Inference v deskripčních logikách

Petr Křemen

FEL ČVUT

laboratory Gerstner

Co nás čeká

1. Inference v deskripčních logikách

2. Inferenční algoritmy
   - Tablový algoritmus pro $\mathcal{ALC}$

Inference v deskripčních logikách

Inferenční problémy pro TBOX

Představili jsme si syntax a sémantiku jazyka $\mathcal{ALC}$. Nyní se podívejme na automatické odvozování. Máme ontologii $\mathcal{K} = (\mathcal{T}, \mathcal{A})$. Pro TBOX $\mathcal{T}$ a koncepty C a D nás zajímá, zda

(ne)splnitelnost koncept C je *nesplnitelný*, tedy $\mathcal{T} \models C \sqsubseteq \bot$?

Inferenční problémy pro TBOX

Představili jsme si syntax a sémantiku jazyka $\mathcal{ALC}$. Nyní se podívejme na automatické odvozování. Máme ontologii $\mathcal{K} = (\mathcal{T}, \mathcal{A})$. Pro TBOX $\mathcal{T}$ a koncepty C a D nás zajímá, zda

(ne)splnitelnost koncept C je *nesplnitelný*, tedy $\mathcal{T} \models C \sqsubseteq \bot$?

subsumpce koncept C *je obecnější* než D, tedy $\mathcal{T} \models D \sqsubseteq C$?

Inferenční problémy pro TBOX

Představili jsme si syntax a sémantiku jazyka $\mathcal{ALC}$. Nyní se podívejme na automatické odvozování. Máme ontologii $\mathcal{K} = (\mathcal{T}, \mathcal{A})$. Pro TBOX $\mathcal{T}$ a koncepty C a D nás zajímá, zda

(ne)splnitelnost koncept C je *nesplnitelný*, tedy $\mathcal{T} \models C \sqsubseteq \bot$?

subsumpce koncept C *je obecnější* než D, tedy $\mathcal{T} \models D \sqsubseteq C$?

ekvivalence dva koncepty C a D jsou *ekvivalentní*, tedy $\mathcal{T} \models C \equiv D$?

Inferenční problémy pro TBOX

Představili jsme si syntax a sémantiku jazyka $\mathcal{ALC}$. Nyní se podívejme na automatické odvozování. Máme ontologii $\mathcal{K} = (\mathcal{T}, \mathcal{A})$. Pro TBOX $\mathcal{T}$ a koncepty C a D nás zajímá, zda

(ne)splnitelnost koncept C je *nesplnitelný*, tedy $\mathcal{T} \models C \sqsubseteq \bot$?

subsumpce koncept C *je obecnější* než D, tedy $\mathcal{T} \models D \sqsubseteq C$?

ekvivalence dva koncepty C a D jsou *ekvivalentní*, tedy $\mathcal{T} \models C \equiv D$?

disjunktnost dva koncepty C a D jsou *disjunktní*, tedy $\mathcal{T} \models C \sqcap D \sqsubseteq \bot$?

Inferenční problémy pro TBOX

Představili jsme si syntax a sémantiku jazyka $\mathcal{ALC}$. Nyní se podívejme na automatické odvozování. Máme ontologii $\mathcal{K} = (\mathcal{T}, \mathcal{A})$. Pro TBOX $\mathcal{T}$ a koncepty C a D nás zajímá, zda

(ne)splnitelnost koncept C je *nesplnitelný*, tedy $\mathcal{T} \models C \sqsubseteq \bot$?

subsumpce koncept C *je obecnější* než D, tedy $\mathcal{T} \models D \sqsubseteq C$?

ekvivalence dva koncepty C a D jsou *ekvivalentní*, tedy $\mathcal{T} \models C \equiv D$?

disjunktnost dva koncepty C a D jsou *disjunktní*, tedy $\mathcal{T} \models C \sqcap D \sqsubseteq \bot$?

Všechny tyto úlohy lze redukovat na kontrolu splnitelnosti jednoho konceptu.

Redukce na nesplnitelnost – příklad

Příklad

Tyto redukce jsou jednoduché, ukažme si, jakým způsobem převést např. subsumpci na nesplnitelnost:

$$
\begin{aligned}
(\mathcal{T} \models C \sqsubseteq D) & && && \text{iff} \\
(\forall \mathcal{I})(\mathcal{I} \models \mathcal{T} & \quad \text{implikuje} && \mathcal{I} \models C \sqsubseteq D) && \text{iff} \\
(\forall \mathcal{I})(\mathcal{I} \models \mathcal{T} & \quad \text{implikuje} && C^{\mathcal{I}} \subseteq D^{\mathcal{I}}) && \text{iff} \\
(\forall \mathcal{I})(\mathcal{I} \models \mathcal{T} & \quad \text{implikuje} && C^{\mathcal{I}} \cap (\Delta^{\mathcal{I}} \setminus D^{\mathcal{I}}) \subseteq \emptyset && \text{iff} \\
(\forall \mathcal{I})(\mathcal{I} \models \mathcal{T} & \quad \text{implikuje} && \mathcal{I} \models C \sqcap \neg D \sqsubseteq \bot && \text{iff} \\
(\mathcal{T} \models C \sqcap \neg D \sqsubseteq \bot) & && &&
\end{aligned}
$$

Redukce ostatních inferenčních úloh na nesplnitelnost je analogická – vyzkoušejte si.

Inferenční problémy pro ABOX

... pro ABOX $\mathcal{A}$, axiom α, koncept C, roli R a individuály a,a_0 nás zajímá

Inferenční problémy pro ABOX

... pro ABOX $\mathcal{A}$, axiom α, koncept C, roli R a individuály a,a_0 nás zajímá

consistency checking zda je ABOX $\mathcal{A}$ konzistentní vzhledem k $\mathcal{T}$, tedy pokud $\mathcal{K}$ je konzistentní.

Inferenční problémy pro ABOX

... pro ABOX $\mathcal{A}$, axiom α, koncept C, roli R a individuály a,a_0 nás zajímá

consistency checking zda je ABOX $\mathcal{A}$ konzistentní vzhledem k $\mathcal{T}$, tedy pokud $\mathcal{K}$ je konzistentní.

instance checking zda platí $\mathcal{T} \cup \mathcal{A} \models C(a)$?

Inferenční problémy pro ABOX

... pro ABOX $\mathcal{A}$, axiom α, koncept C, roli R a individuály a,a_0 nás zajímá

consistency checking zda je ABOX $\mathcal{A}$ konzistentní vzhledem k $\mathcal{T}$, tedy pokud $\mathcal{K}$ je konzistentní.

instance checking zda platí $\mathcal{T} \cup \mathcal{A} \models C(a)$?

role checking zda platí $\mathcal{T} \cup \mathcal{A} \models R(a, a_0)$?

Inferenční problémy pro ABOX

... pro ABOX $\mathcal{A}$, axiom α, koncept C, roli R a individuály a,a_0 nás zajímá

consistency checking zda je ABOX $\mathcal{A}$ konzistentní vzhledem k $\mathcal{T}$, tedy pokud $\mathcal{K}$ je konzistentní.

instance checking zda platí $\mathcal{T} \cup \mathcal{A} \models C(a)$?

role checking zda platí $\mathcal{T} \cup \mathcal{A} \models R(a, a_0)$?

instance retrieval nalezení množiny všech individuálů a_1, pro které $\mathcal{T} \cup \mathcal{A} \models C(a_1)$.

Inferenční problémy pro ABOX

... pro ABOX $\mathcal{A}$, axiom α, koncept C, roli R a individuály a,a_0 nás zajímá

consistency checking zda je ABOX $\mathcal{A}$ konzistentní vzhledem k $\mathcal{T}$, tedy pokud $\mathcal{K}$ je konzistentní.

instance checking zda platí $\mathcal{T} \cup \mathcal{A} \models C(a)$?

role checking zda platí $\mathcal{T} \cup \mathcal{A} \models R(a, a_0)$?

instance retrieval nalezení množiny všech individuálů a_1, pro které $\mathcal{T} \cup \mathcal{A} \models C(a_1)$.

realization nalezení nejspecifičtějšího konceptu C z předem dané množiny konceptů takového, že $\mathcal{T} \cup \mathcal{A} \models C(a)$.

Inferenční problémy pro ABOX

... pro ABOX $\mathcal{A}$, axiom α, koncept C, roli R a individuály a,a_0 nás zajímá

consistency checking zda je ABOX $\mathcal{A}$ konzistentní vzhledem k $\mathcal{T}$, tedy pokud $\mathcal{K}$ je konzistentní.

instance checking zda platí $\mathcal{T} \cup \mathcal{A} \models C(a)$?

role checking zda platí $\mathcal{T} \cup \mathcal{A} \models R(a, a_0)$?

instance retrieval nalezení množiny všech individuálů a_1, pro které $\mathcal{T} \cup \mathcal{A} \models C(a_1)$.

realization nalezení nejspecifičtějšího konceptu C z předem dané množiny konceptů takového, že $\mathcal{T} \cup \mathcal{A} \models C(a)$.

Všechny tyto inferenční problémy, stejně tak jako ověření nesplnitelnosti konceptu lze redukovat na ověření konzistence $\mathcal{A}$ vzhledem k $\mathcal{T}$. Za jakých podmínek a jak ?

Inferenční algoritmy

Inferenční algoritmy

Algoritmy strukturálního porovnávání jsou polynomiální, ale úplné pouze pro některé jednoduché DL bez úplné negace, např. $\mathcal{ALN}$, viz.[BCM$^+$03].

Inferenční algoritmy

Algoritmy strukturálního porovnávání jsou polynomiální, ale úplné pouze pro některé jednoduché DL bez úplné negace, např. $\mathcal{ALN}$, viz.[BCM+03].

Tablové algoritmy jsou SoA algoritmy pro komplexní DL – korektní, úplné, pracující v konečném čase, viz. [HS03], [HS01], [BCM+03].

Inferenční algoritmy

Algoritmy strukturálního porovnávání jsou polynomiální, ale úplné pouze pro některé jednoduché DL bez úplné negace, např. $\mathcal{ALN}$, viz.[BCM+03].

Tablové algoritmy jsou SoA algoritmy pro komplexní DL – korektní, úplné, pracující v konečném čase, viz. [HS03], [HS01], [BCM+03].

jiné ... – např. algoritmy založené na rezoluci [Hab06], převodu na konečné automaty [BCM+03], apod.

Inferenční algoritmy

Algoritmy strukturálního porovnávání jsou polynomiální, ale úplné pouze pro některé jednoduché DL bez úplné negace, např. $\mathcal{ALN}$, viz.[BCM+03].

