CEID

MSc on DATA DRIVEN COMPUTING AND DECISION MAKING (DDCDM)

Reasoning with Description Logics

I. HATZILYGEROUDIS, PROFESSOR EMERITUS

DL System Architecture

Knowledge Base

TBox

$Woman \equiv Person \cap Female$

$Man \equiv Person \cap \neg Female$

…

Abox

Man(BOB)

hasChild(BOB, MARY)

$\neg$Doctor(MARY)

…

Inference System

User Interface

Reasoning in DL

- Concept satisfiability

  Whether a concept C is satisfiable with respect to a TBox T (i.e. does not create a conflict)

- Concept subsumption

  Whether a concept C subsumes another concept D ($C \subseteq D$) with respect to a TBox T

- Concept equivalence

  Whether two concepts C and C are equivalent ($C \equiv D$) with respect to a TBox T

Reasoning in DL

- Concept inconsistency

  Whether a concept C is inconsistent (disjoint) with concept D with respect to TBox T
- Instance checking

  Whether an entity *a* is instance of concept C with respect to a TBox T and an ABox A

The following refer to DL $\mathcal{ALC}$

Satisfiability

- All previous types of reasoning are equivalent to some satisfiability checking:
  - Concept subsumption:

    $C \subseteq D$ iff $C \cap \neg D$ is non-satisfiable

    (C is subsumed by D or D subsumes C)
  - Concept equivalence:

    $C \equiv D$ iff $C \subseteq D$ and $D \subseteq C$, that is

    iff $(C \cap \neg D) \cup (\neg C \cap D)$ is non-satisfiable
  - Concept inconsistency:

    C and D are disjoint iff $C \cap D$ is non-satisfiable
  - Instance checking:

    a is an instance of C iff $\mathcal{A} \cup \{a: \neg C\}$ is non-satisfiable

    ($\mathcal{A}$ is an ABox)

Tableau Reasoning Method

- It is a concept satisfiability checking method
- Process
  1. Convert the concept to Negation Normal Form (NNF)
  2. Apply Completion Rules in arbitrary order until:
     - ✓ encounter a conflict case or
     - ✓ there is no other applicable rule
  3. The concept is satisfied iff a complete and clash-free tableau is produced, i.e. it does not contain $\bot$ nor any pair $\{\neg C, C\}$.

Negative Normal Form (NNF)

- All negations are moved to the level of concept names
- NNF transformation rules (from Zakharyaschev slides)

$$
\begin{aligned}
\neg\top &\equiv \bot \\
\neg\bot &\equiv \top \\
\neg\neg C &\equiv C \\
\neg(C \sqcap D) &\equiv \neg C \sqcup \neg D && \text{(De Morgan's law)} \\
\neg(C \sqcup D) &\equiv \neg C \sqcap \neg D && \text{(De Morgan's law)} \\
\neg\forall R.C &\equiv \exists R.\neg C \\
\neg\exists R.C &\equiv \forall R.\neg C
\end{aligned}
$$

Negative Normal Form (NNF)

Transform the following concept:

$$\neg\exists R.(A \sqcap \neg B) \sqcup \neg\forall R.(\neg A \sqcup \neg B)$$

in equivalent NNF

$\neg\exists R.(A \sqcap \neg B) \sqcup \neg\forall R.(\neg A \sqcup \neg B) \equiv$	(use $\neg\exists R.D \equiv \forall R.\neg D$)
$\forall R.\neg(A \sqcap \neg B) \sqcup \neg\forall R.(\neg A \sqcup \neg B) \equiv$	(use $\neg(A \sqcap D) \equiv \neg A \sqcup \neg D$)
$\forall R.(\neg A \sqcup \neg\neg B) \sqcup \neg\forall R.(\neg A \sqcup \neg B) \equiv$	(use $\neg\neg B \equiv B$)
$\forall R.(\neg A \sqcup B) \sqcup \neg\forall R.(\neg A \sqcup \neg B) \equiv$	(use $\neg\forall R.D \equiv \exists R.\neg D$)
$\forall R.(\neg A \sqcup B) \sqcup \exists R.\neg(\neg A \sqcup \neg B) \equiv$	(use $\neg(C \sqcup D) \equiv \neg C \sqcap \neg D$)
$\forall R.(\neg A \sqcup B) \sqcup \exists R.(\neg\neg A \sqcap \neg\neg B) \equiv$	(use $\neg\neg C \equiv C$)
$\forall R.(\neg A \sqcup B) \sqcup \exists R.(A \sqcap B)$	

