AE4M33RZN, Fuzzy description logic: fuzzyDL reasoner

Radomír Černoch

radomir.cernoch@fel.cvut.cz

18/11/2013

Faculty of Electrical Engineering, CTU in Prague

Plan of the lecture

Finite model property

Definition

A logic is said to have the finite model property if every satisfiable formula of the logic admits a finite model, i.e., a model with a finite domain. [Baader, 2003]

Finite model property

Definition

A logic is said to have the finite model property if every satisfiable formula of the logic admits a finite model, i.e., a model with a finite domain. [Baader, 2003]

- Why is FMP important?

Finite model property

Definition

A logic is said to have the finite model property if every satisfiable formula of the logic admits a finite model, i.e., a model with a finite domain. [Baader, 2003]

- Why is FMP important? Unless FMP holds, we need to be clever about our reasoning algorithms and avoid creating infinite models.

Finite model property

Definition

A logic is said to have the finite model property if every satisfiable formula of the logic admits a finite model, i.e., a model with a finite domain. [Baader, 2003]

- Why is FMP important? Unless FMP holds, we need to be clever about our reasoning algorithms and avoid creating infinite models.
- Does FMP hold in Fuzzy Description Logic?

Finite model property

Definition

A logic is said to have the finite model property if every satisfiable formula of the logic admits a finite model, i.e., a model with a finite domain. [Baader, 2003]

- Why is FMP important? Unless FMP holds, we need to be clever about our reasoning algorithms and avoid creating infinite models.
- Does FMP hold in Fuzzy Description Logic? Unfortunately no.

Witnessed model property

Definition

An interpretation $\mathcal{I}$ is $\circ$-**witnessed** if for all $x \in \Delta$, there is $y \in \Delta$ s.t.

$$(\exists R \cdot C)^{\mathcal{I}}(x) = R^{\mathcal{I}}(x, y) \underset{\circ}{\wedge} C^{\mathcal{I}}(y)$$

and similarly there is a $y \in \Delta$ s.t.

$$(C \sqsubseteq D)^{\mathcal{I}}(y) = C^{\mathcal{I}}(y) \overset{\circ}{\underset{\circ}{\Rightarrow}} D^{\mathcal{I}}(y) .$$

We say that the y is the "witness", because he is responsible for the particular membership degree of $\exists R \cdot C$ (or $C \sqsubseteq D$).

Relationship between FMP and WMP

- It is easy to see that every finite model is a witnessed model,

Relationship between FMP and WMP

- It is easy to see that every finite model is a witnessed model, because all $\sup()$ can be replaced by $\max()$ in the definition of $\exists$.

Relationship between FMP and WMP

- It is easy to see that every finite model is a witnessed model, because all $\sup()$ can be replaced by $\max()$ in the definition of $\exists$.
- **Example:** Assume $\underset{S}{\neg}$ and $\underset{S}{\wedge}$ logic and a concept

$$C = \neg \forall R \cdot A \sqcap \neg \exists R \cdot \neg A \, .$$

We will show that C can be satisfied to the degree 0.5 in an **infinite** model, but **no finite model** (and therefore no witnessed model) can satisfy C to 0.5.

Relationship between FMP and WMP

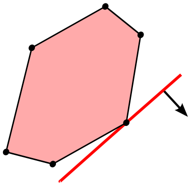
- It is easy to see that every finite model is a witnessed model, because all $\sup()$ can be replaced by $\max()$ in the definition of $\exists$.
- **Example:** Assume $\underset{S}{\neg}$ and $\underset{S}{\wedge}$ logic and a concept

$$C = \neg \forall R \,.\, A \sqcap \neg \exists R \,.\, \neg A \,.$$

  We will show that C can be satisfied to the degree 0.5 in an **infinite** model, but **no finite model** (and therefore no witnessed model) can satisfy C to 0.5.
- Are we hopeless? **No!** In Łukasiewicz logic ($\underset{S}{\neg}, \underset{L}{\wedge}, \overset{R}{\underset{L}{\Rightarrow}}$) we can restrict our reasoning to witnessed and finite models without loosing any information [Hájek, 2005].

Linear programming in a nutshell

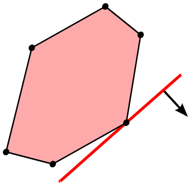
Imagine a 2D space with a convex polygon in the space (x, y). Given constraints $4x + y \geqslant 6, y \leqslant 8, \ldots$, minimize $x - 2y$.



Source: [Wikipedia, 2013]

Linear programming in a nutshell

Imagine a 2D space with a convex polygon in the space (x, y). Given constraints $4x + y \geqslant 6, y \leqslant 8, \ldots$, minimize $x - 2y$.



Source: [Wikipedia, 2013]

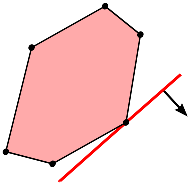
- Usually written in a matrix form

$$\text{maximize } \mathbf{c}^T \cdot x \quad (1)$$

$$\text{subject to } A\,\mathbf{x} \quad (2)$$

Linear programming in a nutshell

Imagine a 2D space with a convex polygon in the space (x, y). Given constraints $4x + y \geqslant 6, y \leqslant 8, \ldots$, minimize $x - 2y$.