Tablové algoritmy jsou SoA algoritmy pro komplexní DL – korektní, úplné, pracující v konečném čase, viz. [HS03], [HS01], [BCM+03].

jiné ... – např. algoritmy založené na rezoluci [Hab06], převodu na konečné automaty [BCM+03], apod.

My si dále podrobněji představíme právě tablové algoritmy.

Tablové algoritmy

- Tablové algoritmy (TA) slouží k ověřování konzistence ABOXu vzhledem k TBOXu. Nejedná se o žádnou novinku v deskripčních logikách – TA byly známy již pro FOL.

Tablové algoritmy

- Tablové algoritmy (TA) slouží k ověřování konzistence ABOXu vzhledem k TBOXu. Nejedná se o žádnou novinku v deskripčních logikách – TA byly známy již pro FOL.
- Hlavní myšlenka je jednoduchá: **"Máme-li ověřit konzistenci daného ABOXu $\mathcal{A}$ vzhledem k TBOXu $\mathcal{T}$, pokusme se sestrojit model $\mathcal{T} \cup \mathcal{A}$. Pokud se nám to podaří, víme, že $\mathcal{A}$ je konzistentní vzhledem k $\mathcal{T}$ "**

Tablové algoritmy

- Tablové algoritmy (TA) slouží k ověřování konzistence ABOXu vzhledem k TBOXu. Nejedná se o žádnou novinku v deskripčních logikách – TA byly známy již pro FOL.
- Hlavní myšlenka je jednoduchá: **"Máme-li ověřit konzistenci daného ABOXu $\mathcal{A}$ vzhledem k TBOXu $\mathcal{T}$, pokusme se sestrojit model $\mathcal{T} \cup \mathcal{A}$. Pokud se nám to podaří, víme, že $\mathcal{A}$ je konzistentní vzhledem k $\mathcal{T}$ "**
- Na každý tablový algoritmus lze vlastně nahlížet jako na *produkční systém* :

Tablové algoritmy

- Tablové algoritmy (TA) slouží k ověřování konzistence ABOXu vzhledem k TBOXu. Nejedná se o žádnou novinku v deskripčních logikách – TA byly známy již pro FOL.
- Hlavní myšlenka je jednoduchá: **"Máme-li ověřit konzistenci daného ABOXu $\mathcal{A}$ vzhledem k TBOXu $\mathcal{T}$, pokusme se sestrojit model $\mathcal{T} \cup \mathcal{A}$. Pokud se nám to podaří, víme, že $\mathcal{A}$ je konzistentní vzhledem k $\mathcal{T}$ "**
- Na každý tablový algoritmus lze vlastně nahlížet jako na *produkční systém* :
  - *Stav* TA (báze dat) je tvořen množinou grafů zúplnění,

Tablové algoritmy

- Tablové algoritmy (TA) slouží k ověřování konzistence ABOXu vzhledem k TBOXu. Nejedná se o žádnou novinku v deskripčních logikách – TA byly známy již pro FOL.
- Hlavní myšlenka je jednoduchá: **"Máme-li ověřit konzistenci daného ABOXu $\mathcal{A}$ vzhledem k TBOXu $\mathcal{T}$, pokusme se sestrojit model $\mathcal{T} \cup \mathcal{A}$. Pokud se nám to podaří, víme, že $\mathcal{A}$ je konzistentní vzhledem k $\mathcal{T}$ "**
- Na každý tablový algoritmus lze vlastně nahlížet jako na *produkční systém* :
  - *Stav* TA (báze dat) je tvořen množinou grafů zúplnění,
  - *inferenční pravidla* (produkční pravidla) implementují sémantiku jednotlivých konstruktů daného jazyka, např. $\exists, \sqcap$, apod., a slouží k populaci grafů zúplnění podle

Tablové algoritmy

- Tablové algoritmy (TA) slouží k ověřování konzistence ABOXu vzhledem k TBOXu. Nejedná se o žádnou novinku v deskripčních logikách – TA byly známy již pro FOL.
- Hlavní myšlenka je jednoduchá: **"Máme-li ověřit konzistenci daného ABOXu $\mathcal{A}$ vzhledem k TBOXu $\mathcal{T}$, pokusme se sestrojit model $\mathcal{T} \cup \mathcal{A}$. Pokud se nám to podaří, víme, že $\mathcal{A}$ je konzistentní vzhledem k $\mathcal{T}$ "**
- Na každý tablový algoritmus lze vlastně nahlížet jako na *produkční systém* :
  - *Stav* TA (báze dat) je tvořen množinou grafů zúplnění,
  - *inferenční pravidla* (produkční pravidla) implementují sémantiku jednotlivých konstruktů daného jazyka, např. $\exists, \sqcap$, apod., a slouží k populaci grafů zúplnění podle
  - zvolené *strategie* pro aplikaci pravidel

Tablové algoritmy

- Tablové algoritmy (TA) slouží k ověřování konzistence ABOXu vzhledem k TBOXu. Nejedná se o žádnou novinku v deskripčních logikách – TA byly známy již pro FOL.
- Hlavní myšlenka je jednoduchá: **"Máme-li ověřit konzistenci daného ABOXu $\mathcal{A}$ vzhledem k TBOXu $\mathcal{T}$, pokusme se sestrojit model $\mathcal{T} \cup \mathcal{A}$. Pokud se nám to podaří, víme, že $\mathcal{A}$ je konzistentní vzhledem k $\mathcal{T}$ "**
- Na každý tablový algoritmus lze vlastně nahlížet jako na *produkční systém* :
  - *Stav* TA (báze dat) je tvořen množinou grafů zúplnění,
  - *inferenční pravidla* (produkční pravidla) implementují sémantiku jednotlivých konstruktů daného jazyka, např. $\exists, \sqcap$, apod., a slouží k populaci grafů zúplnění podle
  - zvolené *strategie* pro aplikaci pravidel

Kontrolní otázka: Co je to RETE algoritmus ?

Grafy zúplnění (completion graphs)

Graf zúplnění je ohodnocený orientovaný graf $G = (V_G, E_G, L_G))$, kde každý uzel $x \in V_G$ je ohodnocen množinou $L_G(x)$ konceptů a každá hrana $\langle x, y \rangle \in E_G$ je ohodnocena množinou hran $L_G(\langle x, y \rangle)$[5]

Grafy zúplnění (completion graphs)

Graf zúplnění je ohodnocený orientovaný graf $G = (V_G, E_G, L_G))$, kde každý uzel $x \in V_G$ je ohodnocen množinou $L_G(x)$ konceptů a každá hrana $\langle x, y \rangle \in E_G$ je ohodnocena množinou hran $L_G(\langle x, y \rangle)$[5]

Přímý spor se vyskytuje v grafu zúplnění $G = (V_G, E_G, L_G))$, pokud $\{A, \neg A\} \subseteq L_G(x)$, nebo $\bot \in L_G(x)$, pro nějaký atomický koncept A a uzel $x \in V_G$

Grafy zúplnění (completion graphs)

Graf zúplnění je ohodnocený orientovaný graf $G = (V_G, E_G, L_G))$, kde každý uzel $x \in V_G$ je ohodnocen množinou $L_G(x)$ konceptů a každá hrana $\langle x, y \rangle \in E_G$ je ohodnocena množinou hran $L_G(\langle x, y \rangle)$[5]

Přímý spor se vyskytuje v grafu zúplnění $G = (V_G, E_G, L_G))$, pokud $\{A, \neg A\} \subseteq L_G(x)$, nebo $\bot \in L_G(x)$, pro nějaký atomický koncept A a uzel $x \in V_G$

Úplný graf zúplnění je takový graf zúplnění $G = (V_G, E_G, L_G))$, na který nelze aplikovat žádné pravidlo z množiny pravidel daného tablového algoritmu.

Grafy zúplnění (completion graphs)

Graf zúplnění je ohodnocený orientovaný graf $G = (V_G, E_G, L_G))$, kde každý uzel $x \in V_G$ je ohodnocen množinou $L_G(x)$ konceptů a každá hrana $\langle x, y \rangle \in E_G$ je ohodnocena množinou hran $L_G(\langle x, y \rangle)$[5]

Přímý spor se vyskytuje v grafu zúplnění $G = (V_G, E_G, L_G))$, pokud $\{A, \neg A\} \subseteq L_G(x)$, nebo $\bot \in L_G(x)$, pro nějaký atomický koncept A a uzel $x \in V_G$

Úplný graf zúplnění je takový graf zúplnění $G = (V_G, E_G, L_G))$, na který nelze aplikovat žádné pravidlo z množiny pravidel daného tablového algoritmu.
Definujme též $\mathcal{I} \models G$ iff $\mathcal{I} \models \mathcal{A}_G$, kde $\mathcal{A}_G$ je ABOX vytvořený z G, který obsahuje

Grafy zúplnění (completion graphs)

Graf zúplnění je ohodnocený orientovaný graf $G = (V_G, E_G, L_G))$, kde každý uzel $x \in V_G$ je ohodnocen množinou $L_G(x)$ konceptů a každá hrana $\langle x, y \rangle \in E_G$ je ohodnocena množinou hran $L_G(\langle x, y \rangle)$[5]

Přímý spor se vyskytuje v grafu zúplnění $G = (V_G, E_G, L_G))$, pokud $\{A, \neg A\} \subseteq L_G(x)$, nebo $\bot \in L_G(x)$, pro nějaký atomický koncept A a uzel $x \in V_G$

Úplný graf zúplnění je takový graf zúplnění $G = (V_G, E_G, L_G))$, na který nelze aplikovat žádné pravidlo z množiny pravidel daného tablového algoritmu.

Definujme též $\mathcal{I} \models G$ iff $\mathcal{I} \models \mathcal{A}_G$, kde $\mathcal{A}_G$ je ABOX vytvořený z G, který obsahuje

- $C(a)$ pro každý vrchol $a \in V_G$ a každý $C \in L_G(a)$ a

Grafy zúplnění (completion graphs)

Graf zúplnění je ohodnocený orientovaný graf $G = (V_G, E_G, L_G))$, kde každý uzel $x \in V_G$ je ohodnocen množinou $L_G(x)$ konceptů a každá hrana $\langle x, y \rangle \in E_G$ je ohodnocena množinou hran $L_G(\langle x, y \rangle)$[5]

Přímý spor se vyskytuje v grafu zúplnění $G = (V_G, E_G, L_G))$, pokud $\{A, \neg A\} \subseteq L_G(x)$, nebo $\bot \in L_G(x)$, pro nějaký atomický koncept A a uzel $x \in V_G$

Úplný graf zúplnění je takový graf zúplnění $G = (V_G, E_G, L_G))$, na který nelze aplikovat žádné pravidlo z množiny pravidel daného tablového algoritmu.