(from Zakharyaschev's slides)

Definitions

- Constraint: Expression of the form «x: C» ή «(x, y): R»
- Constraint system: A non-empty finite set of constraints S
- Completion rules: A transformation S → S′, where S′ is a constraint system which includes S
- Complete system: S is complete if no completion rule can apply to S
- Clash: S includes a class if $\{x: A, x: \neg A\} \subseteq S$, where A is a concept name

Completion Rules

$$S \rightarrow_{\sqcap} S \cup \{ x: C, x: D \}$$

if (a) $x: C \sqcap D$ is in S

(b) $x: C$ and $x: D$ are not both in S

(intersection)

$$S \rightarrow_{\sqcup} S \cup \{ x: E \}$$

if (a) $x: C \sqcup D$ is in S

(b) neither $x: C$ nor $x: D$ is in S

(c) $E = C$ or $E = D$ (branching!)

(union)

(from Zakharyaschev's slides)

Completion Rules

$S \rightarrow_{\forall} S \cup \{ y\colon C \}$

if (a) $x\colon \forall R.C$ is in S
(b) $(x, y)\colon R$ is in S
(c) $y\colon C$ is not in S

(Universality)

$S \rightarrow_{\exists} S \cup \{ (x, y)\colon R,\ y\colon C \}$

if (a) $x\colon \exists R.C$ is in S
(b) there is no z such that both $(x, z)\colon R$ and $z\colon C$ are in S
(c) y is a fresh individual

(Existentiality)

(from Zakharyaschev's slides)

$S \rightarrow_{\sqcap} S \cup \{ x : C,\ x : D \}$
if (a) $x : C \sqcap D$ is in S
(b) $x : C$ and $x : D$ are not both in S

Examples

$S \rightarrow_{\sqcup} S \cup \{ x : E \}$
if (a) $x : C \sqcup D$ is in S
(b) neither $x : C$ nor $x : D$ is in S
(c) $E = C$ or $E = D$ (branching!)

(from Zakharyaschev's slides)

Tableau Algorithm: example

Let Woman $\equiv$ Person $\sqcap$ Female, Mother $\equiv$ Parent $\sqcap$ Female
and Parent $\equiv$ Person $\sqcap$ $\exists$hasChild.Person

Does the concept Woman subsume the concept Mother?
i.e., is the concept $\neg$Woman $\sqcap$ Mother satisfiable?

S_0	$= \{ x : (\neg\text{Person} \sqcup \neg\text{Female}) \sqcap ((\text{Person} \sqcap \exists\text{hasChild.Person}) \sqcap \text{Female}) \}$
$S_0 \rightarrow_{\sqcap} S_1$	$= S_0 \cup \{ x : \neg\text{Person} \sqcup \neg\text{Female},\ x : (\text{Person} \sqcap \exists\text{hasChild.Person}) \sqcap \text{Female} \}$
$S_1 \rightarrow_{\sqcap} S_2$	$= S_1 \cup \{ x : \text{Person} \sqcap \exists\text{hasChild.Person},\ x : \text{Female} \}$
$S_2 \rightarrow_{\sqcap} S_3$	$= S_2 \cup \{ x : \text{Person}, x : \exists\text{hasChild.Person} \}$
$S_3 \rightarrow_{\sqcup} S_{4.1}$	$= S_3 \cup \{ x : \neg\text{Person} \}$ **clash**
$S_3 \rightarrow_{\sqcup} S_{4.2}$	$= S_3 \cup \{ x : \neg\text{Female} \}$ **clash**