Source: [Wikipedia, 2013]

- Usually written in a matrix form

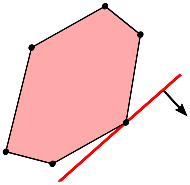
$$\text{maximize } \mathbf{c}^T \cdot x \quad (1)$$

$$\text{subject to } A\ \mathbf{x} \quad (2)$$

- (Mixed) Integer LP allows (some) variables to be **discrete**.

Linear programming in a nutshell

Imagine a 2D space with a convex polygon in the space (x, y). Given constraints $4x + y \geqslant 6, y \leqslant 8, \ldots$, minimize $x - 2y$.



Source: [Wikipedia, 2013]

- Usually written in a matrix form

$$\text{maximize } \mathbf{c}^T \cdot x \tag{1}$$

$$\text{subject to } A\,\mathbf{x} \tag{2}$$

- (Mixed) Integer LP allows (some) variables to be **discrete**.
- LP with real values is in P class, ILP is NP-complete.

Linear programming in a nutshell

Solution of a ((M)I)LP

- One solution (a point in the polytope).
- No solution (the polytope is empty).
- Multiple solutions with equal objective function value.

Linear programming in a nutshell

Solution of a ((M)I)LP

- One solution (a point in the polytope).
- No solution (the polytope is empty).
- Multiple solutions with equal objective function value.

Syntactical notes about fuzzyDL:

- $x \in \mathbb{R}$ will be **real numbers.**
- $y \in \mathbb{N}$ will be **integer numbers.**
- All values x, y will be bounded by $[0, 1]$.

FuzzyDL algorithm overview

- Transforms $\mathcal{K}$ to the **negated-normal-form.**[1]

[1] Makes sure that the negation ¬ appears only in front of concepts using:

$$\text{nnf}(\neg \forall R \cdot C) = \exists R \cdot \text{nnf}(\neg C) \text{ and } \text{nnf}(\neg \exists R \cdot C) = \forall R \cdot \text{nnf}(\neg C) .$$

FuzzyDL algorithm overview

- Transforms $\mathcal{K}$ to the **negated-normal-form.**[1]
- Creates an witnessed interpretation of $\mathcal{K}$.

[1] Makes sure that the negation $\neg$ appears only in front of concepts using:

$$\text{nnf}(\neg \forall R \cdot C) = \exists R \cdot \text{nnf}(\neg C) \text{ and } \text{nnf}(\neg \exists R \cdot C) = \forall R \cdot \text{nnf}(\neg C) .$$

FuzzyDL algorithm overview

- Transforms $\mathcal{K}$ to the **negated-normal-form.**[1]
- Creates an witnessed interpretation of $\mathcal{K}$.
- During its working it creates
  - a **completion forest** and

[1] Makes sure that the negation $\neg$ appears only in front of concepts using:

$$\mathrm{nnf}(\neg \forall R \cdot C) = \exists R \cdot \mathrm{nnf}(\neg C) \text{ and } \mathrm{nnf}(\neg \exists R \cdot C) = \forall R \cdot \mathrm{nnf}(\neg C)\,.$$

FuzzyDL algorithm overview

- Transforms $\mathcal{K}$ to the **negated-normal-form.**[1]
- Creates an witnessed interpretation of $\mathcal{K}$.
- During its working it creates
  - a **completion forest** and
  - a **list** of **linear** constraints $\mathscr{C}$.

[1] Makes sure that the negation $\neg$ appears only in front of concepts using:

$$\text{nnf}(\neg \forall R \cdot C) = \exists R \cdot \text{nnf}(\neg C) \text{ and } \text{nnf}(\neg \exists R \cdot C) = \forall R \cdot \text{nnf}(\neg C) .$$

FuzzyDL algorithm overview

- Transforms $\mathcal{K}$ to the **negated-normal-form.**[1]
- Creates an witnessed interpretation of $\mathcal{K}$.
- During its working it creates
  - a **completion forest** and
  - a **list** of **linear** constraints $\mathscr{C}$.
- Linear constraints $\mathscr{C}$ are solved using any **mixed-integer-linear-programming solver.**

[1] Makes sure that the negation $\neg$ appears only in front of concepts using:

$$\mathrm{nnf}(\neg \forall R \cdot C) = \exists R \cdot \mathrm{nnf}(\neg C) \text{ and } \mathrm{nnf}(\neg \exists R \cdot C) = \forall R \cdot \mathrm{nnf}(\neg C)\,.$$

FuzzyDL algorithm overview

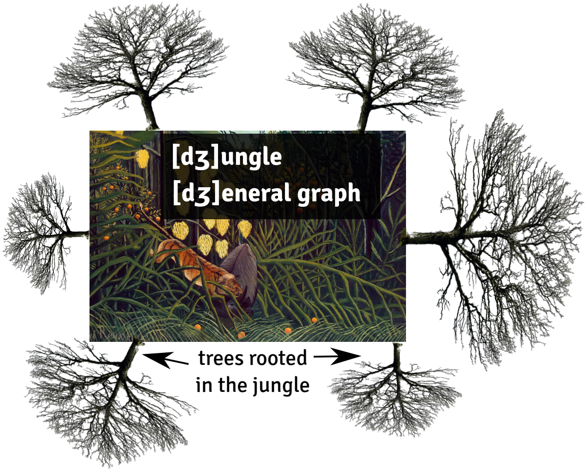
- Transforms $\mathcal{K}$ to the **negated-normal-form.**[1]
- Creates an witnessed interpretation of $\mathcal{K}$.
- During its working it creates
  - a **completion forest** and
  - a **list** of **linear** constraints $\mathscr{C}$.
- Linear constraints $\mathscr{C}$ are solved using any **mixed-integer-linear-programming solver.**

Disclaimer: Not going beyond Ł-logic, no concrete data types.