Definujme též $\mathcal{I} \models G$ iff $\mathcal{I} \models \mathcal{A}_G$, kde $\mathcal{A}_G$ je ABOX vytvořený z G, který obsahuje

- $C(a)$ pro každý vrchol $a \in V_G$ a každý $C \in L_G(a)$ a
- $R(a, b)$ pro každou hranu $\langle a, b \rangle \in E_G$ a každou $R \in L_G(a, b)$ a

Grafy zúplnění (completion graphs)

Graf zúplnění je ohodnocený orientovaný graf $G = (V_G, E_G, L_G))$, kde každý uzel $x \in V_G$ je ohodnocen množinou $L_G(x)$ konceptů a každá hrana $\langle x, y \rangle \in E_G$ je ohodnocena množinou hran $L_G(\langle x, y \rangle)$[5]

Přímý spor se vyskytuje v grafu zúplnění $G = (V_G, E_G, L_G))$, pokud $\{A, \neg A\} \subseteq L_G(x)$, nebo $\bot \in L_G(x)$, pro nějaký atomický koncept A a uzel $x \in V_G$

Úplný graf zúplnění je takový graf zúplnění $G = (V_G, E_G, L_G))$, na který nelze aplikovat žádné pravidlo z množiny pravidel daného tablového algoritmu.

Definujme též $\mathcal{I} \models G$ iff $\mathcal{I} \models \mathcal{A}_G$, kde $\mathcal{A}_G$ je ABOX vytvořený z G, který obsahuje

- $C(a)$ pro každý vrchol $a \in V_G$ a každý $C \in L_G(a)$ a
- $R(a, b)$ pro každou hranu $\langle a, b \rangle \in E_G$ a každou $R \in L_G(a, b)$ a

Pozor na záměnu. Co je to úplný graf v teorii grafů ?

Tablový algoritmus pro $\mathcal{ALC}$ bez TBOXu

máme $\mathcal{K} = (\mathcal{T}, \mathcal{A})$. Uvažujme zatím pro jednoduchost, že $\mathcal{T} = \emptyset$.

Tablový algoritmus pro $\mathcal{ALC}$ bez TBOXu

máme $\mathcal{K} = (\mathcal{T}, \mathcal{A})$. Uvažujme zatím pro jednoduchost, že $\mathcal{T} = \emptyset$.

0 (Předzpracování) Transformuj všechny koncepty, které se vyskytují v $\mathcal{K}$ do tzv. "negační normální formy" (NNF). Jedná se o aplikaci ekvivalentních úprav známých z výrokové a predikátové logiky, po jejichž aplikace se negace $\neg$ vyskytuje pouze před atomickými koncepty. Např. $\neg(A \sqcap B)$ se vyjádří pomocí de Morganových pravidel jako $\neg A \sqcup \neg B$).

Tablový algoritmus pro $\mathcal{ALC}$ bez TBOXu

máme $\mathcal{K} = (\mathcal{T}, \mathcal{A})$. Uvažujme zatím pro jednoduchost, že $\mathcal{T} = \emptyset$.

0 (Předzpracování) Transformuj všechny koncepty, které se vyskytují v $\mathcal{K}$ do tzv. "negační normální formy" (NNF). Jedná se o aplikaci ekvivalentních úprav známých z výrokové a predikátové logiky, po jejichž aplikace se negace $\neg$ vyskytuje pouze před atomickými koncepty. Např. $\neg(A \sqcap B)$ se vyjádří pomocí de Morganových pravidel jako $\neg A \sqcup \neg B$).

1 (Inicializace) Počáteční stav algoritmu je $S_0 = \{G_0\}$, kde $G_0 = (V_{G_0}, E_{G_0}, L_{G_0})$ je obrazem $\mathcal{A}$:

Tablový algoritmus pro $\mathcal{ALC}$ bez TBOXu

máme $\mathcal{K} = (\mathcal{T}, \mathcal{A})$. Uvažujme zatím pro jednoduchost, že $\mathcal{T} = \emptyset$.

0 (Předzpracování) Transformuj všechny koncepty, které se vyskytují v $\mathcal{K}$ do tzv. "negační normální formy" (NNF). Jedná se o aplikaci ekvivalentních úprav známých z výrokové a predikátové logiky, po jejichž aplikace se negace $\neg$ vyskytuje pouze před atomickými koncepty. Např. $\neg(A \sqcap B)$ se vyjádří pomocí de Morganových pravidel jako $\neg A \sqcup \neg B$).

1 (Inicializace) Počáteční stav algoritmu je $S_0 = \{G_0\}$, kde $G_0 = (V_{G_0}, E_{G_0}, L_{G_0})$ je obrazem $\mathcal{A}$:

- pro každý $C(a)$ je $a \in V_{G_0}$ a $C \in L_{G_0}(a)$

Tablový algoritmus pro $\mathcal{ALC}$ bez TBOXu

máme $\mathcal{K} = (\mathcal{T}, \mathcal{A})$. Uvažujme zatím pro jednoduchost, že $\mathcal{T} = \emptyset$.

0 (Předzpracování) Transformuj všechny koncepty, které se vyskytují v $\mathcal{K}$ do tzv. "negační normální formy" (NNF). Jedná se o aplikaci ekvivalentních úprav známých z výrokové a predikátové logiky, po jejichž aplikace se negace $\neg$ vyskytuje pouze před atomickými koncepty. Např. $\neg(A \sqcap B)$ se vyjádří pomocí de Morganových pravidel jako $\neg A \sqcup \neg B$).

1 (Inicializace) Počáteční stav algoritmu je $S_0 = \{G_0\}$, kde $G_0 = (V_{G_0}, E_{G_0}, L_{G_0})$ je obrazem $\mathcal{A}$:

- pro každý $C(a)$ je $a \in V_{G_0}$ a $C \in L_{G_0}(a)$
- pro každý $R(a, b)$ je $\langle a, b \rangle \in E_{G_0}$ a $R \in L_{G_0}(a, b)$

Tablový algoritmus pro $\mathcal{ALC}$ bez TBOXu

máme $\mathcal{K} = (\mathcal{T}, \mathcal{A})$. Uvažujme zatím pro jednoduchost, že $\mathcal{T} = \emptyset$.

0 (Předzpracování) Transformuj všechny koncepty, které se vyskytují v $\mathcal{K}$ do tzv. "negační normální formy" (NNF). Jedná se o aplikaci ekvivalentních úprav známých z výrokové a predikátové logiky, po jejichž aplikace se negace $\neg$ vyskytuje pouze před atomickými koncepty. Např. $\neg(A \sqcap B)$ se vyjádří pomocí de Morganových pravidel jako $\neg A \sqcup \neg B$).

1 (Inicializace) Počáteční stav algoritmu je $S_0 = \{G_0\}$, kde $G_0 = (V_{G_0}, E_{G_0}, L_{G_0})$ je obrazem $\mathcal{A}$:

- pro každý $C(a)$ je $a \in V_{G_0}$ a $C \in L_{G_0}(a)$
- pro každý $R(a, b)$ je $\langle a, b \rangle \in E_{G_0}$ a $R \in L_{G_0}(a, b)$
- množiny $V_{G_0}, E_{G_0}, L_{G_0}$ jsou nejmenší s těmito vlastnostmi.

Tablový algoritmus pro $\mathcal{ALC}$ bez TBOXu (2)

...

2 (Test konzistence) Aktuální stav tablového algoritmu označme S. Pokud každý $G \in S$ obsahuje přímý spor, potom algoritmus končí s výsledkem "NEKONZISTENTNÍ"

Tablový algoritmus pro $\mathcal{ALC}$ bez TBOXu (2)

...

2. (Test konzistence) Aktuální stav tablového algoritmu označme S. Pokud každý $G \in S$ obsahuje přímý spor, potom algoritmus končí s výsledkem "NEKONZISTENTNÍ"
3. (Test modelu) Vybereme jeden $G \in S$, který neobsahuje přímý spor. Je-li G úplný vzhledem k pravidlům uvedeným dále, potom algoritmus končí s výsledkem "KONZISTENTNÍ"

Tablový algoritmus pro $\mathcal{ALC}$ bez TBOXu (2)

...

2. (Test konzistence) Aktuální stav tablového algoritmu označme S. Pokud každý $G \in S$ obsahuje přímý spor, potom algoritmus končí s výsledkem "NEKONZISTENTNÍ"
3. (Test modelu) Vybereme jeden $G \in S$, který neobsahuje přímý spor. Je-li G úplný vzhledem k pravidlům uvedeným dále, potom algoritmus končí s výsledkem "KONZISTENTNÍ"
4. (Aplikace pravidla) Nalezneme pravidlo, které ja na G aplikovatelné - aplikací tohoto pravidla získáme ze stavu S nový stav S' a algoritmus pokračuje bodem 2.

TA pro $\mathcal{ALC}$ bez TBOXu – odvozovací pravidla

$\rightarrow_{\sqcap}$ pravidlo

TA pro $\mathcal{ALC}$ bez TBOXu – odvozovací pravidla

$\rightarrow_{\sqcap}$ pravidlo

if $(C_1 \sqcap C_2) \in L_G(a)$ a $\{C_1, C_2\} \not\subseteq L_G(a)$ pro nějaké $a \in V_G$.

TA pro $\mathcal{ALC}$ bez TBOXu – odvozovací pravidla

$\rightarrow_{\sqcap}$ pravidlo

if $(C_1 \sqcap C_2) \in L_G(a)$ a $\{C_1, C_2\} \not\subseteq L_G(a)$ pro nějaké $a \in V_G$.

then $S' = S \cup \{G'\} \setminus \{G\}$, kde $G' = (V_G, E_G, L_{G'})$, a $L_{G'}(a) = L_G(a) \cup \{C_1, C_2\}$ a jinak se shoduje s L_G.

TA pro $\mathcal{ALC}$ bez TBOXu – odvozovací pravidla

$\rightarrow_{\sqcap}$ pravidlo

if $(C_1 \sqcap C_2) \in L_G(a)$ a $\{C_1, C_2\} \not\subseteq L_G(a)$ pro nějaké $a \in V_G$.

then $S' = S \cup \{G'\} \setminus \{G\}$, kde $G' = (V_G, E_G, L_{G'})$, a $L_{G'}(a) = L_G(a) \cup \{C_1, C_2\}$ a jinak se shoduje s L_G.

