach 4)

**For $\mathcal{ALCO}^+$:**

- ATP systems support $\mathcal{ALCO}^+$ (smaller updated ABoxes)
- We use Otter: supports query-answering

# Not in the Paper (Approach 5, 6)

We have now also tried the following:

- Spartacus — decides hybrid logic, and thus Boolean $\mathcal{ALCO}^{@}$ ABoxes
- MetTeL — decides $\mathcal{ALBO}$, and thus Boolean $\mathcal{ALCO}^{+}$ ABoxes

# Lessons Learned

# ABox Representation

Should you update to CNF or DNF?

*Always update ABoxes to CNF*

# Smaller Updated ABoxes

How to keep ABoxes small at a low cost?

- Detect subsuming disjuncts (syntactically)
- Exploit UNA (syntactically)
- Identify independent assertions (syntactically)
- Detect many determinate updates by detecting subsuming disjuncts

Syntactic techniques are fast — Semantic techniques do not help much

# Reasoning

Which reasoning methods worked?

- DNF based reasoning doesn't work: costly conversion from CNF
- Otter can't keep up: costly conversion to full CNF
- Reduction fast on consistent ABoxes
- DPLL(T) fast on inconsistent ABoxes (but bad at query-answering)
- Spartacus as fast as Reduction and DPLL(T) (no query-answering)
- MetTeL decides $\mathcal{ALCO}^{+}$, but is slower than Otter

# The Big Picture

Dilemma:

- $\mathcal{ALCO}^{+}$: good representation for updated ABoxes
- $\mathcal{ALCO}^{@}$: reasoning works better

What performance do I get now if my ABox ...

- doesn't contain nested quantifiers? Nice.
- does contain nested quantifiers? Not so nice.

*Thanks for your attention! Questions?*

# Ramification Problem

No problem for acyclic TBox: Unfold

Otherwise (general TBox/no action preconditions) we get modified notion of ABox semantics:

- ABox $\mathcal{A}$ is consistent iff
  there's no sequence $\vec{u}$ of updates s.t. $\mathcal{A} \star \vec{u} \equiv \bot$
- $\alpha$ is a consequence of $\mathcal{A}$ iff $\mathcal{A} \cup \neg\alpha$ is inconsistent
- Check initial consistency? Generate plan space.