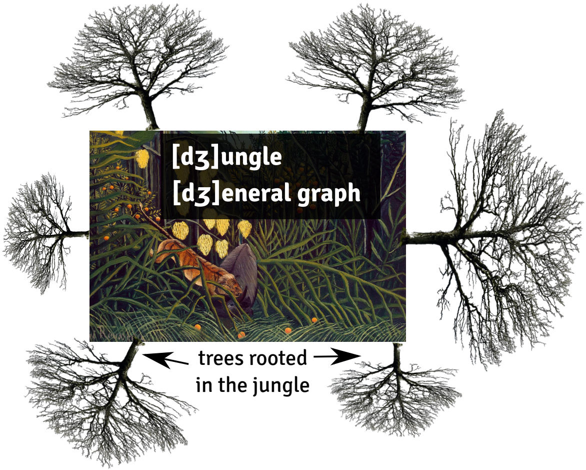
[1] Makes sure that the negation $\neg$ appears only in front of concepts using:

$$\mathrm{nnf}(\neg \forall R \cdot C) = \exists R \cdot \mathrm{nnf}(\neg C) \text{ and } \mathrm{nnf}(\neg \exists R \cdot C) = \forall R \cdot \mathrm{nnf}(\neg C) .$$

Completion-forest informally



Completion-forest informally



Completion forest is a **graph**, that has a general structure (jungle) “in the middle” and many trees, whose root nodes are nodes in the jungle.

Completion-forest formally

The fuzzyDL algorithm starts with creating the “jungle”. It contains all **individuals** (connected by an edge if they are linked by some relation).

Initialization

- Create a new vertex v_a for each **individual** a in the $\mathcal{K}$.

Completion-forest formally

The fuzzyDL algorithm starts with creating the “jungle”. It contains all **individuals** (connected by an edge if they are linked by some relation).

Initialization

- Create a new vertex v_a for each **individual** a in the $\mathcal{K}$.
- Create an edge (v_a, v_b) for each role assertion between a and b.

Completion-forest formally

The fuzzyDL algorithm starts with creating the “jungle”. It contains all **individuals** (connected by an edge if they are linked by some relation).

Initialization

- Create a new vertex v_a for each **individual** a in the $\mathcal{K}$.
- Create an edge (v_a, v_b) for each role assertion between a and b.
- Add a **label** $\langle \mathsf{C}, n \rangle$ to vertex a for each concept assertion $\langle a : \mathsf{C} \mid n \rangle$.

Completion-forest formally

The fuzzyDL algorithm starts with creating the “jungle”. It contains all **individuals** (connected by an edge if they are linked by some relation).

Initialization

- Create a new vertex v_a for each **individual** a in the $\mathcal{K}$.
- Create an edge (v_a, v_b) for each role assertion between a and b.
- Add a **label** $\langle \mathsf{C}, n \rangle$ to vertex a for each concept assertion $\langle a : \mathsf{C} \mid n \rangle$.
- Add a label $\langle \mathsf{R}, n \rangle$ to edge (a, b) for each role assertion $\langle (a, b) : \mathsf{R} \mid n \rangle$.

Forest completion (1)

The reasoner applies each of the following rules sequentially:

A If a vertex v is labeled $\langle C, l \rangle$, add $(x_{v:C} \geqslant l)$ into $\mathscr{C}$.

Forest completion (1)

The reasoner applies each of the following rules sequentially:

- A If a vertex v is labeled $\langle C, l \rangle$, add $(x_{v:C} \geqslant l)$ into $\mathscr{C}$.
- $\bar{A}$ If a vertex v is labeled $\langle \neg C, l \rangle$, add $(x_{v:C} \leqslant 1 - l)$ into $\mathscr{C}$.

Forest completion (1)

The reasoner applies each of the following rules sequentially:

- A If a vertex v is labeled $\langle \mathrm{C}, l \rangle$, add $(x_{v:\mathrm{C}} \geqslant l)$ into $\mathscr{C}$.
- $\bar{\mathrm{A}}$ If a vertex v is labeled $\langle \neg\, \mathrm{C}, l \rangle$, add $(x_{v:\mathrm{C}} \leqslant 1 - l)$ into $\mathscr{C}$.
- R If an edge (v, w) is labeled $\langle \mathrm{R}, l \rangle$, add $(x_{(v,w):\mathrm{R}} \geqslant l)$ into $\mathscr{C}$.