$\rightarrow_{\sqcup}$ pravidlo

TA pro $\mathcal{ALC}$ bez TBOXu – odvozovací pravidla

$\rightarrow_{\sqcap}$ pravidlo

if $(C_1 \sqcap C_2) \in L_G(a)$ a $\{C_1, C_2\} \not\subseteq L_G(a)$ pro nějaké $a \in V_G$.

then $S' = S \cup \{G'\} \setminus \{G\}$, kde $G' = (V_G, E_G, L_{G'})$, a $L_{G'}(a) = L_G(a) \cup \{C_1, C_2\}$ a jinak se shoduje s L_G.

$\rightarrow_{\sqcup}$ pravidlo

if $(C_1 \sqcup C_2) \in L_G(a)$ a $\{C_1, C_2\} \cap L_G(a) = \emptyset$ pro nějaké $a \in V_G$.

TA pro $\mathcal{ALC}$ bez TBOXu – odvozovací pravidla

$\rightarrow_{\sqcap}$ pravidlo

if $(C_1 \sqcap C_2) \in L_G(a)$ a $\{C_1, C_2\} \not\subseteq L_G(a)$ pro nějaké $a \in V_G$.

then $S' = S \cup \{G'\} \setminus \{G\}$, kde $G' = (V_G, E_G, L_{G'})$, a $L_{G'}(a) = L_G(a) \cup \{C_1, C_2\}$ a jinak se shoduje s L_G.

$\rightarrow_{\sqcup}$ pravidlo

if $(C_1 \sqcup C_2) \in L_G(a)$ a $\{C_1, C_2\} \cap L_G(a) = \emptyset$ pro nějaké $a \in V_G$.

then $S' = S \cup \{G_1, G_2\} \setminus \{G\}$, kde $G_{(1|2)} = (V_G, E_G, L_{G_{(1|2)}})$, a $L_{G_{(1|2)}}(a) = L_G(a) \cup \{C_{(1|2)}\}$ a jinak se shoduje s L_G.

TA pro $\mathcal{ALC}$ bez TBOXu – odvozovací pravidla

$\rightarrow_{\sqcap}$ pravidlo

if $(C_1 \sqcap C_2) \in L_G(a)$ a $\{C_1, C_2\} \not\subseteq L_G(a)$ pro nějaké $a \in V_G$.

then $S' = S \cup \{G'\} \setminus \{G\}$, kde $G' = (V_G, E_G, L_{G'})$, a $L_{G'}(a) = L_G(a) \cup \{C_1, C_2\}$ a jinak se shoduje s L_G.

$\rightarrow_{\sqcup}$ pravidlo

if $(C_1 \sqcup C_2) \in L_G(a)$ a $\{C_1, C_2\} \cap L_G(a) = \emptyset$ pro nějaké $a \in V_G$.

then $S' = S \cup \{G_1, G_2\} \setminus \{G\}$, kde $G_{(1|2)} = (V_G, E_G, L_{G_{(1|2)}})$, a $L_{G_{(1|2)}}(a) = L_G(a) \cup \{C_{(1|2)}\}$ a jinak se shoduje s L_G.

$\rightarrow_{\exists}$ pravidlo

TA pro $\mathcal{ALC}$ bez TBOXu – odvozovací pravidla

$\rightarrow_{\sqcap}$ pravidlo

if $(C_1 \sqcap C_2) \in L_G(a)$ a $\{C_1, C_2\} \not\subseteq L_G(a)$ pro nějaké $a \in V_G$.

then $S' = S \cup \{G'\} \setminus \{G\}$, kde $G' = (V_G, E_G, L_{G'})$, a $L_{G'}(a) = L_G(a) \cup \{C_1, C_2\}$ a jinak se shoduje s L_G.

$\rightarrow_{\sqcup}$ pravidlo

if $(C_1 \sqcup C_2) \in L_G(a)$ a $\{C_1, C_2\} \cap L_G(a) = \emptyset$ pro nějaké $a \in V_G$.

then $S' = S \cup \{G_1, G_2\} \setminus \{G\}$, kde $G_{(1|2)} = (V_G, E_G, L_{G_{(1|2)}})$, a $L_{G_{(1|2)}}(a) = L_G(a) \cup \{C_{(1|2)}\}$ a jinak se shoduje s L_G.

$\rightarrow_{\exists}$ pravidlo

if $(\exists R \cdot C) \in L_G(a)$ a neexistuje $b \in V_G$ takové, že $R \in L_G(a, b)$ a současně $C \in L_G(b)$.

TA pro $\mathcal{ALC}$ bez TBOXu – odvozovací pravidla

$\rightarrow_{\sqcap}$ pravidlo

if $(C_1 \sqcap C_2) \in L_G(a)$ a $\{C_1, C_2\} \not\subseteq L_G(a)$ pro nějaké $a \in V_G$.

then $S' = S \cup \{G'\} \setminus \{G\}$, kde $G' = (V_G, E_G, L_{G'})$, a $L_{G'}(a) = L_G(a) \cup \{C_1, C_2\}$ a jinak se shoduje s L_G.

$\rightarrow_{\sqcup}$ pravidlo

if $(C_1 \sqcup C_2) \in L_G(a)$ a $\{C_1, C_2\} \cap L_G(a) = \emptyset$ pro nějaké $a \in V_G$.

then $S' = S \cup \{G_1, G_2\} \setminus \{G\}$, kde $G_{(1|2)} = (V_G, E_G, L_{G_{(1|2)}})$, a $L_{G_{(1|2)}}(a) = L_G(a) \cup \{C_{(1|2)}\}$ a jinak se shoduje s L_G.

$\rightarrow_{\exists}$ pravidlo

if $(\exists R \cdot C) \in L_G(a)$ a neexistuje $b \in V_G$ takové, že $R \in L_G(a, b)$ a současně $C \in L_G(b)$.

then $S' = S \cup \{G'\} \setminus \{G\}$, kde $G' = (V_G \cup \{b\}, E_G \cup \{\langle a, b\rangle\}, L_{G'})$, a $L_{G'}(b) = \{C\}$, $L_{G'}(a, b) = \{R\}$ a jinak se shoduje s L_G.

TA pro $\mathcal{ALC}$ bez TBOXu – odvozovací pravidla

$\rightarrow_{\sqcap}$ pravidlo

if $(C_1 \sqcap C_2) \in L_G(a)$ a $\{C_1, C_2\} \not\subseteq L_G(a)$ pro nějaké $a \in V_G$.

then $S' = S \cup \{G'\} \setminus \{G\}$, kde $G' = (V_G, E_G, L_{G'})$, a $L_{G'}(a) = L_G(a) \cup \{C_1, C_2\}$ a jinak se shoduje s L_G.

$\rightarrow_{\sqcup}$ pravidlo

if $(C_1 \sqcup C_2) \in L_G(a)$ a $\{C_1, C_2\} \cap L_G(a) = \emptyset$ pro nějaké $a \in V_G$.

then $S' = S \cup \{G_1, G_2\} \setminus \{G\}$, kde $G_{(1|2)} = (V_G, E_G, L_{G_{(1|2)}})$, a $L_{G_{(1|2)}}(a) = L_G(a) \cup \{C_{(1|2)}\}$ a jinak se shoduje s L_G.

$\rightarrow_{\exists}$ pravidlo

if $(\exists R \cdot C) \in L_G(a)$ a neexistuje $b \in V_G$ takové, že $R \in L_G(a, b)$ a současně $C \in L_G(b)$.

then $S' = S \cup \{G'\} \setminus \{G\}$, kde $G' = (V_G \cup \{b\}, E_G \cup \{\langle a, b\rangle\}, L_{G'})$, a $L_{G'}(b) = \{C\}$, $L_{G'}(a, b) = \{R\}$ a jinak se shoduje s L_G.

$\rightarrow_{\forall}$ pravidlo

TA pro $\mathcal{ALC}$ bez TBOXu – odvozovací pravidla

$\rightarrow_{\sqcap}$ pravidlo

if $(C_1 \sqcap C_2) \in L_G(a)$ a $\{C_1, C_2\} \not\subseteq L_G(a)$ pro nějaké $a \in V_G$.

then $S' = S \cup \{G'\} \setminus \{G\}$, kde $G' = (V_G, E_G, L_{G'})$, a $L_{G'}(a) = L_G(a) \cup \{C_1, C_2\}$ a jinak se shoduje s L_G.

$\rightarrow_{\sqcup}$ pravidlo

if $(C_1 \sqcup C_2) \in L_G(a)$ a $\{C_1, C_2\} \cap L_G(a) = \emptyset$ pro nějaké $a \in V_G$.

then $S' = S \cup \{G_1, G_2\} \setminus \{G\}$, kde $G_{(1|2)} = (V_G, E_G, L_{G_{(1|2)}})$, a $L_{G_{(1|2)}}(a) = L_G(a) \cup \{C_{(1|2)}\}$ a jinak se shoduje s L_G.

$\rightarrow_{\exists}$ pravidlo

if $(\exists R \cdot C) \in L_G(a)$ a neexistuje $b \in V_G$ takové, že $R \in L_G(a, b)$ a současně $C \in L_G(b)$.

then $S' = S \cup \{G'\} \setminus \{G\}$, kde $G' = (V_G \cup \{b\}, E_G \cup \{\langle a, b\rangle\}, L_{G'})$, a $L_{G'}(b) = \{C\}$, $L_{G'}(a, b) = \{R\}$ a jinak se shoduje s L_G.

$\rightarrow_{\forall}$ pravidlo

if $(\forall R \cdot C) \in L_G(a)$ a existuje $b \in V_G$ takové, že $R \in L_G(a, b)$ a současně $C \notin L_G(b)$.

TA pro $\mathcal{ALC}$ bez TBOXu – odvozovací pravidla

$\rightarrow_{\sqcap}$ pravidlo

if $(C_1 \sqcap C_2) \in L_G(a)$ a $\{C_1, C_2\} \not\subseteq L_G(a)$ pro nějaké $a \in V_G$.

then $S' = S \cup \{G'\} \setminus \{G\}$, kde $G' = (V_G, E_G, L_{G'})$, a $L_{G'}(a) = L_G(a) \cup \{C_1, C_2\}$ a jinak se shoduje s L_G.

$\rightarrow_{\sqcup}$ pravidlo

if $(C_1 \sqcup C_2) \in L_G(a)$ a $\{C_1, C_2\} \cap L_G(a) = \emptyset$ pro nějaké $a \in V_G$.

then $S' = S \cup \{G_1, G_2\} \setminus \{G\}$, kde $G_{(1|2)} = (V_G, E_G, L_{G_{(1|2)}})$, a $L_{G_{(1|2)}}(a) = L_G(a) \cup \{C_{(1|2)}\}$ a jinak se shoduje s L_G.