Thus the concept $\neg$Woman $\sqcap$ Mother is unsatisfiable,
and so Woman subsumes Mother

Examples

(the previous one as a tree)

{x:(¬Person∪ ¬Female)∩((Person∩∃hasChild.Person)∩Female)} S_0

↓ →∩ (κανόνας τομής)

{x:¬Person∪ ¬Female, x:(Person∩∃hasChild.Person)∩Female} S_1

↓ →∩

{x:¬Person∪ ¬Female, x:Person∩∃hasChild.Person, x:Female} S_2

↓ →∩

{x:¬Person∪ ¬Female, x:Person, x:∃hasChild.Person, x:Female} S_3

→∪ (κανόνας ένωσης)

{x:¬Person, x:Person, x:∃hasChild.Person, x:Female} $S_{4.1}$

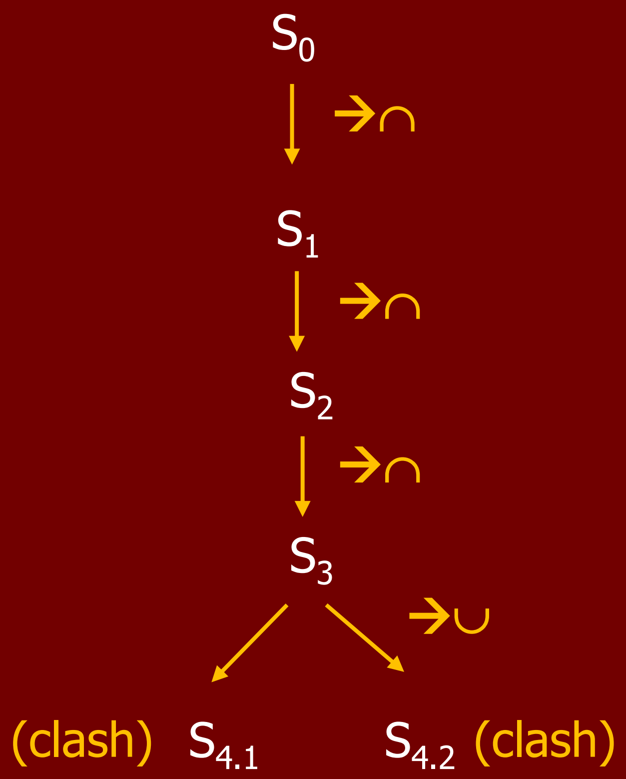
(clash)

{x: ¬Female, x:Person, x:∃hasChild.Person, x:Female} $S_{4.2}$

(clash)

Examples

(the previous one in a more abstract tree)



Examples

Reasoning with ABoxes: example
(from Zakharyaschev's slides)

Given: Sam is a person living in Germany. Sam drinks beer and Deuchars. A Bavarian is a person who lives in Germany, drinks beer and only beer.

Q: Is Sam a Bavarian?

ABox $\mathcal{A}$

sam : Person
sam : ∃livesIn.Germany
sam : ∃drinks.Beer
(sam, deuchars) : drinks

TBox $\mathcal{T}$

$$\text{Bavarian} \equiv \text{Person} \sqcap \exists\text{livesIn.Germany} \sqcap \exists\text{drinks.Beer} \sqcap \forall\text{drinks.Beer}$$

Is sam an instance of Bavarian ?