Forest completion (1)

The reasoner applies each of the following rules sequentially:

- A If a vertex v is labeled $\langle \mathsf{C}, l \rangle$, add $(x_{v:\mathsf{C}} \geqslant l)$ into $\mathscr{C}$.
- $\bar{\mathsf{A}}$ If a vertex v is labeled $\langle \neg\, \mathsf{C}, l \rangle$, add $(x_{v:\mathsf{C}} \leqslant 1 - l)$ into $\mathscr{C}$.
- R If an edge (v, w) is labeled $\langle \mathsf{R}, l \rangle$, add $(x_{(v,w):\mathsf{R}} \geqslant l)$ into $\mathscr{C}$.
- $\perp$ If a vertex v is labeled $\langle \perp, l \rangle$, add $(l = 0)$ into $\mathscr{C}$.

Forest completion (2)

- If a vertex v is labeled $\langle C \sqcap D, l\rangle$, append labels $\langle C, x_1\rangle$, $\langle D, x_2\rangle$ to v and add the following constraints into $\mathscr{C}$ (with fresh x_1, x_2, y):

$$
\begin{aligned}
y &\leqslant 1 - l \\
x_1 &\leqslant 1 - y \\
x_2 &\leqslant 1 - y \\
x_1 + x_2 &= l + 1 - y
\end{aligned}
$$

Forest completion (2)

- $\sqcap$ If a vertex v is labeled $\langle C \sqcap D, l\rangle$, append labels $\langle C, x_1\rangle$, $\langle D, x_2\rangle$ to v and add the following constraints into $\mathscr{C}$ (with fresh x_1, x_2, y):

$$\begin{aligned} y &\leqslant 1 - l \\ x_1 &\leqslant 1 - y \\ x_2 &\leqslant 1 - y \\ x_1 + x_2 &= l + 1 - y \end{aligned}$$

- $\sqcup$ If a vertex v is labeled $\langle C \sqcup D, l\rangle$, append labels $\langle C, x_1\rangle$, $\langle C, x_2\rangle$ to v and add $(x_1 + x_2 = l)$ into $\mathscr{C}$ (with fresh x_1, x_2, y).

Forest completion (3)

∀ If a vertex v is labeled $\langle \forall \mathsf{R} \cdot \mathsf{C}, l_1 \rangle$, an edge (v, w) is labeled $\langle R, l_2 \rangle$ and the rule has not been applied to this pair, then append the label $\langle \mathsf{C}, x \rangle$ to w and add the following constraints into $\mathscr{C}$ (with fresh x, y):

$$l_1 + l_2 - 1 \leqslant x \leqslant y \leqslant l_1 + l_2$$

Forest completion (3)

- $\forall$ If a vertex v is labeled $\langle \forall \mathsf{R} \cdot \mathsf{C}, l_1 \rangle$, an edge (v, w) is labeled $\langle R, l_2 \rangle$ and the rule has not been applied to this pair, then append the label $\langle \mathsf{C}, x \rangle$ to w and add the following constraints into $\mathscr{C}$ (with fresh x, y):

$$l_1 + l_2 - 1 \leqslant x \leqslant y \leqslant l_1 + l_2$$

- $\sqsubseteq$ If $\langle \mathsf{C} \sqsubseteq \mathsf{D} \mid n \rangle \in \mathcal{K}$, and the rule has not been applied to a node v, then append labels $\langle nnf(\neg \mathsf{C}), 1 - x_1 \rangle, \langle \mathsf{D}, x_2 \rangle$ to w and add $(x_1 < x_2 + 1 - n)$ to $\mathscr{C}$.

Forest completion: Example

Consider $\mathcal{K} = \{\langle \exists \mathsf{R} \cdot \mathsf{C} \sqsubseteq \mathsf{D} \,|\, 1 \rangle, \langle (a, b) : \mathsf{R} \,|\, 0.7 \rangle, \langle b : \mathsf{C} \,|\, 0.8 \rangle\}$.
Show that $bdb(\mathcal{K}, a : \mathsf{D}) = 0.5$.

Termination (1)

Unless the rules are applied repeatedly, the algorithm (as explained so far) terminates.

Termination (1)

Unless the rules are applied repeatedly, the algorithm (as explained so far) terminates.

For defining $\exists$ rule, new nodes are added, which needs to refine the terminating condition.

Termination (1)

Unless the rules are applied repeatedly, the algorithm (as explained so far) terminates.

For defining $\exists$ rule, new nodes are added, which needs to refine the terminating condition.

Equivalence of labels

Two lists of labels $[\langle C_1, l_1\rangle, \ldots, \langle C_n, l_n\rangle]$ and $[\langle C_1, l'_1\rangle, \ldots, \langle C_n, l'_n\rangle]$ are equivalent iff either

- l_i and l'_i are variables or
- l_i and l'_i are negated variables or
- l_i and l'_i are equal rationals.

Termination (2)

Directly blocked node

A node is directly blocked iff

- it is outside the “jungle” and
- none of its ancestors are blocked and
- **it has an ancestor with equivalent labels.**

Blocked node

A node is blocked iff either

- it is directly blocked or
- one of its predecessors is blocked.