$\rightarrow_{\exists}$ pravidlo

if $(\exists R \cdot C) \in L_G(a)$ a neexistuje $b \in V_G$ takové, že $R \in L_G(a, b)$ a současně $C \in L_G(b)$.

then $S' = S \cup \{G'\} \setminus \{G\}$, kde $G' = (V_G \cup \{b\}, E_G \cup \{\langle a, b\rangle\}, L_{G'})$, a $L_{G'}(b) = \{C\}$, $L_{G'}(a, b) = \{R\}$ a jinak se shoduje s L_G.

$\rightarrow_{\forall}$ pravidlo

if $(\forall R \cdot C) \in L_G(a)$ a existuje $b \in V_G$ takové, že $R \in L_G(a, b)$ a současně $C \notin L_G(b)$.

then $S' = S \cup \{G'\} \setminus \{G\}$, kde $G' = (V_G, E_G, L_{G'})$, a $L_{G'}(b) = L_G(b) \cup \{D\}$ a jinak se shoduje s L_G.

Je představený TA konečný ?

Konečnost je snadným důsledkem následujících faktů :

- $\mathcal{K}$ je konečná

Je představený TA konečný ?

Konečnost je snadným důsledkem následujících faktů :

- $\mathcal{K}$ je konečná
- stav tablového algoritmu můžeme v každém kroku obohatit o nejvýše jeden graf zúplnění a to pouze po aplikaci $\rightarrow_{\sqcup}$ pravidla. Počet disjunkcí v $\mathcal{K}$ je konečný, tedy i toto pravidlo může být aplikovánou pouze konečněkrát.

Je představený TA konečný ?

Konečnost je snadným důsledkem následujících faktů :

- $\mathcal{K}$ je konečná
- stav tablového algoritmu můžeme v každém kroku obohatit o nejvýše jeden graf zúplnění a to pouze po aplikaci $\rightarrow_{\sqcup}$ pravidla. Počet disjunkcí v $\mathcal{K}$ je konečný, tedy i toto pravidlo může být aplikovánou pouze konečněkrát.
- pro každý graf zúplnění $G = (V_G, E_G, L_G)$ platí, že počet vrcholů v V_G je nejvýše rovna počtu individuálů v $\mathcal{A}$ plus počet existenciálních kvantifikátorů v $\mathcal{A}$.

Je představený TA konečný ?

Konečnost je snadným důsledkem následujících faktů :

- $\mathcal{K}$ je konečná
- stav tablového algoritmu můžeme v každém kroku obohatit o nejvýše jeden graf zúplnění a to pouze po aplikaci $\rightarrow_{\sqcup}$ pravidla. Počet disjunkcí v $\mathcal{K}$ je konečný, tedy i toto pravidlo může být aplikovánou pouze konečněkrát.
- pro každý graf zúplnění $G = (V_G, E_G, L_G)$ platí, že počet vrcholů v V_G je nejvýše rovna počtu individuálů v $\mathcal{A}$ plus počet existenciálních kvantifikátorů v $\mathcal{A}$.
- po aplikaci libovolného z pravidel $\rightarrow_{\sqcap}, \rightarrow_{\exists}, \rightarrow_{\forall}$ se buď v G vytvoří nová hrana, nový vrchol, nebo se přidá ohodnocení k existujícímu vrcholu či hraně. Všechny tyto operace jsou konečné.

Je představený TA korektní ?

- Korektnost ověříme rovněž snadno. Vezmeme-li libovolnou $\mathcal{I} \models \mathcal{A}_{G_i}$, potom musí $\mathcal{I} \models \mathcal{A}_{G_{i+1}}$. Musíme ukázat, že aplikací každého z pravidel zachováme splnitelnost. Jako příklad vezměme $\rightarrow_\exists$ pravidlo:

Je představený TA korektní ?

- Korektnost ověříme rovněž snadno. Vezmeme-li libovolnou $\mathcal{I} \models \mathcal{A}_{G_i}$, potom musí $\mathcal{I} \models \mathcal{A}_{G_{i+1}}$. Musíme ukázat, že aplikací každého z pravidel zachováme splnitelnost. Jako příklad vezměme $\rightarrow_\exists$ pravidlo:
  - Před aplikací pravidla $\rightarrow_\exists$ platilo pro $a \in V_{G_i}$, že $(\exists R \cdot C) \in L_{G_i}(a)$.

Je představený TA korektní ?

- Korektnost ověříme rovněž snadno. Vezmeme-li libovolnou $\mathcal{I} \models \mathcal{A}_{G_i}$, potom musí $\mathcal{I} \models \mathcal{A}_{G_{i+1}}$. Musíme ukázat, že aplikací každého z pravidel zachováme splnitelnost. Jako příklad vezměme $\rightarrow_\exists$ pravidlo:
  - Před aplikací pravidla $\rightarrow_\exists$ platilo pro $a \in V_{G_i}$, že $(\exists R \cdot C) \in L_{G_i}(a)$.
  - Tedy $a^\mathcal{I} \in (\exists R \cdot C)^\mathcal{I}$.

Je představený TA korektní ?

- Korektnost ověříme rovněž snadno. Vezmeme-li libovolnou $\mathcal{I} \models \mathcal{A}_{G_i}$, potom musí $\mathcal{I} \models \mathcal{A}_{G_{i+1}}$. Musíme ukázat, že aplikací každého z pravidel zachováme splnitelnost. Jako příklad vezměme $\rightarrow_\exists$ pravidlo:
  - Před aplikací pravidla $\rightarrow_\exists$ platilo pro $a \in V_{G_i}$, že $(\exists R \cdot C) \in L_{G_i}(a)$.
  - Tedy $a^\mathcal{I} \in (\exists R \cdot C)^\mathcal{I}$.
  - Tedy v $\Delta^\mathcal{I}$ musí existovat nějaký i, pro který $\langle a^\mathcal{I}, i \rangle \in R^\mathcal{I}$ a současně $i \in C^\mathcal{I}$.

Je představený TA korektní ?

- Korektnost ověříme rovněž snadno. Vezmeme-li libovolnou $\mathcal{I} \models \mathcal{A}_{G_i}$, potom musí $\mathcal{I} \models \mathcal{A}_{G_{i+1}}$. Musíme ukázat, že aplikací každého z pravidel zachováme splnitelnost. Jako příklad vezměme $\rightarrow_\exists$ pravidlo:
  - Před aplikací pravidla $\rightarrow_\exists$ platilo pro $a \in V_{G_i}$, že $(\exists R \cdot C) \in L_{G_i}(a)$.
  - Tedy $a^\mathcal{I} \in (\exists R \cdot C)^\mathcal{I}$.
  - Tedy v $\Delta^\mathcal{I}$ musí existovat nějaký i, pro který $\langle a^\mathcal{I}, i \rangle \in R^\mathcal{I}$ a současně $i \in C^\mathcal{I}$.
  - Aplikací pravidla $\rightarrow_\exists$ jsme vytvořili v G_{i+1} nový vrchol b a upravili ohodnocení hrany $\langle a, b \rangle$ a vrcholu b.

Je představený TA korektní ?

- Korektnost ověříme rovněž snadno. Vezmeme-li libovolnou $\mathcal{I} \models \mathcal{A}_{G_i}$, potom musí $\mathcal{I} \models \mathcal{A}_{G_{i+1}}$. Musíme ukázat, že aplikací každého z pravidel zachováme splnitelnost. Jako příklad vezměme $\rightarrow_\exists$ pravidlo:
  - Před aplikací pravidla $\rightarrow_\exists$ platilo pro $a \in V_{G_i}$, že $(\exists R \cdot C) \in L_{G_i}(a)$.
  - Tedy $a^\mathcal{I} \in (\exists R \cdot C)^\mathcal{I}$.
  - Tedy v $\Delta^\mathcal{I}$ musí existovat nějaký i, pro který $\langle a^\mathcal{I}, i \rangle \in R^\mathcal{I}$ a současně $i \in C^\mathcal{I}$.
  - Aplikací pravidla $\rightarrow_\exists$ jsme vytvořili v G_{i+1} nový vrchol b a upravili ohodnocení hrany $\langle a, b \rangle$ a vrcholu b.
  - Stačí tedy, položíme-li $i = b^\mathcal{I}$ a vidíme, že po aplikaci pravidla jsme pouze "zhmotnili" doménový element, který jinak v interpretaci musí být vždy přítomen na základě sémantiky konstruktu $\exists$. Pravidlo je tedy korektní.

Je představený TA korektní ?

- Korektnost ověříme rovněž snadno. Vezmeme-li libovolnou $\mathcal{I} \models \mathcal{A}_{G_i}$, potom musí $\mathcal{I} \models \mathcal{A}_{G_{i+1}}$. Musíme ukázat, že aplikací každého z pravidel zachováme splnitelnost. Jako příklad vezměme $\rightarrow_\exists$ pravidlo:
  - Před aplikací pravidla $\rightarrow_\exists$ platilo pro $a \in V_{G_i}$, že $(\exists R \cdot C) \in L_{G_i}(a)$.
  - Tedy $a^\mathcal{I} \in (\exists R \cdot C)^\mathcal{I}$.
  - Tedy v $\Delta^\mathcal{I}$ musí existovat nějaký i, pro který $\langle a^\mathcal{I}, i \rangle \in R^\mathcal{I}$ a současně $i \in C^\mathcal{I}$.
  - Aplikací pravidla $\rightarrow_\exists$ jsme vytvořili v G_{i+1} nový vrchol b a upravili ohodnocení hrany $\langle a, b \rangle$ a vrcholu b.
  - Stačí tedy, položíme-li $i = b^\mathcal{I}$ a vidíme, že po aplikaci pravidla jsme pouze "zhmotnili" doménový element, který jinak v interpretaci musí být vždy přítomen na základě sémantiky konstruktu $\exists$. Pravidlo je tedy korektní.
- Pro ostatní pravidla bychom korektnost ukázali stejným způsobem.

Je představený TA úplný ?

- Pro dokázání úplnosti musíme zkonstruovat model pro každý úplný graf zúplnění G, který neobsahuje přímý spor. Kanonický model $\mathcal{I}$ můžeme zkonstruovat takto:
  - Doménu $\Delta^{\mathcal{I}}$ budou tvořit všechny vrcholy grafu G.

- Je zřejmé, že $\mathcal{I}$ je modelem $\mathcal{A}_G$. Indukcí "dozadu" podle aplikovaných pravidel lze ukázat, že musí být též modelem každého předchozího kroku a ve výsledku tedy i $\mathcal{A}$.

Je představený TA úplný ?