1. Reduction to ABox consistency:

Sam is an instance of Bavarian iff $\mathcal{A} \cup \{\, \text{sam} : \neg\text{Bavarian} \,\}$ is unsatisfiable

2. NNF of ¬Bavarian:

$$\neg\text{Person} \sqcup \forall\text{livesIn}.\neg\text{Germany} \sqcup \forall\text{drinks}.\neg\text{Beer} \sqcup \exists\text{drinks}.\neg\text{Beer}$$

$S \rightarrow_\forall S \cup \{ y\colon C \}$

if (a) $x\colon \forall R.C$ is in S
(b) $(x,y)\colon R$ is in S
(c) $y\colon C$ is not in S

Examples

$S \rightarrow_\exists S \cup \{ (x,y)\colon R,\ y\colon C \}$

if (a) $x\colon \exists R.C$ is in S
(b) there is no z such that both $(x,z)\colon R$ and $z\colon C$ are in S
(c) y is a fresh individual

Reasoning with ABoxes: example (cont.)

$S_0 = \{$ sam: Person, sam: ∃livesIn.Germany, sam: ∃drinks.Beer, (sam, deuchars): drinks, sam: ¬Person ⊔ ∀livesIn.¬Germany ⊔ ∀drinks.¬Beer ⊔ ∃drinks.¬Beer $\}$

$S_0 \rightarrow_\sqcup S_{1.1} = S_0 \cup \{$ sam: ¬Person $\}$ **clash**

$S_0 \rightarrow_\sqcup S_{1.2} = S_0 \cup \{$ sam: ∀livesIn.¬Germany $\}$

$S_{1.2} \rightarrow_\exists S_{2.2} = S_{1.2} \cup \{$ (sam, x): livesIn, x: Germany $\}$

$S_{2.2} \rightarrow_\forall S_{3.2} = S_{2.2} \cup \{$ x: ¬Germany $\}$ **clash**

$S_0 \rightarrow_\sqcup S_{1.3} = S_0 \cup \{$ sam: ∀drinks.¬Beer $\}$

$S_{1.3} \rightarrow_\exists S_{2.3} = S_{1.3} \cup \{$ (sam, x): drinks, x: Beer $\}$

$S_{2.3} \rightarrow_\forall S_{3.3} = S_{2.3} \cup \{$ x: ¬Beer $\}$ **clash**

$S_0 \rightarrow_\sqcup S_{1.4} = S_0 \cup \{$ sam: ∃drinks.¬Beer $\}$ (...see the next slide)

(from Zakharyaschev's slides)

Examples

Reasoning with ABoxes: example (cont.)

$S_0 = \{$ sam: Person, sam: ∃livesIn.Germany, sam: ∃drinks.Beer, (sam, deuchars): drinks, sam: ¬Person ⊔ ∀livesIn.¬Germany ⊔ ∀drinks.¬Beer ⊔ ∃drinks.¬Beer $\}$

$S_0 \rightarrow_{\sqcup} S_{1.4} = S_0 \cup \{$ sam: ∃drinks.¬Beer $\}$

$S_{1.4} \rightarrow_{\exists} S_{2.4} = S_{1.4} \cup \{$ (sam, x): drinks, x: ¬Beer $\}$

$S_{2.4} \rightarrow_{\exists} S_{3.4} = S_{2.4} \cup \{$ (sam, y): drinks, y: Beer $\}$

$S_{3.4} \rightarrow_{\exists} S_{4.4} = S_{3.4} \cup \{$ (sam, z): livesIn, z: Germany $\}$

$S_{4.4}$ is a complete clash-free constraint system. Therefore,

$\mathcal{A} \cup \{$sam: ¬Bavarian$\}$

is satisfiable and Sam is **not an instance** of Bavarian.

Indeed, the interpretation which is obtained on the fourth branch on the one hand is a model of $\mathcal{A}$; on the other hand it includes the pair of constraints (sam, x): drinks and x: ¬Beer, which contradicts the definition of a Bavarian ('drinks only beer').

Note that nothing would change if we added deuchars: Beer to the ABox.

(from Zakharyaschev's slides)