Forest completion (4)

- $\exists$ If a vertex v is labeled $\langle \exists \mathrm{R} \cdot \mathrm{C}, l \rangle$ and **it is not blocked,** add a new vertex w and an edge (v, w), add labels $\langle C, x_2 \rangle$ to w, and $\langle R, x_1 \rangle$ to (v, w) and the following constraints into $\mathscr{C}$ (with fresh x_1, x_2 and y):

$$
\begin{aligned}
y &\leqslant 1 - l \\
x_1 &\leqslant 1 - y \\
x_2 &\leqslant 1 - y \\
x_1 + x_2 &= l + 1 - y
\end{aligned}
$$

FuzzyDL: Overview

- The instance of MILP is created using constraints $\mathscr{C}$.
- In order to solve $bdb(\mathcal{K}, \langle a : \mathrm{C} \rangle)$,

FuzzyDL: Overview

- The instance of MILP is created using constraints $\mathscr{C}$.
- In order to solve $bdb(\mathcal{K}, \langle a : \mathrm{C} \rangle)$, the objective function is set to minimize x in the MILP instance created for an augmented knowledge base $\mathcal{K} \cup \langle a : \neg \mathrm{C} \mid 1 - x \rangle$.

FuzzyDL: Overview

- The instance of MILP is created using constraints $\mathscr{C}$.
- In order to solve $bdb(\mathcal{K}, \langle a : \mathsf{C} \rangle)$, the objective function is set to minimize x in the MILP instance created for an augmented knowledge base $\mathcal{K} \cup \langle a : \neg\, \mathsf{C} \,|\, 1 - x \rangle$.
- Similarly for $bdb(\mathcal{K}, \langle a : C \sqsubseteq D \rangle)$ the augmented knowledge base is $\mathcal{K} \cup \langle a : \neg\, \mathsf{C} \,|\, 1 - x \rangle$.

FuzzyDL: Overview

- The instance of MILP is created using constraints $\mathscr{C}$.
- In order to solve $bdb(\mathcal{K}, \langle a : \mathsf{C} \rangle)$, the objective function is set to minimize x in the MILP instance created for an augmented knowledge base $\mathcal{K} \cup \langle a : \neg\, \mathsf{C} \,|\, 1 - x \rangle$.
- Similarly for $bdb(\mathcal{K}, \langle a : C \sqsubseteq D \rangle)$ the augmented knowledge base is $\mathcal{K} \cup \langle a : \neg\, \mathsf{C} \,|\, 1 - x \rangle$.
- $\mathcal{K}$ is inconsistent iff the MILP instance has no solution.
- Hence the $bdb(\cdot, \cdot)$ is found if MILP instance has a solution.

FuzzyDL: Conclusion

- FuzzyDL is a tableau algorithm with exactly 1 branch.
  The $\sqcup$ does not cause branching.

FuzzyDL: Conclusion

- FuzzyDL is a tableau algorithm with exactly 1 branch.
  The $\sqcup$ does not cause branching.
- Rules are applied deterministically (to ensure termination).

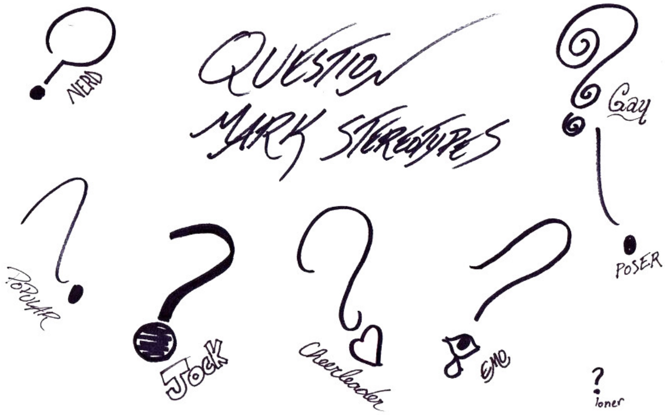
FuzzyDL: Conclusion

- FuzzyDL is a tableau algorithm with exactly 1 branch.
  The $\sqcup$ does not cause branching.
- Rules are applied deterministically (to ensure termination).
- $\mathcal{K}$ is inconsistent if and only if the MILP has no solution.

FuzzyDL: Conclusion

- FuzzyDL is a tableau algorithm with exactly 1 branch.
  The $\sqcup$ does not cause branching.
- Rules are applied deterministically (to ensure termination).
- $\mathcal{K}$ is inconsistent if and only if the MILP has no solution.
- The complexity of reasoning is caused by the integer (y) variables.

Questions?! Ask, please.