- Pro dokázání úplnosti musíme zkonstruovat model pro každý úplný graf zúplnění G, který neobsahuje přímý spor. Kanonický model $\mathcal{I}$ můžeme zkonstruovat takto:
  - Doménu $\Delta^{\mathcal{I}}$ budou tvořit všechny vrcholy grafu G.
  - Pro každý atomický koncept A definujeme $A^{\mathcal{I}} = \{a \mid A \in L_G(a)\}$
- Je zřejmé, že $\mathcal{I}$ je modelem $\mathcal{A}_G$. Indukcí "dozadu" podle aplikovaných pravidel lze ukázat, že musí být též modelem každého předchozího kroku a ve výsledku tedy i $\mathcal{A}$.

Je představený TA úplný ?

- Pro dokázání úplnosti musíme zkonstruovat model pro každý úplný graf zúplnění G, který neobsahuje přímý spor. Kanonický model $\mathcal{I}$ můžeme zkonstruovat takto:
  - Doménu $\Delta^{\mathcal{I}}$ budou tvořit všechny vrcholy grafu G.
  - Pro každý atomický koncept A definujeme $A^{\mathcal{I}} = \{a \mid A \in L_G(a)\}$
  - Pro každou atomickou roli R definujeme $R^{\mathcal{I}} = \{\langle a, b\rangle \mid R \in L_G(a, b)\}$
- Je zřejmé, že $\mathcal{I}$ je modelem $\mathcal{A}_G$. Indukcí "dozadu" podle aplikovaných pravidel lze ukázat, že musí být též modelem každého předchozího kroku a ve výsledku tedy i $\mathcal{A}$.

Několik poznámek k TA

- Proč jsme vlastně používali pojem graf zúplnění ? Nestačí nám držet si stav tablového algoritmu v ABOXu ?

Několik poznámek k TA

- Proč jsme vlastně používali pojem graf zúplnění ? Nestačí nám držet si stav tablového algoritmu v ABOXu ?
  - Pro námi představený algoritmus bychom si ABOXy opravdu vystačili ! My jsme zavedli grafy zúplnění jednak proto, že grafová reprezentace je názornější, především však proto, že v případě TA pro $\mathcal{ALC}$ a složitější logiky nám již ABOX pro reprezentaci stavu reasoneru bohužel nestačí.

Několik poznámek k TA

- Proč jsme vlastně používali pojem graf zúplnění ? Nestačí nám držet si stav tablového algoritmu v ABOXu ?
  - Pro námi představený algoritmus bychom si ABOXy opravdu vystačili ! My jsme zavedli grafy zúplnění jednak proto, že grafová reprezentace je názornější, především však proto, že v případě TA pro $\mathcal{ALC}$ a složitější logiky nám již ABOX pro reprezentaci stavu reasoneru bohužel nestačí.
- Jak je to s komplexitou představeného algoritmu ?

Několik poznámek k TA

- Proč jsme vlastně používali pojem graf zúplnění ? Nestačí nám držet si stav tablového algoritmu v ABOXu ?
  - Pro námi představený algoritmus bychom si ABOXy opravdu vystačili ! My jsme zavedli grafy zúplnění jednak proto, že grafová reprezentace je názornější, především však proto, že v případě TA pro $\mathcal{ALC}$ a složitější logiky nám již ABOX pro reprezentaci stavu reasoneru bohužel nestačí.
- Jak je to s komplexitou představeného algoritmu ?
  - Spokojíme se s konstatováním (bez důkazu), že se jedná o problém ve třídě P-SPACE.

Příklad běhu TA

Example

Ověřme konzistenci ontologie $\mathcal{K}_2 = (\emptyset, \mathcal{A}_2)$, kde $\mathcal{A}_2 = \{(\exists maDite \cdot Muz \sqcap \exists maDite \cdot Prarodic \sqcap \neg\exists maDite \cdot (Muz \sqcap Prarodic))(JAN)\}$.

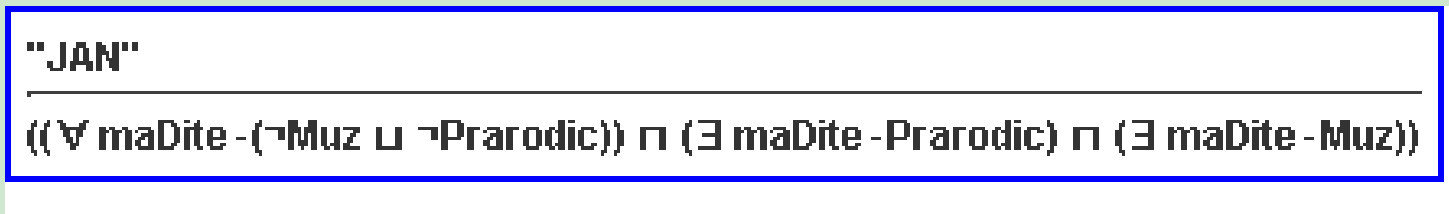
- Převedeme zmíněný koncept do NNF:
  $\exists maDite \cdot Muz \sqcap \exists maDite \cdot Prarodic \sqcap \forall maDite \cdot (\neg Muz \sqcup \neg Prarodic)$

Příklad běhu TA

Example

Ověřme konzistenci ontologie $\mathcal{K}_2 = (\emptyset, \mathcal{A}_2)$, kde $\mathcal{A}_2 = \{(\exists maDite \cdot Muz \sqcap \exists maDite \cdot Prarodic \sqcap \neg\exists maDite \cdot (Muz \sqcap Prarodic))(JAN)\}$.

- Převedeme zmíněný koncept do NNF:
  $\exists maDite \cdot Muz \sqcap \exists maDite \cdot Prarodic \sqcap \forall maDite \cdot (\neg Muz \sqcup \neg Prarodic)$
- Počáteční stav G_0 tablového algoritmu je tedy



Příklad běhu TA (2)

Example

...

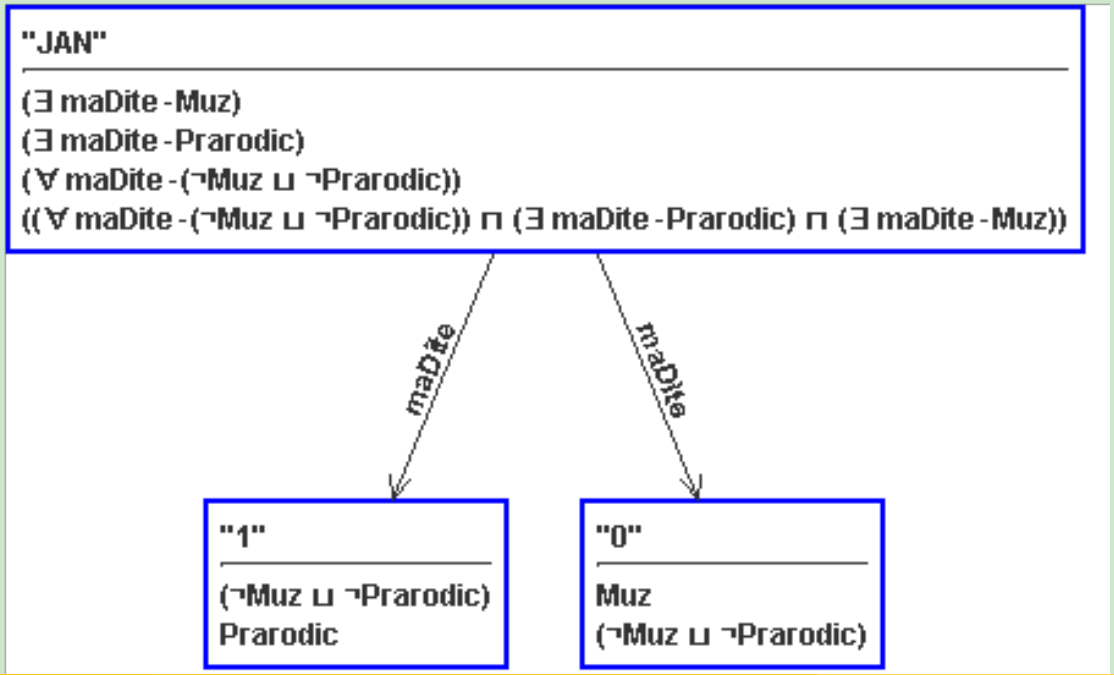
- Nyní se vykonají 4 sekvence kroků 2,3,4 tablového algoritmu, které můžeme znázornit vývojem stavu během kroku 4:

Příklad běhu TA (2)

Example

...

- Nyní se vykonají 4 sekvence kroků 2,3,4 tablového algoritmu, které můžeme znázornit vývojem stavu během kroku 4:
- $\{G_0\} \overset{\sqcap\text{-pravidlo}}{\longrightarrow} \{G_1\} \overset{\exists\text{-pravidlo}}{\longrightarrow} \{G_2\} \overset{\exists\text{-pravidlo}}{\longrightarrow} \{G_3\} \overset{\forall\text{-pravidlo}}{\longrightarrow} \{G_4\}$, kde G_4 je



Příklad běhu TA (3)

Example

. . .

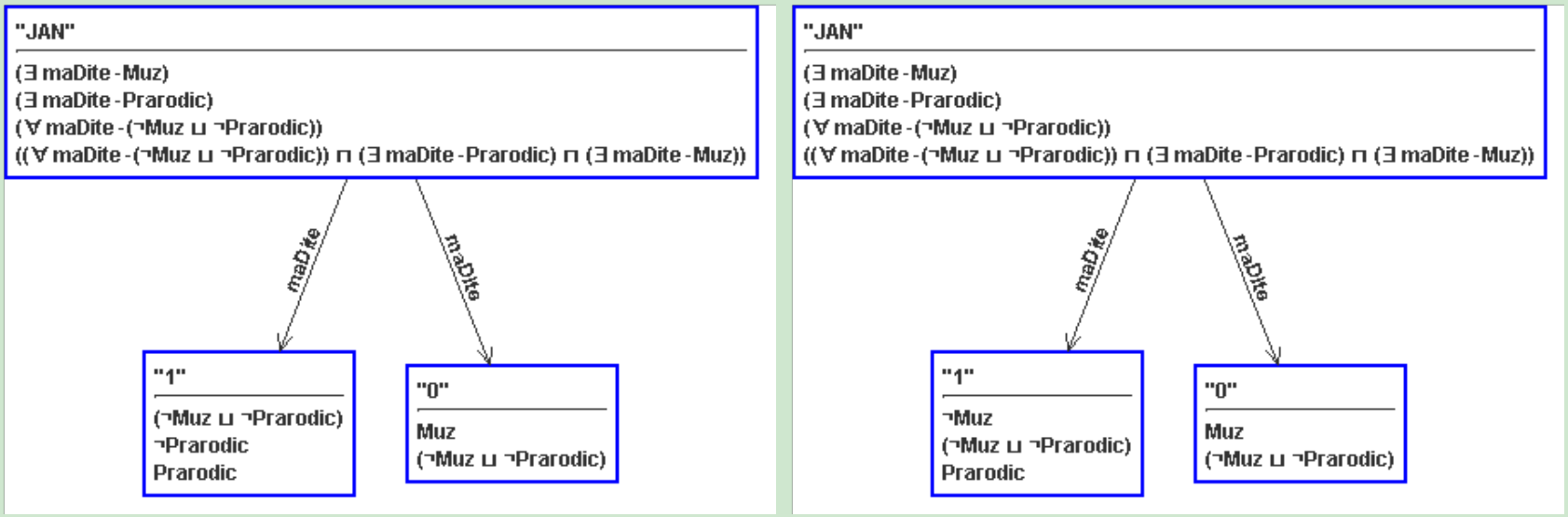
- Dosud jsme prováděli pouze deterministická pravidla (máme stále jediný graf zúplnění). Nyní není žádné deterministické pravidlo již aplikovatelné.

Příklad běhu TA (3)

Example

. . .

- Dosud jsme prováděli pouze deterministická pravidla (máme stále jediný graf zúplnění). Nyní není žádné deterministické pravidlo již aplikovatelné.
- Nyní můžeme použít $\sqcup$-pravidlo, a to na koncept $\neg Muz \sqcup \neg Rodic$ buď v ohodnocení vrcholu "0", nebo v ohodnocení vrcholu "1". Jeho aplikací např. na "1" získáme stav $\{G_5, G_6\}$ (G_5 vlevo, G_6 vpravo)

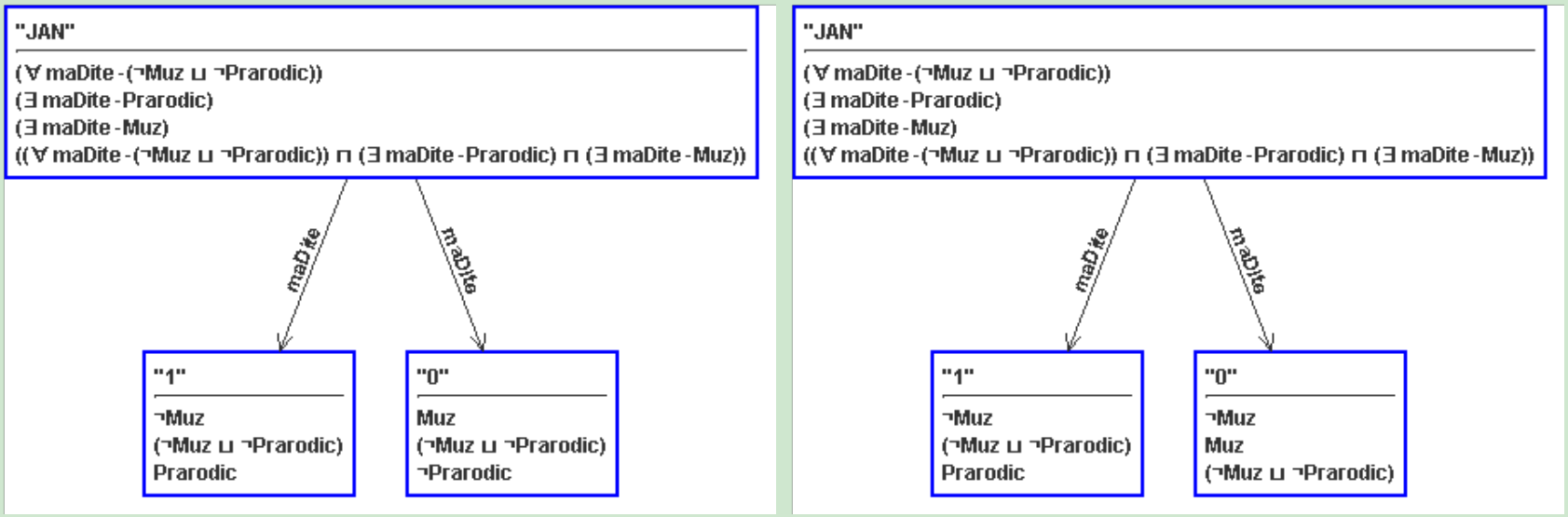


Příklad běhu TA (4)

Example

. . .

- Vidíme, že G_5 obsahuje přímý spor ve vrcholu "1". Zbývá tedy vyšetřit graf G_6. Aplikací $\sqcup$-pravidla získáme stav $\{G_5, G_7, G_8\}$, kde G_7 (vlevo), G_8 (vpravo) vznikly z G_6 :

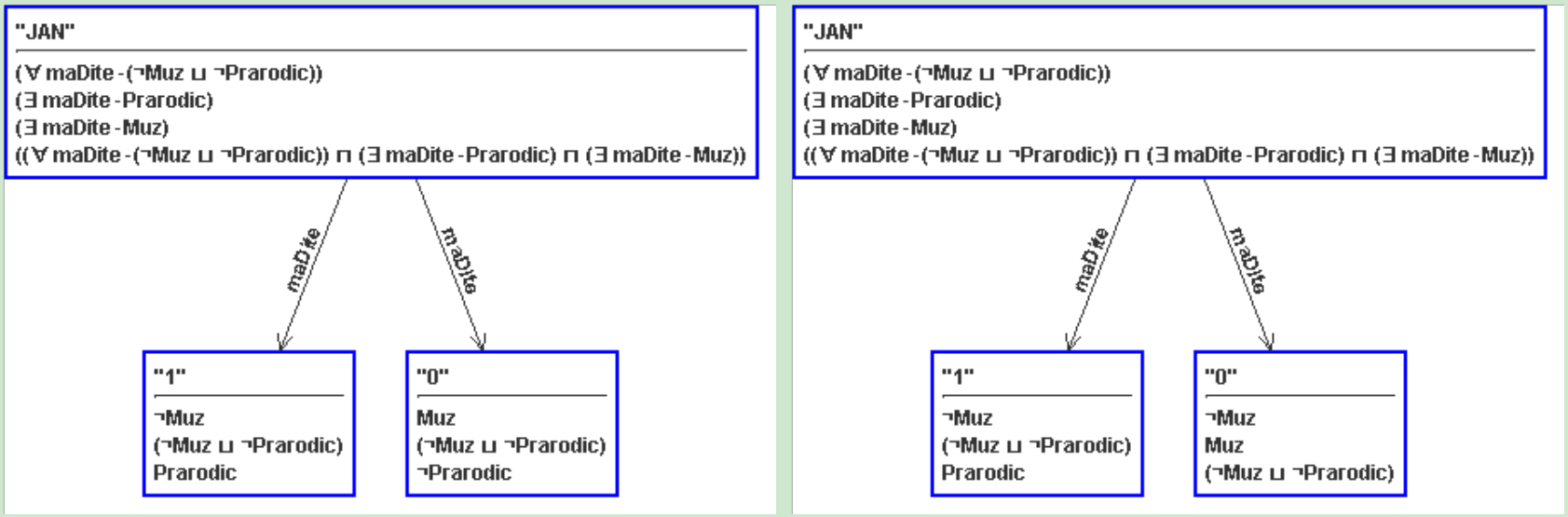


Příklad běhu TA (4)

Example

. . .

- Vidíme, že G_5 obsahuje přímý spor ve vrcholu "1". Zbývá tedy vyšetřit graf G_6. Aplikací $\sqcup$-pravidla získáme stav $\{G_5, G_7, G_8\}$, kde G_7 (vlevo), G_8 (vpravo) vznikly z G_6 :



- G_7 je již úplný a bez přímého sporu.

Příklad běhu TA (5)

Example

... Z G_7 můžeme vytvořit kanonický model $\mathcal{I}_2$. Jedná se o jediný model $\mathcal{K}_2$?

- $\Delta^{\mathcal{I}_2} = \{Jan, i_1, i_2\}$,

Příklad běhu TA (5)

Example

... Z G_7 můžeme vytvořit kanonický model $\mathcal{I}_2$. Jedná se o jediný model $\mathcal{K}_2$?

- $\Delta^{\mathcal{I}_2} = \{Jan, i_1, i_2\}$,
- $maDite^{\mathcal{I}_2} = \{\langle Jan, i_1\rangle, \langle Jan, i_2\rangle\}$,

Příklad běhu TA (5)

Example

... Z G_7 můžeme vytvořit kanonický model $\mathcal{I}_2$. Jedná se o jediný model $\mathcal{K}_2$?

- $\Delta^{\mathcal{I}_2} = \{Jan, i_1, i_2\}$,
- $maDite^{\mathcal{I}_2} = \{\langle Jan, i_1\rangle, \langle Jan, i_2\rangle\}$,
- $Prarodic^{\mathcal{I}_2} = \{i_1\}$,

Příklad běhu TA (5)

Example

... Z G_7 můžeme vytvořit kanonický model $\mathcal{I}_2$. Jedná se o jediný model $\mathcal{K}_2$?

- $\Delta^{\mathcal{I}_2} = \{Jan, i_1, i_2\}$,
- $maDite^{\mathcal{I}_2} = \{\langle Jan, i_1\rangle, \langle Jan, i_2\rangle\}$,
- $Prarodic^{\mathcal{I}_2} = \{i_1\}$,
- $Muz^{\mathcal{I}_2} = \{i_2\}$,

Příklad běhu TA (5)

Example

... Z G_7 můžeme vytvořit kanonický model $\mathcal{I}_2$. Jedná se o jediný model $\mathcal{K}_2$?

- $\Delta^{\mathcal{I}_2} = \{Jan, i_1, i_2\}$,
- $maDite^{\mathcal{I}_2} = \{\langle Jan, i_1\rangle, \langle Jan, i_2\rangle\}$,
- $Prarodic^{\mathcal{I}_2} = \{i_1\}$,
- $Muz^{\mathcal{I}_2} = \{i_2\}$,
- $\text{“}JAN\text{''}^{\mathcal{I}_2} = Jan$, $\text{“}0\text{''}^{\mathcal{I}_2} = i_2$, $\text{“}1\text{''}^{\mathcal{I}_2} = i_1$,

Obecné inkluze

Ukázali jsme si tablový algoritmus pro konzistenci $\mathcal{K} = (\emptyset, \mathcal{A})$. My však již umíme použít stejný algoritmus i pro některé neprázdné TBOXy. Jaké ?

- Jak se však změní situace pro obecnou $\mathcal{K} = (\mathcal{T}, \mathcal{A})$?