Source: `ragtagdoodles.deviantart.com`

Ex: Jim revisited

We will use the Łukasiewicz logic in the following examples ($\sqcap = \sqcap_{Ł}$, ...).

$$\langle jim : \text{Male} \mid 0.9 \rangle \tag{3}$$

$$\langle jim : \text{Female} \mid 0.2 \rangle \tag{4}$$

$$\langle \text{Male} \sqcap \text{Female} \sqsubseteq \bot \mid 1 \rangle \tag{5}$$

Ex: Jim revisited

We will use the Łukasiewicz logic in the following examples ($\sqcap = \sqcap_{Ł}$, ...).

$$\langle jim : \text{Male} \,|\, 0.9 \rangle \quad (3)$$

$$\langle jim : \text{Female} \,|\, 0.2 \rangle \quad (4)$$

$$\langle \text{Male} \sqcap \text{Female} \sqsubseteq \bot \,|\, 1 \rangle \quad (5)$$

The interpretation domain is $\Delta^{\mathcal{I}_1} = \Delta^{\mathcal{I}_2} = \{j\}, jim^{\mathcal{I}_1} = jim^{\mathcal{I}_2} = j$.

$$\text{Male}^{\mathcal{I}_1} = \{(j; 0.9)\} \qquad \text{Male}^{\mathcal{I}_2} = \{(j; 0.9)\}$$

$$\text{Female}^{\mathcal{I}_1} = \{(j; 0)\} \qquad \text{Female}^{\mathcal{I}_2} = \{(j; 0.2)\}$$

Ex: Jim revisited (check your knowledge)

Let's check the interpretation against the definitions...

$\mathcal{I} \models \tau$	$\tau_{(1)}$	$\tau_{(2)}$	$\tau_{(3)}$
$\mathcal{I}_1$	?	?	?
$\mathcal{I}_2$	?	?	?

Ex: Jim revisited (check your knowledge)

Let's check the interpretation against the definitions...

$\mathcal{I} \models \tau$	$\tau_{(1)}$	$\tau_{(2)}$	$\tau_{(3)}$
$\mathcal{I}_1$	yes	no	yes
$\mathcal{I}_2$	yse	yes	no

Ex: Jim revisited (in fuzzyDL)

Let's change the weights and encode the example in fuzzyDL:

```
(instance jim Male 0.4)
(instance jim Female 0.2)

(l-implies (and Male Female) *bottom* 0.9)

(min-instance? jim Male)
(max-instance? jim Male)
(min-instance? jim Female)
(max-instance? jim Female)
```

Let $\langle jim : \text{Male} \,|\, \alpha \rangle$ and $\langle jim : \text{Female} \,|\, \beta \rangle$, what are the bounds on α and β? fuzzyDL shows that $0.4 \leqslant \alpha \leqslant 0.9$ and $0.2 \leqslant \beta \leqslant 0.7$. Why?

Ex: Smokers

Recall the motivational example from the first lecture:

$$\langle \text{symmetric}(\mathsf{friend}) \rangle \tag{6}$$

$$\langle (anna, bill) : \mathsf{friend} \,|\, 1 \rangle \tag{7}$$

$$\langle (bill, cloe) : \mathsf{friend} \,|\, 1 \rangle \tag{8}$$

$$\langle (cloe, dirk) : \mathsf{friend} \,|\, 1 \rangle \tag{9}$$

$$\langle anna : \mathsf{Smoker} \,|\, 1 \rangle \tag{10}$$

$$\langle \exists \mathsf{friend} \cdot \mathsf{Smoker} \sqsubseteq \mathsf{Smoker} \,|\, 0.7 \rangle \tag{11}$$

What are the bounds on $\langle i : \mathsf{Smoker} \rangle$ for $i \in \{anna, bill, cloe, dirk\}$?

Ex: Smokers

What changes if we add

$$\langle dirk : \neg \text{Smoker} \mid 0.7 \rangle \tag{12}$$

(13)

What are the bounds on $\langle i : \neg \text{Smoker} \rangle$ for $i \in \{anna, bill, cloe, dirk\}$?

Ex: Smokers (in fuzzyDL)

```
(implies (some friendOf Smoker) Smoker 0.7)

(symmetric friendOf)
(related anna bill friendOf)
(related bill cloe friendOf)
(related cloe dirk friendOf)

(instance anna Smoker)
(instance dirk (not Smoker) 0.7)

(min-instance? anna Smoker)
(min-instance? bill Smoker)
(min-instance? cloe Smoker)
(min-instance? dirk Smoker)

(max-instance? anna Smoker)
(max-instance? bill Smoker)
(max-instance? cloe Smoker)
(max-instance? dirk Smoker)
```

Concrete data types

The domain $\Delta^{\mathcal{I}}$ is an unordered set. This is good for modelling cathegorical data: e.g. colors, people, ...

General idea: Extended interpretation

But we also need to include real numbers $\mathbb{R}$. The *fuzzy description logic with concrete datatypes* $\mathcal{SHIF}(\mathcal{D})$ uses "abstract objects" and "concrete objects":

$$\Delta^{\mathcal{I}} = \Delta_a^{\mathcal{I}} \cup \mathbb{R}$$

Concrete data types

- *Concrete individuals*, are interpreted as objects from $\mathbb{R}$.

Concrete data types

- *Concrete individuals*, are interpreted as objects from $\mathbb{R}$.
- *Concrete concepts*, are interpreted as subsets from $\mathbb{R}$.

Concrete data types

- *Concrete individuals*, are interpreted as objects from $\mathbb{R}$.
- *Concrete concepts*, are interpreted as subsets from $\mathbb{R}$.
- *Concrete roles*, are interpreted as subsets from $(\Delta_a^{\mathcal{I}} \times \mathbb{R})$.