Obecné inkluze

Ukázali jsme si tablový algoritmus pro konzistenci $\mathcal{K} = (\emptyset, \mathcal{A})$. My však již umíme použít stejný algoritmus i pro některé neprázdné TBOXy. Jaké ?

- Jak se však změní situace pro obecnou $\mathcal{K} = (\mathcal{T}, \mathcal{A})$?
- Máme $\mathcal{T}$ obsahující axiomy ve tvaru $C_i \sqsubseteq D_i$ pro $1 \leq i \leq n$. Takový TBOX lze transformovat do jediného axiomu

$$\top \sqsubseteq \top_C$$

Obecné inkluze

Ukázali jsme si tablový algoritmus pro konzistenci $\mathcal{K} = (\emptyset, \mathcal{A})$. My však již umíme použít stejný algoritmus i pro některé neprázdné TBOXy. Jaké ?

- Jak se však změní situace pro obecnou $\mathcal{K} = (\mathcal{T}, \mathcal{A})$?
- Máme $\mathcal{T}$ obsahující axiomy ve tvaru $C_i \sqsubseteq D_i$ pro $1 \leq i \leq n$. Takový TBOX lze transformovat do jediného axiomu

$$\top \sqsubseteq \top_C$$

kde $\top_C$ označuje koncept $(\neg C_1 \sqcup D_1) \sqcap \ldots \sqcap (\neg C_n \sqcup D_n)$

Obecné inkluze

Ukázali jsme si tablový algoritmus pro konzistenci $\mathcal{K} = (\emptyset, \mathcal{A})$. My však již umíme použít stejný algoritmus i pro některé neprázdné TBOXy. Jaké ?

- Jak se však změní situace pro obecnou $\mathcal{K} = (\mathcal{T}, \mathcal{A})$?
- Máme $\mathcal{T}$ obsahující axiomy ve tvaru $C_i \sqsubseteq D_i$ pro $1 \leq i \leq n$. Takový TBOX lze transformovat do jediného axiomu

$$\top \sqsubseteq \top_C$$

  kde $\top_C$ označuje koncept $(\neg C_1 \sqcup D_1) \sqcap \ldots \sqcap (\neg C_n \sqcup D_n)$
- Pro každý model $\mathcal{I}$ ontologie $\mathcal{K}$ musí každý element $\Delta^{\mathcal{I}}$ patřit do interpretace konceptu na pravé straně. Jak toho docílit ?

Obecné inkluze (2)

Co třeba takto ?

$\rightarrow_{\sqsubseteq}$ pravidlo

Obecné inkluze (2)

Co třeba takto ?

$\rightarrow_{\sqsubseteq}$ pravidlo

if $\top_C \notin L_G(a)$ pro nějaké $a \in V_G$.

Obecné inkluze (2)

Co třeba takto ?

$\rightarrow_{\sqsubseteq}$ pravidlo

if $\top_C \notin L_G(a)$ pro nějaké $a \in V_G$.

then $S' = S \cup \{G'\} \setminus \{G\}$, kde $G' = (V_G, E_G, L_{G'})$, a $L_{G'}(a) = L_G(a) \cup \{\top_C\}$ a jinak se shoduje s L_G.

Obecné inkluze (2)

Co třeba takto ?

$\rightarrow_{\sqsubseteq}$ pravidlo

if $\top_C \notin L_G(a)$ pro nějaké $a \in V_G$.

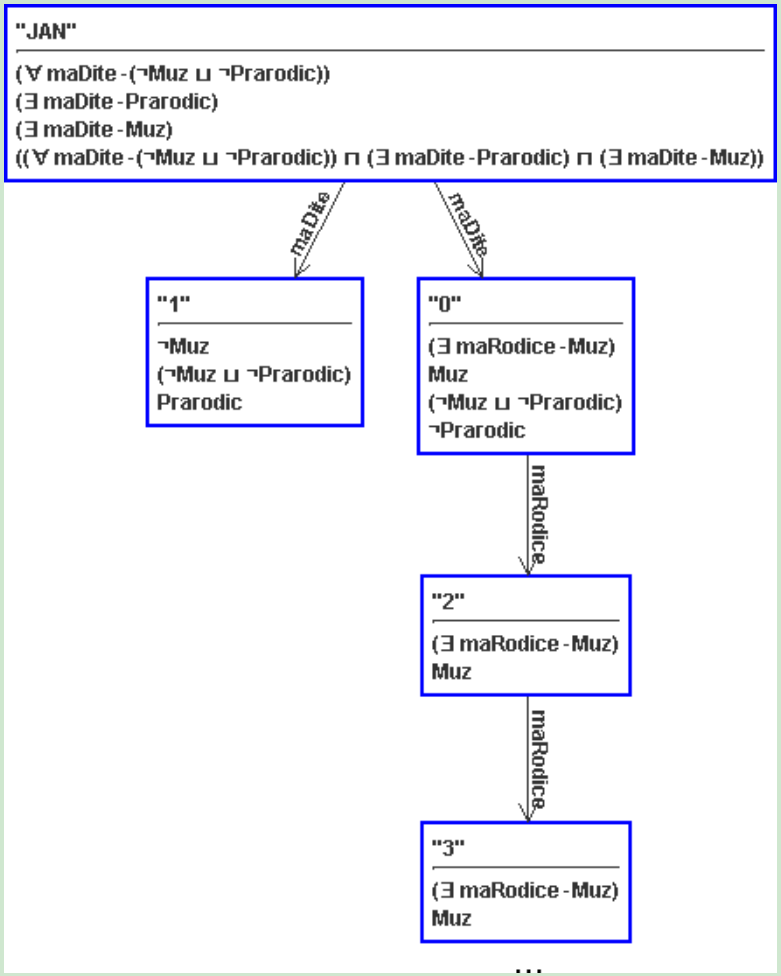
then $S' = S \cup \{G'\} \setminus \{G\}$, kde $G' = (V_G, E_G, L_{G'})$, a $L_{G'}(a) = L_G(a) \cup \{\top_C\}$ a jinak se shoduje s L_G.

Příklad

Uvažujme $\mathcal{K}_3 = (\{Muz \sqsubseteq \exists maRodice \cdot Muz\}, \mathcal{A}_2)$. Potom $\top_C$ je $\neg Muz \sqcup \exists maRodice \cdot Muz$. Použijme nyní dříve představený tablový algoritmus a obohacený o $\rightarrow_{\sqsubseteq}$ pravidlo. Několikerou aplikací sekvence pravidel $\rightarrow_{\sqsubseteq}$, $\rightarrow_{\sqcup}$, $\rightarrow_{\exists}$ na G_7 (který již není úplný vzhledem k $\rightarrow_{\sqsubseteq}$ pravidlu) z předchozího příkladu dostáváme ...

Obecné inkluze (3)

Příklad



. . . tento algoritmus nemusí skončit ☹.

Blokování v TA

- TA se snaží nalézt nekonečný model. Je tedy jej třeba přinutit, aby tento nekonečný model reprezentoval konečným grafem zúplnění.

Blokování v TA

- TA se snaží nalézt nekonečný model. Je tedy jej třeba přinutit, aby tento nekonečný model reprezentoval konečným grafem zúplnění.
- Mechanismu, který vynutí reprezentaci nekonečného modelu konečným grafem zúplnění, se říká *blokování*.

Blokování v TA

- TA se snaží nalézt nekonečný model. Je tedy jej třeba přinutit, aby tento nekonečný model reprezentoval konečným grafem zúplnění.
- Mechanismu, který vynutí reprezentaci nekonečného modelu konečným grafem zúplnění, se říká *blokování*.
- Blokování zajišťuje, že odvozovací pravidla bude možné aplikovat jen dokud se nezačnou jimi prováděné změny "dostatečně pravidelně opakovat".

Blokování v TA

- TA se snaží nalézt nekonečný model. Je tedy jej třeba přinutit, aby tento nekonečný model reprezentoval konečným grafem zúplnění.
- Mechanismu, který vynutí reprezentaci nekonečného modelu konečným grafem zúplnění, se říká *blokování*.
- Blokování zajišťuje, že odvozovací pravidla bude možné aplikovat jen dokud se nezačnou jimi prováděné změny "dostatečně pravidelně opakovat".
- Pro $\mathcal{ALC}$ lze ukázat, že dostačuje tzv. *podmnožinové blokování*:

Blokování v TA

- TA se snaží nalézt nekonečný model. Je tedy jej třeba přinutit, aby tento nekonečný model reprezentoval konečným grafem zúplnění.
- Mechanismu, který vynutí reprezentaci nekonečného modelu konečným grafem zúplnění, se říká *blokování*.
- Blokování zajišťuje, že odvozovací pravidla bude možné aplikovat jen dokud se nezačnou jimi prováděné změny "dostatečně pravidelně opakovat".
- Pro $\mathcal{ALC}$ lze ukázat, že dostačuje tzv. *podmnožinové blokování*:
  - **V grafu zúplnění G je vrchol x (nevyskytující se v ABOXu $\mathcal{A}$) blokován vrcholem y, vede-li orientovaná cesta od y k x a $L_G(x) \subseteq L_G(y)$.**

Blokování v TA

- TA se snaží nalézt nekonečný model. Je tedy jej třeba přinutit, aby tento nekonečný model reprezentoval konečným grafem zúplnění.
- Mechanismu, který vynutí reprezentaci nekonečného modelu konečným grafem zúplnění, se říká *blokování*.
- Blokování zajišťuje, že odvozovací pravidla bude možné aplikovat jen dokud se nezačnou jimi prováděné změny "dostatečně pravidelně opakovat".
- Pro $\mathcal{ALC}$ lze ukázat, že dostačuje tzv. *podmnožinové blokování*:
  - **V grafu zúplnění G je vrchol x (nevyskytující se v ABOXu $\mathcal{A}$) blokován vrcholem y, vede-li orientovaná cesta od y k x a $L_G(x) \subseteq L_G(y)$.**
- Všechna inferenční pravidla je možné aplikovat pouze, není-li vrchol a v jejich definici blokován jiným vrcholem.

Blokování v TA (2)

- V předchozím příkladu tedy blokování zabezpečí, že vrchol "2" je blokován vrcholem "0" a k další expanzi již nedojde. *Jakému modelu takovýto graf zúplnění odpovídá ?*

Blokování v TA (2)

- V předchozím příkladu tedy blokování zabezpečí, že vrchol "2" je blokován vrcholem "0" a k další expanzi již nedojde. *Jakému modelu takovýto graf zúplnění odpovídá ?*
- **Představený TA s podmnožinovým blokováním je již korektní, úplnou a konečnou rozhodovací procedurou pro jazyk $\mathcal{ALC}$.**

Pohrajme si . . .

- http://krizik.felk.cvut.cz/km/dl/index.html