Concrete data types

- *Concrete individuals*, are interpreted as objects from $\mathbb{R}$.
- *Concrete concepts*, are interpreted as subsets from $\mathbb{R}$.
- *Concrete roles*, are interpreted as subsets from $(\Delta_a^{\mathcal{I}} \times \mathbb{R})$.

All non-concrete notions are called *abstract*.

Concrete data types: New concepts

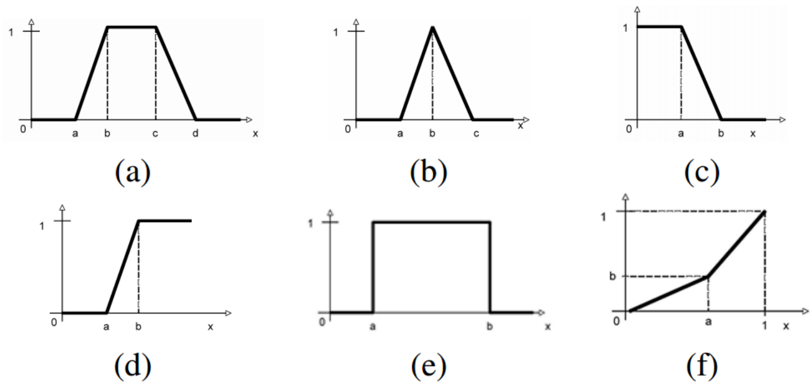


Fig. 1. (a) Trapezoidal function; (b) Triangular function; (c) L-function; (d) R-function; (e) Crisp interval; (f) Linear function.

Ex: Age of parents

```
(related adam bob parent) (related adam eve parent)

(define-fuzzy-concept around23 triangular(0,100, 18,23,26))
(define-fuzzy-concept moreTh17 right-shoulder(0,100, 13,21))
(instance bob (some age around23) 0.9)
(instance eve (some age moreTh17))

(define-fuzzy-concept young left-shoulder(0,100, 17,25))
(define-concept YoungPerson (some age young))

(min-instance? eve YoungPerson) (max-instance? eve YoungPerson)
(min-instance? bob YoungPerson) (max-instance? bob YoungPerson)
(min-instance? adam (all  parent YoungPerson))
(max-instance? adam (all  parent YoungPerson))
(min-instance? adam (some parent YoungPerson))
(max-instance? adam (some parent YoungPerson))
```

Ex: Age of parents

1. What are the bounds on α from $\langle \mathit{eve} : \text{YoungPerson} \,|\, \alpha \rangle$?

Ex: Age of parents

1. What are the bounds on α from $\langle \textit{eve} : \text{YoungPerson} \mid \alpha \rangle$?

   Start by drawing the concept around23, then construct an interpretation. How much freedom do you have when constructing the interpretation?

Ex: Age of parents

1. What are the bounds on α from $\langle \mathit{eve} : \text{YoungPerson} \,|\, \alpha \rangle$?

   Start by drawing the concept around23, then construct an interpretation. How much freedom do you have when constructing the interpretation?

2. Let fuzzyDL reasoner give you both bounds on $\langle i : \text{YoungPerson} \,|\, \beta_i \rangle$ for $i \in \{\mathit{eve}, \mathit{bob}\}$.

   How do you infer the bounds on $\langle \mathit{adam} : \text{YoungPerson} \,|\, \gamma \rangle$?

Ex: Car dealing

1. The buyer wants a **passenger** that costs **less than €26000**.
2. If there is an **alarm system** in the car, **then** he is satisfied with paying no more than **€22300**, but he can go up to **€22750** with a lesser degree of satisfaction.
3. The **driver insurance, air conditioning** and the **black color** are important factors.
4. Preferably the price is no more than **€22000**, but he can go to **€24000** to a lesser degree of satisfaction.

Ex: Car dealing

1. The seller wants to sell no less than **€22000**.
2. Preferably the buyer buys the **insurance plus** package.
3. If the **color is black**, then it is highly possible the car has an **air-conditioning**.

This can be formalized in fuzzy description logic.

Ex: Car dealing

1. The seller wants to sell no less than **€22000**.
2. Preferably the buyer buys the **insurance plus** package.
3. If the **color is black**, then it is highly possible the car has an **air-conditioning**.

This can be formalized in fuzzy description logic.
We have the background knowledge:

$$\langle \mathsf{Sedan} \sqsubseteq \mathsf{PassengerCar} \mid 1 \rangle$$
$$\langle \mathsf{InsurancePlus} = \mathsf{DriverInsurance} \sqcap \mathsf{TheftInsurance} \mid 1 \rangle$$

Ex: Car dealing

The buyer's preferences:

1. $B = \mathsf{PassengerCar} \sqcap \exists\, \mathsf{price} \cdot \leqslant 26000$

Ex: Car dealing

The buyer's preferences:

1. $B = \text{PassengerCar} \sqcap \exists \text{price} \cdot \leqslant 26000$
2. $B_1 = \text{AlarmSystem} \mapsto \exists \text{price} \cdot l.sh.(22300, 22750)$

Ex: Car dealing

The buyer's preferences:

1. $B = \text{PassengerCar} \sqcap \exists \text{price} \cdot \leqslant 26000$
2. $B_1 = \text{AlarmSystem} \mapsto \exists \text{price} \cdot l.sh.(22300, 22750)$
3. $B_2 = \text{DriverInsurance},$

Ex: Car dealing

The buyer's preferences:

1. $B = \text{PassengerCar} \sqcap \exists \text{price} \cdot \leqslant 26000$
2. $B_1 = \text{AlarmSystem} \mapsto \exists \text{price} \cdot l.sh.(22300, 22750)$
3. $B_2 = \text{DriverInsurance}, B_3 = \text{AirCondition},$

Ex: Car dealing

The buyer's preferences:

1. $B = \text{PassengerCar} \sqcap \exists \text{price} \cdot \leqslant 26000$
2. $B_1 = \text{AlarmSystem} \mapsto \exists \text{price} \cdot l.sh.(22300, 22750)$
3. $B_2 = \text{DriverInsurance}, B_3 = \text{AirCondition}, B_4 = \exists \text{color} \cdot \text{Black}$
4. $B_5 = \exists \text{price} \cdot l.sh.(22000, 24000)$

The buyer's preferences:

1. $S = \text{PassengerCar} \sqcap \exists \text{price} \cdot \geqslant 22000$

Ex: Car dealing

The buyer's preferences:

1. $B = \text{PassengerCar} \sqcap \exists \text{price} \cdot \leqslant 26000$
2. $B_1 = \text{AlarmSystem} \mapsto \exists \text{price} \cdot l.sh.(22300, 22750)$
3. $B_2 = \text{DriverInsurance}, B_3 = \text{AirCondition}, B_4 = \exists \text{color} \cdot \text{Black}$
4. $B_5 = \exists \text{price} \cdot l.sh.(22000, 24000)$

The buyer's preferences:

1. $S = \text{PassengerCar} \sqcap \exists \text{price} \cdot \geqslant 22000$
2. $S_1 = \text{InsurancePlus}$

Ex: Car dealing

The buyer's preferences:

1. $B = \text{PassengerCar} \sqcap \exists\, \text{price} \cdot \leqslant 26000$
2. $B_1 = \text{AlarmSystem} \mapsto \exists\, \text{price} \cdot l.sh.(22300, 22750)$
3. $B_2 = \text{DriverInsurance}, B_3 = \text{AirCondition}, B_4 = \exists\, \text{color} \cdot \text{Black}$
4. $B_5 = \exists\, \text{price} \cdot l.sh.(22000, 24000)$

The buyer's preferences:

1. $S = \text{PassengerCar} \sqcap \exists\, \text{price} \cdot \geqslant 22000$
2. $S_1 = \text{InsurancePlus}$
3. $S_2 = (0.5\ (\exists\, \text{color} \cdot \text{Black}) \mapsto \text{AirCondition})$

Ex: Car dealing

We know that S and B are hard constraints and $B_{1..5}$ and $S_{1..2}$ are soft preferences. All the concepts can be “summed up”:

Ex: Car dealing

We know that S and B are hard constraints and $B_{1..5}$ and $S_{1..2}$ are soft preferences. All the concepts can be "summed up":

$$\text{Buy} = B \sqcap (0.1B_1 + 0.2B_2 + 0.1B_3 + 0.4B_4 + 0.2B_5)$$

and

$$\text{Sell} = S \sqcap (0.6S_1 + 0.4S_2)$$

Ex: Car dealing

We know that S and B are hard constraints and $B_{1..5}$ and $S_{1..2}$ are soft preferences. All the concepts can be "summed up":

$$\text{Buy} = B \sqcap (0.1B_1 + 0.2B_2 + 0.1B_3 + 0.4B_4 + 0.2B_5)$$

and

$$\text{Sell} = S \sqcap (0.6S_1 + 0.4S_2)$$

A good choice of $\sqcap$ can make B a hard constraint.

Ex: Car dealing

Optimal match

$$bsb(\mathcal{K}, \text{Buy} \sqcap \text{Sell})$$

Finds the optimal match between a seller and a buyer. (Finds an ideal, imaginary car that maximizes satisfaction of both parties.)

Particular car

$$bdb(\mathcal{K}, \langle auditt : \text{Buy} \sqcap \text{Sell} \rangle)$$

Finds the degree of satisfaction for a particuklar car *audiTT*.

Where to find more examples?

- **Simple examples** are bundled with fuzzyDL installation (`/opt/fuzzydl/` on the `heartofgold` server).
- **Advanced examples** can be found on the fuzzyDL web site: `http://gaia.isti.cnr.it/~straccia/software/fuzzyDL/fuzzyDL.html`

Bibliography

Baader, F. (2003).
The Description Logic Handbook: Theory, Implementation, and Applications.
Cambridge University Press.

Hájek, P. (2005).
Making fuzzy description logic more general.
Fuzzy Sets and Systems, 154(1):1--15.

Wikipedia (2013).
Plagiarism --- Wikipedia, the free encyclopedia.
[Online; accessed 17-November-2013].